ADR-005 : Fusion RRF v2 (pondération question×2 + blending position-aware)

ADR-005 : Fusion RRF v2 (pondération question×2 + blending position-aware)

> *Date : 2026-Q1 — Statut : accepté*

Contexte

Le retrieval combine plusieurs sources :

- Vecteur dense de la question brute (TEI bge-m3)
- Vecteur dense du passage HyDE (Haiku-généré)
- Sparse BM25 sur la question brute
- (Selon intent) chunks META, chunks synergy

Toutes les sources retournent un top-K avec rank et score. Il faut les fusionner en une seule liste classée.

Premier essai (RRF v1, classique) : `score = Σ 1/(60 + rank)` pour chaque chunk présent dans une passe. Inspiré de la littérature standard.

Problèmes constatés :

- **HyDE dilue la précision** : les exact-matches de la question brute sont noyés par les paraphrases HyDE. Si rank 1 dans question + rank 50 dans HyDE → score combiné moyen, pas top-1.
- **Reranker peu confiant** sur du contenu technique abstrait (notes de mécaniques, conditions de victoire) → score reranker ~0 → enterre des bons chunks.

Décision

Fusion RRF v2 avec deux modifications :

1. Pondération question × 2

```
score = 2 × (1 / (60 + rank_question + 1))
      + 1 × (1 / (60 + rank_hyde + 1))
      + 1 × (1 / (60 + rank_bm25 + 1))
```

La question brute pèse double — sa précision compte plus que celle du HyDE qui est une paraphrase.

2. Top-rank bonus

```
if (rank === 0) score += 0.05;
else if (rank === 1 || rank === 2) score += 0.02;
```

Préserve les exact-matches contre la dilution.

3. Blending position-aware (post-rerank)

Au lieu d'écraser le score RRF avec le score reranker :

```
const rrf_position = candidate.rrf_rank;
let weight_rrf, weight_rerank;
if (rrf_position <= 2)        { weight_rrf = 0.75; weight_rerank = 0.25; }
else if (rrf_position <= 9)   { weight_rrf = 0.60; weight_rerank = 0.40; }
else                          { weight_rrf = 0.40; weight_rerank = 0.60; }
```

```
final_score = weight_rrf * rrf_score + weight_rerank * rerank_score;
```

Si la fusion RRF a déjà classé un chunk dans le top, on garde son signal. Si le reranker est confiant et qu'il détrône un mauvais top, son signal compte plus.

Activable / désactivable via `RAG_FUSION_V2_ENABLED` (défaut `true`).

Conséquences

Bonnes

- Les exact-matches de la question survivent au passage HyDE
- Le reranker peu confiant sur du contenu abstrait n'enterre plus systématiquement les bons chunks
- Kill-switch via env var pour A/B test si nécessaire
- Inspiré de tobi/qmd, pattern documenté

Mauvaises

- **Magic numbers** : 0.05, 0.02, 75/25, 60/40, 40/60. Les ratios ont été tunés empiriquement, pas théoriquement.
- **Plus de paramètres à comprendre** pour un dev qui débarque
- Toute modification doit être validée sur le banc d'éval RAG (sinon régression silencieuse)

Tuning

Si tu veux ajuster :

- **Ratios blending** : monter à 85/15 sur top-3 si tu veux encore plus protéger le signal RRF
- **Top-rank bonus** : monter à +0.10 sur rank=0 si exact-matches cruciaux

Toujours `npm run test:rag` avant et après pour comparer la qualité.

Source d'inspiration

- RRF original paper (Cormack 2009)
- tobi/qmd — implémentation référence pour les ratios pondérés

- Banc d'éval RAG interne (boardgame-referee `tests/rag/`)

Revision #1
Created 2026-05-10 15:19:53 UTC by thymon
Updated 2026-05-10 15:19:53 UTC by thymon