Du sens au temps : comprendre la recherche vectorielle avec fraîcheur

Un cours progressif, des vecteurs de base jusqu'à une recherche euclidienne qui combine pertinence et date.

Objectif : score = α × similarité + β × fraîcheur

Contrainte : conserver une recherche native en distance L2

Pour Paul - à partir de notre discussion sur la croissance du vault, les décisions anciennes et le fonctionnement de sqlite-vec.

Ce document fait trois choses

Il démontre les identités utilisées, développe une construction compatible en théorie avec une recherche euclidienne, et expose ses coûts de maintenance. Les paramètres illustratifs ne constituent pas une décision d'architecture.

Statut : étude et support pédagogique. Aucun changement du classement, du schéma SQLite ou du code de production. Les démonstrations sont exactes sur les nombres réels ; l'intégration en float32 et les performances demandent des essais.

COURS / 01

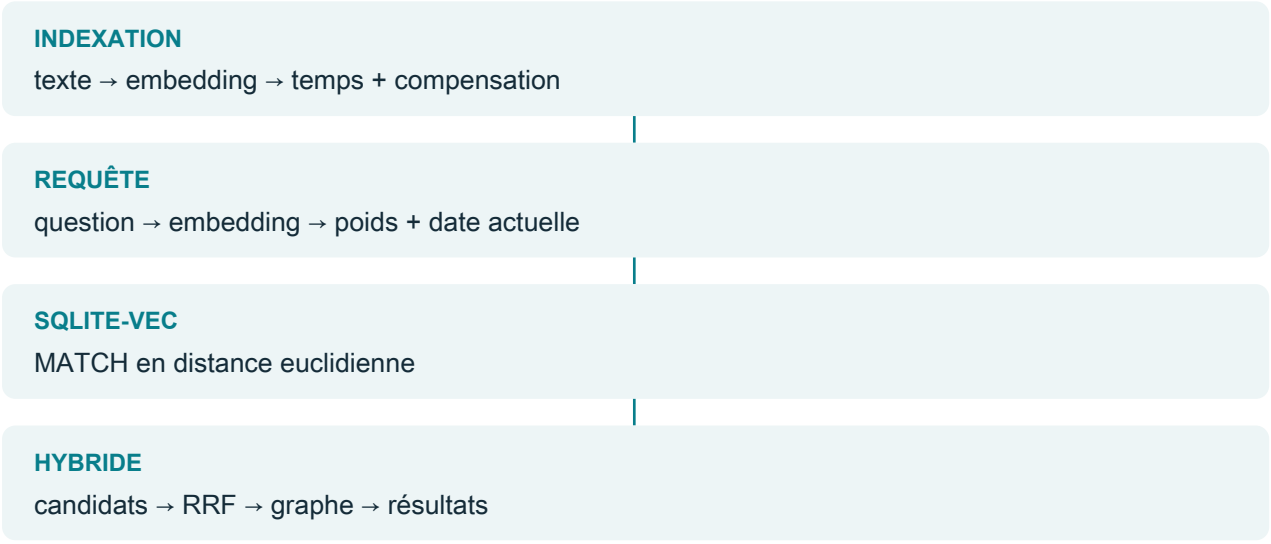
Le problème à résoudre

Comprendre pourquoi un meilleur classement final ne suffit pas toujours.

Dans un grand vault, une décision ancienne peut être très proche de la question et occuper les premières places. Des sessions plus récentes peuvent la modifier sans employer exactement les mêmes mots. Si elles sont éliminées lors de la sélection initiale, un reclassement ultérieur ne les verra jamais.

Deux objectifs distincts

Retrouver les évolutions : faire intervenir la date avant la première coupure. **Interpréter correctement les sources :** vérifier si une décision a été confirmée ou remplacée. La fraîcheur aide le premier objectif ; elle ne prouve pas le second.



Le schéma montre une architecture possible. Aujourd'hui, la branche sémantique ne contient pas cette transformation temporelle. La recherche lexicale et le graphe sont des signaux séparés.

Parcours de lecture

Chapitres 2 à 5 : géométrie. 6 à 10 : calcul de fraîcheur. 11 à 16 : compatibilité euclidienne, bornes et précision numérique. 17 et 18 : exercices et sources.

COURS / 02

Un vecteur et sa longueur

Distinguer la dimension, qui compte les coordonnées, et la norme, qui mesure une longueur.

Un vecteur est une liste de nombres. En deux dimensions, on peut le dessiner comme une flèche. Un embedding contient beaucoup plus de coordonnées ; les mêmes opérations restent valables, même si on ne peut plus le dessiner.

$$v = [3 ; 4]$$

$$\text{dimension}(v) = 2$$

$$||v|| = \sqrt{(3^2 + 4^2)} = 5$$

La normalisation

Normaliser consiste à diviser toutes les coordonnées par la norme. La direction est conservée, mais la nouvelle longueur vaut 1.

$$v_{\text{normalisé}} = [3/5 ; 4/5] = [0{,}6 ; 0{,}8]$$

$$||v_{\text{normalisé}}||^2 = 0{,}36 + 0{,}64 = 1$$

La direction porte alors la comparaison sémantique. Les coordonnées d'un embedding ne sont pas des étiquettes lisibles telles que « réseau » ou « décision » ; elles contribuent ensemble à une représentation apprise.

Pourquoi cela compte ici

L'équivalence entre distance euclidienne et produit scalaire repose sur les normes constantes. Ajouter une coordonnée de temps détruit généralement cette propriété. La compensation étudiée plus loin sert précisément à la restaurer côté notes.

Cas particulier : le vecteur nul ne peut pas être normalisé. La construction du cours suppose des embeddings non nuls de norme 1 ; le moteur doit traiter explicitement les exceptions.

COURS / 03

Le produit scalaire

Une multiplication coordonnée par coordonnée, suivie d'une somme.

$$[a\ ;\ b] \cdot [c\ ;\ d] = a \times c + b \times d$$

$$[1\ ;\ 0] \cdot [0{,}8\ ;\ 0{,}6] = 0{,}8$$

Pour des vecteurs normalisés, le produit scalaire est le cosinus de l'angle entre leurs directions. Une direction identique donne 1, deux directions orthogonales donnent 0, deux directions opposées donnent -1.

Position relative	Produit scalaire
Même direction	1
Directions assez proches	par exemple 0,8
Directions orthogonales	0
Directions opposées	-1

Pourquoi les dimensions ne se mélangent pas

$$[q_1\ ;\ q_2\ ;\ g] \cdot [x_1\ ;\ x_2\ ;\ z]$$

$$= q_1x_1 + q_2x_2 + gz$$

Le facteur temporel g ne multiplie ni x□ ni x□. Les deux premières coordonnées continuent à mesurer le sens ; la dernière produit une contribution temporelle indépendante. La séparation est algébrique, pas une consigne donnée au modèle.

Ce que signifie « ajouter un signal »

Une dimension supplémentaire n'a aucun poids imposé par sa seule présence. Son influence dépend des nombres placés dedans et des coefficients appliqués. On peut construire ces valeurs pour retrouver exactement une formule choisie.

COURS / 04

La distance euclidienne

La formule de Pythagore, étendue à toutes les coordonnées.

$$||q - x||^2 = \text{somme des } (q\square - x\square)^2$$

$$||q - x||^2 = ||q||^2 + ||x||^2 - 2(q \cdot x)$$

La seconde ligne vient du développement de chaque carré : $(a-b)^2 = a^2 + b^2 - 2ab$. En additionnant, on retrouve les deux normes et le produit scalaire.

Lorsque les deux vecteurs sont normalisés

$$||q||^2 = ||x||^2 = 1$$

$$\text{distance}^2 = 2 - 2 \times \text{similarité}$$

Similarité	Distance²	Distance
1	0	0
0,8	0,4	≈ 0,6325
0	2	≈ 1,4142
-1	4	2

La racine carrée est croissante : trier par distance ou par distance au carré donne le même ordre. Maximiser la similarité donne aussi cet ordre, mais dans le sens inverse du nombre mesuré.

La limite de cette équivalence

Si les normes des notes varient, le terme $||x||^2$ varie lui aussi. La distance ne dépend plus seulement du produit scalaire. C’est pourquoi on ne peut pas ajouter le temps puis continuer comme si les vecteurs étaient toujours normalisés.

COURS / 05

Le classement actuel

Ce que fait le code local inspecté pendant cette étude.

La table vectorielle est créée avec vec0 et une colonne float[D], sans métrique explicite. La recherche utilise embedding MATCH avec k. sqlite-vec emploie alors la distance L2 par défaut. Le code transforme ensuite la distance d en score 1/(1+d).

score_actuel = 1 / (1 + distance_L2)

À normes unitaires : distance_L2 = √(2 − 2 × cosinus)

Cette transformation garde le même ordre sémantique. Ses valeurs ne sont cependant pas celles du cosinus : ajouter 0,02 à l'un ou à l'autre n'a pas la même signification.

Sélection, fusion et graphe

Pour une demande de 8 résultats, le moteur hybride sélectionne jusqu'à 64 extraits sémantiques et 64 lexicaux. Leur union contient au plus 128 candidats. Il fusionne leurs rangs avec RRF, ajoute le bonus de graphe et conserve les 8 meilleurs.

taille par branche ≈ max(50 ; 8 × limite)

plafond habituel : 512, et jamais plus que le corpus disponible

score_RRF = poids_sem/(60 + rang_sem)

+ poids_lex/(60 + rang_lex)

Le bonus de graphe vaut 0,5/60 ≈ 0,00833. Il favorise les candidats directement liés à une des cinq premières notes distinctes, dans les deux sens du lien. Les notes de référence elles-mêmes sont exclues du bonus.

Ces chiffres concernent la recherche hybride d'un index. La fédération entre montages ajoute sa propre sélection et son reclassement.

COURS / 06

Définir la fraîcheur

Une fonction contrôlable, séparée de la représentation du texte.

âge = date_requête − date_note

f(âge) = 2^(−âge / h)

h = demi-vie, dans la même unité que l’âge

La demi-vie est le temps nécessaire pour diviser la fraîcheur par deux. Avec h = 365 jours, une note d’un an vaut 0,5 sur ce signal et une note de deux ans vaut 0,25.

Âge, avec h = 1 an	Fraîcheur
0 an	1
1 an	0,5
2 ans	0,25
4 ans	0,0625

Date de création ou de modification ?

Le papier décrit des objets horodatés ; choisir la dernière modification est une adaptation à notre usage. Ce choix mesure une activité sur la note, pas la validité de son contenu. Une correction cosmétique peut rajeunir une décision ancienne.

Les dates futures doivent être traitées : la formule peut alors dépasser 1. On peut refuser une anomalie ou définir une politique de plafonnement, mais cette politique doit aussi rester compatible avec la représentation stockée. On ne peut pas modifier la formule sans examiner ses conséquences.

Poids et demi-vie répondent à deux questions

Le poids répond à « combien la date peut-elle influencer le résultat ? ». La demi-vie répond à « à quelle vitesse cet effet diminue-t-il ? ». Une date repère, introduite plus loin, ne doit répondre à aucune de ces deux questions.

COURS / 07

Additif ou multiplicatif ?

Deux comportements très différents pour une vieille note pourtant pertinente.

Additif : $F = \alpha s + \beta f$, avec $\alpha + \beta = 1$

Multiplicatif : $F = s \times f$

Les deux familles sont étudiées dans le papier TANGO [1]. La différence pratique se voit sans théorie avancée : l'additif borne la contribution temporelle par β lorsque f reste entre 0 et 1 ; le multiplicatif réduit toute la similarité selon l'âge.

s = 0,8 ; h = 1 an	Additif α=0,98	Multiplicatif
Aujourd'hui	0,804	0,8
Il y a un an	0,794	0,4
Il y a deux ans	0,789	0,2

Pourquoi étudier surtout l'additif

Notre intention est de conserver les anciennes références utiles et de favoriser les évolutions lorsque les pertinences sont proches. Un coefficient temporel faible donne une limite explicite à cet arbitrage. Il n'impose pas que seules les notes récentes existent dans les résultats.

Que signifie « 2 % » ?

Ici, β = 0,02 est un coefficient dans une somme de signaux. Ce n'est pas une hausse de 2 % du score initial, ni la garantie que 2 % des résultats changent. Et après une fusion fondée sur les rangs, cela ne devient pas un plafond de 2 % sur le score final.

COURS / 08

Combien peut changer le temps ?

Traduire le coefficient en un écart sémantique concret.

Une note A récente, un peu moins proche du sens, peut dépasser une note B ancienne si son avantage temporel compense son retard sémantique.

$$F_A > F_B$$

$$\alpha(s_A - s_B) + \beta(f_A - f_B) > 0$$

$$s_B - s_A < (\beta/\alpha) \times (f_A - f_B)$$

Avec $\alpha = 0{,}98$ et $\beta = 0{,}02$, le rattrapage maximal est $\beta/\alpha \approx 0{,}020408$ de similarité, car l'écart maximal de fraîcheur est 1. Ce seuil s'exprime dans l'échelle s choisie, ici le cosinus.

Exemple : deux ans d'écart

$f_A = 1$; $f_B = 0{,}25$

écart temporel = 0,75

rattrapage maximal = (0,02/0,98) × 0,75

≈ 0,015306 de cosinus

Une récente à 0,80 peut donc dépasser une ancienne à 0,81, mais pas une ancienne à 0,83 dans cet exemple. Il s'agit d'une propriété de la formule, pas d'une preuve qu'elles traitent la même décision.

Pourquoi un cosinus élevé ne suffit pas

Des notes peuvent parler du même système en donnant des décisions contradictoires. La géométrie aide à les retrouver ensemble ; comprendre la chronologie et les relations « remplace » reste une tâche d'interprétation des sources.

COURS / 09

Séparer la date en deux facteurs

La date repère disparaît du score, mais reste présente dans les nombres intermédiaires.

z = 2^((t − T)/h) facteur stocké pour la note

g = 2^((T − τ)/h) facteur calculé pour la requête

t : date de la note ; τ : date de la recherche ; T : repère

La démonstration

g × z = 2^((T − τ)/h) × 2^((t − T)/h)

= 2^((T − τ + t − T)/h)

= 2^((t − τ)/h)

= fraîcheur

Les puissances de même base se multiplient en additionnant leurs exposants. Le +T et le -T s’annulent. En arithmétique exacte, changer T ne change donc pas le score si les deux facteurs sont recalculés de manière cohérente.

Le temps passe sans modifier les notes

La date t de la note ne change pas chaque matin, donc z reste stocké tel quel. Seul g change avec la date de la requête. Une note réellement modifiée reçoit en revanche un nouveau t et un nouveau z.

Cette factorisation est celle de la réduction STR du papier [1]. Le choix de h intervient des deux côtés : changer la demi-vie implique de revoir les facteurs stockés. Les exemples du cours emploient la même unité de temps partout.

COURS / 10

Pourquoi une seule dimension suffit au score

Construire une somme pondérée avec un produit scalaire.

Prenons q et x, les vecteurs sémantiques normalisés de la question et de la note. On veut $F = \alpha(q \cdot x) + \beta g z$.

Construction symétrique

$$u = [\sqrt{\alpha} \times q \ ; \ \sqrt{\beta} \times g]$$

$$v = [\sqrt{\alpha} \times x \ ; \ \sqrt{\beta} \times z]$$

$$u \cdot v = \alpha(q \cdot x) + \beta g z = F$$

Les racines carrées apparaissent parce que les coefficients sont présents des deux côtés : $\sqrt{\alpha} \times \sqrt{\alpha} = \alpha$. Dans cette version, α et β sont intégrés aux notes stockées ; les modifier impose de reconstruire leurs représentations.

Construction asymétrique

$$u = [\alpha \times q \ ; \ \beta \times g]$$

$$v = [x \ ; \ z]$$

$$u \cdot v = \alpha(q \cdot x) + \beta g z = F$$

Cette deuxième version produit exactement le même score. Les poids ne sont plus stockés dans la note, donc on peut les changer côté requête. Mais ce produit scalaire n'est pas encore une recherche L2 correcte : la norme de v dépend de z. Le chapitre suivant résout cette difficulté.

Point clé : « asymétrique » signifie que question et note ne sont pas préparées par la même formule. Cela n'interdit pas de calculer une distance euclidienne entre elles.

COURS / 11

La borne sur la norme

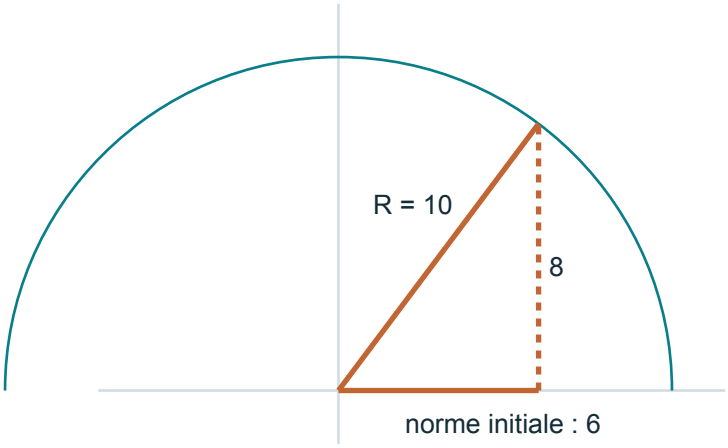
Pourquoi on ajoute une coordonnée qui ne représente ni du sens ni du temps.

Soit v un vecteur de note déjà transformé. On choisit un rayon R tel que $||v|| \leq R$ pour toutes les notes. Puis on ajoute une coordonnée de compensation.

$$P = [v\ ;\ \sqrt{(R^2 - ||v||^2)}]$$

$$||P||^2 = ||v||^2 + R^2 - ||v||^2 = R^2$$

Toutes les notes ont maintenant la même norme R. Cette compensation est une technique de réduction du produit scalaire vers une recherche euclidienne [2].



La compensation place toutes les notes sur une même sphère.

$6^2 + 8^2 = 10^2$

La condition indispensable

Le nombre sous la racine doit être positif ou nul. Si une nouvelle note possède une norme supérieure à R, la construction n'est plus valide avec ce rayon. Cela n'empêche pas toute recherche vectorielle ; cela impose une autre borne ou une reconstruction de cette représentation.

R est une longueur, pas un nombre de dimensions. Une note peut avoir 4096 coordonnées et une norme de 1. Ajouter deux coordonnées ne transforme pas cette norme en 4098.

COURS / 12

La preuve de compatibilité L2

Le passage qui permet à sqlite-vec de conserver une distance euclidienne.

À partir de u, le vecteur de requête, et de v, le vecteur de note, on construit :

$$Q = [u ; 0]$$

$$P = [v ; \sqrt{(R^2 - ||v||^2)}]$$

Développons la distance

$$||Q - P||^2 = ||u - v||^2 + (R^2 - ||v||^2)$$

$$= ||u||^2 + ||v||^2 - 2(u \cdot v) + R^2 - ||v||^2$$

$$= ||u||^2 + R^2 - 2F$$

Pour une même question, $||u||^2 + R^2$ est identique pour toutes les notes. Si F augmente, la distance diminue. Le classement par distance euclidienne reproduit donc exactement celui du score F, sur les nombres réels.

Ce que sqlite-vec doit connaître

Il ne doit pas comprendre la date ni la formule de fraîcheur. Il reçoit des vecteurs préparés et calcule sa distance L2 habituelle. La signification du résultat vient de notre construction.

Ne pas normaliser à nouveau

Une normalisation arbitraire après cette préparation changerait les coefficients et détruirait l'identité attendue. Les embeddings sémantiques d'origine peuvent rester normalisés ; les représentations de recherche sont un objet distinct.

Cette preuve concerne l'équivalence du score et du classement. Elle ne garantit ni le rappel d'un éventuel index approximatif ni les performances d'une implémentation particulière.

COURS / 13

Une version réglable sans réindexer le poids

Deux dimensions supplémentaires, et des coefficients placés uniquement dans la requête.

Supposons $||x|| = ||q|| = 1$ et $0 \le z \le 1$. Pour la version asymétrique, $||[x;z]||^2 = 1 + z^2 \le 2$. On peut donc choisir $R = \sqrt{2}$.

Note stockée $P = [x ; z ; \sqrt{(1 - z^2)}]$

Requête $Q = [\alpha q ; \beta g ; 0]$

$||P||^2 = 1 + z^2 + 1 - z^2 = 2$

La distance recherchée

$||Q - P||^2 = 2 + \alpha^2 + \beta^2 g^2$

$- 2[\alpha(q \cdot x) + \beta g z]$

Changer α ou β modifie la norme de la requête, mais ce changement est commun à toutes les notes de cette recherche. L'ordre reste exactement celui du nouveau score. Les vecteurs de notes et leur compensation ne changent pas.

Paramètre modifié	Reconstruction de cette représentation ?
Poids α et β	Non, ils sont côté requête
Demi-vie h	Oui, elle définit z
Date repère T	Oui, elle définit z
Texte ou date d'une note	Mise à jour de cette note

Avec $\beta = 0$ et $\alpha > 0$, on retrouve le classement sémantique initial. Les valeurs de distance restent différentes de celles de l'ancien index : les consommateurs de scores doivent être adaptés, même lorsque l'ordre est identique.

Un exemple numérique complet

Quatre coordonnées suffisent pour vérifier toute la chaîne à la main.

Prenons un sens en deux dimensions : q = [1;0]. La note A est récente et un peu moins similaire que B. On fixe α = 0,98, β = 0,02, h = 1 an et T = τ pour simplifier les chiffres ; donc g = 1.

x_A = [0,80 ; 0,60] ; âge_A = 0 ; z_A = 1

x_B = [0,81 ; $\sqrt{(1 - 0{,}81^2)}$] ; âge_B = 2 ans ; z_B = 0,25

Les vecteurs préparés

P_A = [0,80 ; 0,60 ; 1 ; 0]

P_B ≈ [0,81 ; 0,58643 ; 0,25 ; 0,96825]

Q = [0,98 ; 0 ; 0,02 ; 0]

Scores et distances

	A récente	B ancienne
Similarité	0,80	0,81
F = 0,98s + 0,02f	0,804	0,7988
$\|\|Q\|\|^2 + 2$	2,9608	2,9608
Distance² = 2,9608 − 2F	1,3528	1,3632
Distance	≈ 1,16310	≈ 1,16756

La note A a le score le plus élevé et la distance la plus faible. Sans fraîcheur, B serait devant. L'écart sémantique est assez petit pour que la date fasse la différence.

Choisir T = τ ne convient pas comme stratégie permanente sans réencodage : c'est seulement un choix pédagogique pour cet exemple figé.

COURS / 15

Choisir une date dans le futur

Une borne connue, au prix d'un horizon à maintenir.

Si on choisit T dans le futur, toutes les notes dont la date t ≤ T vérifient z = 2^((t-T)/h) ≤ 1. La borne du chapitre 13 est donc connue sans calculer le maximum réel des normes.

$$t \leq T \Rightarrow z \leq 1 \Rightarrow ||[x\ ;\ z]|| \leq \sqrt{2}$$

Est-ce que T change le classement ?

En calcul exact, non : gz ne dépend pas de T. Mais les distances numériques intermédiaires changent, et leur précision peut changer. La date repère est un choix de représentation, pas un réglage de pertinence.

T − aujourd'hui, h = 1 an	g	Terme $\beta^2 g^2$, avec β = 0,02
1 an	2	0,0016
5 ans	32	0,4096
10 ans	1024	419,4304
20 ans	1 048 576	≈ 439 804 651

Le dernier cas crée des distances au carré énormes dont les petits écarts utiles peuvent être masqués par les arrondis. Quelques années et vingt ans ne sont donc pas équivalents. Le rapport horizon/demi-vie est déterminant.

Quand une date dépasse le repère

Alors z peut dépasser 1 et $\sqrt{(1-z^2)}$ n'est plus réel. Il faut renouveler le repère, adopter une borne plus grande avec une autre compensation, ou gérer plusieurs générations. On ne doit pas écrêter z silencieusement : cela changerait la formule recherchée.

La précision numérique

Un petit nombre n’est pas automatiquement un nombre imprécis.

Les nombres flottants gardent une précision relative limitée. Un facteur temporel petit peut encore être représenté correctement. Le problème apparaît notamment lorsque des écarts minuscules doivent être distingués au milieu de valeurs bien plus grandes.

La compensation du temps, en théorie et en pratique

petit facteur note × grand facteur requête = fraîcheur correcte

distance² = grand terme commun − 2 × score utile

La première ligne est une identité exacte. Dans la seconde, le grand terme commun peut dominer l’échelle des distances. Même si le moteur calcule une somme de carrés sans soustraction explicite, les écarts entre distances doivent rester représentables à cette échelle.

Ce qu’il faut mesurer

Comparer le classement obtenu avec les vecteurs stockés en float32 au score de référence calculé en float64. Varier le repère, la demi-vie, les poids et des notes aux scores très proches. Mesurer les inversions et les égalités induites par les arrondis.

Une note ancienne qui reçoit une contribution temporelle presque nulle correspond d’abord au comportement voulu. Il faut distinguer cette décroissance d’une erreur d’arrondi, d’un dépassement de capacité ou d’un passage à zéro par sous-dépassement.

Conclusion pratique

Le repère futur peut simplifier la borne, mais il crée une obligation de suivi. Aucun horizon optimal n’a été mesuré ici pour le modèle et le corpus réels. Le cours démontre une possibilité ; il ne valide pas ce réglage en production.

COURS / 17

Exercices et corrigés

Refaire les étapes importantes sans dépendre de la conversation.

1. Norme et normalisation

Calculer la norme de [6;8], puis normaliser ce vecteur. **Réponse :** norme 10 ; vecteur [0,6;0,8].

2. Sens et distance

Pour deux vecteurs unitaires de produit scalaire 0,9, calculer la distance au carré. **Réponse :** 2 - 2×0,9 = 0,2 ; distance ≈ 0,4472.

3. Fraîcheur

Avec une demi-vie de six mois, quelle est la fraîcheur après deux ans ? **Réponse :** quatre demi-vies ; 2^(-4) = 1/16 = 0,0625.

4. La borne

Une note transformée a une norme de 3. Pour R=5, quelle coordonnée ajouter ? **Réponse :** √(25-9) = 4. Pour une note de norme 6, ce même R est invalide.

5. Changer le poids sans changer l’index

Avec P=[x;z;√(1-z²)] et Q=[αq;βg;0], que vaut la norme au carré de P ? **Réponse :** 2. Elle ne dépend ni de z ni des coefficients de la requête. C’est ce qui permet de les changer sans reconstruire P.

6. Changer le repère

Si T augmente de deux demi-vies, z est divisé par 4 et g est multiplié par 4. Leur produit reste identique. Les vecteurs stockés changent néanmoins, et les valeurs intermédiaires peuvent avoir une précision différente.

COURS / 18

Sources, vérifications et limites

Distinguer les résultats publiés, les preuves du cours et les points encore ouverts.

Sources primaires

[1] Wei, Luo, Xu, Yang et Palpanas. Time-Decayed Vector Search in the Rhythm of TANGO: Jointly Modeling Semantic Similarity and Temporal Freshness. arXiv:2609.00548v1, 2026. Sections 2.2 et 2.3 : objectifs temporels et réduction STR. Le cours ne reproduit pas l'architecture Chronos/TANGO.
Lire le papier sur arXiv

[2] Bachrach et al. Speeding Up the Xbox Recommender System Using a Euclidean Transformation for Inner-Product Spaces. RecSys 2014. Réduction du produit scalaire à une recherche euclidienne.
Publication Microsoft Research

[3] sqlite-vec. Documentation du projet et réponse du mainteneur sur la métrique L2 par défaut de vec0.
Métrique par défaut et recherche MATCH
Référence des fonctions de distance

Vérification locale

Code inspecté le 17 octobre 2026 : deep-obsidian-index/src/search.rs et sqlite.rs, plus la fédération serveur. Le script joint vérifie par calcul aléatoire l'identité distance² = constante - 2F pour la variante asymétrique, ainsi que l'invariance du score selon le repère. Il ne teste pas une intégration sqlite-vec en production.

Statut des propositions

Les exemples, la variante asymétrique et les développements détaillés sont des constructions pédagogiques vérifiées algébriquement. Le poids de 2 %, la demi-vie d'un an, l'horizon et le protocole de migration ne sont pas des recommandations validées par un benchmark du vault. Aucun moteur n'a été modifié.