

INALCO M2 Master ingénierie multilingue 2025

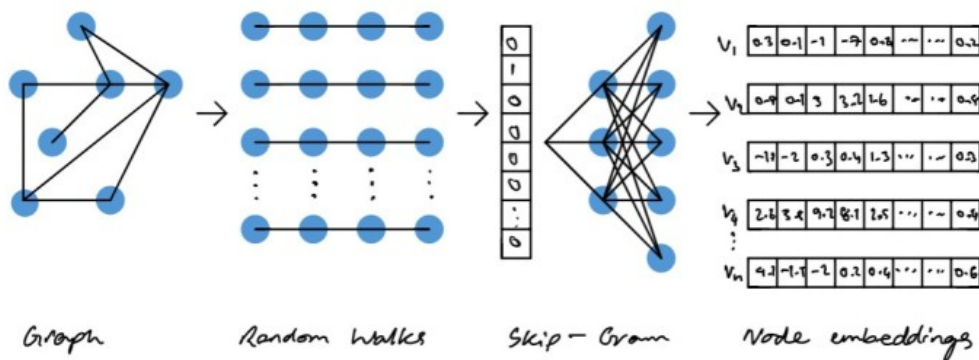
Ingénierie des connaissances : Contrôle n°2 :

NOM
PRÉNOM

I) Révisions

1) Décrivez les étapes principales de la Data preparation et leur utilité.
Qu'est ce qu'un connecteur ? À quoi cela sert-il ?

2) Comment peut-on construire les embeddings des nœuds d'un graphe ?



Expliquez le fonctionnement de la méthode node2vec. Quelles sont ses limitations et comment peut-on les dépasser ?

II) Nouveaux chapitres

3) La fonction mystère : Voici une fonction vraiment chouette, mais j'ai oublié son nom !

```
def my_little_algo(data, n = 10, max_iters=100):
    keys, vectors = zip(*data.items())
    vectors = np.array(vectors)
    centroids = vectors[random.sample(range(len(vectors)), n)]

    for _ in range(max_iters):
        clusters = {i: [] for i in range(n)}
        for i, vec in enumerate(vectors):
            cluster = np.argmin([np.linalg.norm(vec - c) for c in centroids])
            clusters[cluster].append(i)

        new_centroids = np.array([np.mean(vectors[clusters[i]], axis=0) if clusters[i] else centroids[i]
                                  for i in range(n)])

        if np.allclose(centroids, new_centroids):
            break
        centroids = new_centroids

    return {keys[i]: cluster for cluster, indices in clusters.items() for i in indices}
```

Quel est cet algorithme ? Expliquez son fonctionnement global, expliquez quels sont ces paramètres puis expliquez ce qu'il fait ligne à ligne.

4) Voici une classe python qui sert à réaliser des clusterings hiérarchiques.

```
class HierarchicalClustering:
    def __init__(self, data: List[Any], similarity_func: Callable, termination_cond: Callable):
        self.data = data
        self.similarity_func = similarity_func
        self.termination_cond = termination_cond
        self.clusters = [[item] for item in data] # Start with each item in its own cluster

    def run(self):
        while not self.termination_cond(self.clusters):
            max_sim = float('-inf')
            pair_to_merge = (None, None)
            for i in range(len(self.clusters)):
                for j in range(i + 1, len(self.clusters)):
                    sim = self.similarity_func(self.clusters[i], self.clusters[j])
                    if sim > max_sim:
                        max_sim = sim
                        pair_to_merge = (i, j)
            if pair_to_merge == (None, None):
                i, j = pair_to_merge
                new_cluster = self.clusters[i] + self.clusters[j]
                del self.clusters[j]
                del self.clusters[i]
                self.clusters.append(new_cluster)
        return self.clusters
```

Expliquez l'importance de la condition de terminaison et de la mesure de similarité lorsqu'on fait un clustering hiérarchique.

Quel est le type des variables `similarity_func` et `termination_cond` ?

Implémentez une condition de terminaison pour que l'algorithme s'arrête quand on a obtenu `k` clusters.

5) A quoi sert un système de recommandation ? Quelles sont les données que l'on peut utiliser pour construire un système de recommandation ? Présentez plusieurs approches possibles.

6) Définissez ce qu'est le Graph RAG.

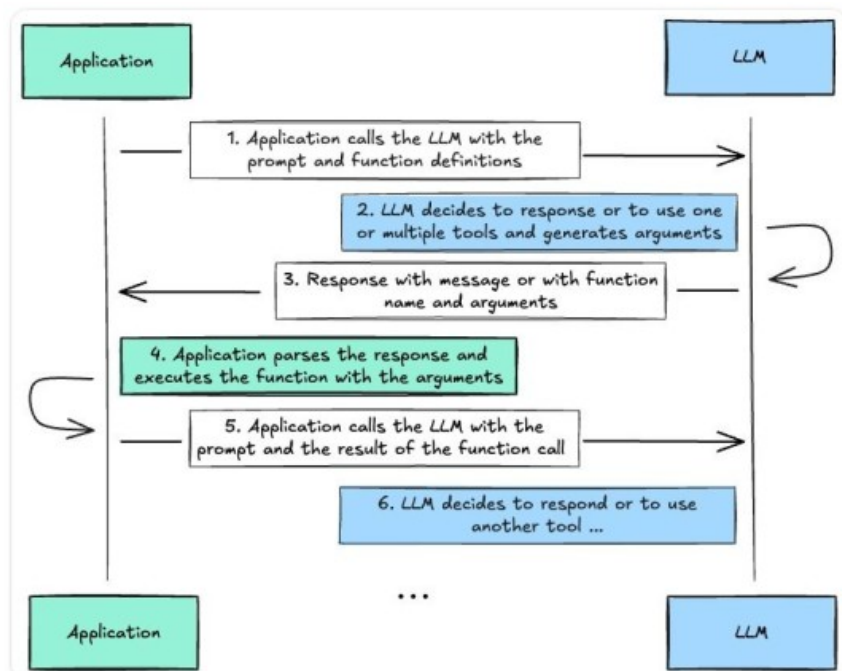
Quelles sont les différentes approches qui ont été explorées pour l'implémenter?

7) Quels sont les composants nécessaires pour construire un agent IA ?

Expliquez leur rôle. Présentez l'architecture ReAct à l'aide d'un schéma.

8) Qu'est ce que le *function calling* (ou *tool calling*) ?

Expliquez les étapes de ce mécanisme en précisant quelle programme l'exécute (LLM, application...)




III) Culture scientifique et lecture d'articles.

9) Expliquez la stratégie nommée « Graph RAG » présentée dans le papier « From Local to Global: A Graph RAG Approach to Query-Focused Summarization ». Quelles sont les limites de cette approche ?

10) Quels sont les différents paramètres qui peuvent jouer sur la compréhension d'un graphe par un LLM (4 points de réponse minimum attendus)

BONUS :

le retour de Jérémie. Commentez cette publication à partir de vos connaissances.



Jérémie Ravenel ↑ Nouveaux posts
⚡ Building bridges @naas.ai Universal Data & AI Platform | Research ...
[Accéder à mon site web](#)
2 j • 🌐

...

×

Why are agents just the tip of the iceberg?

Because behind every intelligent agent lies a powerful agentic infrastructure, the invisible system that makes everything possible.

Everyone is talking about AI agents right now. But focusing only on the agent is like praising a great dish without acknowledging the kitchen, the chef, or the ingredients. The real innovation happens beneath the surface.

Here are the 7 core components of an agentic infrastructure:

1. Agents: the interface layer, executing tasks and interacting with users.
2. Integrations: the bridges connecting the agent to external systems, tools, and APIs.
3. Pipelines: the flow of data from raw sources to usable outputs.
4. Workflows: the logic and orchestration that guide how tasks unfold across tools and teams.
5. Ontologies: the brain of the system, ensuring shared understanding across humans and machines.
6. Analytics: the feedback loop that helps the system learn, optimize, and make informed decisions.
7. Apps: the productized outputs that package value into tangible, usable interfaces.

You don't build agents.
You build systems that enable agents to operate with intelligence, autonomy, and reliability.

That's where the magic really happens.
That's how AI stops being a gimmick and becomes infrastructure.