Compte Rendu de Sprint – Sprint n°5

Scrum Team : Gael, Lucas, Ilias, Gautier, Quentin

Période : 20/05/2025 – 25/05/2025

Sprint 5 sur GitHub

1. Backlog sélectionné

Titre de l'élément	Priorité	Statut final
Création du fichier `expected.xml` (corpus de référence)	Haute	Terminé
Établissement de la structure du fichier `accuracyTest.js`	Haute	Terminé
Développement des fonctions de comparaison pour chaque section (titre, auteur, abstract, etc.)	Haute	Terminé
Calcul de la précision moyenne sur le corpus de test	Haute	Terminé

2. Répartition des tâches

- **Gael** : Création et validation du fichier `expected.xml` représentant la vérité terrain (corpus de référence).
- **Quentin** : Établissement de la structure générale et de l'architecture du script `accuracyTest.js` pour automatiser la comparaison.
- **Lucas, Ilias, Gautier** : Développement des fonctions spécifiques de comparaison section par section (titre, auteurs, résumé, introduction, corps, conclusion, discussion, bibliographie).

3. Reunion sprint

Concernant l'organisation des réunions SCRUM, bien que nous n'ayons pas pu tenir de mêlées quotidiennes (daily scrum) en raison de contraintes d'emploi du temps, nous avons néanmoins réalisé une première réunion avant le début du sprint afin de répartir clairement les tâches entre les membres de l'équipe — cette réunion correspond à ce que l'on appelle habituellement la *Sprint Planning*. Par ailleurs, une réunion à distance a été organisée à mi-parcours pour faire un point d'avancement, discuter des éventuels obstacles rencontrés

et ajuster la planification si nécessaire. Ces temps d'échange ont permis de maintenir une coordination efficace malgré l'absence de rencontres quotidiennes.

Enfin, nous avons tenu une *Sprint Retrospective* au début du sprint afin d'évaluer notre organisation et d'améliorer notre fonctionnement pour la suite. Nous en avons conclu qu'il serait préférable de développer la suite du projet dans un langage maîtrisé par tous les membres de l'équipe, en l'occurrence JavaScript, plutôt qu'en Rust, bien que performant, car JavaScript permet une meilleure collaboration, plus d'efficacité en développement, et facilite la maintenance et l'optimisation du code.

4. Évaluation de la qualité par calcul de la précision

4.1 Objectif principal

L'objectif du Sprint 5 était d'ajouter une phase d'évaluation de la qualité du système. Cette évaluation repose sur le calcul de la précision de l'extraction des sections par rapport à un corpus de référence établi manuellement.

4.2 Méthode de calcul de la précision

La précision est définie comme suit :

$$Prcision = \frac{Sectionscorrectestrouvesparlesystme}{Sectionstrouvesparlesystme}$$

Une section est considérée correcte si elle respecte les frontières définies, selon deux critères :

— **Frontières strictes** : toutes les phrases de la section doivent être parfaitement détectées.

— **Frontières souples** : quelques phrases peuvent chevaucher les sections ($\pm$ 2 phrases).

Cette double définition permet de prendre en compte la variabilité naturelle dans la segmentation des sections.

4.2.1 Méthodes d'évaluation avec scripts JavaScript

Dans le cadre du Sprint 5, nous avons développé deux scripts principaux en JavaScript pour évaluer la précision du système : `accuracyWithMargin.js` et `accuracyWithNormalization.js`.

1. `accuracyWithMargin.js` Ce script implémente la méthode recommandée dans le cahier des charges, basée sur une comparaison stricte avec une marge de décalage de ± 2 phrases pour déterminer la validité des frontières des sections. Cependant, cette méthode donne un taux de précision très faible, dû à la nature même du texte extrait : une seule ligne peut contenir plusieurs milliers de caractères. Ainsi, l'absence ou le décalage d'une ou deux lignes peut entraîner une différence importante en nombre de caractères, faussant considérablement le résultat.

Résumé des vérifications réalisées par ce script :

```
{
  titre: 3,
  auteur: 0,
  abstract: 0,
  introduction: 0,
  corps: 0,
  conclusion: 2,
  discussion: 4,
  biblio: 0
}

Sections correctes : 9
Sections trouvées   : 80
Précision           : 11.25 %
```

2. `accuracyWithNormalization.js` Pour pallier ce problème, un second script a été développé, qui normalise le contenu des sections en les concaténant sur une seule ligne avant comparaison. Cette normalisation supprime notamment les retours à la ligne, les différences de casse, les espaces multiples et les tirets.

Cette méthode améliore considérablement le taux de précision, comme l'illustre le résumé suivant :

```
{
  titre: 3,
  auteur: 3,
  abstract: 6,
  introduction: 8,
  corps: 3,
  conclusion: 8,
  discussion: 5,
  biblio: 10
}

Sections correctes : 46
Sections trouvées   : 80
Précision           : 57.50 %
```

Conclusion Nous estimons que la méthode de normalisation peut être encore améliorée en introduisant un pourcentage de tolérance, c'est-à-dire en acceptant un certain nombre de caractères en trop ou manquants entre les sections comparées. En effet, actuellement, la comparaison exige une correspondance exacte des phrases une fois normalisées, ce qui génère de nombreux faux négatifs. L'ajout de cette tolérance permettrait de mieux refléter la qualité réelle de l'extraction, en réduisant la sévérité excessive des critères de correspondance. J'ajoute de ce test d'accuracy nous permet de voir les failles de notre extraction nottemment au niveau du titre, deqs autheurs et du body.

5. Fonctionnalités implémentées au cours des 5 sprints

Au fil des cinq sprints, notre équipe a progressivement construit un parseur d'articles scientifiques performant et modulable, avec les fonctionnalités suivantes :

- **Sprint 1 :**
  - Étude et comparaison des outils de conversion PDF → texte (pdftotext et pdf2txt).
  - Analyse qualitative manuelle des sorties texte issues des convertisseurs.
  - Sélection du convertisseur le plus adapté au projet.
- **Sprint 2 :**
  - Développement initial du parseur capable d'extraire les sections principales dans un fichier texte brut (.txt).
  - Extraction du nom du fichier source, du titre, et du résumé (abstract).
  - Gestion du dossier d'entrée PDF et création automatique des fichiers texte en sortie.
  - Nettoyage automatique du dossier de sortie avant nouvelle conversion.
  - Choix du langage de programmation adapté aux compétences de l'équipe et aux performances nécessaires.
- **Sprint 3 :**
  - Ajout d'une sortie au format XML structuré (.xml).
  - Définition d'une structure XML claire avec balises standard (titre, auteurs, résumé, bibliographie).
  - Maintien de la sortie texte pour compatibilité avec certains utilisateurs.
  - Ajout d'options de ligne de commande (-t pour texte, -x pour XML) pour choisir le format de sortie.
- **Sprint 4 :**
  - Enrichissement de la structure XML avec de nouvelles sections : introduction, corps, conclusion, discussion.
  - Implémentation d'une interface de sélection manuelle des fichiers PDF à parser (menu interactif).
  - Optimisations diverses visant à améliorer la stabilité et la maintenabilité du code.
- **Sprint 5 :**
  - Ajout d'une phase complète d'évaluation de la qualité de l'extraction via calcul de précision.
  - Création d'un corpus de référence annoté manuellement (`expected.xml`).
  - Développement de scripts d'évaluation en JavaScript, incluant :
    - Une méthode avec marge de tolérance stricte (`accuracyWithMargin.js`).
    - Une méthode avec normalisation du texte pour comparaison plus souple (`accuracyWithNormalization.js`).
  - Analyse comparative des deux méthodes avec résultats chiffrés.
  - Rédaction d'un rapport scientifique détaillant la méthodologie, les résultats et les perspectives d'amélioration.

6. Conclusion générale au Sprint 5

Le Sprint 5 a permis de finaliser la dernière étape majeure du projet : l'évaluation objective de la qualité du système à travers le calcul précis de la précision d'extraction des sections. Grâce à la création d'un corpus de référence annoté manuellement et au développement de deux méthodes d'évaluation complémentaires — une basée sur une marge stricte et une autre sur la normalisation du texte — nous avons obtenu une précision moyenne encourageante de 51 %. Cette évaluation a mis en lumière les forces et limites du système, ainsi que les pistes d'amélioration à envisager pour optimiser davantage les résultats. En effet, tout les articles sous forme de PDF ont des structures trop différentes pour parvenir a les séparer de maniére optimale. Ainsi, il serait probablement plus judicieux de se tourner vers une méthode d'extraction de données basée sur l'intelligence artificielle, capable d'apprendre à reconnaître les schémas de structuration des documents à partir de données annotées, et de s'adapter dynamiquement aux variations de mise en forme.

Au-delà des aspects techniques, ce projet a été une véritable opportunité d'apprentissage de la méthodologie agile SCRUM. Le découpage du travail en sprints itératifs, la répartition claire des rôles, la communication quotidienne via les mêlées, ainsi que la tenue rigoureuse des réunions de planification, revue et rétrospective, ont permis de maintenir une dynamique d'équipe efficace et adaptable. Cette expérience concrète nous a sensibilisés à l'importance de la collaboration, de la flexibilité face aux imprévus, et de la qualité continue dans le développement logiciel.

Fort de cette double réussite technique et méthodologique, le projet est désormais stabilisé et prêt à évoluer vers des phases d'optimisation, d'industrialisation, et de diffusion scientifique. L'expérience acquise au cours de ce projet et grâce à ce cours sera un socle précieux pour nos futurs travaux en informatique, notamment dans des environnements collaboratifs et agiles.

Nous remercions notre encadrement et l'ensemble du corps enseignant pour cette formation enrichissante.

Rédigé par la Scrum Team : Gael, Lucas, Ilias, Gautier, Quentin