

Knowledge Management & Graphs Concepts

Fabien FURFARO

24 juillet 2025

1 Présentation du contexte et du besoin

- Historique
- Défis actuels et objectifs
- Composants d'un KMS moderne
- Cas d'usage : Curiosity
- Bénéfices et limites
- Exemple concret : aéronautique
- Approche Capgemini « Trust before technology »

2 Présentation des compétences nécessaires

- Théorie et concepts fondamentaux
 - Graphes et connaissance
 - Sémantique, ontologies
 - Vectorisation et recherche sémantique
 - IA et NLP
- Mathématiques et algorithmes de base
- Exemple pratique de requête sur un LPG

La gestion des connaissances (Knowledge Management) a évolué en plusieurs phases :

- Avant 1980 : pratiques informelles, catalogage manuel, échanges oraux.
- 1980-1995 : premiers outils informatiques et bases de connaissances structurées.
- 1995-2010 : plateformes collaboratives, moteurs de recherche spécialisés.
- 2010 et après : NLP/NLU, graph databases, transformers, accent sur gouvernance et modélisation.

- **Fragmentation** de l'information et silos de données.
- **Hétérogénéité** des systèmes.
- **Perte du savoir** par turnover ou absence de capitalisation.

Objectif : créer des plateformes « Google-like », capables de retrouver et structurer la connaissance avec un **KMS** (Knowledge Management System) s'appuyant sur curation humaine, ontologie métier, indexation croisée et technologies avancées (ex : knowledge graph, IA).

Composants d'un KMS moderne

- Modèles formels, ontologies.
- Moteurs de règle, systèmes experts.
- Knowledge graph, sémantique.
- IA (NLP/NLU, entity recognition, retrieval).
- Interfaces LLM/RAG, intégration outils métier.

- KMS basé sur un **Linked Property Graph (LPG)** développé en C#.
- Détection automatique et pondération des entités (NLP).
- Recherche par similarité (subgraph embedding, vectorisation texte).
- Rapidité sur grands volumes (plusieurs centaines de milliers de docs).

- Gain de temps, pertinence, efficacité de support client
- Sécurité du patrimoine informationnel
- Défis persistants de désilotage, capitalisation experte, automatisation des process

Exemple concret : aéronautique

Déploiement progressif chez un avionneur : de 15 000 utilisateurs internes à plus de 100 000 clients externes, grâce à la puissance du knowledge graph pour résoudre des problématiques techniques complexes.

- Diagnostic métier et technique approfondi, co-construction humaine + technologique, prototypage rapide permettant d'incarner la valeur ajoutée.

- **Graphe orienté ou non orienté** : $G = (V, E)$ avec V = nœuds (entités), E = arêtes (relations).
- **Linked Property Graph (LPG)** : chaque nœud/arête peut avoir des propriétés clé-valeur.
- **Requête sur graphe** : trouver des sous-graphes isomorphes, calcul de chemins, recherche de motifs.

- **Ontologie** : formalise les concepts et relations d'un domaine (ex : OWL, RDF Schema) ; utile pour donner du sens et permettre des inférences au-delà du simple graphe.

- **Embeddings** : vectorisation des nœuds/document associés pour calculer la similarité (cosinus, euclidienne) :

$$\cos(\theta) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\| \|\mathbf{v}_2\|}$$

- Techniques courantes : Word2Vec, GloVe, transformers (BERT/LLM).

- Extraction d'entités : reconnaissance automatique des entités dans un texte.
- Similarité sémantique : utiliser des modèles de langage pour relier de nouveaux cas à l'historique.

- Algorithme de parcours de graphe (DFS, BFS).
- Pagerank pour déterminer l'importance d'un nœud.
- Recherche de plus courts chemins (Dijkstra, A^*).
- Recherche de sous-graphes similaires (isomorphisme de sous-graphe, graph embedding).

Exemple d'algorithme : BFS en C#

```
1 Queue<Node> queue = new Queue<Node>();
2 HashSet<Node> visited = new HashSet<Node>();
3 queue.Enqueue(startNode);
4
5 while (queue.Count > 0)
6 {
7     Node current = queue.Dequeue();
8     if (!visited.Contains(current))
9     {
10         visited.Add(current);
11         foreach (Node neighbor in current.Neighbors)
12         {
13             queue.Enqueue(neighbor);
14         }
15     }
16 }
```

Exemple de similarité cosinus entre deux embeddings

```
1 public double CosineSimilarity(double[] v1, double[]  
   v2)  
2 {  
3     double dot = 0.0, mag1 = 0.0, mag2 = 0.0;  
4     for (int i = 0; i < v1.Length; i++)  
5     {  
6         dot += v1[i] * v2[i];  
7         mag1 += v1[i] * v1[i];  
8         mag2 += v2[i] * v2[i];  
9     }  
10    return dot / (Math.Sqrt(mag1) * Math.Sqrt(mag2));  
11 }
```


Formule de base :

$$PR(u) = \frac{1 - d}{N} + d \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

avec d = facteur d'amortissement (classiquement 0.85), N = nombre total de nœuds.

Définition d'un nœud et d'une relation en C#

```
1 public class Node
2 {
3     public string Id { get; set; }
4     public Dictionary<string, object> Properties { get
        ; set; }
5 }
6
7 public class Relationship
8 {
9     public Node From { get; set; }
10    public Node To { get; set; }
11    public string Type { get; set; }
12    public Dictionary<string, object> Properties { get
        ; set; }
13 }
```

Recherche des voisins d'un nœud (extraction de sous-graphe)

```
1 public List<Node> GetNeighbors(Node node, List<  
    Relationship> relationships)  
2 {  
3     return relationships  
4         .Where(r => r.From == node)  
5         .Select(r => r.To)  
6         .ToList();  
7 }
```