

Fuzzy matching — logique de correspondance du grading Accounting

À quoi sert le fuzzy matching ?

Le grading doit retrouver quelle facture ou écriture attendue correspond à une ligne ou section étudiante, même si son identifiant manque ou est erroné. Le matching choisit cette correspondance; les correcteurs évaluent ensuite les valeurs, options, dates, références, soldes et l'ordre. Un match trouvé ne rend donc pas automatiquement la réponse correcte et son score de ressemblance n'est pas la note finale.

Cette logique est exécutée dans le script serveur BrainCloud

`node.js/backend/braincloud/server_scripts/_utils/by_module/byModule_utils_grade_module.js`, à l'intérieur de `byModuleGradeModule()`. Voir [Grading, reassemble et calcul des scores](#) pour la pondération finale.

Le helper commun : matchUidsToAnswers

Signature : `matchUidsToAnswers(uids, answersByKey, getStrictAnswerKey, getMatchScore)`. Il reçoit les UID étudiants ordonnés, les réponses candidates indexées par clé, une fonction d'identification stricte et une fonction de ressemblance. Il retourne uniquement `answerKeyByUid`; une entrée sans correspondance est absente de ce résultat.

flowchart TD

A[Entrées étudiantes et réponses candidates] --> B[Passage strict dans l'ordre des UID]

B --> C[Réserver chaque clé stricte une seule fois]

C --> D[Entrées et réponses restantes]

```

D --> E[Seuil initial 1]
E --> F[Meilleure réponse disponible au-dessus du seuil pour chaque entrée]
F --> G[Consommer la réponse et retirer l'entrée appariée]
G --> H{Au moins un match dans ce passage ?}
H -->|Oui| F
H -->|Non| I{Score restant au moins égal à 0.5 ?}
I -->|Oui| J[Abaissier le seuil au meilleur score observé]
J --> F
I -->|Non| K[Retourner la correspondance partielle]

```

1. Priorité stricte

Le passage strict traite toutes les entrées avant tout fuzzy matching. Une clé non nulle est retenue si elle n'a pas déjà été utilisée. Elle est retirée des réponses restantes. Une deuxième entrée demandant la même clé passe dans le lot fuzzy; elle ne peut pas réutiliser cette réponse, mais peut être associée à une autre.

La validation métier de la clé appartient au callback. `getStrictBillUid` lit `bill_id`, retrouve la facture dans `_billsByNumber` et vérifie `getBillAffectedPageUid()`. Un identifiant valide pour la page est prioritaire même si les montants ou la date ressemblent davantage à une autre facture; ces champs seront corrigés ensuite.

2. Passages fuzzy

1. Commencer à `currentThreshold = 1`.
2. Pour chaque UID restant, parcourir les réponses disponibles et retenir le plus grand score satisfaisant `score ≥ currentThreshold`.
3. Consommer immédiatement la réponse retenue. Aucun retour arrière ne réaffecte un match déjà choisi.
4. Lorsqu'un passage n'a produit aucun match, abaisser le seuil au plus grand score inférieur au seuil courant mais supérieur ou égal à `0.5`.
5. Arrêter si aucune paire admissible ne reste, ou si l'un des lots est vide.

Le seuil descend vers les scores réellement observés : ce n'est pas une suite fixe $1 \rightarrow 0,9 \rightarrow 0,8$. Une égalité conserve la première réponse rencontrée, car une nouvelle candidate doit satisfaire `score > bestScore`. Le résultat dépend donc de l'ordre des UID et des clés des réponses.

Qui utilise le helper commun ?

Correcteur	Unité et priorité stricte	Score fuzzy
------------	---------------------------	-------------

<code>computeAuxJournalScoreBase</code> : achats, ventes, retraits, encaissements, paie	Section; première row contenant <code>bill_id</code> → facture de cette page	<code>calculateMatchScore</code> : partenaire, mois/année, comptes et montants
<code>computePettyCashJournalScore</code>	Row; <code>bill_id</code> → facture de cette page	<code>calculateRowMatchScore</code> : mois/année et comptes/montants
<code>computeSubledgersJournalTableScore</code> : clients/fournisseurs	Row; clé construite table/partenaire + send/receive + <code>bill_id</code> , hors <code>upfront_payment</code>	<code>calculateRowAmountMatchScore</code> : mois/année, sens et total de la facture
<code>computeLedgerAccountTableScore</code>	Row admissible du compte; <code>bill_id</code> présent dans ses réponses attendues	<code>calculateRowAmountMatchScore</code> : mois/année, sens et montant de l'écriture de ce compte

Le helper n'est pas branché sur tous les correcteurs du fichier. Les stocks et le General Journal possèdent d'autres règles.

Comment sont calculés les scores ?

Journaux auxiliaires : section

`calculateMatchScore` retourne 0 si la facture ne concerne pas cette page. Les poids sont partenaire 0.3, date 0.3, comptes 0.4. Le partenaire correspond à l'expéditeur ou au destinataire de la facture; la date compare le mois et l'année. Les poids disponibles sont additionnés au dénominateur.

Les comptes sont extraits des rows, en excluant `date`, `partner_uid`, `bill_id`, `report_to_account`, `account_uid`. Les colonnes débit/crédit sont rattachées à `account_uid`, avec un signe négatif pour le crédit. Pour un compte présent dans la facture, le code attribue une contribution de présence ou de sens d'impact, puis une contribution de proximité du montant. La proximité utilise `max(0, 1 - abs(saisi - attendu) / attendu)` lorsque l'attendu est non nul.

Détails de l'implémentation actuelle : le test de date utilise `if (sectionMonth && sectionYear)`. Comme janvier vaut 0 dans `Date.getMonth()`, le facteur date est omis en janvier. Le cumul de contributions par compte peut aussi produire un score supérieur à 1 : il n'est pas borné. Il faut conserver ces détails lorsqu'on diagnostique un résultat, plutôt que supposer une probabilité normalisée.

Petite caisse : row et comptes

`calculateRowMatchScore` utilise date `0.33` et comptes `0.66`. Les comptes renseignés sont comparés à ceux de la facture; `date, partner_uid, bill_id, petty_cash_total` sont exclus. Un compte absent de la facture contribue 0. Un montant attendu nul et saisi nul contribue 1; sinon le code utilise la proximité relative du montant. La contribution moyenne des comptes est pondérée par 0.66.

La date compare uniquement mois/année, avec un test de présence différent de celui des sections. Sans facteur disponible, le résultat est 0. Le partenaire et le total petite caisse ne servent pas au matching de cette row.

Subledgers et ledger : row et montant

`calculateRowAmountMatchScore` utilise date `0.33` et montant `0.66`. Le sens saisi doit être le sens attendu; si débit et crédit sont tous deux renseignés, le débit est prioritaire. Pour un attendu non nul, la contribution montant est `max(0, 1 - abs(saisi - attendu) / abs(attendu))`. Les facteurs présents sont renormalisés par leur somme; sans facteur, le score vaut 0.

Une date présente mais différente reste dans le dénominateur avec une contribution nulle. Une date absente est exclue. Si seul le mois/année correspond et que le montant est incorrect, le score est environ 1/3 : insuffisant pour le seuil 0.5. Si la date est absente et le sens/montant sont corrects, le score peut atteindre 1.

Exemple chiffré

Écriture attendue : débit 100 en septembre. L'étudiant saisit débit 90 dans le même mois et la même année, sans identifiant strict valide.

```
proximité montant = 1 - |90 - 100| / 100 = 0.9
score = (0.33 × 1 + 0.66 × 0.9) / (0.33 + 0.66)
       = 0.9333...
```

La row peut donc être appariée lors d'un passage fuzzy, si la réponse reste disponible. Le montant 90 est ensuite comparé à 100 dans le grading des cellules; cet appariement n'accorde pas 93,33 % de la note.

Ce qui a changé dans le ledger le 30 septembre

Le ledger construit les réponses attendues par compte et numéro de facture ou formulaire, avec sens, montant, page de référence et date. Le nouveau helper remplace le fallback local d'appariement et permet de retrouver une écriture à partir de son sens, montant et mois/année quand le `bill_id` ne permet pas de l'identifier.

Seules les rows ayant une `reference` participent à cet appariement. Avec les journaux auxiliaires actifs, la référence doit contenir `general_journal`; sans eux, toutes les rows portant une référence sont candidates. Les reports agrégés des journaux auxiliaires et les réponses de régularisation/clôture conservent leur traitement spécialisé. La recherche ne traverse pas tous les comptes : chaque table possède ses candidates.

Après matching, le correcteur prépare `bill_id`, `reference`, la date, le débit/crédit attendu, puis corrige le solde et l'ordre. Les réponses non rencontrées peuvent produire `table_bill_missing` ou `table_form_missing`. Une référence invalide peut produire `row_reference_missing`.

Le General Journal garde un algorithme distinct

`computeGeneralJournalScore` ne passe pas par `matchUidsToAnswers`. Il construit `wantedExplanationBySectionUids` pour factures, formulaires et relations de régularisation/clôture, avec ses propres passages de seuil, priorités et traitement des doublons.

`getMatchRateForGeneralJournalSection` pondère la ressemblance des comptes par 4, puis ajoute l'explication sélectionnée et la date exacte lorsqu'elles sont disponibles. Les factures/formulaires comparent aussi montants et sens; les relations peuvent comparer la structure des comptes sans leurs valeurs. Le plancher des passages est `0.4` (le code arrête avant un passage dont le prochain seuil est ≤ 0.4), et une recherche complémentaire de relation utilise `0.8`. Ces seuils ne remplacent pas le 0.5 du helper commun.

Ambiguïtés et diagnostic

Le helper est glouton : il cherche une bonne paire au seuil courant, pas la meilleure affectation globale de toutes les rows. Exemple : $R1 \rightarrow A = 0.9$, $R1 \rightarrow B = 0.8$, $R2 \rightarrow A = 0.85$, $R2 \rightarrow B = 0.1$. Il peut retenir $R1 \rightarrow A$ puis laisser $R2$ sans match, alors que $R1 \rightarrow B$ et $R2 \rightarrow A$ permettraient deux correspondances. Les égalités dépendent de l'ordre. Une valeur NaN ne satisfait aucune comparaison de seuil.

1. Identifier le correcteur de page et la liste exacte des candidates.
2. Vérifier si un `bill_id` valide a verrouillé une réponse au passage strict.
3. Examiner les champs présents : page, partenaire, date, compte, débit/crédit et montant selon le helper.

4. Recalculer le score avec les poids réellement disponibles et vérifier le seuil.
 5. Contrôler l'ordre des UID, les doublons et les réponses consommées par d'autres entrées.
 6. Examiner ensuite `errors`, `expectedValues` et les scores bruts; vérifier la pondération dans `pages_settings` seulement après.
-

Revision #7

Created 2026-09-30 14:32:35 UTC by Mathis

Updated 2026-09-30 14:48:39 UTC by Mathis