

Veille Technologique

Technique

13 octobre 2026 – Fréquence : Bi-hebdomadaire (Mardi et Jeudi)

Synthèse & Sujet du Jour

Docling, Granite & OpenShift AI : Ingestion multimodale, décodage guidé et extraction structurée RAG

La performance des architectures RAG dépend de la fidélité de l'ingestion documentaire. Les flux PDF bruts brisent l'ordre sémantique au profit de l'ordre d'impression matriciel (Tm). Docling résout cette fracture via des modèles de vision de layout et son format intermédiaire DoclingDocument. L'extraction de schémas JSON avec IBM Granite sur OpenShift AI éclaire l'arbitrage critique entre décodage guidé (vLLM / llguidance / xgrammar) et guardrails défensifs pour neutraliser les hallucinations forcées.

1. Périmètre

Famille de veille : 2. Métiers, Compétences & Technologies (Indicateur Qualiopi 24).

Fréquence d'exécution : Bi-hebdomadaire (Mardi et Jeudi à 08h30).

Diffusion : Radar Web Aperta (FR) — LinkedIn Aperta Scientia — X @ApertaScientia.

2. Sujet du Jour — Article Publié sur le Radar Web

Dans la construction de pipelines RAG (*Retrieval-Augmented Generation*) et d'architectures d'IA d'entreprise, la qualité des représentations vectorielles dépend directement de la fidélité de l'ingestion documentaire. Les organisations disposent de volumes massifs de données stratégiques captives de formats bureautiques statiques (PDF, Word, feuilles de calcul). Or, le traitement naïf de ces flux se heurte à une fracture systémique : la divergence entre l'**ordre d'impression** (*printing order*) et l'**ordre de lecture** (*reading order*).

1. La fracture documentaire : ordre d'impression vs ordre de lecture

Le format PDF standard n'a pas été conçu pour la structuration sémantique de données, mais pour la fidélité de restitution graphique sur imprimante. Le flux de contenu d'un PDF brut est piloté par des opérateurs matriciels (tels que Tm qui positionne le curseur à des coordonnées cartésiennes précises X et Y).

En conséquence, l'ordre sériel dans le flux binaire reflète la séquence d'exécution graphique et non la hiérarchie logique. Un titre de section peut être encodé après un pied de page ou une marge latérale. Les découpages en colonnes et les tableaux complexes sont désintégrés en fragments incohérents par les outils de parsing naïfs (pypdf, OCR sans analyse structurelle), polluant irrémédiablement le chunking et la recherche vectorielle.

2. L'approche hybride de Docling : Vision de layout et modèle intermédiaire

Développée par IBM Research et soutenue par Red Hat, la bibliothèque **Docling** implémente une approche hybride associant des parseurs multi-formats (35 formats bureautiques) et des modèles de vision neuronaux dédiés à l'analyse de mise en page (*Document Layout Analysis*) :



- **Segmentation par modèle de vision** : Docling exploite un réseau entraîné sur plus de 150 000 documents annotés (tel que `docling-layout-heron`) pour prédire les boîtes englobantes (*bounding boxes*) et leur assigner un label sémantique (`[section_header]`, `[text]`, `[table]`, `[caption]`).
- **Projection géométrique inverse** : Les coordonnées Tm du flux d'impression PDF sont projetées et réassociées à leurs boîtes englobantes parentes, restaurant rigoureusement l'ordre de lecture original.
- **Représentation intermédiaire DoclingDocument** : La structure résultante est matérialisée dans un modèle objet hiérarchique Pydantic préservant l'arbre des sections et les items de texte ordonnés, avant export vers un format Markdown assaini.

3. Extraction structurée : Décodage guidé vs Guardrails défensifs

Une fois le document exporté en Markdown, l'étape suivante consiste à extraire un schéma JSON typé via un modèle compact tel qu'**IBM Granite 3.1** ou **Granite 4:3B**. Deux approches d'ingénierie s'opposent :

- **Décodage guidé au runtime (vLLM / llguidance / xgrammar)** : Un masque de logits dynamique est appliqué à chaque génération de token. Les tokens violant la grammaire du JSON Schema sont exclus de la distribution. **Écueil critique** : si une propriété requise par le schéma est absente du document d'origine, le décodage guidé interdit l'omission ou la valeur nulle ; le modèle est mathématiquement contraint d'**halluciner une valeur plausible** pour valider le contrat d'API.
- **Guardrails défensifs et boucle de retry** : L'approche robuste privilégie une émission fluide suivie de validateurs déterministes (nettoyage de code fences, clamping sur vocabulaires contrôlés, typage Pydantic avec nullabilité explicite) et d'une boucle de rétroaction renvoyant les erreurs de schéma au modèle.

4. Vectorisation ciblée et évitement de la dilution d'embedding

Moyenner l'intégralité d'un document complexe dans un unique vecteur dense (ex. 1536 dimensions) provoque une **dilution d'embedding** (*Embedding Dilution*), où la superposition d'informations hétérogènes annule les signaux discriminants dans l'espace latent. L'architecture décompose les entités en facettes modulaires (rôle, compétences, domaine) indexées indépendamment dans **PostgreSQL / pgvector** ou **Milvus sur ODF** avec recherche hybride (dense + dense + BM25).

5. Opérationnalisation sur Red Hat OpenShift AI (RHOAI)

La plateforme OpenShift industrialise cette architecture via des primitives cloud-native éprouvées :

- **GPU Operator** : Ordonnancement accéléré des conteneurs Docling pour un traitement des layouts à haut débit.
- **ServingRuntimes vLLM & KServe** : Gestion déclarative du cycle de vie des modèles IBM Granite avec gestion fine du KV-cache.
- **Data Science Pipelines (Kubeflow Pipelines v2)** : Automatisation du flux continu d'ingestion et d'indexation vectorielle.
- **OpenShift Data Foundation (ODF)** : Stockage persistant souverain sur volumes Ceph RBD chiffrés et NooBaa S3.

3. Impacts Pédagogiques & Cursus Aperta Scientia



- **Cursus AS300 (AI Platform Engineer — 399h)** : Ingénierie complète des pipelines de données pour l'IA — ingestion multimodale haute fidélité avec **Docling**, ServingRuntimes **vLLM** et modèles **IBM Granite** sur **Red Hat OpenShift AI (RHOAI)**, arbitrage technique *Guided Decoding* (llguidance) vs guardrails défensifs, prévention de l'*embedding dilution* en architectures RAG, et persistance vectorielle sur **Milvus** / **ODF**.
- **Cursus AS200 (DevOps Platform Engineer — 399h)** : Déploiement déclaratif GitOps des ServingRuntimes avec **OpenShift GitOps (ArgoCD)**, cycle de vie du **GPU Operator**, isolation multi-tenant des namespaces d'ingestion et sécurisation des API d'inférence.
- **Compétence clé validée** : Concevoir une chaîne d'ingestion RAG résiliente, évaluer l'impact des contraintes grammaticales d'inférence sur la véracité des données et opérer des ServingRuntimes GPU sur Kubernetes.

4. Sources & Références Techniques

- **Red Hat Developer Blog** — *Build an AI matching engine with Docling, Granite & OpenShift AI* : <https://developers.redhat.com/articles/2026/10/12/build-ai-matching-engine-docling-granite-openshift-ai> (consulté le 13 octobre 2026)
- **Docling Layout Analysis Research Paper** — *Advanced Layout Analysis Models for Docling* (arXiv:2509.11720) : <https://arxiv.org/abs/2509.11720>
- **Docling Project** — *Repository & Models* : <https://github.com/DS4SD/docling> — <https://huggingface.co/docling-project/docling-layout-heron>
- **vLLM Documentation** — *Structured Outputs (llguidance & xgrammar)* : https://docs.vllm.ai/en/stable/features/structured_outputs/
- **Red Hat OpenShift AI** — *Documentation officielle* : https://docs.redhat.com/en/documentation/red_hat_openshift_ai

Contacts & Communauté :

- **Direction des Formations** : learning@apertascientia.io
- **Référent Handicap & Bien-être (Care)** : care@apertascientia.io
- **X (Twitter)** : [@ApertaScientia](https://twitter.com/ApertaScientia)
- **Communauté Discord** : discord.gg/eDJzR5Vk85