RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

UNIVERSITÉ ABOU BAKR BELKAID - TLEMCEN

Faculté des Sciences

Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

OPTION: INTELLIGENCE ARTIFICIELLE (I.A)

THÈME

Conception et Développement d'une Application Intelligente pour l'Optimisation des Emplois du Temps

Réaliser par:

- **Allam Aya**

Présenté le 26 juin 2025 devant le jury composé de:

- Mme CHAOUCHE RAMDANE Lamia (Président)
- M. SMAHI Mohammed Ismail (Encadrant)
- M. BELABED Amine (Examinateur)

ANNÉE UNIVERSITAIRE 2024/2025

Je dédie ce modeste travail à

Mes chers parents, pour leur amour inconditionnel, leurs sacrifices et leurs prières qui m'ont toujours accompagné.

À ma famille, pour leur soutien constant et leur présence bienveillante.

À mes enseignants, pour la qualité de leur enseignement et leur engagement.

À mes ami(e)s, pour leur aide précieuse, leurs encouragements et les moments partagés.

À toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce travail.

Remerciement

Avant tout, je rends grâce à **ALLAH** qui m'a toujours soutenu par Sa grande générosité, en m'accordant le courage, la volonté et la détermination nécessaires pour mener à bien tous mes projets d'études.

Je remercie également **Monsieur M. I. SMAHI** pour son encadrement, ses conseils avisés, sa disponibilité et son accompagnement tout au long de ce travail.

J'exprime ma sincère gratitude aux professeurs **Madame R. L. Chaouche** et **Monsieur A. Belabed** pour les connaissances qu'ils m'ont transmises et pour avoir accepté de présider mon jury.

Par ailleurs, je tiens à remercier l'ensemble du corps enseignant et administratif pour leur soutien tout au long de mon parcours académique.

Enfin, je suis profondément reconnaissant(e) envers ma famille et mes amis pour leur amour, leurs encouragements constants et leur présence bienveillante.

Résumé

Dans ce travail, nous avons proposé une approche innovante pour optimiser l'organisation des emplois du temps au sein du milieu académique. Cette optimisation a été réalisée grâce à l'application de l'intelligence artificielle, en particulier les techniques du traitement du langage naturel. Notre travail repose sur trois contributions principales. La première est la mise en place d'un chatbot intelligent capable de transformer automatiquement des questions formulées en langage naturel en requêtes SQL. L'idée est d'assurer une grande adaptabilité aux évolutions futures, minimisant ainsi de futurs interventions humaine directes. La deuxième contribution est l'optimisation automatique des emplois du temps universitaires. Pour cela, nous nous sommes appuyé sur la théorie des jeux afin de modéliser les préférences et les contraintes des différents acteurs impliqués. Cette approche permet de générer des emplois du temps plus équitables et cohérents en tenant compte des conflits potentiels et des besoins spécifiques de chacun. Enfin, la troisième contribution concerne le développement d'une interface conversationnelle simple et intuitive, permettant une interaction fluide entre les utilisateurs (étudiants, enseignants, administrateurs) et le système, même sans connaissances techniques avancées. Nous avons mené plusieurs expérimentations, principalement durant les phases de génération de requêtes SQL et d'optimisation des plannings, dans le but d évaluer l'éfficacité de notre approche. Les résultats obtenus confirment la pertinence de notre solution , tant sur le plan de la précision des requêtes produites que sur la qualité des emplois du temps générés.

Mots-clés: Intelligence artificielle, Traitement du langage naturel, Chatbot intelligent, Requêtes SQL, Optimisation des emplois du temps, Théorie des jeux

ملخص

يتناول هذا العمل مقاربة مُجَدِّدَة لتحسين إدارة الشؤْون الأكاديمية باستخدام الذكاءْ الاصطناعي، في هذا العمل، اقترحنا منهجًا مبتكرًا يهدف إلى تحسين تنظيم الجداول الزمنية في الوسط الأكاديمي. وقد تم تحقيق هذا التحسين من خلال توظيف تقنيات الذكاءْ الاصطناعي، وخاصة تقنيات معالجة اللغات الطبيعية. يرتكز عملنا على ثلاث مساهمات رئيسية. أولًا، تطوير روبوت محادثة ذكي قادر على تحويل الأسْئلة المطروحة باللغة الطبيعية تلقائيًا إلى استعلامات SQL. وتتمثل الفكرة في ضمان قدرة كبيرة على التكيّف مع التطورات المستقبلية، مما يحد من الحاجة إلى التدخلات البشرية المباشرة لاحقًا. ثانيًا، تحسين الجداول الزمنية الجامعية بشكل آلي. ولتحقيق ذلك، اعتمدنا على نظرية الألعاب لنمذجة تفضيلات وقيود مختلف الأطراف المعنية. وتسمح هذه المقاربة بإنتاج جداول زمنية أكثر عدلًا واتساقًا، تأخذ في الحسبان التعارضات المحتملة واحتياجات كل طرف. ثالثًا، تطوير واجهة محادثة بسيطة وبديهية، تتيح تفاعلًا سلسًا بين المستخدمين)سواءْ طلابًا، أو أساتذة، أو إداريين(والنظام، حتى من دون امتلاك خلفية تقنية متقدمة. وقد أجرينا العديد من التجارب، خاصة خلال مراحل توليد الاستعلامات SQL وتحسين الجداول الزمنية، بهدف تقييم فعالية منهجيتنا. وأكدت النتائج المحققة مدى أهمية الحل المقترح، سواءْ من حيث دقة الاستعلامات المنتجة أو من حيث جودة الجداول الزمنية التي تم إنشاؤها

الكلمات الدالة :الذكاءْ الاصطناعي، معالجة اللغة الطبيعية، روبوت محادثة ذكي، استعلامات SQL، تحسين الجداول الزمنية، نظرية الألعاب، واجهة محادثة.

Abstract

In this work, we proposed an innovative approach to optimize the organization of academic timetables. This optimization was achieved through the application of artificial intelligence, particularly natural language processing techniques. Our work is based on three main contributions. The first is the implementation of an intelligent chatbot capable of automatically transforming questions formulated in natural language into SQL queries. The idea is to ensure high adaptability to future developments, thereby minimizing the need for direct human intervention. The second contribution is the automatic optimization of university timetables. To achieve this, we relied on game theory to model the preferences and constraints of the various stakeholders involved. This approach enables the generation of fairer and more coherent timetables by considering potential conflicts and the specific needs of each party. Finally, the third contribution concerns the development of a simple and intuitive conversational interface, allowing smooth interaction between users (students, teachers, administrators) and the system, even without advanced technical knowledge. We conducted several experiments, mainly during the SQL query generation and timetable optimization phases, to evaluate the effectiveness of our approach. The results obtained confirm the relevance of our solution, both in terms of the accuracy of the generated queries and the quality of the produced timetables.

Keywords: Artificial Intelligence, Natural Language Processing, Intelligent Chatbot, SQL Queries, Timetable Optimization, Game Theory

Table des Matières

Remerciement **v**

Remerciements **v**

Résumé **vii**

ملخص **ix**

Abstract **xi**

Table des Matières **xiv**

Liste des Figures **xv**

Liste des Tableaux **xvii**

1 Introduction **1**

2 Fondements théoriques et conceptuels **3**

2.1 Introduction 4

2.2 Problématique de l'organisation temporelle 4

2.2.1 Cadre organisationnel et académique 4

2.2.2 Dimensions structurelles de la planification 7

2.2.3 Enjeux d'optimisation et de contraintes 8

2.3 L'intelligence artificielle et le langage naturel 10

2.3.1 Principes fondamentaux et définitions 10

2.3.2 Apprentissage du langage 11

2.3.3 Modèles modernes de génération de texte : GPT-2 et T5 11

2.4 Optimisation multi-acteurs et théorie des jeux 12

2.4.1 Fondements théoriques et application en IA 12

2.4.2 Interaction stratégique dans la planification 12

2.5 Conclusion 13

3 Description de l'Architecture du Système **15**

3.1 Introduction 16

3.2 Vision et but du système développé 16

3.3 Vue d'ensemble du fonctionnement 16
3.3.1 Étapes générales : de la question à la requête SQL 16
3.3.2 Circulation de l'information dans le système 17
3.3.3 Exemple simplifié d'interaction 17
3.4 Intérêt d'un modèle pré-entraîné dans notre système 18
3.5 T5 : Le cœur transformeur de notre générateur SQL 19
3.5.1 Présentation du modèle T5 19
3.5.2 Architecture du modèle T5 20
3.6 Modélisation de l'architecture du NL2SQL 20
3.6.1 Architecture générale du système 20
3.6.2 Processus Opérationnel du Chatbot 21
3.7 Vers des Emplois du Temps Équilibrés 22
3.7.1 l'intérêt d'utiliser la théorie des jeux ? 22
3.7.2 Modélisation via la théorie des jeux 22
3.7.3 Optimisation Multicritère des Emplois du Temps -Méthodologie- 23
3.8 Conclusion 25

4 Expérimentation et interprétation des résultats 27
4.1 Introduction 28
4.2 Environnement expérimental 28
4.2.1 Outils logiciels et bibliothèques 28
4.2.2 Jeux de données exploités 31
4.3 Implémentation du modèle T5 pour la génération NL2SQL 32
4.3.1 Prétraitement des données 32
4.3.2 Entraînement du modèle T5 33
4.3.3 Évaluation des performances (BLEU, Exact Match) 33
4.3.4 Comparaison avec des approches alternatives 36
4.4 Optimisation de l'emploi du temps 37
4.4.1 Modélisation du problème d'emploi du temps par la théorie des jeux . . . 37
4.4.2 Présentation et analyse des résultats 38
4.5 Visualisation des résultats dans l'application 42
4.6 Conclusion 47

5 Conclusion 49

Bibliography 51

Liste des Figures

2.1 Le rôle de l'ordonnancement : rendre réelle la planification à travers un emploi du temps exécutable [67]. 6

3.1 Architecture simplifiée illustrant l'exemple d'interaction du chatbot NL2SQL . . 18
3.2 Comparaison des architectures des modèles GPT, BERT et T5 [59, 57, 17, 1]. . . . 19
3.3 Architecture du système NL2SQL basé sur le modèle T5 21
3.4 Algorithme d'optimisation des emplois du temps 24

4.1 Courbe d'évolution de la loss (entraînement et validation) du modèle T5 sur WikiSQL 35
4.2 Évolution des scores BLEU et Exact Match sur WikiSQL au fil des époques 35
4.3 Visualisation 3D des emplois du temps générés et filtrés (Pareto et Nash) 39
4.4 Comparaison des scores de solutions Pareto selon les critères définis 40
4.5 La page d'authentification 42
4.6 La page de sélection du rôle 42
4.7 La page de création de compte 42
4.8 Le tableau de bord de l'administrateur 43
4.9 Gestion des salles disponibles 43
4.10 Modification du type de salle 43
4.11 Modification du nom d'une salle 43
4.12 Visualisation de l'emploi du temps généré dans l'application 44
4.13 La page de sélection 44
4.14 Exemple de question mal interprétée 45
4.15 Requête SQL générée automatiquement 45
4.16 Création d'emploi du temps 45
4.17 La gestion des promotions 46
4.18 Accueil étudiant 46
4.19 Emploi du temps 46
4.20 visualisation d'emploi du tempsEnseignant 47
4.21 Détaills de séance 47

Liste des Tableaux

2.1 Comparaison entre planification, ordonnancement et emploi du temps 8

4.1 Langages de programmation utilisés dans le projet 29
4.2 Frameworks et plateformes utilisés dans le projet 29
4.3 Bibliothèques Python utilisées dans le projet 30
4.4 Comparaison entre les jeux de données WikiSQL, Spider et Text-to-SQL 32
4.5 Résultats d'entraînement et de validation sur le dataset WikiSQL 34
4.6 Résultats d'entraînement du modèle T5 sur le dataset Text-to-SQL 34
4.7 Comparaison entre questions, requêtes SQL générées et requêtes réelles 36
4.8 Comparaison des performances (**Exact Match**) sur le dataset **WikiSQL** (test) . . . 36
4.9 Scores des joueurs pour trois emplois du temps générés 37
4.10 Solution générée sans prise en compte des préférences 37
4.11 Exemple d'un emploi du temps hebdomadaire 39
4.12 Comparaison entre le meilleur emploi du temps et la moyenne par critère 40

1

Introduction

Dans un monde en constante évolution numérique, les établissements d'enseignement supérieur sont de plus en plus amenés à moderniser leurs outils de gestion afin de répondre aux besoins croissants en efficacité, en accessibilité et en automatisation. Parmi les nombreuses tâches administratives et pédagogiques, la gestion des données académiques, l'interrogation des bases de données, ainsi que la planification des emplois du temps représentent des défis majeurs, à la fois techniques et organisationnels. Ces opérations, bien que primordiales, génèrent fréquemment des complexités et une consommation de temps excessive lorsqu'elles sont gérées manuellement ou à travers des interfaces rigides et difficilement exploitables.

L'émergence de l'intelligence artificielle, et plus précisément du traitement automatique du langage naturel (NLP), ouvre la voie à de nouvelles solutions permettant de repenser l'interaction entre l'humain et la machine. Grâce à des modèles de génération de texte comme GPT, T5, claude, etc., il devient possible d'interpréter des instructions formulées en langage naturel et de les convertir en requêtes exploitables par un système informatique, rendant l'accès aux données plus fluide, naturel et intelligent.

C'est dans ce contexte que s'inscrit ce travail, qui propose de développer un système intelligent basé sur l'intelligence artificielle (IA) pour améliorer la gestion académique au sein des établissements universitaires. Pour atteindre ces objectifs, trois contributions principales et distinctives sont apportées (en respectant l'ordre chronologique de réalisation).

1. En premier lieu, il s'agit du développement d'un chatbot 'performant' capable de transformer automatiquement les questions formulées en langage naturel en requêtes SQL, permettant ainsi aux utilisateurs d'interroger une base de données académique sans connaissances techniques préalables.
2. Notre deuxième contribution réside dans l'intégration d'une interface conversationnelle facilitant l'interaction avec les différentes fonctionnalités du système, en simplifiant la navigation et l'accès à l'information.
3. Enfin, la troisième contribution concerne la génération automatique des emplois du temps,

répondant aux contraintes des utilisateurs et générée à partir de requêtes simples en langage naturel.

Le mémoire est organisé de manière à présenter progressivement les concepts, les choix techniques ainsi que les différentes expérimentations réalisées afin de mener à des résultats concrets et exploitables.

1. Le premier chapitre est dédié aux fondements théoriques et conceptuels liés à la planification des tâches dans les milieux académiques, à l'ordonnancement, ainsi qu'aux notions de base de l'intelligence artificielle et du traitement du langage naturel.
2. Le deuxième chapitre détaille l'architecture générale du système que nous avons développé, en présentant ses composants clés et leur mode de fonctionnement.
3. Quant au dernier chapitre, il est dédié à l'implémentation pratique, aux tests effectués, à l'analyse approfondie des résultats obtenus, et aux perspectives d'amélioration du système, concluant ainsi notre démarche avec des réflexions sur l'avenir

Ce travail s'inscrit donc dans une démarche de modernisation des processus académiques, en mettant l'intelligence artificielle au service de l'enseignement supérieur, afin de proposer des outils plus intuitifs, intelligents et adaptés aux besoins actuels.

2

Fondements théoriques et conceptuels

Sommaire

2.1 Introduction 4
2.2 Problématique de l'organisation temporelle 4
2.2.1 Cadre organisationnel et académique 4
2.2.2 Dimensions structurelles de la planification 7
2.2.3 Enjeux d'optimisation et de contraintes 8
2.3 L'intelligence artificielle et le langage naturel 10
2.3.1 Principes fondamentaux et définitions 10
2.3.2 Apprentissage du langage 11
2.3.3 Modèles modernes de génération de texte : GPT-2 et T5 11
2.4 Optimisation multi-acteurs et théorie des jeux 12
2.4.1 Fondements théoriques et application en IA 12
2.4.2 Interaction stratégique dans la planification 12
2.5 Conclusion 13

2.1 Introduction

L'organisation temporelle dans les établissements universitaires pose de réels défis en raison de nombreuses contraintes pédagogiques, humaines et matérielles. Générer des emplois du temps cohérents demande donc des approches capables de concilier flexibilité et efficacité.

Dans ce chapitre, nous présentons les notions clés liées à la planification des différentes taches académiques, avant d'introduire l'apport de l'intelligence artificielle et du traitement du langage naturel (NLP) pour automatiser certaines tâches. Nous terminons par une brève introduction à la théorie des jeux, utile dans un contexte multi-acteurs pour améliorer les prises de décision dans la génération d'emplois du temps.

2.2 Problématique de l'organisation temporelle

L'organisation temporelle constitue une composante essentielle dans la gestion des systèmes éducatifs. Elle repose principalement sur trois notions complémentaires : la planification [21], l'ordonnancement [48] et la construction d'un emploi du temps [4].

2.2.1 Cadre organisationnel et académique

Dans les établissements d'enseignement supérieur, la planification et l'organisation des activités académiques s'appuient sur plusieurs mécanismes interdépendants. Il est donc essentiel de clarifier les notions fondamentales qui structurent l'organisation temporelle dans le contexte universitaire.

La planification des activités académiques

Dans les établissements d'enseignement supérieur, la planification constitue un levier organisationnel central pour articuler les contraintes pédagogiques, matérielles et humaines. Selon Mintzberg [41], la planification peut être définie comme un système formel de coordination des activités, visant à optimiser l'allocation des ressources dans un environnement contraint.

Cette approche trouve une application directe dans le contexte universitaire, où les plannings assurent la cohérence des enseignements, l'efficience des ressources et la qualité du service éducatif.

Typologies des plannings en milieu universitaire : En milieu académique (institutions de l'enseignement supérieur), la littérature met en évidence divers types de plannings, différenciés par leur objectif et leur granularité temporelle [3, 15].

1. **Planning de présence (ou planning des ressources)** : Il s'agit d'un document recensant les disponibilités des enseignants, des salles et des étudiants, sans détailler les contenus pédagogiques. Il constitue une base essentielle pour éviter les conflits d'occupation et optimiser l'utilisation des infrastructures [3].
2. **Planning des activités pédagogiques** : Ce type de planning détaille les séances d'enseignement (cours magistraux, travaux dirigés, travaux pratiques) en fonction des contraintes de formation. Il peut être décliné :

— *À court terme* : plannings journaliers ou hebdomadaires, adaptés à la gestion immédiate des enseignements [43].
— *À long terme* : plannings semestriels ou annuels, servant à répartir les charges pédagogiques conformément aux règles fixées par le Code de l'Éducation [13].
Des variantes terminologiques existent :
— *Tableau de service* : affectation nominative des enseignants à des créneaux précis [18].
— *Grille horaire* : représentation synthétique de l'emploi du temps par créneaux et par semaine [15].

Classification selon la cyclicité Bertrand et Gruber proposent une autre grille de lecture des plannings, fondée sur leur cyclicité [6] :

— **Planning cyclique** : caractérisé par la répétition régulière des mêmes séquences (cas des licences classiques).
— **Planning acyclique** : varie en fonction des sessions (cas des formations en alternance ou professionnalisantes).

Cette distinction permet de mesurer le degré de *flexibilité pédagogique* requis dans la conception des plannings.

Niveaux hiérarchiques de la planification : La planification dans les établissements d'enseignement supérieur s'inscrit dans une hiérarchie d'actions stratégiques et opérationnelles, s'étendant de la vision long terme jusqu'à la mise en œuvre concrète des emplois du temps. On distingue généralement trois niveaux de planification, chacun répondant à des objectifs précis [67] :

— **Gestion de la demande** (12 à 24 mois) : ce niveau vise à anticiper les besoins pédagogiques et logistiques, notamment en fonction des prévisions d'inscriptions, des évolutions de l'offre de formation et des ressources humaines disponibles.
— **Planification stratégique** (3 à 12 mois) : elle permet de répartir les ressources (enseignants, salles, équipements) en fonction des priorités institutionnelles et des contraintes structurelles, dans le but de garantir une cohérence pédagogique sur l'année universitaire.
— **Plan de charge** (1 à 3 mois) : il s'agit d'une planification à moyen terme destinée à équilibrer les volumes horaires, préparer les semestres et optimiser l'usage des ressources tout en tenant compte des contraintes internes (absences prévues, périodes d'examen, etc.).

L'ordonnancement dans l'enseignement supérieur

Dans les systèmes d'enseignement supérieur, **l'ordonnancement** s'impose comme un mécanisme opérationnel essentiel pour traduire les objectifs de la planification en emplois du temps exécutables. Il s'agit, selon Pinedo, de "l'allocation temporelle optimale des tâches à des ressources limitées, en tenant compte de contraintes multiples" [54]. L'ordonnancement intervient ainsi en aval de la planification stratégique, au moment où les contraintes pédagogiques (durées des séances, enchaînements logiques), humaines (disponibilités des enseignants, contraintes individuelles) et matérielles (capacités des salles, équipements spécialisés) doivent être traduites en séquences concrètes et cohérentes

dans le temps. Dans le cadre universitaire, cette étape requiert une modélisation fine des contraintes et des priorités, souvent résolue à l'aide de méthodes combinatoires telles que la programmation par contraintes, les heuristiques ou les algorithmes gloutons [9, 63]. On distingue plusieurs formes d'ordonnancement selon les objectifs visés: l'ordonnancement à objectif unique (minimisation des conflits, maximisation de l'occupation des salles), ou multi-objectifs (équilibre des charges horaires, préférences individuelles, équité entre groupes)[11]. Ce processus est particulièrement critique dans les contextes à forte densité d'activités, tels que les formations en alternance ou les instituts techniques, où la marge de manœuvre temporelle est réduite. L'efficacité de l'ordonnancement conditionne donc directement la qualité perçue du service éducatif et la satisfaction des parties prenantes [30]. La figure 2.1 illustre l'importance de l'ordonnancement dans le processus de planification, en mettant en évidence son rôle clé dans la concrétisation d'un emploi du temps réalisable à partir des intentions pédagogiques [67].

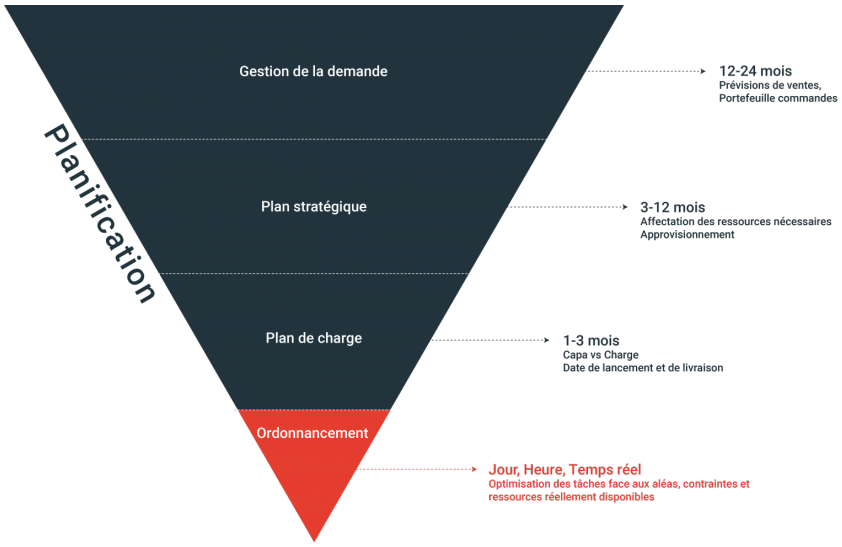

Figure 2.1 – Le rôle de l'ordonnancement : rendre réelle la planification à travers un emploi du temps exécutable [67].

L'emploi du temps universitaire

L'emploi du temps est l'outil final qui rend visibles et opérationnelles les décisions issues de la planification stratégique et de l'ordonnancement. Il organise de manière précise les différentes activités d'enseignement (cours, TD, TP) dans le temps (jours, heures) et dans l'espace (salles, laboratoires), en tenant compte des contraintes humaines, matérielles et pédagogiques.

Dans le cadre universitaire, il joue un rôle central dans le bon fonctionnement des formations. Un emploi du temps efficace doit non seulement être techniquement faisable, mais aussi équilibré, respectueux des besoins des étudiants et des enseignants, et adapté aux ressources disponibles. Il doit par exemple éviter les conflits de salle, limiter les périodes creuses et respecter la charge horaire prévue pour chaque module [6, 43].

Plus qu'un simple tableau d'horaires, l'emploi du temps reflète l'organisation pédagogique d'un établissement. Il résulte d'un processus complexe qui cherche à concilier les logiques institutionnelles (règlements, maquettes de formation), les réalités matérielles (nombre de salles, capacités), et les préférences des utilisateurs (disponibilités, fatigue, rythme d'apprentissage) [67, 3, 30].

2.2.2 Dimensions structurelles de la planification

Afin de mieux cerner la spécificité de l'emploi du temps universitaire, il est essentiel d'en distinguer les composantes structurelles. Cette partie aborde les différences conceptuelles entre les niveaux de planification, ainsi que la formulation précise du problème traité.

Différence entre planification, ordonnancement et emploi du temps

Ces trois notions sont complémentaires dans la conception des activités pédagogiques. En effet, la **planification** définit les grandes lignes stratégiques et les ressources nécessaires, tandis que l'**ordonnancement** traduit ces objectifs en séquences temporelles réalisables selon les contraintes, par contre l'**emploi du temps** concrétise ces décisions dans un format exploitable par les usagers. Le tableau ci-dessous (Tableau 2.1) synthétise les différences principales entre ces concepts.

Présentation du problème d'emploi du temps

Le problème d'emploi du temps désigne le processus d'affectation d'un ensemble de tâches à différentes ressources disponibles (enseignants, groupes, salles, etc.) sur des créneaux horaires bien définis, tout en respectant un ensemble varié de contraintes [9]. Chaque tâche représente une unité de travail élémentaire qui demande à la fois une certaine durée d'exécution et une ou plusieurs ressources spécifiques pour sa réalisation. On distingue généralement les ressources humaines (enseignants, étudiants) et les ressources matérielles (salles de cours), chacune étant soumise à des plages de disponibilité généralement fixées à l'avance.

Dans ce contexte, construire un emploi du temps revient à organiser un ensemble d'affectations en respectant une période temporelle donnée, de façon à coordonner les ressources disponibles avec les tâches à exécuter. Ce problème concerne tout particulièrement les établissements éducatifs, où les cours, travaux dirigés et travaux pratiques doivent être planifiés selon les disponibilités des intervenants, les préférences individuelles, ainsi que les capacités matérielles (comme le nombre de salles).

Les contraintes à prendre en compte peuvent fortement varier selon les objectifs et les spécificités du système considéré. De manière générale, on distingue deux grandes familles de contraintes [34, 56, 11] :

— **Les contraintes dures**, qui sont impératives. Leur non-respect entraîne une invalidité de l'emploi du temps généré. Par exemple, un enseignant ne peut pas être affecté à deux cours simultanément, ou une salle ne peut pas être utilisée par deux groupes au même moment.

Aspect	Planification	Ordonnancement	Emploi du temps
Définition	Système formel de coordination des activités visant à optimiser l'allocation des ressources dans un environnement contraint [41].	Allocation temporelle optimale des tâches à des ressources limitées, en tenant compte de contraintes multiples [54].	Représentation concrète des décisions prises en amont sous forme de planning visible [9, 63].
Objectif principal	Répartir les ressources sur une période longue (année, semestre) selon les besoins institutionnels [3].	Générer une séquence de tâches réalisables en tenant compte des contraintes temporelles et logiques [54].	Organiser les séances de manière lisible pour les usagers finaux [30].
Type d'action	Stratégique / Prévisionnelle [54, 67]	Tactique / Organisationnelle [54]	Opérationnelle / Exécutable [67]
Contraintes traitées	Institutionnelles : offre de formation, capacités humaines et matérielles [3].	Techniques : conflits de ressources, enchaînements logiques, équilibrage [11].	Pratiques : conflits d'horaire, lisibilité, satisfaction des usagers [63].
Lien aux ressources	Estimation globale des besoins (enseignants, salles) [3].	Affectation optimale selon les contraintes temporelles et logiques [54].	Affectation finale concrète dans un calendrier [30].

Table 2.1 – Comparaison entre planification, ordonnancement et emploi du temps

— **Les contraintes souples**, dites aussi de préférence. Leur respect est souhaitable mais non obligatoire. Elles reflètent souvent des préférences individuelles ou organisationnelles, comme éviter les cours très tôt le matin ou regrouper les séances sur une même demi-journée.

Dans notre projet, la prise en compte conjointe de ces deux types de contraintes est essentielle pour produire des emplois du temps à la fois valides sur le plan organisationnel et satisfaisants sur le plan humain. Cela nécessite l'utilisation d'algorithmes capables de trouver des compromis intelligents entre exigences strictes et préférences flexibles [33].

2.2.3 Enjeux d'optimisation et de contraintes

La construction d'un emploi du temps pertinent et équilibré repose sur la prise en compte de multiples contraintes. Cette partie explore la complexité du problème d'optimisation ainsi que les différentes contraintes à satisfaire.

La complexité des problèmes d'emploi du temps

La génération d'emplois du temps dans les établissements éducatifs est un problème reconnu comme *NP-complet*, ce qui signifie qu'il appartient à une classe de problèmes particulièrement difficiles à résoudre en informatique théorique [14, 12].

Cette classification repose sur deux caractéristiques principales. D'une part, la validation d'une solution proposée peut se faire efficacement : il suffit de vérifier que l'emploi du temps respecte l'ensemble des contraintes définies (disponibilité des ressources, absence de conflits, etc.). D'autre part, trouver la solution optimale parmi toutes les configurations possibles demande un effort computationnel considérable, car le nombre de combinaisons augmente de façon exponentielle avec la taille du problème [5].

La complexité intrinsèque provient de la gestion simultanée d'un grand nombre de variables : enseignants, groupes d'étudiants, modules, salles, créneaux horaires, et types de séances (cours, TD, TP). Chaque élément est contraint par des règles de disponibilité, de compatibilité ou de préférence. Ces contraintes peuvent parfois entrer en conflit ou se révéler difficilement conciliables, accentuant encore la difficulté du problème [26].

Par exemple, la planification impliquant 15 enseignants, 20 groupes d'étudiants, 8 salles et 40 créneaux horaires correspond à un espace de recherche contenant plusieurs milliards de configurations possibles. Chaque ajout de contrainte supplémentaire — telle que l'interdiction de cours avant 10h pour certains enseignants, ou l'exigence de salles spécifiques pour les travaux pratiques — réduit l'espace de solutions valides tout en augmentant le coût algorithmique pour les identifier [14].

Cette complexité justifie le recours à des méthodes avancées comme les algorithmes génétiques, les métaheuristiques ou les approches hybrides pour produire des solutions satisfaisantes dans un temps raisonnable [26].

Contraintes de planification

L'élaboration d'un emploi du temps pour un établissement d'enseignement supérieur implique la prise en compte d'un ensemble riche et structuré de contraintes. Ces contraintes sont classiquement divisées en deux grandes catégories : les **contraintes dures (hard constraints)** et les **contraintes souples (soft constraints)**, selon qu'elles soient obligatoires ou simplement préférentielles [42].

Contraintes dures (Hard Constraints) Ce sont les contraintes strictes qui doivent absolument être respectées pour assurer la faisabilité d'un emploi du temps [12, 9] :

- **Contraintes liées aux locaux :**
  - Une salle ne peut accueillir qu'une seule séance à la fois.
  - Le nombre de séances simultanées ne doit pas dépasser le nombre de salles disponibles.
  - Certaines séances (ex. TP) nécessitent des salles spécialisées (laboratoires, salles informatiques).
  - La capacité d'une salle doit couvrir le nombre d'étudiants du groupe.
- **Contraintes relatives aux enseignants :**
  - Un enseignant ne peut dispenser qu'une seule séance à la fois.
  - Il ne peut intervenir simultanément dans plusieurs sections ou promotions.
  - Le volume horaire hebdomadaire maximal doit être respecté.
- **Contraintes pédagogiques :**
  - Un module ne doit être enseigné qu'une seule fois par semaine pour un groupe, sauf

si plusieurs séances sont prévues (CM + TD).
  - Certaines matières nécessitent un ordre pédagogique (ex. CM avant TD/TP).
  - Toutes les séances ont une durée fixe (ex. 1h30), interdisant les chevauchements.
- **Contraintes structurelles :**
  - Il ne peut y avoir de conflits de ressources : une salle ou un enseignant ne peut être affecté à deux séances simultanées.
  - Les sections d'une même promotion ne peuvent partager un enseignant ou une salle au même moment, sauf en cas de cours commun.

Contraintes souples (Soft Constraints) Les contraintes souples ne sont pas obligatoires, mais leur satisfaction améliore considérablement la qualité de l'emploi du temps, tant du point de vue organisationnel que du confort des utilisateurs [40, 39] :

- **Préférences des enseignants :**
  - Éviter les séances tôt le matin (ex. avant 9h) ou trop tard dans la journée.
  - Regrouper les séances sur des journées consécutives.
  - Libérer certains jours (ex. le jeudi pour la recherche ou les déplacements).
- **Cohérence et qualité de vie :**
  - Éviter les périodes creuses pour les enseignants et les étudiants.
  - Respecter une pause fixe institutionnelle (ex. 13h00 – 13h30).
  - Minimiser les déplacements entre bâtiments dans un laps de temps court.
  - Ne pas espacer deux séances par un seul créneau libre (sauf souhait explicite).
- **Confort pédagogique :**
  - Assurer une répartition équilibrée des séances sur la semaine.
  - Regrouper les séances d'un même module dans des journées proches.
  - Répartir équitablement l'utilisation des salles spécialisées.

La bonne gestion de ces contraintes permet de générer des emplois du temps à la fois viables et acceptables pour l'ensemble des acteurs concernés [12].

2.3 L'intelligence artificielle et le langage naturel

L'intelligence artificielle (IA) occupe aujourd'hui une place centrale dans la résolution de nombreux problèmes complexes, notamment dans le domaine de l'interaction homme-machine. Parmi ses branches les plus prometteuses figure le traitement automatique du langage naturel (TALN ou NLP pour *Natural Language Processing*)), qui permet aux machines de comprendre, analyser et générer du texte humain. Avant d'explorer les modèles avancés utilisés dans ce projet, il est essentiel de revenir sur les bases de l'IA et des approches NLP.

2.3.1 Principes fondamentaux et définitions

L'**IA** désigne l'ensemble des théories, méthodes et technologies qui visent à permettre à des machines d'accomplir des fonctions cognitives comparables à celles de l'être humain, telles que l'apprentissage, le raisonnement, la planification ou la perception. Selon Russell et Norvig, "l'intelligence artificielle consiste à concevoir des agents capables d'agir de manière rationnelle

dans un environnement donné" [62]. Elle englobe un large spectre d'approches, incluant l'apprentissage automatique (*machine learning*), l'apprentissage profond (*deep learning*) et les systèmes à base de règles. Comme le précisent Goodfellow et al., "l'apprentissage profond a permis d'importants progrès dans des domaines comme la vision, le langage et la reconnaissance vocale" [22].

Le **TALN** est une branche de l'IA qui vise à permettre aux ordinateurs de comprendre, interpréter, manipuler et générer du langage humain. Jurafsky et Martin le définissent comme "la technologie qui permet à des machines d'analyser, de produire et d'interagir avec le langage naturel humain" [32]. Il combine des connaissances linguistiques, informatiques et statistiques pour traiter des textes ou des conversations dans des applications variées comme les traducteurs automatiques, les chatbots ou les systèmes de résumé automatique. Cambria et White soulignent que "le NLP est en constante évolution, passant de simples analyses syntaxiques à une compréhension sémantique profonde du langage" [10].

2.3.2 Apprentissage du langage

Aujourd'hui, les modèles d'intelligence artificielle comme GPT-2 ou T5 sont capables de produire du texte très proche de celui d'un humain. Mais ce qu'on oublie souvent, c'est qu'ils apprennent à le faire en suivant, sans qu'on le leur dise, les règles du langage humain. En fait, ils utilisent des structures comme la grammaire, le sens des mots ou encore le contexte des phrases, ce qui vient directement de la linguistique. Comme l'expliquent [31], ces modèles apprennent "des représentations syntaxiques et sémantiques"même sans qu'on leur donne ces règles de manière directe.

Par exemple, GPT-2 [58] apprend à deviner le mot suivant dans une phrase, simplement en lisant beaucoup de textes. À force, il finit par comprendre comment les phrases sont construites et produit du texte qui a du sens. T5 [59], lui, transforme toutes les tâches (comme traduire ou résumer un texte) en une seule tâche de génération de texte. Ce modèle s'appuie lui aussi sur des formes du langage apprises dans les données.

Donc même si ces modèles sont basés sur des techniques d'apprentissage automatique, ils utilisent des bases solides venues de la linguistique. Cela montre que la langue et la manière dont on la comprend jouent un rôle essentiel dans les progrès de l'IA aujourd'hui.

2.3.3 Modèles modernes de génération de texte : GPT-2 et T5

Les modèles de génération de texte comme GPT-2 et T5 ont fortement contribué aux avancées récentes en traitement du langage naturel. GPT-2, développé par OpenAI, est un modèle qui prédit le mot suivant dans une phrase, ce qui lui permet de créer des textes cohérents et naturels sans supervision directe [58]. De son côté, T5 (Text-to-Text Transfer Transformer) reformule toutes les tâches liées au langage en un problème unique de génération de texte, ce qui facilite l'apprentissage et l'adaptation à différentes applications [59]. Ces modèles sont largement utilisés dans les systèmes de dialogue et les assistants virtuels, car ils peuvent comprendre et générer des réponses pertinentes en langage naturel, améliorant ainsi l'interaction avec les utilisateurs [75].

2.4 Optimisation multi-acteurs et théorie des jeux

Les problématiques d'optimisation impliquant plusieurs agents ou utilisateurs nécessitent des approches capables de modéliser les interactions et les compromis entre ces acteurs. C'est dans ce contexte que la théorie des jeux trouve toute sa pertinence. Elle permet d'étudier les comportements stratégiques au sein de systèmes partagés, notamment à travers des concepts fondamentaux comme l'équilibre de Nash, qui décrit une situation stable où aucun acteur n'a intérêt à changer de stratégie unilatéralement, et l'optimum de Pareto, qui représente un état dans lequel aucune amélioration n'est possible pour un agent sans détériorer la situation d'un autre [47, 52].

2.4.1 Fondements théoriques et application en IA

La **théorie des jeux** représente une branche des mathématiques appliquées qui s'intéresse à l'étude des interactions stratégiques entre plusieurs agents rationnels. Selon Osborne et Rubinstein, elle permet de modéliser et d'analyser des situations où les choix d'un acteur influencent ceux des autres, dans un contexte où chaque participant cherche à maximiser ses propres gains [49].

Introduite initialement en économie, cette théorie a progressivement été adoptée dans d'autres disciplines comme la biologie, les sciences sociales et plus récemment, l'intelligence artificielle [44, 66]. Dans ce cadre, elle constitue un outil puissant pour comprendre et concevoir des systèmes où plusieurs entités doivent coopérer ou rivaliser.

Parmi les concepts fondamentaux de la théorie des jeux, l'**équilibre de Nash** occupe une place centrale. Il décrit une situation stable où aucun joueur ne peut améliorer son gain en changeant unilatéralement de stratégie [47]. Un autre principe important est l'**optimalité de Pareto**, qui correspond à un état où il est impossible d'améliorer la situation d'un individu sans détériorer celle d'un autre [52].

En intelligence artificielle, la théorie des jeux joue un rôle fondamental dans la modélisation des interactions entre agents intelligents. Elle est notamment utilisée pour élaborer des mécanismes de coordination, de négociation ou encore de partage de ressources dans les systèmes multi-agents, où chaque agent poursuit ses propres objectifs tout en tenant compte des actions des autres [66]. Ce type de modélisation est particulièrement pertinent dans les environnements complexes où les agents doivent tenir compte non seulement de leurs objectifs, mais aussi de ceux des autres.

2.4.2 Interaction stratégique dans la planification

Dans ce travail , l'organisation des emplois du temps est modélisée comme un jeu multi-acteurs, impliquant divers participants: enseignants, étudiants et salles. chacun de ces acteurs a ses propres objectifs et contraintes. Pour répondre à ces enjeux, nous utilisons de concepts fondamentaux de la théorie des jeux. Deux notions sont particulièrement pertinentes :

Équilibre de Nash: Un emploi du temps est dit en équilibre de Nash lorsque aucun acteur ne peut améliorer sa situation sans dégrader celle d'un autre [47]. Cela garantit une certaine

stabilité, car chaque acteur est satisfait de son emploi du temps étant donné celui des autres

Optimalité de Pareto: Une solution est considérée comme optimale au sens de pareto si l'on ne peut améliorer la satisfaction d'un acteur sans réduire celle d'un autre[52]. Ce principe permet de filtrer les solutions injustes ou déséquilibrées.

En combinant ces deux critères, nous obtenons des solutions de planification à la fois stables (grâce à l'équilibre de Nash) et équitables (grâce à l'optimalité de Pareto). Chaque solution est représentée sous forme de vecteur multi-objectif indiquant le degré de satisfaction de chaque acteur.Le système sélectionne ensuite les emplois du temps les mieux équilibrés en tenant compte de ces interactions stratégiques

2.5 Conclusion

Dans ce chapitre, nous avons présenté la problématique de la planification des emplois du temps dans le contexte universitaire, en soulignant la complexité liée aux multiples contraintes à prendre en compte. Pour y faire face, dans la suite de ce travail, nous explorons les apports de l'intelligence artificielle, notamment à travers le traitement automatique du langage naturel (NLP) et l'utilisation de modèles comme **GPT-2** et **T5**. Enfin, une introduction à la théorie des jeux et aux approches multi-acteurs sera proposée afin d'améliorer l'équilibre et la qualité des emplois du temps générés.

3

Description de l'Architecture du Système

Sommaire

3.1 Introduction . . . 16
3.2 Vision et but du système développé . . . 16
3.3 Vue d'ensemble du fonctionnement . . . 16
3.3.1 Étapes générales : de la question à la requête SQL . . . 16
3.3.2 Circulation de l'information dans le système . . . 17
3.3.3 Exemple simplifié d'interaction . . . 17
3.4 Intérêt d'un modèle pré-entraîné dans notre système . . . 18
3.5 T5 : Le cœur transformeur de notre générateur SQL . . . 19
3.5.1 Présentation du modèle T5 . . . 19
3.5.2 Architecture du modèle T5 . . . 20
3.6 Modélisation de l'architecture du NL2SQL . . . 20
3.6.1 Architecture générale du système . . . 20
3.6.2 Processus Opérationnel du Chatbot . . . 21
3.7 Vers des Emplois du Temps Équilibrés . . . 22
3.7.1 l'intérêt d'utiliser la théorie des jeux ? . . . 22
3.7.2 Modélisation via la théorie des jeux . . . 22
3.7.3 Optimisation Multicritère des Emplois du Temps -Méthodologie- . . . 23
3.8 Conclusion . . . 25

3.1 Introduction

Dans ce chapitre, nous présentons l'architecture générale du système développé dans le cadre de ce projet, combinant génération automatique de requêtes SQL à partir du langage naturel et optimisation des emplois du temps. Nous décrivons dans un premier temps les composantes principales du module *NL2SQL*, les étapes de transformation d'une question utilisateur en requête exploitable, ainsi que l'intégration du modèle *T5*, coeur du processus de génération. La seconde partie du chapitre s'intéresse à la problématique de l'optimisation des emplois du temps. Nous montrons comment la modélisation inspirée de la théorie des jeux permet de produire des solutions équilibrées, en prenant en compte les préférences des différents acteurs et les contraintes du système. Cette vue d'ensemble a pour but de clarifier le fonctionnement global du système et de justifier les choix méthodologiques adoptés.

3.2 Vision et but du système développé

L'objectif principal du système développé est de permettre aux utilisateurs d'interagir avec une base de données en utilisant un langage naturel, sans avoir besoin de maîtriser le langage SQL. Cette approche vise à rendre les systèmes d'information plus accessibles, en particulier pour les utilisateurs non techniques, en traduisant automatiquement les requêtes formulées en langage humain vers des instructions compréhensibles par la base de données. Le système cherche à proposer des requêtes SQL qui correspondent au mieux à l'intention exprimée par l'utilisateur, tout en respectant la structure du schéma de la base de données. Toutefois, cette conversion repose sur une interprétation automatique du langage, qui peut être influencée par l'ambiguïté des phrases ou la complexité du schéma, ce qui signifie que la précision des requêtes générées peut varier. Le développement de ce type de chatbot s'inscrit dans une tendance plus large d'amélioration de l'accessibilité aux systèmes de données via des interfaces conversationnelles intelligentes [2].

3.3 Vue d'ensemble du fonctionnement

Cette section vise à présenter de manière synthétique le fonctionnement global du chatbot développé, en décrivant les grandes étapes de traitement de l'information, depuis la question posée par l'utilisateur jusqu'à la génération de la requête SQL correspondante. L'objectif est d'expliquer comment les différents composants interagissent sans entrer encore dans les détails techniques des algorithmes.

3.3.1 Étapes générales : de la question à la requête SQL

Le processus de traitement dans le chatbot suit un enchaînement logique d'étapes qui peuvent être résumées comme suit :

— **Entrée utilisateur** : L'utilisateur soumet une question en langage naturel concernant les données d'une base.

— **Prétraitement** : Le texte est nettoyé et encodé afin d'être compatible avec les modèles de traitement du langage.
— **Compréhension de la question et du contexte** : Le modèle identifie les entités clés de la question et prend en compte le schéma de la base de données pour guider la génération.
— **Génération de la requête SQL** : Un modèle de transformation produit la requête SQL correspondant à l'intention exprimée.
— **Retour à l'utilisateur** : La requête SQL est affichée (ou exécutée selon le cas d'usage), fournissant une réponse à la demande initiale.

3.3.2 Circulation de l'information dans le système

La circulation de l'information dans le système se fait de façon séquentielle. L'entrée utilisateur traverse un pipeline constitué de modules spécialisés. Le schéma de la base est intégré dans l'entrée pour guider le modèle dans la génération d'une requête conforme aux structures des tables, colonnes et relations. Cette approche permet d'adapter dynamiquement les requêtes aux spécificités de chaque base de données.

3.3.3 Exemple simplifié d'interaction

Prenons un exemple simple d'interaction avec le chatbot dans le contexte de la gestion d'un emploi du temps universitaire.

— **Utilisateur** : *Quels cours sont programmés le mardi matin ?*
— **Système** :
  — Correction grammaticale et orthographique de la requête en français.
  — Traduction de la question corrigée en anglais : *What courses are scheduled on Tuesday morning?*
  — Utilisation du schéma de la base de données :
    — `seances(id_seance, module, jour, heure_debut, salle)`
  — Génération de la requête SQL correspondante :

```
SELECT module, heure_debut, salle
FROM seances
WHERE jour = 'mardi' AND heure_debut < '12:00';
```

Cet exemple illustre le parcours de l'information dans le système, depuis la question initiale de l'utilisateur jusqu'à la génération de la requête SQL. Pour mieux comprendre l'organisation et les interactions entre les différents composants, la figure 3.1 présente une architecture simplifiée du chatbot.

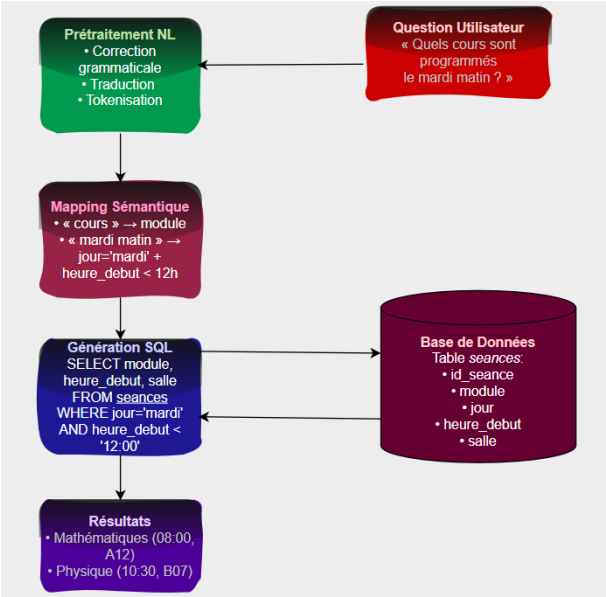

Figure 3.1 – Architecture simplifiée illustrant l'exemple d'interaction du chatbot NL2SQL

3.4 Intérêt d'un modèle pré-entraîné dans notre système

Les modèles de langage pré-entraînés tels que BERT, GPT et T5 jouent un rôle fondamental dans la tâche de génération de requêtes SQL à partir de questions en langage naturel [17, 59, 58]. Grâce à leur architecture basée sur les transformeurs, ces modèles ont démontré une capacité remarquable à modéliser les structures syntaxiques et sémantiques [69], ainsi que le contexte global d'un énoncé [8], ce qui leur permet non seulement de saisir l'intention sous-jacente d'une question, mais aussi de générer des requêtes SQL précises, y compris lorsque ces dernières sont formulées de manière complexe ou ambiguë.

Dans le contexte de la génération de requêtes SQL à partir de questions en langage naturel, les modèles pré-entraînés offrent plusieurs avantages clés : capacité à gérer des formulations variées, correction implicite des erreurs de langage, robustesse aux ambiguïtés, et alignement sémantique avec les structures relationnelles de la base de données. Des travaux comme [74, 35, 27] ont montré l'efficacité de ces approches dans les tâches de Text-to-SQL, y compris sur des benchmarks [1] exigeants comme Spider [76].

Parmi ces modèles, T5 se distingue particulièrement par son architecture de type encodeur-décodeur et son cadre unifié de traitement de texte sous forme de traduction de tâche. Grâce à sa flexibilité et sa puissance, T5 est bien adapté à notre problématique de génération de requêtes SQL à partir du langage naturel [64].

1. Un *benchmark* est un jeu de données standardisé utilisé pour tester et comparer les performances des modèles d'apprentissage automatique sur une tâche précise

3.5 T5 : Le cœur transformeur de notre générateur SQL

Avant d'entrer dans les détails techniques du modèle T5, il est important de comprendre pourquoi ce modèle a été choisi comme base pour notre système de génération automatique de requêtes SQL. Dans cette section, nous allons donc présenter le modèle T5, son principe de fonctionnement général, ainsi que les raisons pour lesquelles il représente une solution pertinente pour la tâche NL2SQL.

3.5.1 Présentation du modèle T5

Le modèle **T5** (*Text-To-Text Transfer Transformer*), introduit par Raffel et ses collègues [59], cherche à unifier toutes les tâches du traitement automatique du langage naturel (TALN) en les formulant comme une simple transformation de texte. Cela signifie que des tâches comme la traduction, la classification, le résumé ou encore la génération de texte peuvent toutes être résolues en convertissant un texte d'entrée en un texte de sortie [59].

Contrairement à d'autres modèles comme **BERT** [17], qui est spécialisé dans la compréhension de texte, ou **GPT** [8], conçu pour la génération de texte, T5 utilise une architecture de type *encoder-decoder*. Cette architecture lui permet à la fois de comprendre un texte et d'en générer un autre, ce qui le rend particulièrement adapté aux tâches complexes comme la conversion de questions en langage naturel vers des requêtes SQL [71]. La Figure 3.2 illustre les trois architectures de ces trois modèles que nous avons cité.

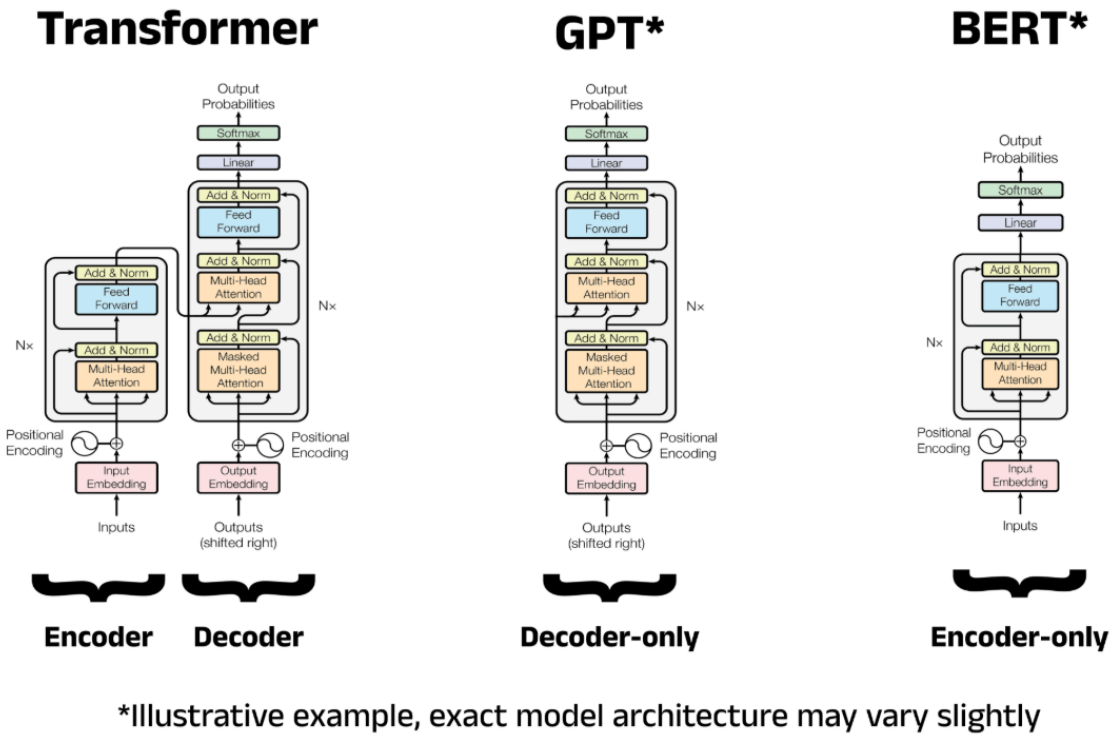

Figure 3.2 – Comparaison des architectures des modèles GPT, BERT et T5 [59, 57, 17, 1].

Le modèle T5 a aussi été pré-entraîné sur un grand corpus de textes web appelé **C4 (Colossal Clean Crawled Corpus)**, qui a été soigneusement nettoyé pour garantir une bonne qualité

linguistique [59]. Grâce à cet apprentissage sur de très grandes quantités de données, T5 est capable de bien saisir le sens et la structure du langage.

Avec son approche text-to-text, son architecture performante et son apprentissage à grande échelle, T5 est devenu un outil puissant et polyvalent, notamment pour des applications comme la génération de requêtes SQL à partir de questions en langage naturel (NL2SQL) [35].

3.5.2 Architecture du modèle T5

Le modèle **T5** (*Text-to-Text Transfer Transformer*) repose sur une architecture de type *encoder-decoder* (voir Figure 3.2), inspirée du transformeur standard introduit par Vaswani et al. [71]. Cette structure permet au modèle de traiter des tâches où la compréhension et la génération de texte doivent coexister. Contrairement à **BERT** [17], qui utilise uniquement la partie encodeur pour comprendre le texte, et à **GPT-2** [58], qui utilise uniquement la partie décodeur pour générer du texte, T5 combine les deux composants pour offrir une flexibilité maximale dans le traitement des tâches de traitement automatique du langage naturel.

Cette combinaison permet à T5 de convertir toutes les tâches de langage en un format uniforme de type *text-to-text*, facilitant ainsi l'apprentissage et l'adaptation à diverses applications, telles que la traduction, la classification, le résumé ou la génération de requêtes SQL à partir du langage naturel.

3.6 Modélisation de l'architecture du NL2SQL

Avant d'entrer dans les détails , nous avons d'abord représenté l'architecture globale du système pour mieux comprendre comment les différentes étapes se connectent entre elles.

3.6.1 Architecture générale du système

Le fonctionnement global de notre système peut être résumé en quatre étapes principales, comme il est illustré par la Figure 3.3.

1. **Pré-traitement** : Fusion des textes (question et schéma) en une seule chaîne. Tokenisation avec `T5Tokenizer` (chaque mot devient un identifiant numérique). Génération des vecteurs `input_ids` et `attention_mask`.
2. **Tokenisation** : Chaque mot ou sous-mot est converti en un ID (ex. `"select"` $\rightarrow$ 23). Les masques permettent d'ignorer les `padding tokens`.
3. **Fine-tuning du modèle T5** : Entraînement supervisé du modèle T5 sur des paires (question + schéma, requête SQL). Optimisation avec l'algorithme AdamW. Évaluation avec les métriques BLEU et Exact Match.
4. **Post-traitement** : Décodage des séquences d'identifiants en texte lisible. Suppression des tokens spéciaux (`<pad>`, `<eos>`, etc.). Nettoyage du texte généré (ex. `"SelectT nAme"` $\rightarrow$ `"select name"`). Évaluation des performances (BLEU, Exact Match).

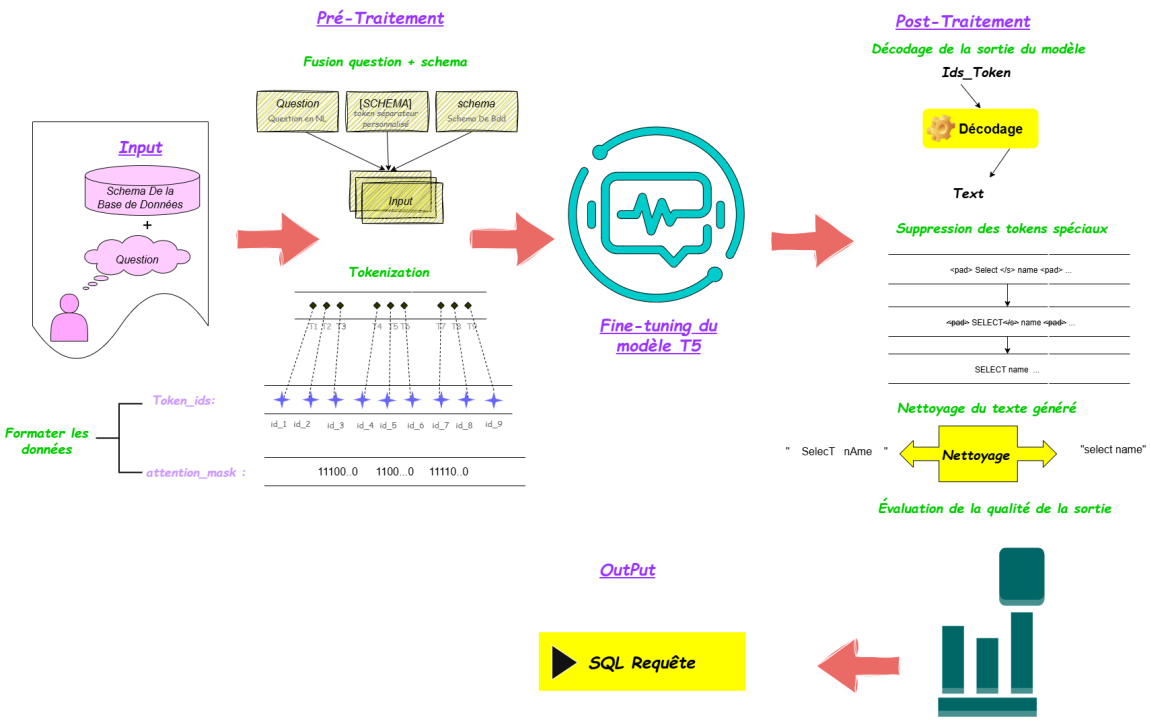

Figure 3.3 – Architecture du système NL2SQL basé sur le modèle T5

3.6.2 Processus Opérationnel du Chatbot

L'algorithme 3.1 suivant résume le fonctionnement général de notre chatbot de traduction NL2SQL.

Algorithm 3.1 : Génération de requêtes SQL à partir de questions avec T5

Entrée : Question utilisateur en langage naturel

Sortie : Requête SQL générée

function T5_NL2SQL_Chatbot(question)
    $schema \leftarrow$ ExtraireSchemaBaseDeDonnées
    $input \leftarrow$ Concaténer(question, "[SCHEMA]", schema)
    $input\_tokens \leftarrow$ Tokenizer(input)
    $attention\_mask \leftarrow$ GénérerMask(input_tokens)
    $ids\_sortie \leftarrow$ T5.Generate(input_tokens, attention_mask)
    $requte\_sql \leftarrow$ Tokenizer.Decode(ids_sortie, skip_special_tokens=True)
    $requte\_nettoye \leftarrow$ NettoyerTexte(requête_sql)
    $score \leftarrow$ Évaluer(requête_nettoyée, référence)
    return $requte\_nettoye$
end function

Cet algorithme détaille le processus de conversion d'une question posée en langage naturel par un utilisateur en une requête SQL exploitable. Le processus commence par l'extraction du schéma de la base de données pertinente, qui fournit le contexte nécessaire à la compréhension de la question.

Ensuite, la question de l'utilisateur est concaténée avec le schéma extrait pour former une en-

trée unique. Cette entrée est ensuite tokenisée, c'est-à-dire convertie en une séquence de jetons numériques, que le modèle T5 peut traiter. Un masque d'attention est généré pour indiquer au modèle les parties de l'entrée sur lesquelles il doit se concentrer.

Le cœur de l'algorithme repose sur la fonction T5.Generate qui, à partir des jetons d'entrée et du masque d'attention, génère des identifiants de sortie. Ces identifiants sont ensuite décodés pour reconstruire la requête SQL en texte clair. Une étape de nettoyage de texte est appliquée à la requête SQL générée pour assurer sa validité et sa lisibilité.

Enfin, une phase d'évaluation optionnelle peut être réalisée en comparant la requête nettoyée à une référence, permettant de mesurer la performance du modèle. L'algorithme retourne la requête SQL nettoyée, prête à être exécutée sur la base de données.

3.7 Vers des Emplois du Temps Équilibrés

L'élaboration d'emplois du temps optimaux est un défi complexe, que ce soit dans le contexte académique, industriel ou personnel. Traditionnellement abordée par des méthodes d'optimisation classiques, cette problématique gagne en profondeur et en pertinence grâce à l'application de la théorie des jeux.

3.7.1 L'intérêt d'utiliser la théorie des jeux ?

L'optimisation d'un emploi du temps universitaire implique plusieurs acteurs ayant des objectifs potentiellement conflictuels : enseignants, étudiants, et ressources matérielles (salles). La théorie des jeux permet de modéliser ces interactions en considérant chaque acteur comme un joueur rationnel qui cherche à maximiser sa satisfaction, ou à minimiser les conflits liés à ses contraintes. L'utilisation de cette approche nous permet de modéliser ces interactions entre les différentes entités (étudiants, professeurs, machines ou tâches) comme des joueurs cherchant à maximiser leur propre utilité, tout en tenant compte des contraintes et des objectifs des autres. En explorant les concepts d'équilibre de Nash ainsi que l'optimum de Pareto, la théorie des jeux offre un cadre puissant pour analyser les dynamiques de décision et concevoir des emplois du temps qui ne sont pas seulement efficaces, mais aussi stables et équitables pour toutes les parties prenantes.

3.7.2 Modélisation via la théorie des jeux

Appliquer la théorie des jeux aux emplois du temps transforme l'optimisation en un problème d'interactions stratégiques. Les éléments clés sont les joueurs (entités comme étudiants, professeurs), les stratégies (leurs actions possibles, ex: choix de créneau), les fonctions de gain (ou utilités) qui mesurent leurs préférences ainsi que la notion d'équilibre de Nash. Nous détaillerons ci-dessous ces différents éléments.

Joueurs:

- $\mathcal{J}_1$: Les enseignants (préférences horaires, type de séances : Cours, TD, TP).
- $\mathcal{J}_2$: Les étudiants (continuité des séances, réduction des périodes creuses).
- $\mathcal{J}_3$: Les salles (optimisation de l'occupation, évitement des conflits).

Stratégies: Chaque stratégie correspond à une configuration d'emploi du temps E, c'est-à-dire une affectation des séances aux créneaux horaires.

Fonctions de gain (ou utilités): Pour chaque joueur $j \in \{\mathcal{J}_1, \mathcal{J}_2, \mathcal{J}_3\}$, on définit une fonction d'utilité $U_j E \in 0, 100$, mesurant le degré de satisfaction du joueur selon une grille d'évaluation.

$$U_{\text{prof}} E \in 0, 100$$
$$U_{\text{etud}} E \in 0, 100$$
$$U_{\text{salle}} E \in 0, 100$$

Ces fonctions sont calculées à partir de critères tels que :
— Les conflits d'horaires
— Le respect des préférences
— La continuité des cours
— L'occupation équilibrée des ressources

Optimalité de Pareto: Une solution E^* est dite **non dominée** ou **optimale au sens de Pareto** s'il n'existe pas de solution alternative E' telle que :

$$\forall j,\, U_j E' \geq U_j E^* \quad \text{et} \quad \exists\, i \,:\, U_i E' > U_i E^*$$

Autrement dit, une solution est optimale au sens de Pareto lorsqu'aucune autre solution ne permet d'améliorer l'utilité d'un acteur sans détériorer celle d'aucun autre [51].

Équilibre de Nash: Un emploi du temps E^* est dit en **équilibre de Nash** s'il n'existe pas de stratégie alternative E' telle que :

$$U_j E' > U_j E^* \quad \text{sans que} \quad \exists\, i \neq j \,:\, U_i E' < U_i E^*$$

Autrement dit, aucun joueur ne peut améliorer son utilité sans dégrader celle d'au moins un autre joueur [46].

3.7.3 Optimisation Multicritère des Emplois du Temps -Méthodologie-

Pour aboutir à un emploi du temps optimal et équitable, nous employons une approche systématique qui combine la génération de solutions avec des techniques de décision multicritère. Le processus débute par la génération d'un grand ensemble de solutions valides d'emploi du temps, assurant une exploration exhaustive des possibilités. Chacune de ces solutions est ensuite évaluée avec des fonctions d'utilité Uj , qui quantifient la satisfaction des différents acteurs impliqués. Afin d'identifier les compromis les plus pertinents, nous appliquons un filtrage de Pareto, ne retenant que les solutions non dominées — celles pour lesquelles il est impossible d'améliorer la satisfaction d'un acteur sans en dégrader une autre. Finalement, la sélection de la solution la plus équilibrée parmi le front de Pareto est effectuée, visant un compromis qui se rapproche d'un équilibre de Nash, où aucun acteur n'aurait intérêt à changer sa stratégie unilatéralement.

La figure 3.4 illustre notre approche pour élaborer des emplois du temps qui satisfassent au mieux les différents acteurs impliqués dans l'élaboration d'un emploi du temps. La mise en oeuvre de cette architecture est concrétisée via l'Algortthme 3.2 , où nous présentons notre processus modulé en quatre étapes clés, visant à trouver un équilibre optimal face à des objectifs souvent contradictoires.

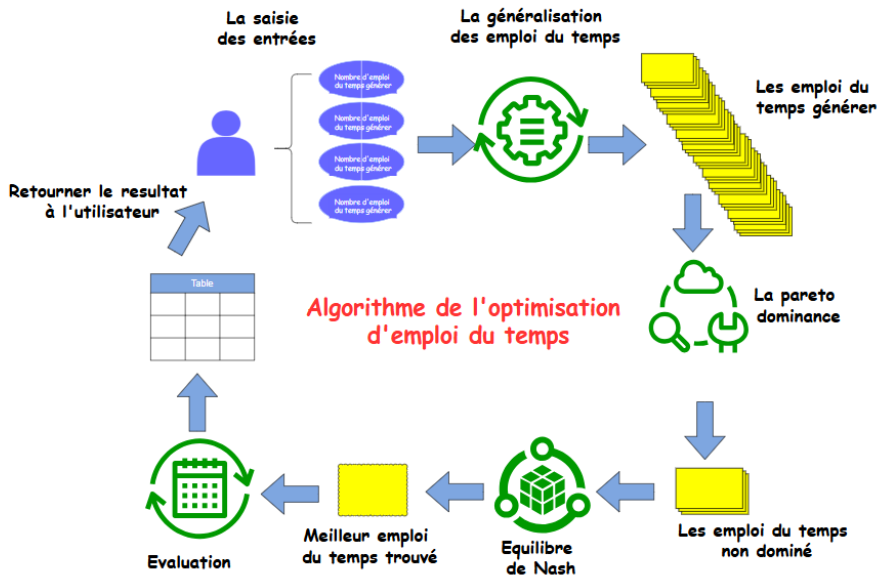

Figure 3.4 – Algorithme d'optimisation des emplois du temps

1. Le cheminement commence par la génération d'un vaste ensemble de solutions valides d'emploi du temps. Cette phase est cruciale, elle assure que l'exploration d'un large éventail de possibilités, en respectant toutes les contraintes de base (pas de chevauchements, disponibilité des ressources, etc.). L'objectif ici est de créer un "réservoir" riche de scénarios potentiels, plutôt que de se limiter à quelques options évidentes.
2. Une fois ces solutions générées, chaque emploi du temps est évalué en utilisant des fonctions d'utilité U_j. Ces fonctions sont au cœur de l'approche multicritère : elles mesurent la "satisfaction" ou le "gain" pour chaque acteur (étudiant, professeur, salle, etc.) pour un emploi du temps donné. Par exemple, un professeur pourrait avoir une utilité plus élevée s'il a ses cours regroupés sur peu de jours, tandis qu'un étudiant préférerait des pauses équilibrées.
3. L'étape suivante est l'application du filtrage de Pareto. C'est ici que nous identifions les solutions les plus pertinentes. Une solution est dite "non dominée" (ou fait partie du front de Pareto) si l'on ne peut pas améliorer la satisfaction d'un acteur sans en dégrader au moins une autre. En d'autres termes, ces solutions représentent les meilleurs compromis possibles ; il n'y a pas d'emploi du temps qui soit "meilleur" pour tout le monde simultanément sur tous les critères.
4. Enfin, parmi les solutions du front de Pareto, une sélection est effectuée en se basant sur un critère de compromis : l'objectif est de choisir la solution qui se rapproche le plus d'un

équilibre de Nash, c'est-à-dire une situation où aucun acteur n'aurait intérêt à modifier unilatéralement sa partie de l'emploi du temps, étant donné les choix des autres. Cela garantit un emploi du temps stable et équitable pour l'ensemble des intervenants.

Algorithm 3.2 Optimisation Multicritère des Emplois du Temps

Entrée : Promotions, Professeurs, Salles, MaxSolutions, MaxIterations
Sortie : $s_{optimal}$ (solution équilibrée)

Initialisation : $S \leftarrow \emptyset$ ▷ Solutions générées
for $i = 1$ à MaxSolutions **do** ▷ Génération aléatoire de solutions valides
    $s_{\text{new}} \leftarrow$ GénérerEDTValide(Promotions, Professeurs, Salles)
    $S \leftarrow S \cup \{s_{\text{new}}\}$
end for
for chaque solution $s \in S$ **do** ▷ Évaluation des solutions (multi-acteurs)
    $U_{prof}, U_{etud}, U_{salle} \leftarrow$ CalculerScores(s)
end for
$P \leftarrow \emptyset$ ▷ Front Pareto
for chaque solution $s_i \in S$ **do** ▷ Filtrage de Pareto
    $domin \leftarrow$ Faux
    for chaque solution $s_k \in S$, $s_k \neq s_i$ **do**
        if s_k domine s_i sur tous les critères **then**
            $domin \leftarrow$ Vrai **; break**
        end if
    end for
    if non $domin$ **then**
        $P \leftarrow P \cup \{s_i\}$
    end if
end for
$s_{optimal} \leftarrow$ TrouverÉquilibreNash(P) ▷ Sélection par équilibre de Nash
return $s_{optimal}$

3.8 Conclusion

Dans ce chapitre, nous avons présenté l'architecture complète de notre système *NL2SQL* basé sur le modèle pré-entraîné *T5*, ainsi que son extension vers l'optimisation des emplois du temps. Nous avons détaillé les étapes de traitement allant du prétraitement des questions jusqu'à la génération automatisée de requêtes SQL. L'utilisation du modèle *T5* s'est révélée pertinente, grâce à sa capacité à comprendre et formuler des requêtes complexes à partir du langage naturel. En seconde partie, nous avons introduit une méthodologie d'optimisation multicritère basée sur la théorie des jeux, permettant de générer des emplois du temps équilibrés. Cette approche offre une réponse originale et robuste aux besoins académiques, en conciliant adaptabilité, équité et efficacité. Ces fondations méthodologiques constituent une base solide pour l'implémentation concrète du système, ainsi que pour les phases d'expérimentation et d'évaluation présentées dans les chapitres suivants.

4

Expérimentation et interprétation des résultats

Sommaire

4.1 Introduction 28
4.2 Environnement expérimental 28
4.2.1 Outils logiciels et bibliothèques 28
4.2.2 Jeux de données exploités 31
4.3 Implémentation du modèle T5 pour la génération NL2SQL 32
4.3.1 Prétraitement des données 32
4.3.2 Entraînement du modèle T5 33
4.3.3 Évaluation des performances (BLEU, Exact Match) 33
4.3.4 Comparaison avec des approches alternatives 36
4.4 Optimisation de l'emploi du temps 37
4.4.1 Modélisation du problème d'emploi du temps par la théorie des jeux . . . 37
4.4.2 Présentation et analyse des résultats 38
4.5 Visualisation des résultats dans l'application 42
4.6 Conclusion 47

4.1 Introduction

Dans ce chapitre, nous présentons la phase expérimentale de notre travail, qui vise à démontrer l'efficacité de l'approche que nous avons proposée pour l'optimisation des emplois du temps académiques à travers une application concrète. Cette application repose sur la combinaison de techniques d'optimisation inspirées de la théorie des jeux (équilibre de Nash et dominance de Pareto) avec un système intelligent de génération automatique de requêtes SQL à l'aide d'un chatbot de type NL2SQL.

4.2 Environnement expérimental

Le système proposé repose sur un environnement expérimental soigneusement structuré, combinant des outils logiciels adaptés et des jeux de données pertinents, indispensables à la mise en oeuvre, à l'entraînement et à la validation des différents modules développés.

4.2.1 Outils logiciels et bibliothèques

Dans le cadre de ce travail, nous avons mobilisé un ensemble de langages, de frameworks et de bibliothèques adaptés aux différentes composantes du projet, à savoir : la création d'une application mobile d'optimisation des emplois du temps, le développement d'un chatbot intelligent capable de générer des requêtes SQL, et l'intégration d'algorithmes d'optimisation.

Les outils sélectionnés ont été choisis pour leur efficacité, leur compatibilité avec les modèles de traitement automatique du langage naturel (NLP) et leur facilité d'intégration dans des environnements modernes de développement. Cette section détaille les principaux outils employés, répartis en trois catégories :

— Les **langages de programmation**, qui constituent la base du développement logiciel ;
— Les **frameworks et plateformes**, utilisés pour la création d'interfaces, d'APIs et l'entraînement de modèles ;
— Les **bibliothèques Python**, qui permettent de traiter les données, de manipuler les modèles de langage et d'appliquer des techniques de machine learning ou d'optimisation.

Les trois tableaux suivants présentent ces outils avec leur description, leur logo, ainsi que les références associées.

Langages de programmation utilisés

Le tableau 4.1 présente les principaux langages de programmation utilisés dans le projet, en précisant pour chacun leur rôle spécifique. Il inclut également les logos correspondants pour une meilleure identification visuelle.

1. Python Software Foundation. *Python Programming Language*. `https://www.python.org/`. Consulté en 2025
2. Google. *Dart Programming Language*. `https://dart.dev/`. Consulté en 2025
3. W3Schools. *SQL Tutorial - Structured Query Language*. `https://www.w3schools.com/sql/`. Consulté en 2025

Langage	Description	Logo
Python	Langage de programmation polyvalent et interprété, largement utilisé pour le traitement automatique du langage naturel, l'intelligence artificielle, les algorithmes d'optimisation et la manipulation de données [1].	
Dart	Langage optimisé pour le développement côté client, notamment pour les interfaces mobiles via Flutter. Il permet de créer des applications performantes avec une seule base de code [2].	
SQL	Langage standard pour l'interrogation et la manipulation de bases de données relationnelles. Il est utilisé ici pour extraire les données d'emplois du temps [3].	

Table 4.1 – Langages de programmation utilisés dans le projet

Frameworks et environnements utilisés

Le tableau 4.2 répertorie les principaux frameworks et plateformes mobilisés dans le projet, chacun étant accompagné d'une brève description de son usage spécifique. Les logos facilitent l'identification rapide des outils.

Outil	Description	Logo
Flutter	Framework open-source développé par Google, utilisé pour créer des applications mobiles, web et desktop à partir d'un seul code source. Il est utilisé ici pour développer l'interface utilisateur [4].	
FastAPI	Framework web moderne pour construire des API rapides et performantes en Python, basé sur Starlette et Pydantic. Il permet la création d'interfaces RESTful robustes avec une documentation automatique [5].	
Node.js	Environnement d'exécution JavaScript côté serveur, souvent utilisé pour créer des serveurs légers ou manipuler les fichiers côté backend [6].	
Kaggle	Plateforme d'exécution de notebooks Python dans le cloud, utilisée ici pour l'entraînement et l'évaluation des modèles NLP (T5) [7].	

Table 4.2 – Frameworks et plateformes utilisés dans le projet

4. Google. *Flutter - Build apps for any screen*. `https://flutter.dev/`. Consulté en 2025
5. Sebastián Ramírez. *FastAPI*. `https://fastapi.tiangolo.com/`. Consulté en juin 2025
6. Node.js Foundation. *Node.js - JavaScript runtime built on Chrome's V8 engine*. `https://nodejs.org/`. Consulté en 2025
7. Kaggle Inc. *Kaggle - Your Home for Data Science*. `https://www.kaggle.com/`. Consulté en 2025

Bibliothèques Python utilisées

Le tableau 4.3 recense les principales bibliothèques Python utilisées dans le cadre du projet. Il en précise les fonctions essentielles en traitement de données, visualisation, apprentissage automatique et traitement du langage naturel.

Bibliothèque	Description	Logo
pandas	Permet de manipuler des données structurées sous forme de tableaux (DataFrames), largement utilisée pour le nettoyage et l'analyse de données [8].	
matplotlib	Bibliothèque de visualisation 2D en Python, utilisée pour tracer des courbes et visualiser les résultats [9].	
textblob	Outil de traitement de texte simple basé sur NLTK, utile pour la correction grammaticale et l'analyse de sentiments [10].	
langdetect	Librairie Python pour détecter automatiquement la langue d'un texte. Basée sur l'algorithme de Google [11].	aucun
scikit-learn	Bibliothèque de machine learning populaire, utilisée ici pour le découpage en train/test et d'autres outils statistiques [12].	
torch	Framework de deep learning utilisé pour entraîner les modèles (GPT-2, T5). Fournit des API pour les tenseurs et les réseaux de neurones [13].	
transformers	Librairie de Hugging Face pour intégrer des modèles pré-entraînés comme GPT-2 ou T5 facilement dans un pipeline NLP [14].	
nltk	Toolkit NLP complet pour le traitement du langage naturel : tokenisation, étiquetage, parsing, etc. [15].	

Table 4.3 – Bibliothèques Python utilisées dans le projet

4.2.2 Jeux de données exploités

Pour entraîner et tester le chatbot NL2SQL, plusieurs jeux de données spécialisés ont été envisagés. Deux jeux très utilisés dans la communauté sont **WikiSQL** et **Spider**.

WikiSQL est un dataset massif contenant plus de 80 000 paires question/requête SQL [78]. Il a l'avantage d'être simple à exploiter et largement adopté dans les projets de génération de requêtes SQL. La plupart des requêtes sont de type `SELECT ... FROM table ...`, ce qui est bien adapté aux systèmes ayant une seule table comme base. Toutefois, WikiSQL ne contient pas une spécification explicite du schéma relationnel, ce qui le rend moins pertinent dans un contexte multi-tables ou complexe.

Spider [76], en revanche, est un benchmark plus exigeant et diversifié, couvrant plus de 200 schémas de bases de données et nécessitant une bonne capacité de généralisation de la part du modèle. Cependant, sa complexité élevée le rend difficile à intégrer pour un projet initial, surtout dans un cadre de développement rapide.

Après évaluation de ces deux sources, nous avons finalement opté pour un jeu de données open-source plus adapté à mes besoins : le dataset **Text-to-SQL** disponible sur Kaggle[70]. Il présente plusieurs avantages : il contient une structure claire avec les colonnes `question`, `schema` et `query`, ce qui facilite le fine-tuning d'un modèle de type encodeur-décodeur tel que T5. De plus, il est déjà pré-nettoyé et prêt à être utilisé, ce qui réduit considérablement le temps de préparation des données. Il est particulièrement adapté à un projet où le chatbot interagit avec un seul schéma cohérent d'emploi du temps universitaire. Le tableau 4.4 illustre une Comaparaison entre les trois datasets

8. The Pandas Development Team. *pandas - Python Data Analysis Library*. `https://pandas.pydata.org/`. Consulté en 2025
9. John D. Hunter. *Matplotlib: Visualization with Python*. `https://matplotlib.org/`. Consulté en 2025. 2023
10. Steven Loria. *TextBlob: Simplified Text Processing*. `https://textblob.readthedocs.io/`. Consulté en 2025
11. Shuyo Nakatani. *Langdetect: Language detection library ported from Google's language-detection*. `https://pypi.org/project/langdetect/`. Consulté en 2025
12. Scikit-learn Developers. *Scikit-learn: Machine Learning in Python*. `https://scikit-learn.org/`. Consulté en 2025. 2023
13. Adam Paszke **and** et al. *PyTorch: An Open Source Machine Learning Framework*. `https://pytorch.org/`. Consulté en 2025
14. Hugging Face. *Transformers: State-of-the-art Natural Language Processing*. `https://huggingface.co/transformers`. Consulté en 2025
15. Steven Bird, Edward Loper **and** Ewan Klein. *NLTK: Natural Language Toolkit*. `https://www.nltk.org/`. Consulté en 2025. 2023

Critère	**WikiSQL** [78]	**Spider** [76]	**Text-to-SQL)** [70]
Nombre de questions	~80 000	~10 000	61 948 (valeurs uniques)
Structure du schéma	Une seule table par exemple	Multi-tables (relations complexes)	Une ou plusieurs tables avec schéma explicite
Complexité des requêtes	Faible à moyenne	Élevée (requêtes imbriquées, jointures)	Moyenne à élevée
Format des données	question+table + SQL	question+schéma relationnel+SQL	question+schema (texte) + SQL
Adapté aux projets simples	Oui	Non (trop complexe)	Oui
Avantages	Taille importante, simple à manipuler	Très réaliste, évalue la généralisation	Schéma clair, données prétraitées, bonne couverture
Limites	Génère souvent `SELECT ... FROM table` sans contexte riche	Trop exigeant sans traitement complexe	Dataset limité et peu adapté à l'entraînement

Table 4.4 – Comparaison entre les jeux de données WikiSQL, Spider et Text-to-SQL

4.3 Implémentation du modèle T5 pour la génération NL2SQL

L'implémentation du modèle T5 pour la génération de requêtes SQL à partir du langage naturel repose sur plusieurs étapes clés, allant de la préparation des données jusqu'à l'évaluation des performances. Ce qui suit présente le processus complet d'entraînement, ainsi que les résultats obtenus et leur comparaison avec d'autres approches.

4.3.1 Prétraitement des données

Avant de procéder à l'entraînement du modèle, nous avons effectué une phase essentielle de préparation des données. Le jeu de données utilisé dans ce projet est le dataset `Text-to-SQL` proposé sur Kaggle [70]. Ce jeu contient des paires *question / requête SQL*, accompagnées d'une description du schéma de la base de données.

L'objectif de ce prétraitement est de formater les entrées dans un format compatible avec le modèle T5, qui repose sur une approche text-to-text [59]. Pour chaque exemple, nous avons donc construit une chaîne de caractères combinant la question en langage naturel et le schéma associé, séparés par un marqueur spécial `[SCHEMA]` :

```
Input  : How many courses are scheduled on Monday? [SCHEMA] seances(id_seance, module, jour,
Output : SELECT COUNT(*) FROM seances WHERE jour = 'Monday';
```

Cette structure permet au modèle d'apprendre à faire le lien entre les termes du langage naturel et les colonnes spécifiques de la base de données.

Ensuite, nous avons divisé les données en trois ensembles :

— **Ensemble d'entraînement** (80 %) : pour ajuster les poids du modèle.

— **Ensemble de validation** (10 %) : pour suivre la performance et éviter le surapprentissage.

— **Ensemble de test** (10 %) : pour évaluer la qualité finale des prédictions.

Enfin, nous avons utilisé le tokenizer de T5 fourni par Hugging Face pour convertir les textes en séquences numériques. Chaque entrée est tronquée ou complétée jusqu'à une longueur maximale de 512 tokens pour les questions, et 150 tokens pour les requêtes SQL. Les masques d'attention et les étiquettes de sortie ont été générés automatiquement pour chaque exemple.

4.3.2 Entraînement du modèle T5

Pour la génération de requêtes SQL à partir du langage naturel, nous avons choisi le modèle **T5-base**, une version intermédiaire du modèle *Text-to-Text Transfer Transformer* introduit par Raffel et al. [59]. Ce modèle repose sur une architecture de type *encoder-decoder* dérivée du Transformer [71], et a été pré-entraîné sur le corpus **C4** (Colossal Clean Crawled Corpus), un ensemble massif de pages web filtrées pour la qualité linguistique [59].

L'entraînement a été effectué sur la plateforme Kaggle, en utilisant le framework `PyTorch` et les bibliothèques `transformers` de Hugging Face. Le modèle a été fine-tuné sur le jeu de données prétraité `Text-to-SQL` [70] selon les spécifications suivantes :

— **Modèle utilisé** : `t5-base`

— **Optimiseur** : AdamW [38]

— **Taux d'apprentissage** : 3×10^{-5}

— **Nombre d'époch**s : 7

— **Batch size** : 4

— **Longueur maximale d'entrée** : 256 tokens

— **Longueur maximale de sortie** : 150 tokens

Les données ont été chargées à l'aide d'un `DataLoader` personnalisé. Durant chaque itération, le modèle reçoit une question concaténée à un schéma en entrée, et génère la requête SQL en sortie. Le modèle est entraîné à minimiser la fonction de perte (`loss`) entre la séquence prédite et la séquence cible, en utilisant le mode `teacher forcing`.

Après chaque époque, la perte d'entraînement et la perte de validation ont été enregistrées pour évaluer la convergence du modèle. Un score BLEU est également calculé sur l'ensemble de validation, afin de mesurer la similarité entre les requêtes générées et les requêtes attendues.

4.3.3 Évaluation des performances (BLEU, Exact Match)

Après l'entraînement du modèle T5, une phase d'évaluation a été réalisée pour mesurer la qualité des requêtes SQL générées à partir des questions en langage naturel. Deux métriques

principales ont été utilisées : **BLEU** et **Exact Match (EM)**.

1. Score BLEU (Bilingual Evaluation Understudy)

Le *BLEU score* est une mesure automatique classiquement utilisée pour évaluer la qualité de textes générés, notamment en traduction automatique [50]. Il compare les n-grammes de la requête prédite à ceux de la requête attendue (référence). Un score proche de 1 indique une forte similarité lexicale.

Dans le cadre de ce projet, la fonction `sentence_bleu` de la bibliothèque `nltk` a été utilisée avec un lissage de type `SmoothingFunction().method4`, pour éviter les scores nuls dans les cas de très courtes séquences.

2. Exact Match (EM) La métrique *Exact Match* permet d'évaluer si la requête SQL générée par le modèle correspond exactement à la requête attendue. Il s'agit d'une comparaison stricte entre les deux chaînes de caractères : si elles sont identiques, le score est 1, sinon 0. Le score global est obtenu en calculant la moyenne de ces résultats sur l'ensemble des exemples.

Cette métrique est largement utilisée dans le domaine du NLP pour les tâches nécessitant une réponse précise, comme la génération de texte structuré ou la réponse à des questions [60].

3. Résultats obtenus

nous avons entraîné le modèle T5 pendant 7 époques sur le dataset WikiSQL. Après chaque époque, nous avons évalué la performance à l'aide de la loss sur l'ensemble d'entraînement et de validation, ainsi que des scores BLEU et Exact Match. Le tableau ci-dessous résume ces résultats, et une évaluation finale a été réalisée sur le jeu de test.

Table 4.5 – Résultats d'entraînement et de validation sur le dataset WikiSQL

Époque	Train Loss	Val Loss	BLEU	Exact Match (%)
1	0.0565	0.0134	0.8948	72.14
2	0.0149	0.0110	0.9097	75.20
3	0.0112	0.0102	0.9166	76.31
4	0.0091	0.0101	0.9167	76.64
5	0.0077	0.0101	0.9210	77.18
6	0.0066	0.0105	0.9228	77.62
7	0.0058	0.0104	0.9216	77.73
Résultats sur le jeu de test			**0.9186**	**77.46**

Table 4.6 – Résultats d'entraînement du modèle T5 sur le dataset Text-to-SQL

Époque	Train Loss	Val Loss	BLEU Score	Exact Match (%)
1	0.0526	0.0487	0.5675	27.88
2	0.0465	0.0464	0.5761	28.72
3	0.0425	0.0453	0.5852	30.73
4	0.0392	0.0441	0.5951	31.58
5	0.0366	0.0429	0.6013	33.13
6	0.0341	0.0423	0.6097	33.96
7	0.0319	0.0418	0.6157	35.21

Visualisation des performances

Les courbes ci-dessous permettent de visualiser l'évolution de la performance du modèle T5 durant l'apprentissage sur le dataset WikiSQL. La **Figure 4.1** illustre la décroissance de la fonction de perte (*loss*) pour les ensembles d'entraînement et de validation. La **Figure 4.2** montre quant à elle la progression des scores *BLEU* et *Exact Match* au fil des époques.

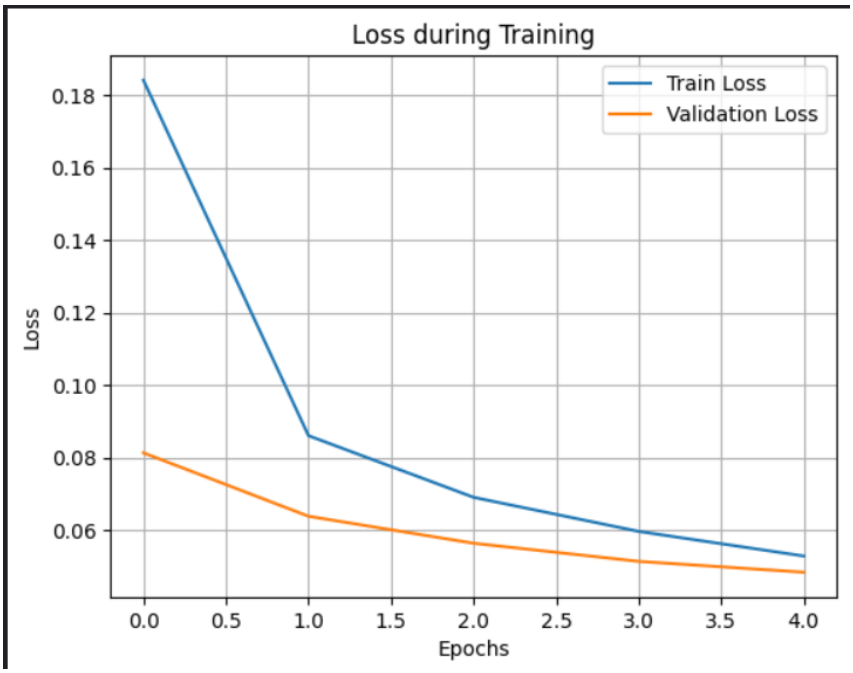

Figure 4.1 – Courbe d'évolution de la loss (entraînement et validation) du modèle T5 sur WikiSQL

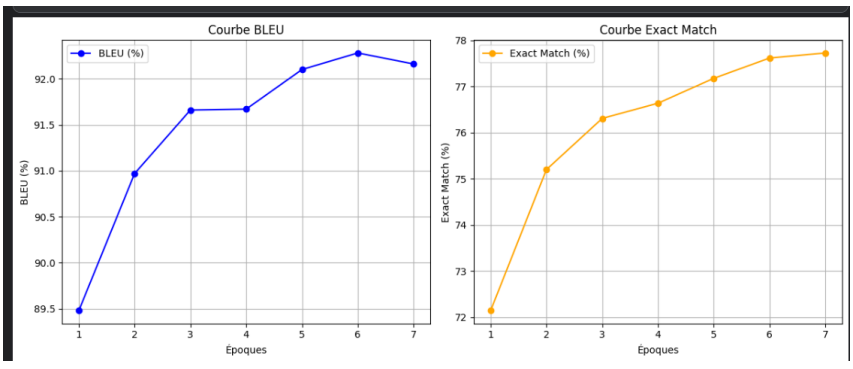

Figure 4.2 – Évolution des scores BLEU et Exact Match sur WikiSQL au fil des époques

Exemple de question test générée par le modèle

Pour illustrer les résultats obtenus, le **tableau ??** présente des questions en langage naturel , pour chacune, la requête SQL générée par le modèle T5 et la requête correcte attendue.

Question	SQL générée	SQL réelle
What is terrence ross' nationality	SELECT Nationality FROM table WHERE Player = Terrence Ross	SELECT Nationality FROM table WHERE Player = Terrence Ross
What clu was in toronto 1995-96	SELECT School/Club Team FROM table WHERE Years in Toronto = 1995-96	SELECT School/Club Team FROM table WHERE Years in Toronto = 1995-96
which club was in toronto 2003-06	SELECT School/Club Team FROM table WHERE Years in Toronto = 2003-06	SELECT School/Club Team FROM table WHERE Years in Toronto = 2003-06
how many schools or teams had jalen rose	SELECT COUNT School/Club Team FROM table WHERE Player = Jen Rose	SELECT COUNT School/Club Team FROM table WHERE Player = Jalen Rose
Where was Assen held?	SELECT circuit FROM table WHERE No = Assen	SELECT Round FROM table WHERE Circuit = Assen
What was the number of race that Kevin Curtain won?	SELECT MAX No FROM table WHERE Race winner = Kevin Curtain	SELECT COUNT No FROM table WHERE Pole Position = Kevin Curtain
What was the date of the race in Misano?	SELECT Date FROM table WHERE Circuit = Misano	SELECT Date FROM table WHERE Circuit = Misano

Table 4.7 – Comparaison entre questions, requêtes SQL générées et requêtes réelles

En comparaison, l'entraînement du même modèle T5 sur le dataset Text-to-SQL a donné des résultats sensiblement inférieurs, notamment au niveau des scores *Exact Match* et *BLEU*, avec un plafonnement autour de 35 % pour l'*Exact Match* après 6 époques. Cela s'explique en partie par la complexité plus élevée du dataset Text-to-SQL, qui contient des schémas de bases de données variés et parfois complexes, contrairement au dataset WikiSQL où chaque requête cible une seule table standard, généralement nommée `table`, ce qui simplifie la tâche du modèle.

4.3.4 Comparaison avec des approches alternatives

Table 4.8 – Comparaison des performances (**Exact Match**) sur le dataset **WikiSQL** (test)

Modèle (Année)	Architecture	EM (%)
notre modèle	T5	**77.46**
SQLNet [78]	Seq2Seq	59.4
TypeSQL [77]	Seq2Seq	68.0
IRNet [25]	Seq2Seq	74.1
BRIDGE [36]	BERT	70.1
RAT-SQL [73]	BERT	75.6
SmBoP [61]	BERT	79.9
PICARD [64]	T5	**82.3**

4.4 Optimisation de l'emploi du temps

La suite du travail porte sur l'optimisation de l'emploi du temps. Cette partie présente la modélisation adoptée ainsi que les résultats obtenus après expérimentation.

4.4.1 Modélisation du problème d'emploi du temps par la théorie des jeux

L'optimisation d'un emploi du temps universitaire implique plusieurs acteurs ayant des objectifs potentiellement conflictuels : enseignants, étudiants, et ressources matérielles (salles). La théorie des jeux permet de modéliser ces interactions en considérant chaque acteur comme un joueur rationnel qui cherche à maximiser sa satisfaction, ou à minimiser les conflits liés à ses contraintes. Comme illustrer dans le chapitre précédent via l'algorithme 3.2 la solution consiste en quatre étapes essentielles:

1. Génération d'un grand ensemble de solutions valides d'emploi du temps.
2. Évaluation de chaque solution avec les fonctions U_j.
3. Application du **filtrage de Pareto** pour retenir les solutions non dominées.
4. Sélection de la solution la plus équilibrée parmi le front de Pareto, représentant un compromis proche d'un équilibre de Nash.

Solution	U_{prof}	U_{etud}	U_{salle}
A	92	86	60
B	85	91	70
C	88	89	88

Table 4.9 – Scores des joueurs pour trois emplois du temps générés

Exemple de résultats La solution C est considérée comme un bon équilibre car :

— Elle maximise l'utilité globale de manière équilibrée.
— Aucun joueur ne peut améliorer significativement sa situation sans désavantager les autres.
— Elle appartient au front de Pareto.

Interprétation des résultats

— **Solution A** : Très favorable aux enseignants, mais mauvaise répartition des salles.
— **Solution B** : Avantage les étudiants, mais utilisation des ressources moyenne.
— **Solution C** : Équilibre global, plus réaliste et acceptable en contexte institutionnel.

Solution	U_{prof}	U_{etud}	U_{salle}
Aléatoire	53	61	50

Table 4.10 – Solution générée sans prise en compte des préférences

Comparaison avec une solution aléatoire Cette solution montre de nombreux conflits, des trous inutiles, et une mauvaise répartition des ressources. Cela justifie le recours à une approche stratégique et multi-objectif.

Récapitulatif et observations La théorie des jeux permet de :

— Intégrer plusieurs objectifs contradictoires de façon équitable.

— Modéliser les préférences des acteurs.

— Trouver un équilibre via les concepts de Nash et de Pareto.

Elle offre ainsi un cadre puissant et souple pour aborder les problèmes complexes d'optimisation multi-objectif comme l'emploi du temps universitaire.

4.4.2 Présentation et analyse des résultats

Dans cette partie, nous présentons les résultats obtenus après avoir appliqué le modèle d'optimisation basé sur la théorie des jeux. nous avons généré plusieurs(ex:10000) emplois du temps et évalué chacun d'eux selon les préférences des différents acteurs impliqués (enseignants, étudiants et gestion des salles). Les solutions ont ensuite été filtrées à l'aide du concept de dominance de Pareto. Nous illustrons cette démarche par des exemples concrets accompagnés de tableaux explicatifs, afin de mieux comprendre les compromis réalisés et la pertinence des solutions obtenues.

Illustration des solutions générées

Affichage de l'emploi du temps : Dans cet exemple, nous avons simulé la génération d'emplois du temps pour une promotion comportant une seule spécialité (Informatique), elle-même composée d'une seule section. .

Chaque spécialité est associée à un **ensemble de sept modules**, répartis comme suit :

— Les modules 1 et 3 comprennent des séances de **cours**, de **TD** et de **TP**.

— Les modules 2 et 7 incluent des **cours** et des **TP**.

— Les modules 4 et 5 sont constitués de **cours** et de **TD**.

— Le module 6 est dispensé uniquement sous forme de **cours magistraux**.

Dans notre simulation, la spécialité contient **5 groupes TD/TP**, et les **8 enseignants** sont affectés aléatoirement aux modules et types de séances, en respectant leurs domaines de compétence et leurs disponibilités.

Les séances sont planifiées dans un parc de **salles variées**, composé de :

— 2 salles de cours

— 5 salles de TD

— 5 salles de TP

Comme illustré dans le tableau 4.11, l'emploi du temps présente une répartition équilibrée des séances.

Evaluation de l'emploi du temps: L'analyse des résultats repose sur trois visualisations complémentaires, présentées progressivement.

— Visualisation 3D des solutions générées

Table 4.11 – Exemple d'un emploi du temps hebdomadaire

Jour	Créneau	Module	Type	Professeur	Salle	Groupe
Dimanche	10H 10H ...	Module 1 Module 7 ...	TP Cours ...	Prof 6 Prof 3 ...	Lab 5 Amphi 1 ...	G1 G1,G2,G3,G4,G5 ...
Lundi	10H 11H30 ...	Module 1 Module 1 ...	TP TD ...	Prof 5 Prof 4 ...	Lab 1 Salle 1 ...	G2 G1 ...
Mardi	10H 11H30 ...	Module 3 Module 2 ...	Cours TP ...	Prof 4 Prof 3 ...	Amphi 2 Lab 4 ...	G1,G2,G3,G4,G5 G3 ...
Mercredi	10H 10H ...	Module 3 Module 3 ...	TD TD ...	Prof 6 Prof 5 ...	Salle 2 Salle 4 ...	G1 G3 ...
Jeudi	10H 10H ...	Module 1 Module 3 ...	TD TP ...	Prof 3 Prof 8 ...	Salle 1 Lab 5 ...	G5 G1 ...

La **Figure 4.3** représente toutes les solutions générées dans un espace à trois dimensions :

— Axe **X** : taux de satisfaction des professeurs,

— Axe **Y** : taux de satisfaction des étudiants,

— Axe **Z** : taux d'utilisation des salles.

Chaque point bleu correspond à un emploi du temps généré par le système. Après application du filtre de dominance de Pareto, seuls les points rouges sont conservés comme solutions optimales. Parmi elles, la solution marquée par une étoile jaune est considérée comme la plus équilibrée selon le concept de **l'équilibre de Nash**.

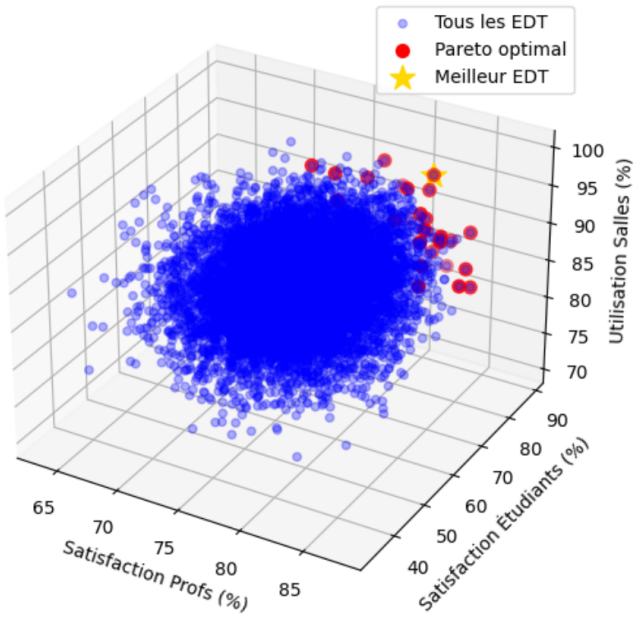

Figure 4.3 – Visualisation 3D des emplois du temps générés et filtrés (Pareto et Nash)

— Comparaison des solutions Pareto par critères

La **Figure 4.4** illustre la performance de plusieurs solutions Pareto sur quatre critères clés :

— **Professeurs** : niveau de satisfaction,
— **Étudiants** : niveau de satisfaction,
— **Salles** : taux d'occupation,
— **Continuité** : séances regroupées sans périodes creuses.

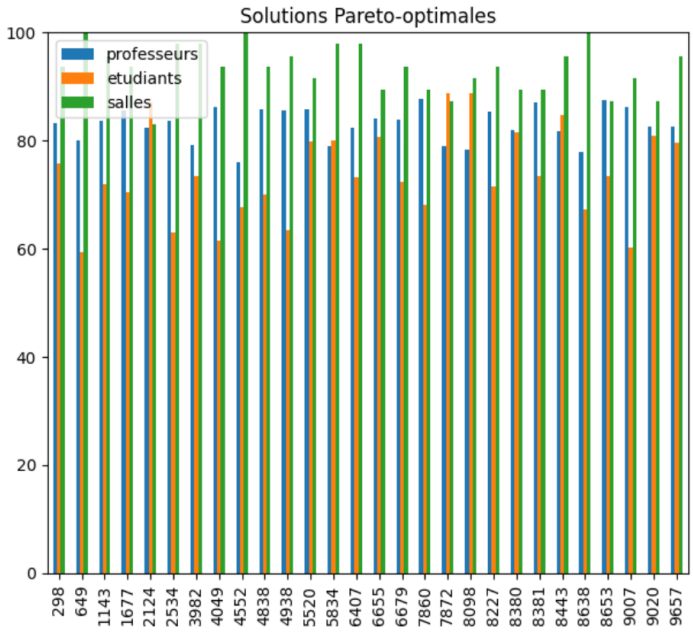

Figure 4.4 – Comparaison des scores de solutions Pareto selon les critères définis

— Comparaison chiffrée avec la moyenne des solutions

La **Table 4.12** présente une comparaison entre la meilleure solution (selon Nash) et la moyenne des autres solutions générées par l'algorithme. On observe que la solution retenue est supérieure à la moyenne pour l'ensemble des critères considérés.

Critère	Meilleur EDT	Moyenne
Professeurs	81.8%	76.7%
Étudiants	84.7%	61.0%
Salles	95.7%	86.9%

Table 4.12 – Comparaison entre le meilleur emploi du temps et la moyenne par critère

— Analyse des conflits

En complément de l'analyse multicritère, le système vérifie également l'absence de conflits liés aux ressources, tels que l'utilisation simultanée d'une même salle, d'un même professeur ou la présence d'un même groupe à plusieurs endroits au même moment.

```
======================================================================
 CONFLITS DÉTECTÉS DANS L'EMPLOI DU TEMPS
======================================================================
```

```
 CONFLITS DE SALLES (même salle au même moment):

Salle salle 4 le Mercredi à 08:30:
- 1 er année licence /section 1: module 1 (td) avec prof 2 pour les groupes g4
- 1 er année licence /section 1: module 3 (td) avec prof 5 pour les groupes g5


 CONFLITS DE PROFESSEURS (même prof au même moment):

Professeur prof 5 le Lundi à 10:00:
- 1 er année licence /section 1: module 1 en lab 1 avec les groupes g2
- 1 er année licence /section 1: module 1 en lab 4 avec les groupes g3

Professeur prof 3 le Jeudi à 11:30:
- 1 er année licence /section 1: module 2 en lab 5 avec les groupes g5
- 1 er année licence /section 1: module 4 en salle 2 avec les groupes g1

CONFLITS DE GROUPES (même groupe dans même promo/section au même moment):

• Groupe g3 (Promo: 1 er année licence , Section: section 1) le Dimanche à 08:30:
   - Module: module 1 (td)
     Professeur: prof 4
     Salle: salle 2
   - Module: module 4 (cours)
     Professeur: prof 2
     Salle: amphi 1

• Groupe g1 (Promo: 1 er année licence , Section: section 1) le Dimanche à 10:00:
   - Module: module 1 (tp)
     Professeur: prof 6
     Salle: lab 5
   - Module: module 7 (cours)
     Professeur: prof 3
     Salle: amphi 1

• Groupe g4 (Promo: 1 er année licence , Section: section 1) le Mardi à 10:00:
   - Module: module 3 (cours)
     Professeur: prof 4
     Salle: amphi 2
   - Module: module 3 (tp)
     Professeur: prof 1
```

```
        Salle: lab 5

• Groupe g3 (Promo: 1 er année licence , Section: section 1) le Mercredi à 10:00:
  - Module: module 3 (td)
    Professeur: prof 5
    Salle: salle 4
  - Module: module 5 (td)
    Professeur: prof 7
    Salle: salle 1

• Groupe g2 (Promo: 1 er année licence , Section: section 1) le Mardi à 08:30:
  - Module: module 3 (tp)
    Professeur: prof 3
    Salle: lab 3
  - Module: module 7 (tp)
    Professeur: prof 6
    Salle: lab 2
======================================================================
```

Ces résultats confirment la pertinence de la solution retenue par le système, qui parvient à concilier les intérêts des différents acteurs impliqués dans l'organisation des emplois du temps.

4.5 Visualisation des résultats dans l'application

Dans cette dernière partie, nous présentons l'interface utilisateur de l'application développée. Elle permet d'interagir avec le système de manière intuitive, via un chatbot intelligent et des pages dédiées à la consultation des emplois du temps générés. Ci-dessous, nous montrons quelques captures d'écran illustrant les fonctionnalités principales de l'application.

— **Authentification des utilisateurs:** L'application comprend un système d'authentification sécurisé permettant d'identifier chaque utilisateur selon son rôle (étudiant, enseignant ou administrateur). Les figures 4.5, 4.6 et 4.7 illustrent respectivement la page d'authentification, la sélection du rôle de l'utilisateur et la création d'un nouveau compte.

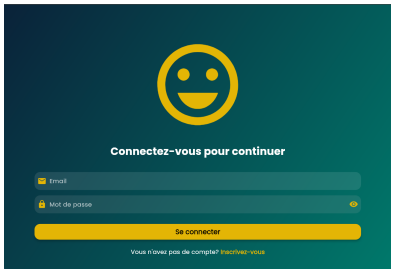

Figure 4.5 – La page d'authentification

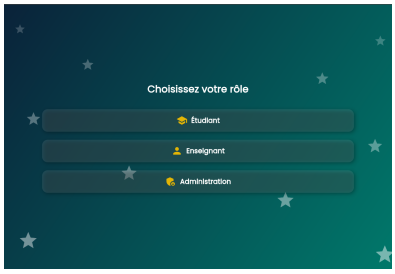

Figure 4.6 – La page de sélection du rôle

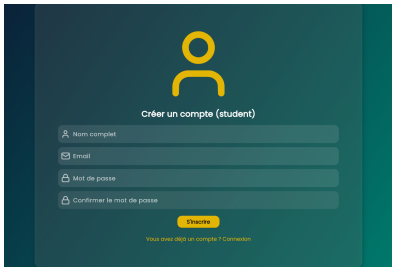

Figure 4.7 – La page de création de compte

— **Interface Administrateur** L'administrateur a accès à un tableau de bord complet qui lui permet de gérer tous les éléments du système, comme les utilisateurs, les groupes, les modules, les enseignants et les emplois du temps. C'est la partie la plus riche en fonctionnalités, car elle lui permet de configurer entièrement les données de l'établissement. La figure 4.8 présente l'interface du tableau de bord de l'administrateur, à partir de laquelle il peut gérer l'ensemble des entités du système.

Figure 4.8 – Le tableau de bord de l'administrateur

Création d'un emploi du temps:

L'administrateur peut générer automatiquement un emploi du temps en suivant une série d'étapes structurées. Il commence par définir les salles disponibles, avec la possibilité de modifier leur nom et de sélectionner leur type (cours, TD, TP). La figure 4.9 illustre cette phase initiale de configuration.

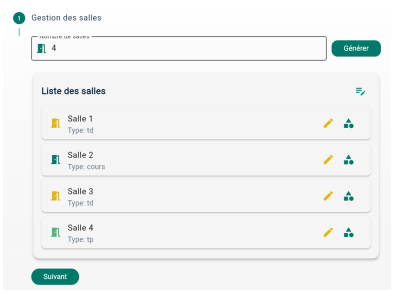

Figure 4.9 – Gestion des salles disponibles

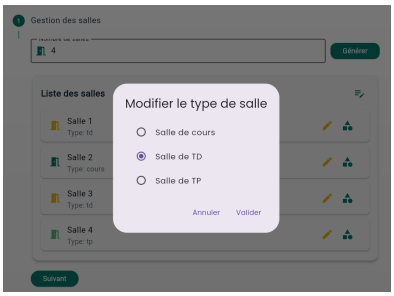

Figure 4.10 – Modification du type de salle

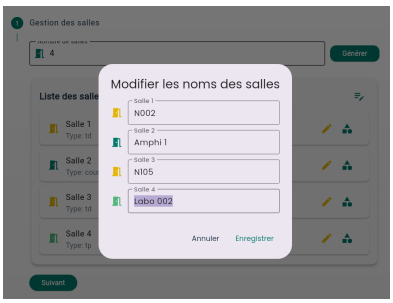

Figure 4.11 – Modification du nom d'une salle

De la même manière, l'administrateur procède à la saisie des enseignants, à la

création des promotions et de leurs spécialités, à la définition des sections et des groupes correspondants, ainsi qu'à l'ajout des modules. Enfin, il assigne les enseignants aux modules selon leur spécialité ou leur rôle pédagogique.
Une fois l'emploi du temps généré, l'administrateur peut le visualiser directement dans l'interface. La figure 4.12 présente l'interface de visualisation de l'emploi du temps généré, telle qu'elle apparaît dans l'application côté administrateur.

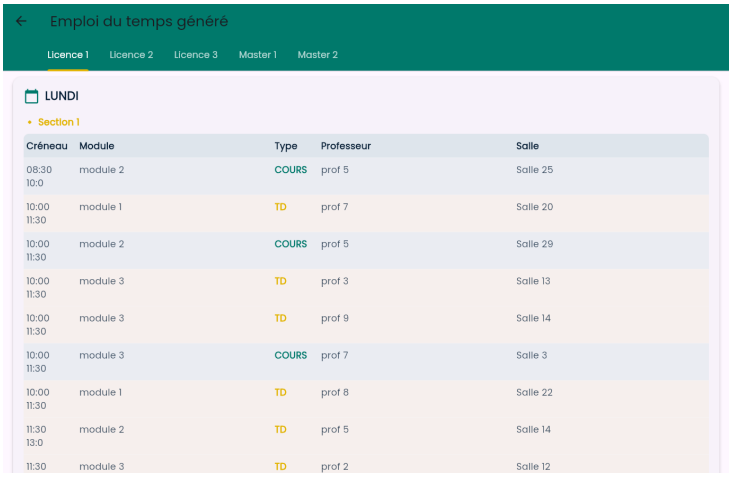

Figure 4.12 – Visualisation de l'emploi du temps généré dans l'application

L'application propose également un chatbot destiné à l'administrateur, qui permet de poser des questions pour consulter ou créer un emploi du temps. La figure 4.13 illustre la page de sélection qui permet à l'administrateur d'accéder au chatbot pour consulter ou générer un emploi du temps via un dialogue interactif.

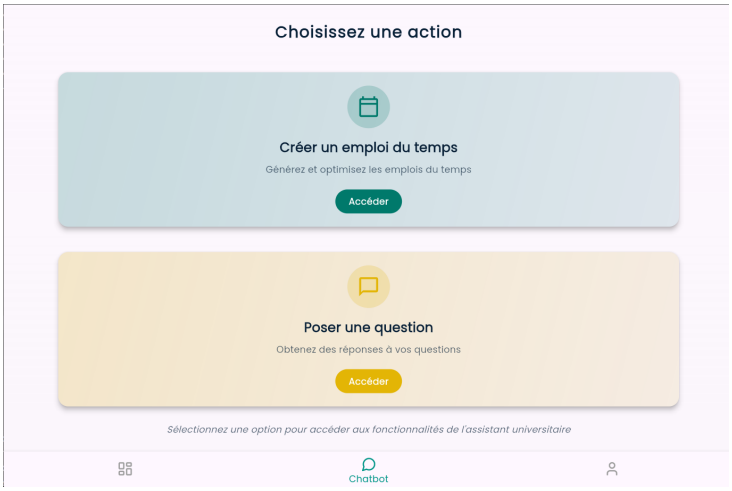

Figure 4.13 – La page de sélection

Pour l'instant, ce chatbot n'affiche pas de réponses formulées en langage naturel. Il transforme directement la question en une requête SQL, exécute cette requête sur la base de données, puis affiche le résultat extrait. Toutefois, nous avons commencé à entraîner une version plus avancée, capable de répondre en langage naturel à l'avenir. Les figures 4.14 et 4.15 illustrent l'interface actuelle du chatbot, qui convertit les questions en

requêtes SQL exécutées directement sur la base de données, sans génération de réponse en langage naturel.

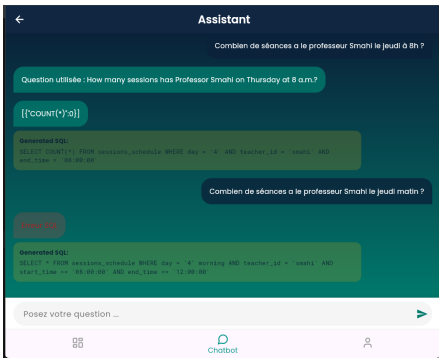

Figure 4.14 – Exemple de question mal interprétée

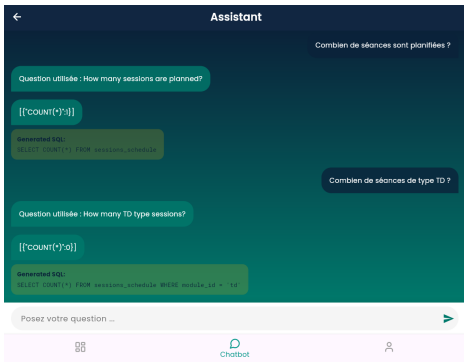

Figure 4.15 – Requête SQL générée automatiquement

Dans le cas de la création d'un emploi du temps, le chatbot agit comme un assistant conversationnel. Il pose des questions successives à l'utilisateur.
L'utilisateur répond étape par étape, sans passer par les formulaires classiques. Une fois toutes les informations saisies, le système génère automatiquement l'emploi du temps La figure 4.16 montre le fonctionnement du chatbot en mode conversationnel, où l'utilisateur renseigne progressivement les informations nécessaires à la génération automatique de l'emploi du temps, sans passer par des formulaires classiques.

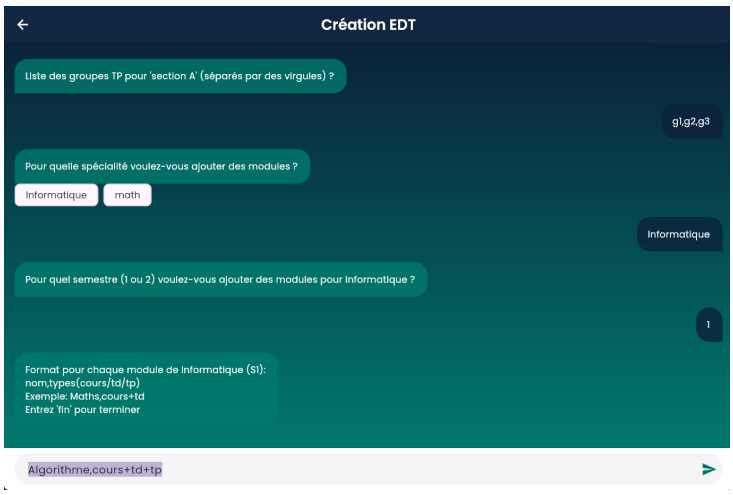

Figure 4.16 – Création d'emploi du temps

L'administrateur commence par créer les promotions, qui correspondent aux différentes années de formation. Chaque promotion peut être associée à une ou plusieurs spécialités. La structure hiérarchique est donc construite de manière progressive :

Promotion –> Spécialités –> Sections –> Groupes La figure 4.17 illustre l'interface de gestion des promotions, point de départ de la structuration hiérarchique des formations incluant spécialités, sections et groupes.

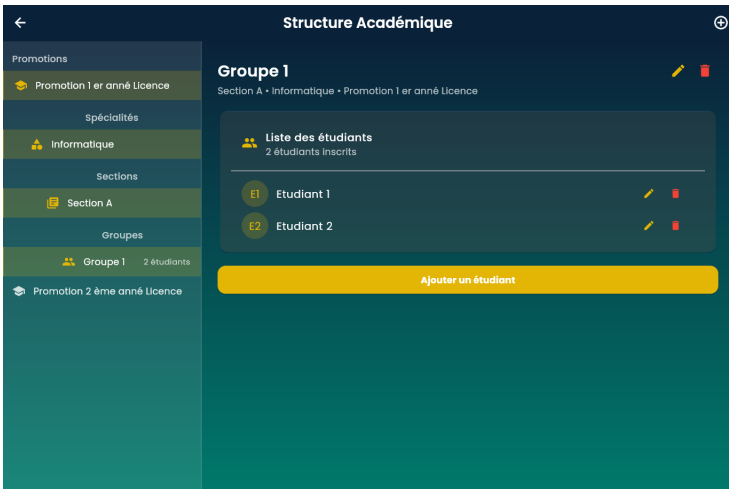

Figure 4.17 – La gestion des promotions

— **Interface Étudiant**

La page d'accueil de l'étudiant affiche la prochaine séance prévue, ou un message personnalisé si aucune séance n'est programmée. Elle met également en avant les événements à venir, tels que les conférences ou les formations. Une seconde page permet à l'étudiant de consulter l'ensemble de son emploi du temps, organisé par jour et par créneau horaire Les figures 4.18 et 4.19 présentent l'interface dédiée à l'étudiant, incluant la page d'accueil avec les informations essentielles (prochaine séance, événements) ainsi que la page de consultation de l'emploi du temps complet par jour et créneau horaire.

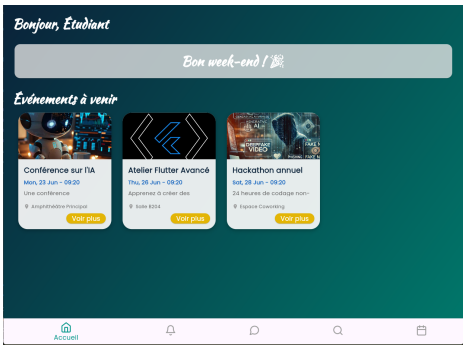

Figure 4.18 – Accueil étudiant

Figure 4.19 – Emploi du temps

— **Interface Enseignant**

L'enseignant peut consulter son emploi du temps de la semaine avec tous les détails : modules, groupes, salles et types de séances. Cela lui permet de visualiser facilement ses cours à venir. Les figures 4.20 et 4.21 illustrent l'interface dédiée à l'enseignant, lui permettant de consulter l'ensemble de son emploi du temps hebdomadaire ainsi que les détails de chaque séance, tels que le module, le groupe, la salle et le type de séance.

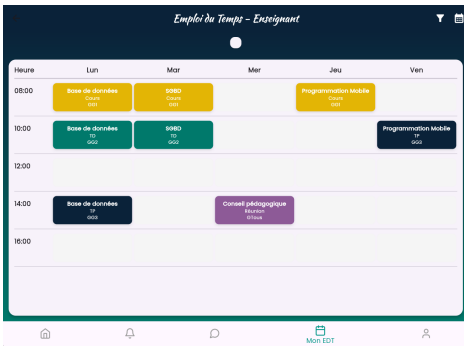

Figure 4.20 – visualisation d'emploi du tempsEnseignant

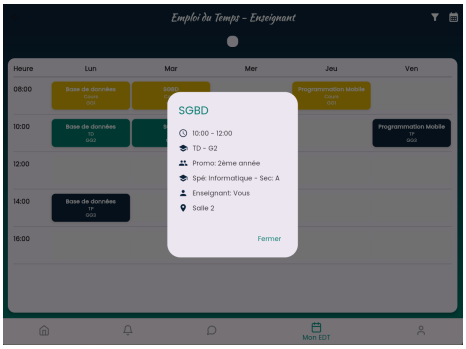

Figure 4.21 – Détaills de séance

4.6 Conclusion

Ce chapitre a validé les principaux composants du système proposé. Le modèle NL2SQL a obtenu de bons résultats sur les jeux de données testés, et l'optimisation des emplois du temps a permis de générer des plannings équilibrés respectant les contraintes.Ces résultats forment une base prometteuse, avec encore des améliorations possibles, notamment pour affiner la génération des requêtes et améliorer l'optimisation.

5

Conclusion

Ce travail s'est inscrit dans le cadre de la recherche de solutions intelligentes pour améliorer la gestion des emplois du temps universitaires, une tâche réputée complexe et chronophage. En combinant un chatbot capable d'interagir en langage naturel avec un système d'optimisation multicritère fondé sur des principes issus de la théorie des jeux, nous avons proposé une approche innovante et adaptable, centrée sur les besoins réels des utilisateurs. Le système développé permet non seulement de simplifier l'accès aux données via des requêtes automatisées, mais aussi de générer des emplois du temps équilibrés selon plusieurs critères pertinents.

Bien que le système ne prétende pas offrir une solution parfaite à tous les cas de figure, les résultats obtenus montrent qu'il peut constituer une véritable aide à la décision dans un contexte académique réel. Il représente une avancée significative vers une planification plus intelligente, plus souple et plus centrée sur l'utilisateur.

À l'avenir, plusieurs améliorations peuvent être envisagées : rendre les réponses du chatbot plus naturelles et explicatives, permettre la modification manuelle des résultats générés, ou encore intégrer des techniques d'apprentissage automatique pour affiner les préférences selon l'historique des utilisateurs. L'intégration dans un environnement cloud permettrait également une meilleure accessibilité et une évolutivité à plus grande échelle.

En somme, cette première version jette les bases d'un outil intelligent et prometteur, qui pourra être enrichi pour répondre encore mieux aux défis quotidiens de la gestion académique.

Bibliography

[2] Ion Androutsopoulos, Graeme D Ritchie **and** Peter Thanisch. "Natural language interfaces to databases–An introduction". **in***Natural Language Engineering*: 1.1 (1995), **pages** 29–81. DOI: 10.1017/S135132490000005X.

[4] Pratima Bansal **andothers**. "Temporal work: The strategic organization of time". **in***Strategic Organization*: 20.1 (2022), **pages** 6–19.

[5] Abeer Bashab **andothers**. "Optimization techniques in university timetabling problem: Constraints, methodologies, benchmarks, and open issues". **in***Computers, Materials Continua*: 74.1 (2023), **pages** 123–145.

[6] Christian Bertrand **and** Stéphane Gruber. *La gestion du temps dans les organisations éducatives*. Bruxelles: De Boeck Supérieur, 2016.

[8] Tom B. Brown **andothers**. "Language Models are Few-Shot Learners". **in***Advances in Neural Information Processing Systems*: 33 (2020), **pages** 1877–1901.

[9] Edmund K Burke **and** Sanja Petrovic. "Recent research directions in automated timetabling". **in***European Journal of Operational Research*: 140.2 (2002), **pages** 266–280. DOI: 10.1016/S0377-2217(01)00238-4.

[10] Erik Cambria **and** Bebo White. "Jumping NLP Curves: A Review of Natural Language Processing Research". **in***IEEE Computational Intelligence Magazine*: 9.2 (2014), **pages** 48–57. DOI: 10.1109/MCI.2014.2307227.

[11] Michael W Carter **and** Gilbert Laporte. "Recent developments in practical examination timetabling". **in***Selected papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling*: 1996, **pages** 3–21.

[12] Sara Ceschia, Luca Di Gaspero **and** Andrea Schaerf. "Educational timetabling: Problems, benchmarks, and state-of-the-art results". **in***European Journal of Operational Research*: 301.2 (2022), **pages** 1–15.

[14] Tim B Cooper **and** Jeffrey H Kingston. "The complexity of timetable construction problems". **in***Practice and Theory of Automated Timetabling*: Springer. 1996, **pages** 281–295.

[17] Jacob Devlin **andothers**. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". **in***arXiv preprint arXiv:1810.04805*: (2018).

[21] Alfonso Gerevini, Alessandro Saetti **and** Ivan Serina. "An approach to temporal planning and scheduling in domains with predictable exogenous events". **in***Journal of Artificial Intelligence Research*: 25 (2006), **pages** 187–231.

[22] Ian Goodfellow, Yoshua Bengio **and** Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN: 9780262035613. URL: http://www.deeplearningbook.org.

[25] Jiaqi Guo **andothers**. "IRNet: A General Purpose Approach for Immutable Input Validation". **in***arXiv preprint arXiv:1906.05378*: (2019).

[26] Achini Herath **and** Dawn Wilkins. "A comparative study on solving university timetabling problems with emphasis on genetic algorithms". **in***Issues in Information Systems*: 25.4 (2024), **pages** 392–408.

[27] Jonathan Herzig **andothers**. "TaPas: Weakly supervised table parsing via pre-training". **in***ACL*: 2020.

[31] Daniel Jurafsky **and** James H. Martin. *Speech and Language Processing*. 3rd. https://web.stanford.edu/~jurafsky/slp3/. Pearson, 2021.

[32] Daniel Jurafsky **and** James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 3rd. Draft edition online. Pearson, 2025. URL: https://web.stanford.edu/~jurafsky/slp3/.

[33] Amit Kumar **and** David M. Shanker. "A Constraint-based Approach to University Timetabling". **in***International Conference on Artificial Intelligence*: 2009.

[34] Roger Lewis. "A survey of metaheuristic-based techniques for university timetabling problems". **in***European Journal of Operational Research*: 179.1 (2008), **pages** 1–15. DOI: 10.1016/j.ejor.2006.04.003.

[35] Xi Victoria Lin **and** Dragomir Radev. "Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing". **in***Findings of EMNLP*: 2020.

[36] Xi Victoria Lin, Richard Socher **and** Caiming Xiong. "BRIDGE: Text-to-SQL with Intermediate Representations". **in***arXiv preprint arXiv:2010.02840*: (2020).

[38] Ilya Loshchilov **and** Frank Hutter. "Decoupled Weight Decay Regularization". **in***arXiv preprint arXiv:1711.05101*: (2017).

[39] Zhipeng Lü **and** Jin-Kao Hao. "Adaptive Tabu Search for course timetabling". **in***European Journal of Operational Research*: 200.1 (2010), **pages** 235–244.

[40] Barry McCollum. "A perspective on bridging the gap between theory and practice in university timetabling". **in***Practice and Theory of Automated Timetabling VI*: Springer, 2007, **pages** 3–23.

[41] Henry Mintzberg. *The Rise and Fall of Strategic Planning*. New York: Free Press, 1994.

[42] T. Muller, T. Schiex **and** S. De Givry. "Soft constraints: A survey". **in***Principles and Practice of Constraint Programming*: Springer, 2003, **pages** 1–41.

[43] Christine Musselin. *Le travail universitaire : Une sociologie des universités au travail*. Paris: Presses de Sciences Po, 2008.

[44] Roger B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1991.

[46] John F. Nash. "Equilibrium Points in N-Person Games". **in***Proceedings of the National Academy of Sciences*: 36.1 (1950), **pages** 48–49.

[47] John F. Nash. "Non-Cooperative Games". **in***Annals of Mathematics*: 54.2 (1951), **pages** 286–295.

[48] Sourd Fra Néron Emmanuel Baptiste Philippe. *Modèles et Algorithmes en Ordonnancement*. Ellipses, 2004.

[49] Martin J. Osborne **and** Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[50] Kishore Papineni **andothers**. "BLEU: a method for automatic evaluation of machine translation". **in***Proceedings of the 40th annual meeting of the Association for Computational Linguistics*: ACL. 2002, **pages** 311–318.

[51] Vilfredo Pareto. *Cours d'économie politique*. Republié en 1964 par Librairie Droz. F. Rouge, Lausanne, 1896.

[52] Vilfredo Pareto. *Manuale di Economia Politica*. Società editrice libraria, 1906.

[54] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 2016.

[56] Rong Qu **and** Edmund K. Burke. "An Examination of Future Directions in Automated Timetabling". **in***Annals of Operations Research*: 153.1 (2007), **pages** 41–53. DOI: 10.1007/s10479-006-0133-0.

[57] Alec Radford **andothers**. "Improving Language Understanding by Generative Pre-Training". **in***OpenAI preprint*: (2018).

[58] Alec Radford **andothers**. "Language models are unsupervised multitask learners". **in***OpenAI Blog*: (2019). https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[59] Colin Raffel **andothers**. "Exploring the limits of transfer learning with a unified text-to-text transformer". **in***Journal of Machine Learning Research*: 21.140 (2020), **pages** 1–67.

[60] Pranav Rajpurkar **andothers**. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". **in***Proceedings of EMNLP*: 2016, **pages** 2383–2392.

[61] Ohad Rubin **and** Jonathan Berant. "SmBoP: Semi-autoregressive Bottom-up Semantic Parsing". **in***arXiv preprint arXiv:2010.12412*: (2021).

[62] Stuart Russell **and** Peter Norvig. *Artificial Intelligence: A Modern Approach*. 4th. Pearson, 2020. ISBN: 9780134610993.

[63] Andrea Schaerf. "A survey of automated timetabling". **in***Artificial Intelligence Review*: 13.2 (1999), **pages** 87–127.

[64] Thomas Scholak, Daniel Andor **and** Jonathan Berant. "PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models". **in***arXiv preprint arXiv:2109.05052*: (2021).

[66] Yoav Shoham **and** Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[69] Ian Tenney, Dipanjan Das **and** Ellie Pavlick. "BERT Rediscovers the Classical NLP Pipeline". **in***Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*: Association for Computational Linguistics, 2019, **pages** 4593–4601.

[71] Ashish Vaswani **andothers**. "Attention is All You Need". **in***Advances in neural information processing systems*: 2017, **pages** 5998–6008.

[73] Bailin Wang **andothers**. "RAT-SQL: Relation-Aware Schema Encoding for Text-to-SQL". **in***arXiv preprint arXiv:1911.04942*: (2020).

[74] Pengcheng Yin **and** Graham Neubig. "TaBERT: Pretraining for joint understanding of textual and tabular data". **in***ACL*: 2020.

[75] Tom Young **andothers**. "Recent Trends in Deep Learning Based Natural Language Processing". **in***IEEE Computational Intelligence Magazine*: 13.3 (2018), **pages** 55–75.

[76] Tao Yu **andothers**. "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task". **in***EMNLP*: 2018.

[77] Tao Yu **andothers**. "TypeSQL: Knowledge-Based Type-Aware Neural Text-to-SQL Generation". **in***arXiv preprint arXiv:1804.09769*: (2018).

[78] Victor Zhong, Caiming Xiong **and** Richard Socher. "Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning". **in***Proceedings of ACL*: 2017.

Résumé

Dans ce travail, nous avons proposé une approche innovante pour optimiser l'organisation des emplois du temps au sein du milieu académique. Cette optimisation a été réalisée grâce à l'application de l'intelligence artificielle, en particulier les techniques du traitement du langage naturel. Notre travail repose sur trois contributions principales. La première est la mise en place d'un chatbot intelligent capable de transformer automatiquement des questions formulées en langage naturel en requêtes SQL. L'idée est d'assurer une grande adaptabilité aux évolutions futures, minimisant ainsi de futurs interventions humaine directes. La deuxième contribution est l'optimisation automatique des emplois du temps universitaires. Pour cela, nous nous sommes appuyé sur la théorie des jeux afin de modéliser les préférences et les contraintes des différents acteurs impliqués. Cette approche permet de générer des emplois du temps plus équitables et cohérents en tenant compte des conflits potentiels et des besoins spécifiques de chacun. Enfin, la troisième contribution concerne le développement d'une interface conversationnelle simple et intuitive, permettant une interaction fluide entre les utilisateurs (étudiants, enseignants, administrateurs) et le système, même sans connaissances techniques avancées. Nous avons mené plusieurs expérimentations, principalement durant les phases de génération de requêtes SQL et d'optimisation des plannings, dans le but d évaluer l'éfficacité de notre approche. Les résultats obtenus confirment la pertinence de notre solution , tant sur le plan de la précision des requêtes produites que sur la qualité des emplois du temps générés.

Mots-clés: Intelligence artificielle, Traitement du langage naturel, Chatbot intelligent, Requêtes SQL, Optimisation des emplois du temps, Théorie des jeux

ملخص

يتناول هذا العمل مقاربة مُجَدِّدَة لتحسين إدارة الشؤون الأكاديمية باستخدام الذكاء الاصطناعي، في هذا العمل، اقترحنا منهجًا مبتكرًا يهدف إلى تحسين تنظيم الجداول الزمنية في الوسط الأكاديمي. وقد تم تحقيق هذا التحسين من خلال توظيف تقنيات الذكاء الاصطناعي، وخاصة تقنيات معالجة اللغات الطبيعية. يرتكز عملنا على ثلاث مساهمات رئيسية. أولًا، تطوير روبوت محادثة ذكي قادر على تحويل الأسئلة المطروحة باللغة الطبيعية تلقائيًا إلى استعلامات SQL. وتتمثل الفكرة في ضمان قدرة كبيرة على التكيّف مع التطورات المستقبلية، مما يحد من الحاجة إلى التدخلات البشرية المباشرة لاحقًا. ثانيًا، تحسين الجداول الزمنية الجامعية بشكل آلي. ولتحقيق ذلك، اعتمدنا على نظرية الألعاب لنمذجة تفضيلات وقيود مختلف الأطراف المعنية. وتسمح هذه المقاربة بإنتاج جداول زمنية أكثر عدلًا واتساقًا، تأخذ في الحسبان التعارضات المحتملة واحتياجات كل طرف. ثالثًا، تطوير واجهة محادثة بسيطة وبديهية، تتيح تفاعلًا سلسًا بين المستخدمين)سواء طلابًا، أو أساتذة، أو إداريين(والنظام، حتى من دون امتلاك خلفية تقنية متقدمة. وقد أجرينا العديد من التجارب، خاصة خلال مراحل توليد الاستعلامات SQL وتحسين الجداول الزمنية، بهدف تقييم فعالية منهجيتنا. وأكدت النتائج المحققة مدى أهمية الحل المقترح، سواء من حيث دقة الاستعلامات المنتجة أو من حيث جودة الجداول الزمنية التي تم إنشاؤها

الكلمات الدالة :الذكاء الاصطناعي، معالجة اللغة الطبيعية، روبوت محادثة ذكي، استعلامات SQL، تحسين الجداول الزمنية، نظرية الألعاب، واجهة محادثة.

Abstract

In this work, we proposed an innovative approach to optimize the organization of academic timetables. This optimization was achieved through the application of artificial intelligence, particularly natural language processing techniques. Our work is based on three main contributions. The first is the implementation of an intelligent chatbot capable of automatically transforming questions formulated in natural language into SQL queries. The idea is to ensure high adaptability to future developments, thereby minimizing the need for direct human intervention. The second contribution is the automatic optimization of university timetables. To achieve this, we relied on game theory to model the preferences and constraints of the various stakeholders involved. This approach enables the generation of fairer and more coherent timetables by considering potential conflicts and the specific needs of each party. Finally, the third contribution concerns the development of a simple and intuitive conversational interface, allowing smooth interaction between users (students, teachers, administrators) and the system, even without advanced technical knowledge. We conducted several experiments, mainly during the SQL query generation and timetable optimization phases, to evaluate the effectiveness of our approach. The results obtained confirm the relevance of our solution, both in terms of the accuracy of the generated queries and the quality of the produced timetables.

Keywords: Artificial Intelligence, Natural Language Processing, Intelligent Chatbot, SQL Queries, Timetable Optimization, Game Theory