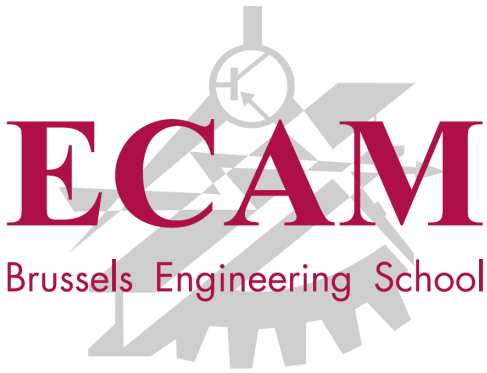


École Centrale des Arts et Métiers

Mise en place d'un tutoriel intelligent et d'outils d'aide à la création de questions d'examens par des algorithmes de machine learning

Julien Kessels

En vue de l'obtention du diplôme de Master en Sciences de l'Ingénieur Industriel orientation Informatique

Promoteur: Dr Ir. Sébastien Combéfis

Superviseur: M. Ing. Rémy Taymans

Année académique 2018-2019

Remerciements

Je tiens à remercier vivement toutes les personnes qui ont contribué au succès de mon travail de fin d'études et qui m'ont aidé lors de la rédaction de ce mémoire.

Tout d'abord, j'adresse mes remerciements à mon professeur et promoteur, le Dr Ir. Sébastien Combéfis qui m'a beaucoup aidé dans ma recherche de stage et m'a permis de rencontrer EDITx. Merci en particulier pour son écoute et ses nombreux conseils lors de la réalisation du projet et la rédaction de mon rapport.

Je tiens aussi à remercier vivement mon superviseur, M. Ing. Rémy Taymans, qui m'a donné de précieux conseils et qui a toujours été disponible quand j'avais des questions.

Je remercie également toute l'équipe d'EDITx pour son accueil, son esprit d'équipe et la bonne ambiance, qui m'ont accompagné durant mon stage et ce travail de fin d'études.

Enfin, je tiens à remercier toutes les autres personnes qui m'ont conseillé plus ponctuellement : mes parents, ma famille et mes camarades de promotion.

L'entreprise présentée dans ce rapport :

EDITx

Rue des Francs, 79

1040 Etterbeek, Belgium

Tel : *+32 2 240 57 80*

Email : feedback@editx.eu

Résumé

Dans le cadre de mon travail de fin d'études réalisé chez EDITx, société organisant des concours informatiques, j'ai dans un premier temps développé une série d'**outils d'aide au processus de création de questions** pour ses concours. Grâce au framework de machine learning Tensorflow et de la librairie Python Scikit-learn, j'ai utilisé des méthodes de clustering et divers autres algorithmes pour automatiser l'évaluation du langage de programmation de la question, l'évaluation de sa difficulté ainsi que la détection de questions identiques ou similaires.

Ensuite, j'ai mis en place un système de **tutoriel intelligent** en exploitant les résultats des utilisateurs aux concours, c'est-à-dire leurs réponses aux questions. Grâce à l'Item Response Theory (IRT) le tutoriel proposera une suite de questions personnalisées pour chaque utilisateur selon son niveau, évalué grâce aux questions de la base de données d'EDITx et les réponses des autres utilisateurs, et selon le type de langage dans lequel l'utilisateur aimerait progresser.

Les outils proposés sont implémentés sous forme d'un prototype de plateforme d'analyse et ont été évalués sur base de données collectées par EDITx afin de mesurer leur performance.

Mots clés: IA, machine learning, data mining, IT, smart tutorial, Item Response Theory, Duplicate detection

Cahier des charges relatif au travail de fin d'études de

Kessels Julien, inscrit en 2ème Master, orientation informatique

- Année académique : 2018-2019
- Titre provisoire :

  Réalisation d'un tutoriel intelligent d'amélioration de compétences IT à l'aide de techniques de machine learning

- Objectifs à atteindre :

  - Design et implémentation d'un outil qui proposera automatiquement un tutoriel à l'utilisateur comportant des questions choisies de manière autonome pour améliorer une compétence IT spécifiée par l'utilisateur.

  L'outil comportera plusieurs parties:

  - Outil d'aide à la rédaction de questions (Identification du niveau de difficulté et des compétences ciblées par une question)

  - Identification de questions proches par rapport aux compétences entraînées

  - Le tutoriel en tant que tel accessible aux utilisateurs

- Principales étapes :

  - Data analytics des données existances
  - Etablissement d'un modèle machinle learning
  - Entraînement et évaluation du modèle
  - Documentation des outils produits

Fait en trois exemplaires à Etterbeek, le 16 novembre 2018

L'étudiant	*Le tuteur*	*Le promoteur*
Julien Kessels	Rémy Taymans	Sébastien Combéfis
	GEI	EDITx
Signature :	Signature :	Signature :

Table des matières

1 Introduction

1.1 Présentation de l'entreprise

Fondée en 2017, la société bruxelloise EDITx organise des concours informatiques à travers l'Europe. Via la plateforme en ligne d'EDITx[1], des utilisateurs peuvent participer aux concours proposés.

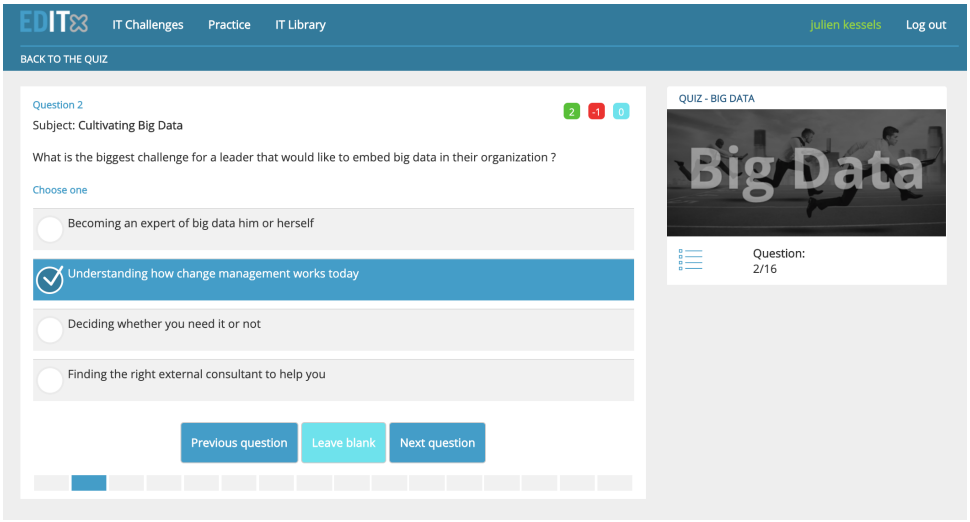


FIGURE 1 – Plateforme de concours EDITx

Les concours se déroulent en deux étapes :
La première phase, en ligne, consiste en un questionnaire d'une vingtaine de questions à choix multiples, spécifiques à un domaine de l'informatique. Si la personne se retrouve dans les top challengers du premier tour, elle est invitée à participer à une finale organisée chez le sponsor du concours, habituellement dans leur quartier général. Les challenges organisés par EDITx sont majoritairement sponsorisés par des entreprises, mais il arrive aussi que des écoles, comme par exemple la Vlerick School, souhaitent organiser un concours.

Outre la visibilité en terme de publicité, ces entreprises montrent leur intérêt pour une certaine technologie comme par exemple Angular, pour le Challenge sponsorisé par Google, ou C# pour celui organisé par Microsoft. L'intérêt pour les écoles est de se faire connaître mais également promouvoir un nouveau master, par exemple.

1. `www.editx.eu`

De plus, à la fin du concours, l'entreprise a accès aux résultats des participants ainsi qu'à leur données de contact, ce qui lui permet d'envoyer des offres de recrutement aux candidats qui lui semblent intéressants.

FIGURE 2 – Exemples de concours proposés sur la plateforme d'EDITx

1.2 Enjeux

Depuis peu, EDITx est confronté à un accroissement massif et rapide d'utilisateurs enregistrés sur la plateforme ainsi que du nombre de questions de concours soumises par les contributeurs. Les contributeurs, qui sont souvent des professeurs du monde entier, utilisent les mêmes textbooks ou sources d'inspiration pour certaines matières. Pour chaque concours, un jury est sélectionné pour choisir une série de questions pertinentes pour le concours, mais les utilisateurs risqueraient de souvent retomber sur le même genre de questions.
Ce succès entraîne de nouvelles problématiques que la société se doit d'adresser.

• Il est primordial d'**améliorer la gestion des questions des concours**. Un utilisateur peut, en effet, s'inscrire en tant qu'auteur ou contributeur sur la plateforme et soumettre ses propres questions. L'infrastructure et la gestion autour de la soumission de questions n'étaient pas fort développées et introduisaient des problèmes évidents : Doublons entre questions dans la base de données, mauvaise estimation du niveau de difficulté d'une question, erreurs d'encodage, ...
C'est d'autant plus difficile pour les personnes du jury qui ne prennent pas forcément le temps d'examiner la librairie des questions et ne savent pas non plus estimer le niveau de difficulté des questions.

• Pour maintenir et booster **l'intérêt de la plateforme**, celle-ci nécessite d'élargir son champ d'activités de sorte que les utilisateurs reviennent régulièrement visiter le site et participer aux concours.

• Il serait intéressant d'exploiter cette grande base de données de questions pour autre chose qu'uniquement les challenges. Il y a un réel potentiel de tenter de rentabiliser, valoriser, voire monnayer d'une façon ou d'une autre ces données.

1.3 Solutions proposées

Pour faire face à ces problématiques, nous avons mis en place dans le cadre de ce travail **un Dashboard d'aide à la création de questions de concours** pour les contributeurs. Ce dashboard aura pour but, à travers quelques outils d'analyse, d'aider l'auteur à créer de nouvelles questions à la fois complètes et intéressantes.

Dans une deuxième étape, nous avons conçu et implémenté une nouvelle activité pour le site d'EDITx : **Un tutoriel intelligent et adapté au niveau de compétences de l'utilisateur pour l'apprentissage d'un langage de programmation.** Ce tutoriel aura pour but d'offrir un nouvel outil sur le site d'EDITx et ainsi accroître l'intérêt de la plateforme et d'attirer encore plus d'utilisateurs. Le tutoriel pourrait également servir d'outil pour créer des formations à la carte pour les entreprises.

Ce mémoire débutera par une analyse de projets similaires déjà existants. Nous discuterons de l'architecture à mettre en place pour structurer de manière efficace le travail réalisé. Ensuite nous développerons les solutions proposées en expliquant les algorithmes et en illustrant les résultats. Nous finirons ce travail en passant en revue nos objectifs, nos solutions et propositions de pistes pour aller plus loin.

2 État de l'art

Dans ce chapitre, nous allons explorer les différentes techniques retrouvées dans la littérature qui pourraient être pertinentes pour aborder notre problème.
Ces techniques vont être brièvement mises en contexte, analysées et comparées pour ensuite déterminer si leur implémentation serait utile dans notre cas.

2.1 Dashboard Auteurs

Les outils que nous voulons apporter au Dashboard Auteurs sont divers. Nous aimerions pouvoir automatiser quelques processus dans l'encodage des questions. Le premier a pour but de trouver le langage de programmation de la nouvelle question. Le deuxième outil aidera l'auteur à déterminer la difficulté de la nouvelle question. Enfin, le dernier s'assurera que la question n'existe pas déjà telle quelle ou légèrement modifiée dans la base de données.

2.1.1 Classification des questions

La classification de phrases en différents groupes n'est pas quelque chose de nouveau. Ce challenge récurrent a déjà suscité et suscite toujours beaucoup de recherches.

En 2011, Chew, Nagano et Mikami ont développé un détecteur de langage[1] qui peut reconnaître la langue des textes de sites internet avec un taux de succès de 94%. Pour ce faire, ils utilisent un modèle amélioré de *n-gram*.

Clive Souter[2] et son équipe sont allés un peu plus loin dans leur réflexion. Après avoir testé d'autres caractéristiques comme la fréquence de différents mots, ils ont conclu que la solution optimale était le modèle *n-gram* à trois mots consécutifs, les *trigrams*.

Cependant, les deux articles diffèrent un peu par rapport à notre cas. Les *n-gram*s fonctionnent effectivement très bien pour la classification de langues parlées comme le français ou l'anglais, mais moins pour des langages de programmation comme Python ou C#.

En 2018, Rathima et Divakar ont mis en place un système[3] utilisant la technique de machine learning K-Means pour extraire les mots importants de leurs textes et former des clusters. Cette technique s'intéresse plus à la sémantique qu'à la structure de la phrase.

Cela s'avère être très intéressant dans notre cas. Si nous arrivons à déterminer les mots-clés qui reviennent souvent dans les différents langages de programmation, nous pourrons les regrouper en clusters. Par exemple, dans le langage de programmation Java, il y a deux mots-clés qui apparaissent fréquemment : "System.out.println()" et "HashMap". K-Means saura précisément faire le lien entre les deux et les mettre dans le même cluster. Nous utiliserons la méthode de K-Means pour notre set de données en espérant qu'elle pourra nous aider à différencier les différents langages de programmation.

2.1.2 Estimation de la difficulté des questions

Estimer la difficulté d'une question est un processus très complexe qui nécessite beaucoup d'informations, a priori. Le souci est que la difficulté n'est pas une caractéristique objective et dépend fortement du contexte et de la personne à qui elle est posée. Une question peut être facile pour une personne mais compliquée pour une autre. Un concours est typiquement constitué de 10 questions faciles, 6 moyennes et 4 difficiles. L'estimation correcte du niveau de difficulté est donc primordiale pour le jury des concours.

Une équipe du Harbin Institute of Technology a tenté de s'attaquer à ce problème en 2014[4]. Leur méthode pour estimer la difficulté de questions est appelée *Regularized Competition Model (RCM)* qui combine des comparaisons question-utilisateur et l'analyse sémantique des questions. Pour estimer la difficulté de nouvelles questions, ils utilisent la technique de K-Nearest Neighbors (KNN) en tirant parti de plusieurs caractéristiques de la question (taille, contenu, ..). Par la technique de KNN, ils arrivent en effet à obtenir de bons résultats sur les deux sets de données. L'algorithme KNN présente dans notre cas malheureusement un inconvénient majeur : il n'apprend rien du set d'entraînement. À chaque fois que nous voulons classifier une nouvelle question selon sa difficulté, l'algorithme doit recalculer les distances (ou les neighbors) avec les nouvelles données (test set + nouvelle question). Pour cette raison, on dit aussi souvent que KNN est un *"Lazy Learner"*. Cela est problématique, car le serveur n'a pas les ressources pour refaire l'entraînement à chaque nouvelle question.

L'estimation de la difficulté de questions est finalement souvent laissée à l'auteur. Par des méthodes de probabilités et statistiques, il est possible de mettre en place des algorithmes pour aider l'auteur à mesurer le niveau de compétence de son public. Il pourra ensuite adapter sont estimation de niveau de difficulté de ses questions par rapport à cela.

2.1.3 Détection de questions doublons

Il est évident qu'à force de rajouter de nouvelles questions, des doublons vont apparaître dans la base de données de la plateforme. Cela remplirait la DB pour rien et l'appauvrirait au point à rendre le travail du jury à sélectionner les bonnes questions pour un concours plus compliqué.

En 2017, Quora, un site question-réponse qui compte plus de 38 millions[2] de questions, a lancé un challenge à ses utilisateurs :
Une récompense de 12 500€[3] pour la personne qui trouve la meilleure solution pour détecter les doublons dans leur base de données.

Les gagnants du concours ont mis en place un système très sophistiqué et complexe[5]. Après avoir vectorisé et extrait des caractéristiques des phrases, ils ont entraîné un modèle de réseaux neuronaux par fréquence de termes. Par des algorithmes de classification comme K-Nearest Neighbors ou Support Vector Machine, ils arrivaient à détecter des doublons avec une précision de 80%.

Dans un autre cas d'utilisation, B. Martins utilise des techniques de calcul de similitudes, correspondances sémantiques ainsi que Random Forest pour détecter les doublons. D'après son article[6], cela lui a permis d'obtenir une précision de 97.45%.

Dans le top 10 des meilleurs participants du concours de Quora, un utilisateur a subdivisé sa méthode de détection de doublons en plusieurs algorithmes indépendants. En plus de la distance de Manhattan ou Euclidienne, il utilise également la similarité cosinus pour détecter deux questions similaires.

Pour notre cas, nous nous focaliserons également sur des algorithmes indépendants et plus petits. Notre set de données est malheureusement trop petit pour pouvoir appliquer des algorithmes de machine learning aussi sophistiqués que le Random Forest.

2. https://www.quora.com/How-many-questions-have-been-asked-on-Quora-1
3. https://www.kaggle.com/c/quora-question-pairs/overview/prizes

2.2 Tutoriel intelligent

La conception d'un tutoriel intelligent dans le monde académique est un sujet que l'on retrouve souvent. Il n'est pourtant pas présenté sous le nom de *Tutoriel intelligent* mais plutôt d' *Évaluation intelligente*. Le premier a pour but d'**améliorer** les connaissances d'un utilisateur, le deuxième vise à **évaluer** ses connaissances. Dans les deux cas le système est basé sur le lien entre question et utilisateur, ou plus spécifiquement, le lien entre difficulté d'une question et niveau de compétence de l'utilisateur.

Sherlock[7] est un système de génération semi-automatique de tests. Créé par Dong Liu and Chenghua Lin en 2015, Sherlock utilise des technologies de machine learning et de compréhension sémantique pour générer ses tests. D'ailleurs, les deux auteurs mettent en avant la capacité de leur système à contrôler la difficulté du test. Pour évaluer la difficulté d'une question, ils utilisent la Linked Data Semantic Distance (LDSD). Cet algorithme se base sur le principe de similitude entre phrases et conviendrait plus précisément pour la première partie du mémoire. L'évaluation du niveau de difficulté reste très vague dans leur cas ("Easy", "Medium" et "Difficult"). Pour notre tutoriel, nous avons besoin d'une valeur plus précise pour d'avantage différencier les questions et leur difficulté.

La difficulté d'une question dépend fondamentalement du niveau de compétence d'une personne. Beaucoup dénoncent en effet la méthode classique d'évaluation et cherchent d'autres moyens plus efficaces et plus corrects d'évaluer le niveau d'une personne. L'approche typique pour mesurer le niveau de qualification d'une personne est de développer un test contenant une série de questions (items). Chacune de ces questions va pouvoir mesurer de manière indépendante le niveau de qualification du candidat. Les réponses à un "item" sont présentées sous la forme d'un choix multiple, de façon à ce que l'examinateur puisse attribuer un score de 1 pour une bonne réponse ou un score de 0 pour une mauvaise réponse. Pour un test classique, le candidat obtiendrait un score équivalent à la somme de tous ses points. En réalité, cette méthode perçoit l'aptitude à répondre correctement à chaque question individuellement, non pas au test dans sa globalité.

Xinming An et Yiu-Fai Yung ont publié un article abordant une méthode pour établir un score précis de tests et le développement de nouvelles questions de tests [8]. Pour ce faire, ils utilisent les modèles IRT (Item Response Theory) dont ils détaillent les différents modèles et les appliquent à un cas concret, les LSAT (Law Scholastic Aptitude Test). L'Item Response Theory permet de dissocier la **difficulté d'une question** et l'**aptitude d'une personne**, deux caractéristiques a priori non discernables comme expliqué plus haut. Autre avantage considérable est qu'il n'est pas nécessaire qu'un utilisateur ait répondu à toutes les questions de l'examen pour évaluer son niveau. Ceci est très important pour notre travail car nous n'avons pas qu'un set de quelques questions auxquelles tous les utilisateurs ont répondu mais bien 2500 questions pour 7000 utilisateurs. Il est évident que chaque utilisateur n'aura pas répondu à toutes les questions. Ce qui nous amène au dernier avantage mentionné dans l'article : La difficulté d'une question n'est pas uniquement déterminée par le "*subset*" des utilisateurs qui ont répondu à cette question mais bien par l'analyse confondue de toutes les questions et toutes les réponses des utilisateurs.

En 2006, Dimitris Rizopoulos, professeur de la KUL, a développé un package R[4] pour utiliser les modèles de l'Item Response Theory[9]. Après avoir mis en avant la théorie des modèles et l'implémentation du package, il procède par une illustration des capacités et des fonctionnalités du package. Dans une de ses expériences, il utilise un set de données semblable au nôtre et réussit à obtenir le résultat souhaité et prédit par la théorie : les modèles calculent la difficulté des questions et le niveau d'aptitude d'une personne ayant répondu aux questions.

Pour la mise en place du tutoriel intelligent, nous allons implémenter les techniques de l'Item Response Theory qui, selon nous, est un choix judicieux. L'IRT nous donnera non seulement le niveau de difficulté de chaque question mais également d'autres caractéristiques comme la devinabilité (pourcentage pour lequel un utilisateur répond correctement à la question par chance ou déduction). En combinant ces caractéristiques et le niveau d'aptitude individuel des utilisateurs, nous pourrons construire un tutoriel sur mesure et performant.

4. Langage de programmation

3 Architecture et API

Comme pour chaque grand projet de software ou data science, il est essentiel de planifier une structure architecturale et de disposer de données convenables et utiles. Cette partie du travail traitera des étapes mises en oeuvre pour établir un environnement sain, efficace et extensible pour de futures applications.

L'architecture que nous avons mise en place est représentée sur la figure 3. Les différentes parties et les liens entre elles sont expliqués plus bas.

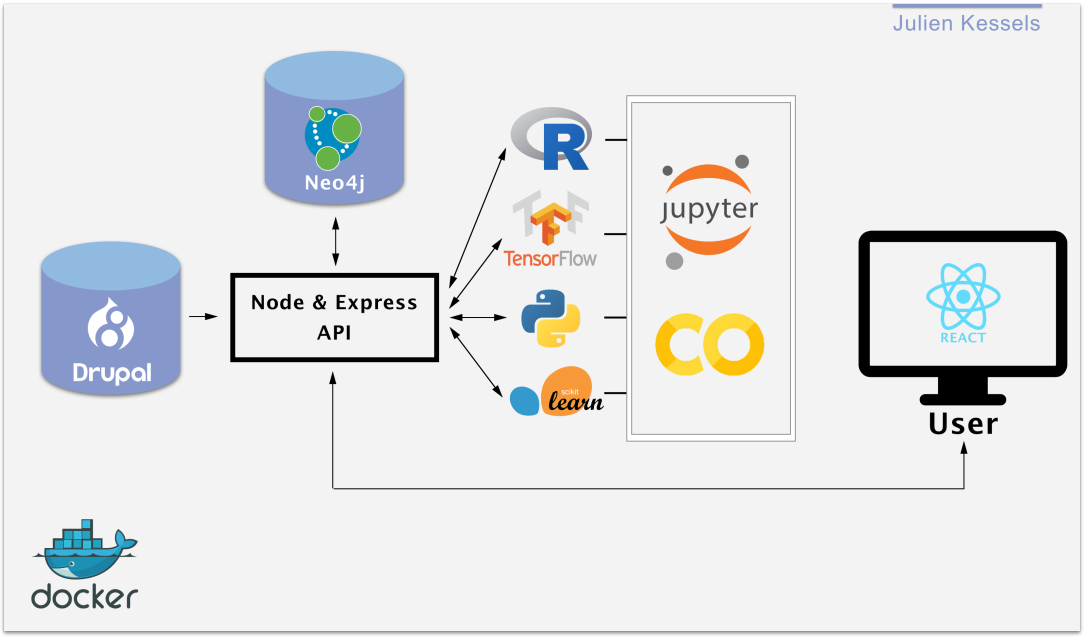


FIGURE 3 – Architecture de la plateforme

API

Les données que nous utiliserons, à travers divers algorithmes, pour mettre en place le *Dashboard Auteur* et le *Smart Tutorial* proviennent d'une base de données **Drupal**. L'ensemble de la plateforme d'EDITx a été conçue sur le CMS (Content Management System) Drupal. Cette façon de faire présente des avantages et désavantages : La création du site et la gestion de la base de données sont entièrement gérées par le CMS, ce qui réduit considérablement la charge de travail pour le maintien du site. Ces données sont cependant enregistrées sur un grand nombre de tables (> 1000) et cela rend l'extraction d'informations pour nos applications très lourde.

Choix et avantages : Il nous a paru indispensable de créer une API pour jouer le rôle de bridge entre les données brutes de Drupal et données adéquates à un traitement ultérieur. La création d'une API au lieu d'un script de migration de données vers une autre base de données mieux structurée semble être un choix plus judicieux puisque l'API communique directement avec la base de données Drupal, à jour à tout instant, et non avec une base de données qui doit être actualisée constamment pour refléter les changements sur la base de données d'origine.
Nous avons utilisé les librairies Node et Express pour implémenter l'API. La figure 4 regroupe une partie des différentes requêtes couramment implémentées.
Nous y retrouvons des requêtes pour obtenir les informations des utilisateurs, comme par exemple les questions auxquelles ils ont répondu. Nous pouvons aussi récupérer les questions créées par les contributeurs/auteurs pour ensuite faire le lien entre les questions et les utilisateurs, ce qui nous permettra de générer un tableau regroupant tous les utilisateurs avec à chaque fois les questions auxquelles ils ont bien ou pas bien répondu.

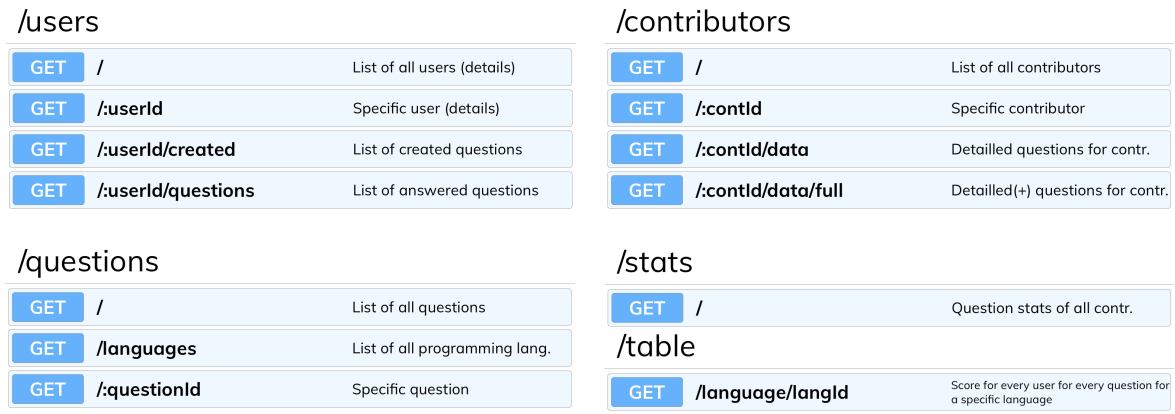


FIGURE 4 – API

Base de données NoSQL

La version 1 du tutoriel intelligent fonctionne sur base d'une base de données NoSQL, dans notre cas les bases de données graphe. Dans notre architecture, nous utiliserons le système de gestion de base de données open source Neo4j connecté directement à l'API pour toutes les requêtes pouvant profiter d'une base de données graphe.

Choix et avantages : Comme cela sera expliqué dans le chapitre 5.1, les bases de données graphe sont idéales lorsqu'on veut implémenter un système de suggestions. Quant au choix du système de gestion Neo4j, nous avons préféré celui-là car il est connu pour sa capacité de scale-up("grandir") facilement ce qui est crucial dans notre cas, comme EDITx a de plus en plus d'utilisateurs. De plus, Neo4j est connu pour sa conformité ACID pour assurer la prévisibilité des requêtes basées sur les relations.

Calculs

Le cerveau de l'architecture est constitué d'une série de scripts accessibles à l'API pour y effectuer des calculs complexes. Nous utiliserons une combinaison des librairies de machine learning *Scikit-learn* et *TensorFlow* ainsi que du langage de programmation adapté pour les statistiques *R*. Les scripts n'ayant pas besoin de beaucoup de ressources calculatoires tourneront en local sur le serveur d'EDITx tandis que les scripts d'entraînement des modèles de machine learning tourneront dans le Cloud, ou sur Google Colab.

Choix et avantages : Parmis les librairies les plus connues dans le monde du data science, Scikit-learn et TensorFlow nous aideront à la mise en place de nos modèles et algorithmes. TensorFlow est un peu plus "bas-level" que Scikit-learn et nous permet de construire les modèles à la main tandis que la librairie Scikit-learn implémente déjà une série de modèles prêts à l'emploi. Les deux librairies travaillent bien de manière complémentaire et formeront le point de départ de nos scripts de calculs. Nous profiterons du Hardware spécifique (GPU haut de gamme) de Google Colab pour entraîner nos modèles et réduire drastiquement le temps d'entraînement.

Front-end

Le front-end de la plateforme visible à l'utilisateur est entièrement codé en *React*. Par des requêtes HTTP vers l'API, la plateforme affiche du contenu différent selon la personne connectée.

Choix et avantages : Le choix d'utiliser la librairie React est très simple. Premièrement, React utilise JSX, une extension syntaxique de Javascript qui permet de mixer du HTML et du Javascript. Comme nous utilisons déjà du Javascript pour programmer notre API autant rester, par simplicité, dans le même environnement. Deuxièmement, React est très proche de son homologue React-Native utilisé dans le mobile. Si un jour, EDITx décide d'étendre son champ d'activité dans le mobile, nous pourrons le faire rapidement et simplement.

Conteneurisation

Pour faire tourner nos différentes parties de l'architecture de façon fiable et prédictive sur une grande variété de machines hôtes, nous utiliserons le logiciel de conteneurisation **Docker**.
Choix et avantages : Docker permet de facilement et rapidement déployer une application sur un serveur qui ne tourne pas forcément sur le même OS. Son système de conteneurisation au lieu de virtualisation est une forme plus légère qui s'appuie sur certaines parties de la machine hôte pour son fonctionnement. Cette approche permet d'accroître la flexibilité et la portabilité d'exécution d'une application.
Comme EDITx détient beaucoup de données personnelles de ses utilisateurs, il est essentiel d'adapter la sécurité de son site web en conséquent. Docker permet de grandement améliorer la sécurité en séparant les différentes parties (API, Calculs, Frontend, ..) en plusieurs conteneurs. En effet, si une partie (conteneur) est compromise par une attaque informatique, les autres conteneurs restent intacts.
Docker présente pourtant plusieurs désavantages[5]. À la base, Docker n'est pas prévu pour des applications possédant une partie graphique, mais comme dans notre cas il s'agit surtout d'une interface graphique statique, cela ne devrait pas poser trop de problèmes.

5. https://bit.ly/2YGjWed

4 Aide à la création de questions

Dans cette première partie, nous allons mettre en place un Dashboard accessible uniquement par les auteurs. Le but de ce Dashboard est d'offrir quelques outils qui assisteront l'auteur lors de la création de nouvelles questions de concours.

4.1 Language Detector

Le premier outil présenté est un détecteur automatique du langage de programmation de la question. Est-ce que la nouvelle question est supposée tester les connaissances en Python, en Java, ou autre ? Cet outil est surtout un moyen de valider les entrées de l'auteur car celui-là sait à priori pour quel type de catégorie il écrit la question.

Tokenization

Il faut savoir qu'il n'est pas possible d'analyser directement du texte en machine learning. Les ordinateurs fonctionnent brillamment lorsqu'il s'agit de nombres. Si on veut analyser du texte comme nous voulons le faire avec les questions, il faut d'abord encoder chaque mot de la phrase en nombres.

• La première manière de le faire est appelée *One-Hot-Encoding*. Cette technique encode chaque mot d'une phrase en un vecteur propre(voir exemple sur la figure 5) que nos modèles pourront mieux interpréter.

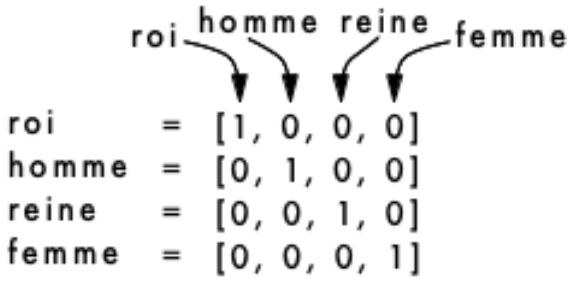


FIGURE 5 – Encodage par One-Hot-Encoding

Le problème de cette technique est que chaque mot, représenté dans l'espace grâce à son vecteur propre correspondant, se trouvera à la même distance des autres mots.

$$d = |\mathbf{x} - \mathbf{y}| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} = 1.41$$

avec par exemple $\mathbf{x} = [1,0,0,0]$
$\mathbf{y} = [0,1,0,0]$

Il ne sera donc pas possible d'en ressortir des similitudes entre mots car la similitude est une caractéristique qu'on peut comparer à une distance.

• La deuxième technique est appelée **Word2Vec**. Cette méthode est plus fastidieuse car elle nécessite d'entraîner un modèle de réseaux neuronaux qui prend en compte les voisins des mots de la même phrase. Pour chaque mot d'une phrase, on obtient un vecteur de deux dimensions représentant un point dans le plan. Les distances entre ces points ne sont cette fois plus les-mêmes pour chaque combinaison et nous pouvons donc plus facilement retrouver le lien entre deux mots.
Les figures ci-dessous montrent un exemple d'encodage des mots roi, homme, reine et femme par Word2Vec en partant d'un set de données "stories15k.txt" [6] et leur représentation dans le plan.

mot unique	One-Hot-Encoding	Word2Vec [X,Y]
roi	[1, 0, 0, 0]	[1, 2]
homme	[0, 1, 0, 0]	[1, 3]
reine	[0, 0, 1, 0]	[5, 1]
femme	[0, 0, 0, 1]	[5, 2]

FIGURE 6 – Encodage avec One-Hot-Encoding et Word2Vec

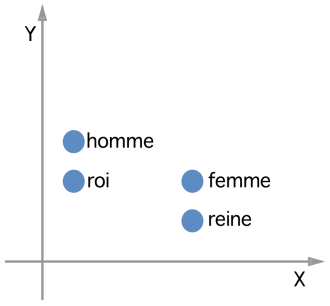


FIGURE 7 – Représentation avec Word2Vec

À vue d'oeil, on peut former deux groupes de similitude : Homme et roi ainsi que femme et reine. L'algorithme K-Means expliqué à la page suivante va nous permettre d'automatiser la détection de ces groupes.

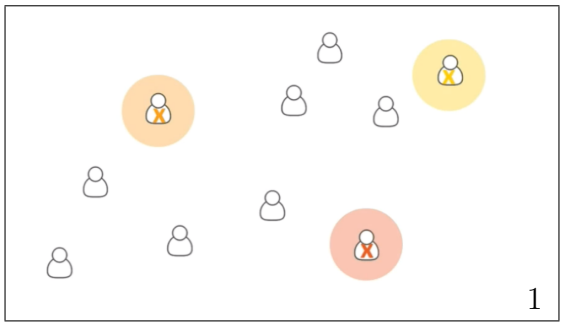
6. https://medium.com/arvind-internet/applying-word2vec-on-our-catalog-data-2d74dfee419d

L'algorithme

K-Means est un algorithme de clustering non supervisé, c'est-à-dire qu'il s'applique à des données dites "non-labeled" et permet, dans notre cas, de former des groupes de mots qui se ressemblent. Chaque groupe ainsi constitué représentera un langage spécifique. Ceci est exactement ce qu'il nous faut quand un auteur ajoute une nouvelle question. L'algorithme attribuera automatiquement un tag de langage à la question créée selon les mots de la phrase.

Les étapes de l'algorithme K-Means sont illustrées ci-dessous. Chaque "bonhomme" correspond à un mot.

1. Choix de N centres initiaux

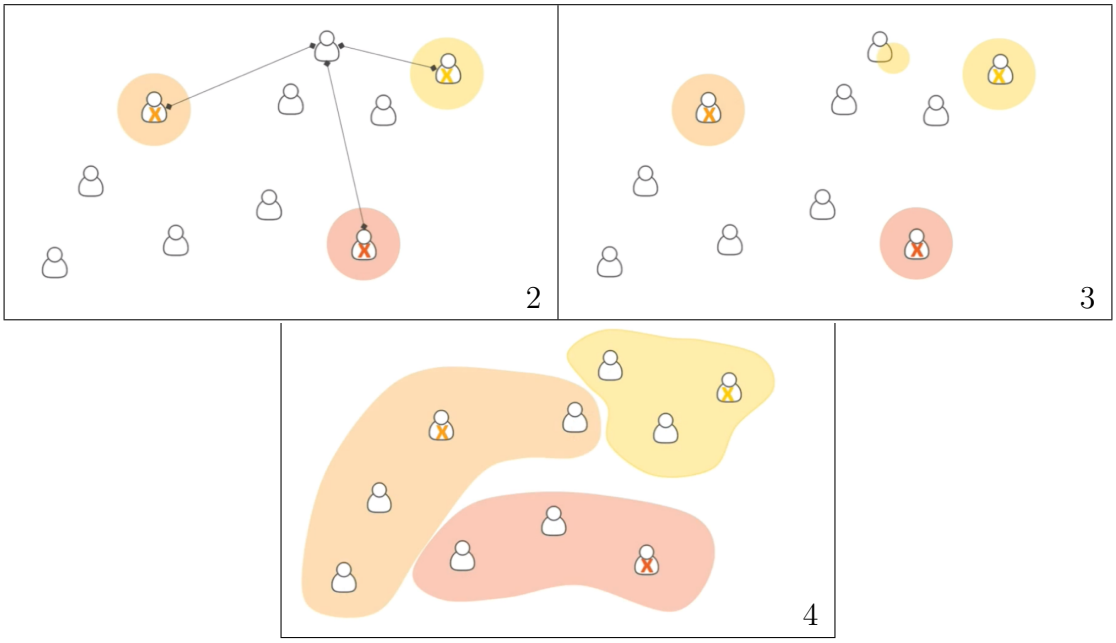


On commence par choisir au hasard N (ici N=3) centres appelés centroïdes. Les centroïdes se dénotent $c_1, c_2, ..., c_k$. On a :

$$C = c_1, c_2, ..., c_k$$

avec C, le set de tous les centroïdes.

2-4. Première catégorisation

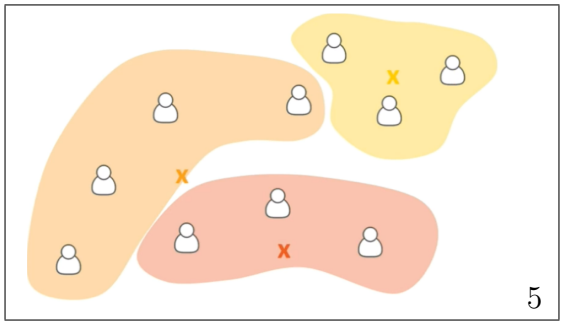


Ensuite on assigne les entrées au centre le plus proche en calculant la distance euclidienne entre les points et chaque centre.

$$arg \min_{c_i \in C} dist(c_1, x)^2$$

avec $dist(.)$, la distance euclidienne.

5. Recalcule des centres

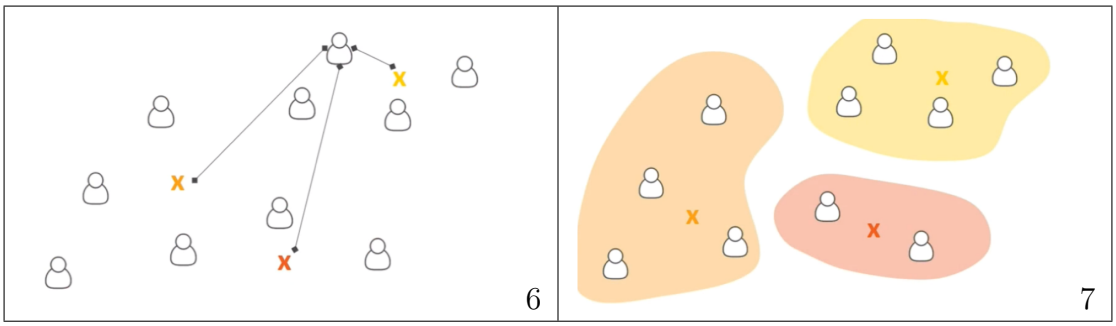


On recalcule les nouveaux centres qui correspondent au centre de gravité pour chaque set.

$$c_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i$$

avec S_i le set de tous les points attribués au centroïde c_i.

6-7. Recatégorisation



On recommence les étapes 2-5 jusqu'à ce que les centres ne bougent plus.
Nous avons alors trouvé les centres optimaux et catégorisé les entrées selon ces centres.

Implémentation et résultats

L'algorithme K-Means est implémenté dans Scikit-learn, une librairie Python qui regroupe une multitude d'algorithmes de machine learning.
Les entrées de l'algorithme sont un dump d'environ 300 questions des langages "Bebras", "Java", "Python", "PHP", ".NET", "UX/Usability".
80% de ces questions vont faire partie du learning set, donc les données pour entraîner le modèle.
Les 20 autres pourcents sont dédiées à tester notre modèle.

• Dans le premier test, nous avons configuré le modèle pour identifier cinq clusters (k=5). La figure 8 affiche une liste des cinq clusters générés avec les mots représentatifs des différents clusters. Les résultats pour les 60 autres questions (test set) sont en grande partie concluants.

```
Top terms per cluster:
Cluster 0:  console  writeline  code  output  foo  following  var  println  int  new  10  lt  exception  bar  add  echo  null  php  write  class
Cluster 1:  print  execution  data  following  does  program  code  int  def  instruction  len  hold  range  variables  lambda  values  length  value
Cluster 2:  following  method  class  net  true  use  correct  used  value  type  string  does  statement  statements  int  code  een  object  new
Cluster 3:  public  int  string  static  main  void  class  println  args  new  code  return  output  following  test  hello  program  result  private
Cluster 4:  le  la  les  et  une  il  est  des  en  dans  castor  sur  pour  par  que  au  deux  se  castors  sont
Cluster 5:  gt  lt  list  integer  stream  string  public  print  final  arrays  aslist  map  add  filter  does  new  static  args  main  foreach


Prediction
Question is: Bebras. Would put in cluster [4]
Question is: Java. Would put in cluster [5]
Question is: Python. Would put in cluster [1]
Question is: PHP. Would put in cluster [0]
Question is: .NET. Would put in cluster [3]
Question is: UX/Usability. Would put in cluster [2]
```

FIGURE 8 – Top words pour chaque cluster, k = 5

Nous obtenons un taux de classification correct de 71%. La plus grande source d'erreur s'avère être la mauvaise classification entre .NET et Java.

• Dans le deuxième test, nous avons réduit le nombre de clusters à trois (k=3). Ce test est intéressant pour voir comment notre modèle classifierait les questions s'il y a moins de clusters que prévu. En effet, nous savons à priori qu'il devrait y avoir cinq clusters parce que les 300 questions proviennent de cinq langages de programmation différents.

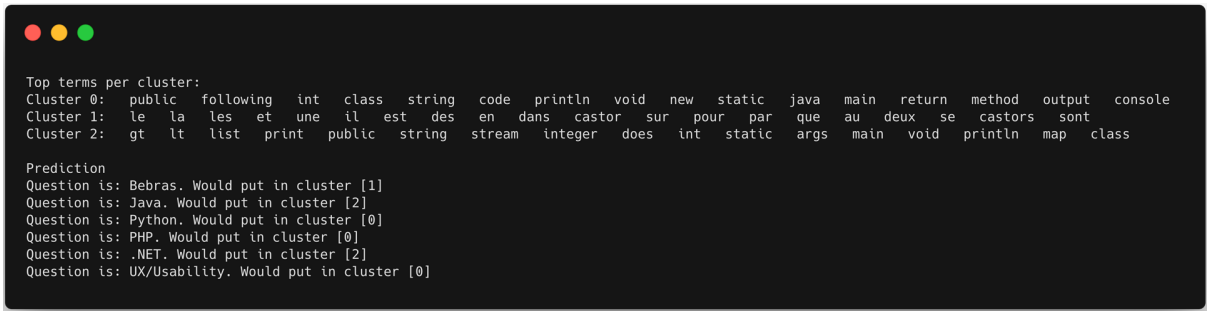


FIGURE 9 – Top words pour chaque cluster, k = 3

Des résultats obtenus, on voit effectivement que l'algorithme aurait tendance à placer une question Java et .NET dans le même cluster. C'est bien ce qu'on retrouvait dans notre premier test et qui était la source d'erreur prédominante.

Optimisation de la classification

Les mots présents et récurrents dans toutes les questions comme "the", "a", "an", "in", ne vont pas aider l'algorithme à correctement classifier les questions. Ces mots-là vont au contraire augmenter la similitude entre les questions et fausser nos résultats. En natural language processing (NLP), ces mots sont appelés **Stop Words**. Des listes de *Stop Words* en anglais existent sur internet[7]. Pour optimiser notre modèle, nous passons par une phase de pré-processing consistant à éliminer ces mots dans chaque question pour que l'algorithme puisse se concentrer sur les mots essentiels.

Grâce à cette technique, nous observons effectivement une augmentation significative du taux de classification correcte (78%) avec les mêmes 60 questions.

7. https://gist.github.com/sebleier/554280

4.2 Difficulty Estimation Alert

Le deuxième outil du Dashboard Auteur **renseigne l'auteur sur sa capacité à bien estimer le niveau de difficulté de ses questions**. Lorsqu'un auteur ajoute une nouvelle question sur le site d'EDITx, il doit définir sa difficulté parmi les options "Easy", "Medium" et "Expert". Cependant, la difficulté d'une question est un concept subjectif, car un individu n'a pas forcément les mêmes connaissances qu'un autre. L'estimation de la difficulté d'une question reste une pratique assez vague car l'auteur n'est pas non plus conscient du niveau des utilisateurs.

Sur base de toutes les questions existantes créées par l'auteur ainsi que des scores obtenus par chaque utilisateur pour ces questions, l'algorithme va mesurer si la difficulté prédite pour ces questions est proche de la difficulté réelle retenue par les utilisateurs.
Pour ce faire nous allons définir deux valeurs : Le **Target Score** (t_s) et le **Real Score** (r_s).
Le **Target Score** est une valeur exprimant de taux de réussite attendu par l'auteur. Le target score peut valoir trois valeurs qui ont été choisies arbitrairement pour les trois niveaux de difficulté :

- Easy : $t_s = 75$ (75% de taux de réussite pour une question facile)
- Medium : $t_s = 50$ (50% de taux de réussite pour une question medium)
- Expert : $t_s = 25$ (25% de taux de réussite pour une question difficile)

Le **Real Score** est le taux de réussite réellement obtenu par les utilisateurs pour une question.

Pour chaque question nous calculons la distance

$$d = t_s - r_s$$

Si un auteur a bien prédit le niveau de difficulté de sa question, nous obtenons une distance proche de 0. Nous avons défini que si la valeur absolue de la distance vaut moins que 20, l'auteur a bien estimé le niveau de difficulté de sa question. Exemple :
Un auteur crée une question et indique qu'elle est difficile, c'est-à-dire qu'il s'attend à un taux de réussite de 25% (Target Score = 25). Par la suite, on observe que le taux de réussite des utilisateurs vaut 32% (Real Score = 32). La distance entre Target Score et Real Score vaut -7. La valeur absolue de -7 étant plus petite que 20, nous estimons que l'auteur a bien estimé la difficulté de la question.

La figure 10 affiche toutes les questions créées par un auteur. Sur l'axe des X, nous avons la distance obtenue entre Target Score et Real Score. En Y, on retrouve le nombre de questions ayant eu la même distance.
Les traits pointillés sur la figure délimitent les questions dont la difficulté a bien été estimée ($-20 < d < 20$) et les questions de difficulté mal estimée.

Idéalement, on aimerait bien avoir une forme de cloche avec une haute concentration autour de $X = 0$ ($d = 0$), ce qui signifie que la difficulté d'une question a été parfaitement bien estimée.

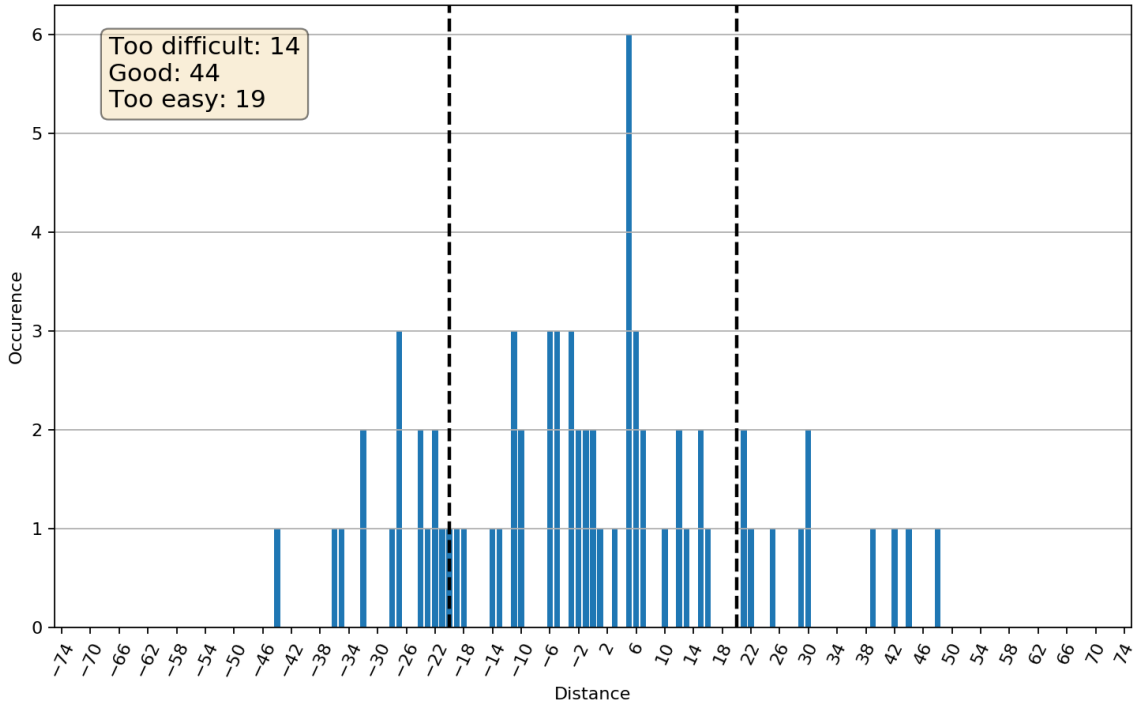


FIGURE 10 – Estimation correcte de la difficulté des questions

Si $d > 0$, la question était plus difficile qu'estimé par l'auteur. En revanche, si $d < 0$, la question était plus facile.

Les figures 11 et 12 montrent deux auteurs qui ont respectivement sous- et surestimé la difficulté de leurs questions.

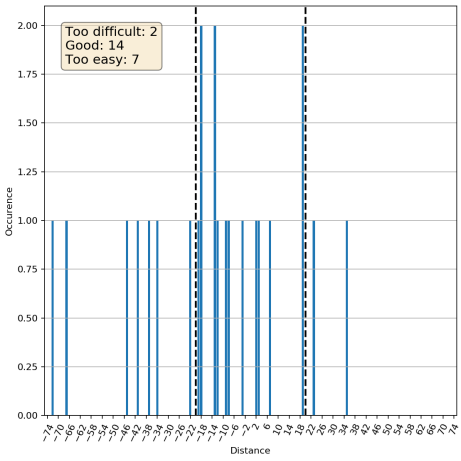


FIGURE 11 – Questions plus faciles qu'estimé par l'auteur

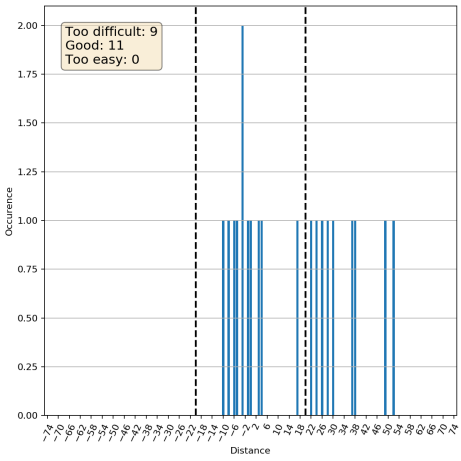


FIGURE 12 – Questions plus difficiles qu'estimé par l'auteur

Lorsqu'un auteur accédera au Dashboard Auteur pour créer une nouvelle question, cet outil l'alertera sous forme d'un Popup dans le cas où l'algorithme aura détecté qu'il estime mal la difficulté de ses questions.
Grâce à cette information, il pourra agir en conséquence et adapter son estimation de niveau de difficulté pour ses futures questions.

4.3 Duplicate Detector

Le troisième outil du Dashboard Auteur vise à détecter des questions déjà existantes dans la base de données et ainsi à empêcher la création de doublons ou questions similaires. Il existe une multitude de façons et d'algorithmes pour analyser les similitudes entre phrases.
Nous en avons choisi quatre : Levenshtein, Manhattan Distance, Euclidian Distance et Cosine Similarity.
Le fait d'avoir quatre algorithmes travaillant en même temps va nous permettre de valider ou non les résultats des autres algorithmes. Cette validation croisée nous donne une sécurité supplémentaire quant aux résultats obtenus.

Comme pour le "détecteur automatique de langage de programmation" vu dans le chapitre 4.1, il faut transformer les phrases en une série de mots encodés grâce à la technique de *Word2Vec*. On va ensuite pouvoir calculer des similitudes entre ces mots encodés représentés par des points dans un plan bi-dimensionnel (voir figures 14, 15 et 16)

Levenshtein

La distance de Levenshtein est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'un mot à un autre.
Cet algorithme est assez puissant pour des phrases courtes mais perd en précision pour des phrases de plus en plus longues.

		S	a	t	u	r	d	a	y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

FIGURE 13 – Levenshtein

Manhattan Distance

La distance de Manhattan s'exprime comme suit :

$$d = (x_2 - x_1) + (y_2 - y_1)$$

avec $P_1(x_1, y_1)$ l'emplacement dans le plan du premier mot et $P_2(x_2, y_2)$ l'emplacement dans le plan du deuxième mot.

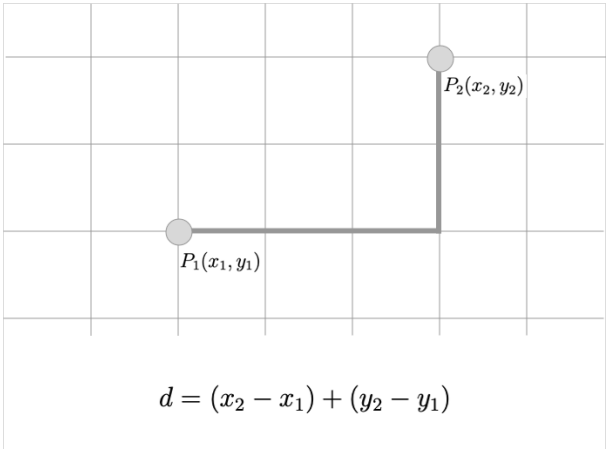


FIGURE 14 – Manhattan Distance

Euclidian Distance

La distance euclidienne s'exprime comme suit :

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Ici, la distance est calculée comme avec la formule de Pythagore tandis que pour la distance de Manhattan, il s'agit d'une distance de valeur linéaire (pas d'intervention d'exposants). En pratique les résultats sont similaires, avec une tendance favorable pour la distance de Manhattan.

La distance Euclidienne est également supposée donner de meilleurs résultats pour une dimensionalité (longeur de phrase) plus petite par rapport à la distance de Manhattan. Nous vérifierons cette théorie dans la partie *"Implémentation et résultats"* de ce chapitre.

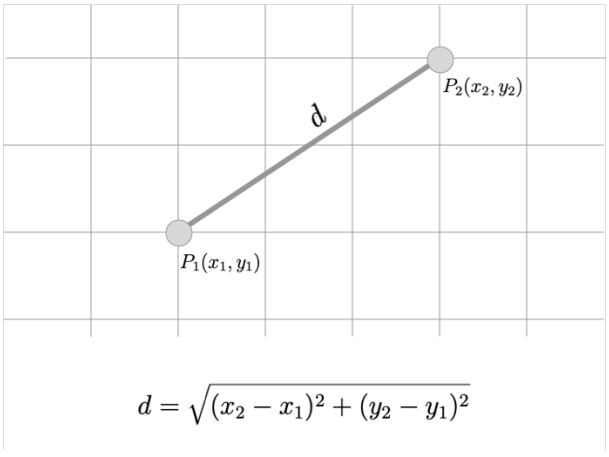


FIGURE 15 – Euclidian Distance

Cosine Similarity

La similarité cosinus s'exprime comme suit :

$$similarity = \cos(\theta) = \frac{\mathbf{A} * \mathbf{B}}{\|\mathbf{A}\| \, \|\mathbf{B}\|}$$

Cet algorithme s'intéresse plus à la similarité directionnelle que des différences exprimées en distance. La similarité cosinus est plus efficace pour interpréter le sens logique plutôt que des nombres. D'ailleurs la taille des phrases influence peu les résultats de cette méthode, en opposition à la distance de Levenshtein dont la précision se détériore rapidement en fonction de la taille des phrases.

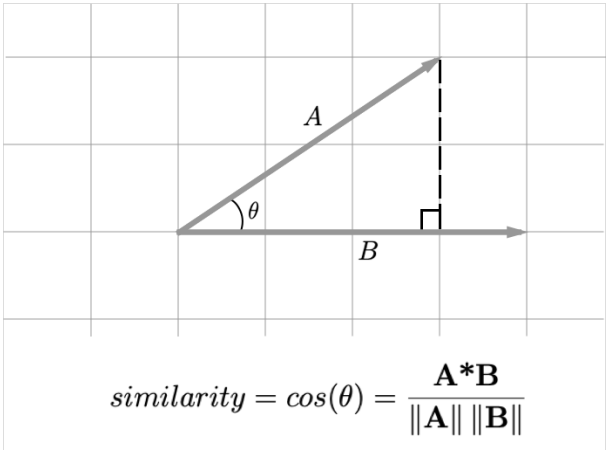


FIGURE 16 – Cosine Similarity

Implémentation et résultats

Les algorithmes présentés ont été programmés en Python et tournent sur le conteneur dédié aux calculs (cf. Architecture). Nous avons composé une série de 20 questions, 10 tirées de la base de données que nous avons légèrement modifiées, 10 rédigées de zéro sans connaissance des questions existantes. Chaque question est passée par les quatre algorithmes.
Dans le cas où un algorithme trouve une question semblable dans la base de données, il nous indique la question déjà existante qu'il trouve ressemblante à la nouvelle question.

Pour tester et valider notre implémentation, nous avons pour chacune des 20 questions analysé la réponse des différents algorithmes et vérifié manuellement s'ils avaient raison.
Les trois cas de figure pouvaient donc se présenter à ce moment :

- Correct : Une question similaire a été détectée et c'est bien le cas / La nouvelle question n'a pas enclenché d'alerte et il n'y a effectivement pas de question similaire existante
- Type I Error : L'algorithme a détecté une question similaire alors que les deux questions ne sont pas similaires
- Type II Error : L'algorithme n'a pas détecté de question similaire, mais il y en avait déjà une fortement semblable dans la base de données

Le tableau suivant résume les résultats obtenus.

	Levenshtein	Manhattan	Euclidian	Cosine Similarity
Correct	9	12	11	15
Type I Error	4	5	6	2
Type II Error	7	3	3	3

Comme prédit lors de la présentation des algorithmes, la distance de Levenshtein donne des résultats médiocres. Ceci est dû à la taille des phrases qui peuvent parfois faire quelques lignes. La similarité cosinus donne des résultats très convaincants la plupart du temps. Trois doublons sur quatre peuvent être détectés avec cet algorithme.

Pour tenter de valider la théorie exposée plus haut, à savoir que la méthode de distance euclidienne est plus efficace que la méthode de la distance de Manhattan pour des phrases plus courtes, nous avons testé les deux algorithmes avec 5 questions de nombre de mots allant de 10 à 40. La figure 17 illustre les résultats obtenus.

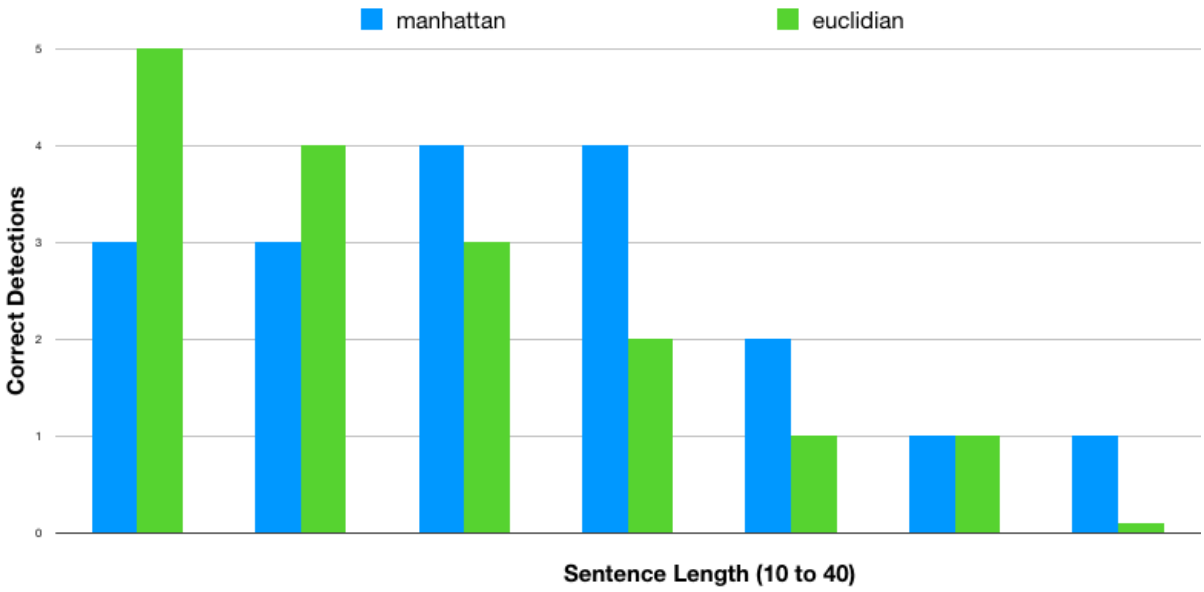


FIGURE 17 – Nombre de détections correctes de doublons des méthodes de distances de Manhattan et d'Euclidian en fonction du nombre de mots dans une phrase

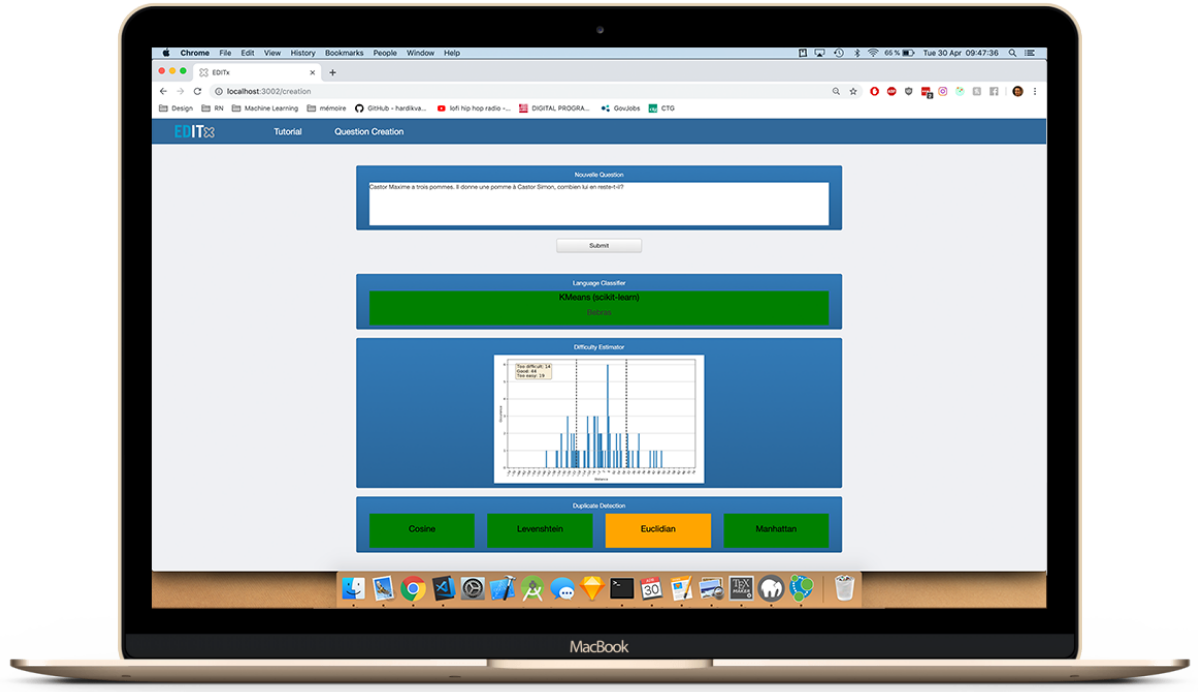
Sur base de la figure 17 on peut effectivement observer une efficacité décroissante (nombre de détections correctes) de la méthode de distance euclidienne dont le taux de réussite baisse

continuellement plus la phrase s'allonge en opposition à la distance de Manhattan dont la performance augmente un peu avant de retomber. C'est une observation qu'il faut absolument prendre en compte dans notre étude de détection de doublons : il faut favoriser le résultat obtenu par la distance euclidienne pour des phrases plus courtes et favoriser la distance de Manhattan pour des phrases plus longues.

On voit également dans les deux cas qu'à partir de 25 mots, les deux algorithmes s'écroulent et ne donnent plus de résultats satisfaisants. Pour de longues phrases, il vaut mieux prendre en compte la similarité cosinus qui est moins sensible à la taille des phrases.

4.4 Dashboard

Une fois identifié sur la plateforme d'EDITx, l'auteur peut accéder à la page *Question Creation* et ajouter une nouvelle question.
Après avoir appuyé sur le bouton "Submit", le front-end fait une rapide validation syntaxique de la question pour éviter tout type d'attaque à la SQL Injection, XSS Attacks ou autre. Ensuite, il envoie la question au conteneur de calculs via une requête à l'API qui se charge de faire le lien entre le front-end et les calculs.
Une fois que les trois outils d'aide à la création de questions ont fini leur calculs, le front-end est mis à jour et affiche les résultats à l'auteur.



• Le champ de détection du langage de programmation affiche le langage déterminé par l'algorithme K-Means. Ce champ peut être remplacé par l'auteur s'il pense que l'algorithme s'est trompé.

• Le champ d'alerte de l'estimation du niveau de difficulté des questions de l'auteur affiche le graphe avec ses anciennes questions et les distances. Si l'algorithme détecte que l'auteur estime mal le niveau de difficulté de ses questions, un popup apparaîtra pour signaler la situation à l'auteur.

• Les champs des algorithmes de détection de doublons peuvent avoir une couleur de background différente.

— Vert : Aucun doublon détecté

— Orange : Doublon possible détecté. Possibilité de cliquer sur le champ pour afficher la question qui pose problème.

5 Tutoriel intelligent

Il existe plusieurs façons d'implémenter un module de tutoriel.
On peut d'ailleurs comparer cet outil à un système de suggestion comme la suggestion d'articles sur Amazon en analysant les liens entre utilisateurs. C'est cette technique qui sera étudiée et réalisée dans la première version.
La deuxième version se basera sur une approche plus pragmatique de la vision d'un tutoriel. De par les modèles de l'Item Response Theory, la version 2 proposera une suite de questions dont la difficulté aura préalablement été calculée pour offrir un tutoriel d'apprentissage d'un langage de programmation, adapté aux utilisateurs.

5.1 Version 1 - GraphQL

Si on représente les questions et les utilisateurs comme des entités, on peut créer des relations entre ces entités.

Les noeuds représentés en violet sont les utilisateurs et ceux en rouge les questions. L'arête entre les deux indique qu'un utilisateur a répondu correctement à cette question.

Une question peut être répondue par plusieurs personnes. Cela implique que ces personnes sont donc aussi en relation par l'intermédiaire de cette question.
Le point fort des bases de données graphe est de parcourir et analyser ces relations rapidement. Grâce à nos analyses et observations nous proposerons un tutoriel non pas axé sur l'idée de progresser dans un langage de programmation, mais plutôt sur les centres d'intérêts.

Le critère du choix des questions proposées à l'utilisateur pour le tutoriel est la **ressemblance entre l'utilisateur et un autre**.

Ce genre de représentation est celui des bases de données graphes comme Neo4j.

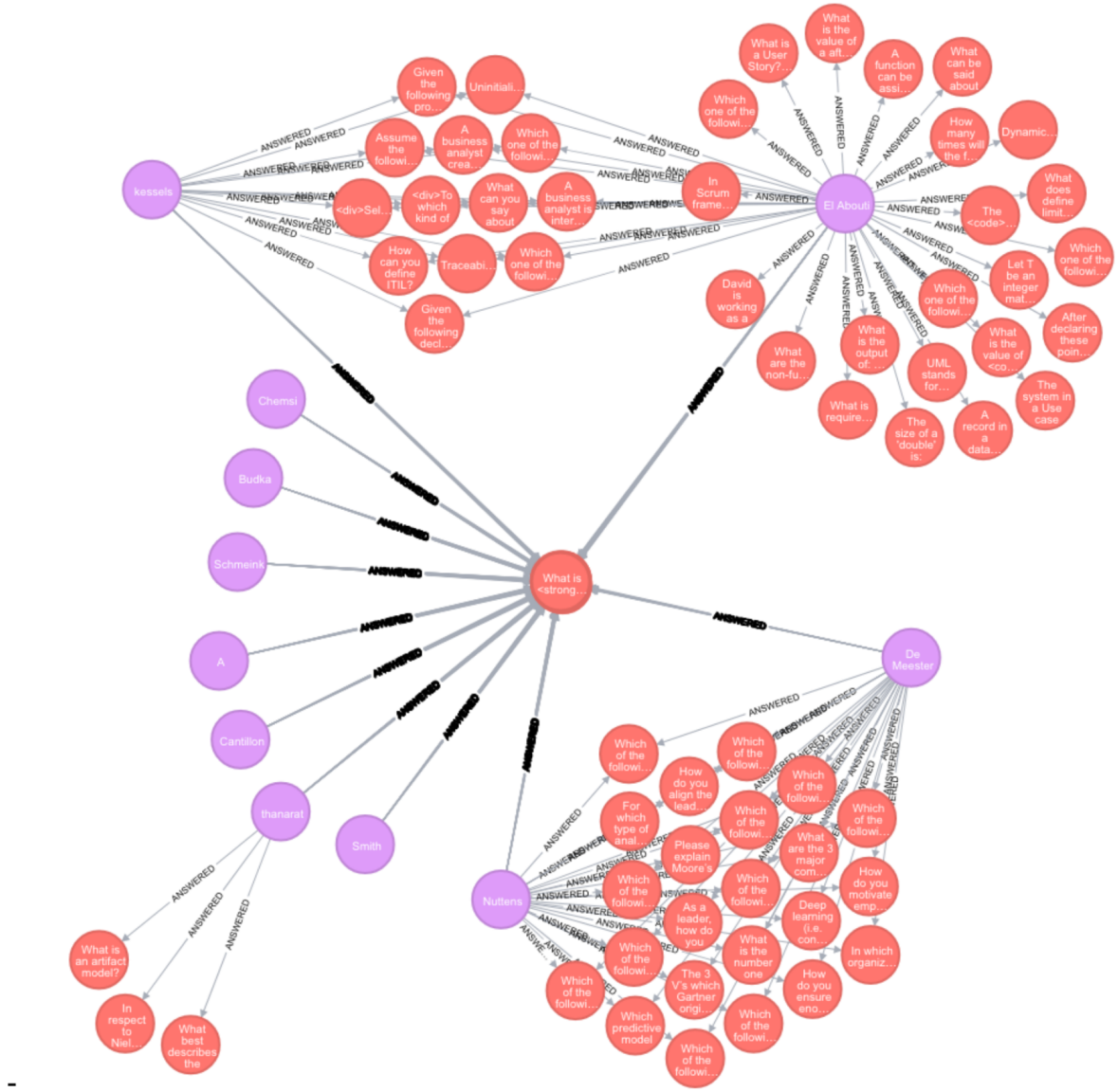


FIGURE 18 – Représentation d'un graphe avec les relations user-question

Voici les étapes suivies pour le choix des questions posées pour un certain utilisateur.

— Récupération des questions réalisées par l'utilisateur.

— Récupération des utilisateurs qui ont répondu à ces questions

En neo4j, ces deux étapes correspondent aux requêtes :

```
(me:User {id: myID})-(:ANSWERED_CORRECTLY) -> (question:Question) RETURN question.id
(q:Question {id: question_id}) <- (:ANSWERED_CORRECTLY)-(other:User) RETURN other.id
```

FIGURE 19 – Cypher query

De ces requêtes on retrouve une liste d'utilisateurs pour chaque question. Il suffit ensuite d'identifier l'utilisateur qui revient le plus souvent dans la liste (avec le plus de questions bien répondues en commun). Cette personne est définie comme la personne ayant le plus de centres d'intérêt communs avec la personne connectée actuellement.

Après avoir trouvé la personne la plus semblable, on récupère toutes les questions auxquelles elle a bien répondu et le tutoriel proposera ces questions à notre utilisateur connecté.

Pourquoi GraphQL ?

Le point fort de neo4j est de traverser rapidement des graphes. Pour réaliser des requêtes de ce type en relationnel, il faudrait faire des requêtes avec beaucoup de jointures entre les tables mais le moteur n'est pas assez performant pour le faire en pratique. Selon Stackoverflow[8], pour une requête avec 12 jointures, l'ordre entre les différentes jointures peut monter jusqu'à 28,158,588,057,600. A priori, le seul facteur limitant pour le nombre de jointures est la disponibilité des ressources du serveur(mémoire).

8. https://stackoverflow.com/questions/2558667/what-is-the-maximum-number-of-joins-allowed-in-sql-

Conclusion

Après avoir essayé ce tutoriel avec une série de testeurs, les résultats ont montré un réel intérêt. Les testeurs y ont retrouvé des questions intéressantes et pertinentes, à chaque fois focalisées autour de leurs centres d'intérêt.

Pour donner un exemple concret, on pourrait avoir un utilisateur doué en Python et en R. Sur base de ses bonnes réponses à des questions dans ces deux branches, le tutoriel identifierait un expert en machine learning comme étant la personne la plus semblable à l'utilisateur. Ils auraient en effet beaucoup de bonnes réponses aux questions en commun. Peut-être que l'expert en machine learning a également répondu à une série de questions autour d'autres domaines du Big Data. Le tutoriel proposerait dans ce cas là sûrement des questions de cette branche à notre utilisateur connecté.

5.2 Version 2 - Item Response Theory

Dans ce chapitre nous allons mettre en place une autre méthode, plus élaborée, pour établir un tutoriel d'apprentissage de langage de programmation. Cette méthode, appelée Item Response Theory (IRT) mettra d'avantage l'accent sur le niveau d'une personne, ainsi que sur le niveau de difficulté des questions pour aboutir à un tutoriel plus sophistiqué.

Comme discuté dans la section de l'état de l'art, Item Response Theory est une approche différente de la méthode classique d'évaluation pour laquelle un candidat obtient un score final équivalent à la somme de toutes les questions bien répondues. Cette méthode perçoit l'aptitude à répondre correctement à chaque question individuellement et non pas au test dans sa globalité.

Dans ce qui suit, nous allons analyser Item Response Theory en expliquant le principe plus en détail. Après avoir exploré les différents modèles de IRT, nous appliquerons celui paraissant le plus approprié à nos données et exploiterons les résultats pour définir un tutoriel puissant et personnalisé pour chaque utilisateur, en fonction du langage de programmation dans lequel il souhaite progresser.

5.2.1 Item Characteristic Curve (ICC)

Reprenons le cas d'un examen comprenant une série de questions.
Une hypothèse raisonnable est que chaque candidat répondant à ces questions possède une partie des compétences nécessaires pour répondre correctement. **Ce niveau de compétence d'une personne est aussi appelé aptitude** et est dénoté par la lettre θ. Nous verrons par la suite qu'il est possible de calculer l'aptitude d'une personne par les modèles d'IRT.
Dans IRT, la valeur de l'aptitude est généralement bornée entre -5 pour une personne "étourdie" et 5 pour une personne "intelligente". Pour chaque niveau de compétence, on peut calculer la probabilité que le candidat réponde correctement à une question. Cette probabilité est notée $P(\theta)$ et est illustrée sur la figure 20.

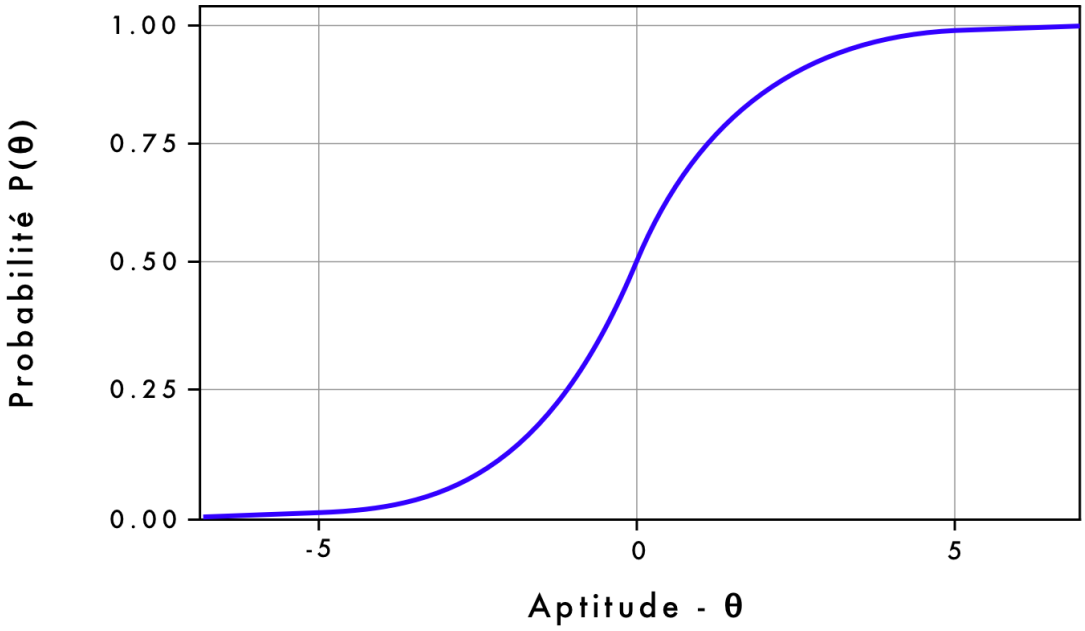


FIGURE 20 – Item Characteristic Curve

Sur la figure 20, on peut voir que la probabilité $P(\theta)$ de répondre correctement à une question est petite pour un candidat de faible compétence et élevée pour un candidat à compétence élevée. Cette courbe en forme de S décrit la relation entre la probabilité de bien répondre à une question et l'échelle de compétence (aptitude).

En Item Response Theory, elle est connue sous le nom de **Item Characteristic Curve**. Chaque question d'un test aura sa propre Item Characteristic Curve.

Cette courbe est le modèle le plus simple de la fonction logistique cumulative, dont l'équation s'écrit :

$$P(\theta) = \frac{1}{1 + e^{-\theta}}$$

avec θ : aptitude

"La probabilité de répondre correctement à une question en fonction de l'aptitude."

Dans les prochains sous-chapitres, nous allons voir les différents modèles de l'Item Response Theory, qui se basent sur la fonction logistique cumulative, mais modifiée pour extraire certaines caractéristiques des questions.

ICC à un paramètre : Difficulté

Le modèle de la fonction logistique à un paramètre a été publié pour la première fois par Georg Rasch, d'où le nom **Rasch Model**. Dans ce modèle, il a introduit le paramètre **b**, appelé **difficulté**, qui est le seul paramètre modifiant la forme de notre courbe.

$$P(\theta) = \frac{1}{1 + e^{-1(\theta - b)}}$$

avec b : difficulté

θ : aptitude

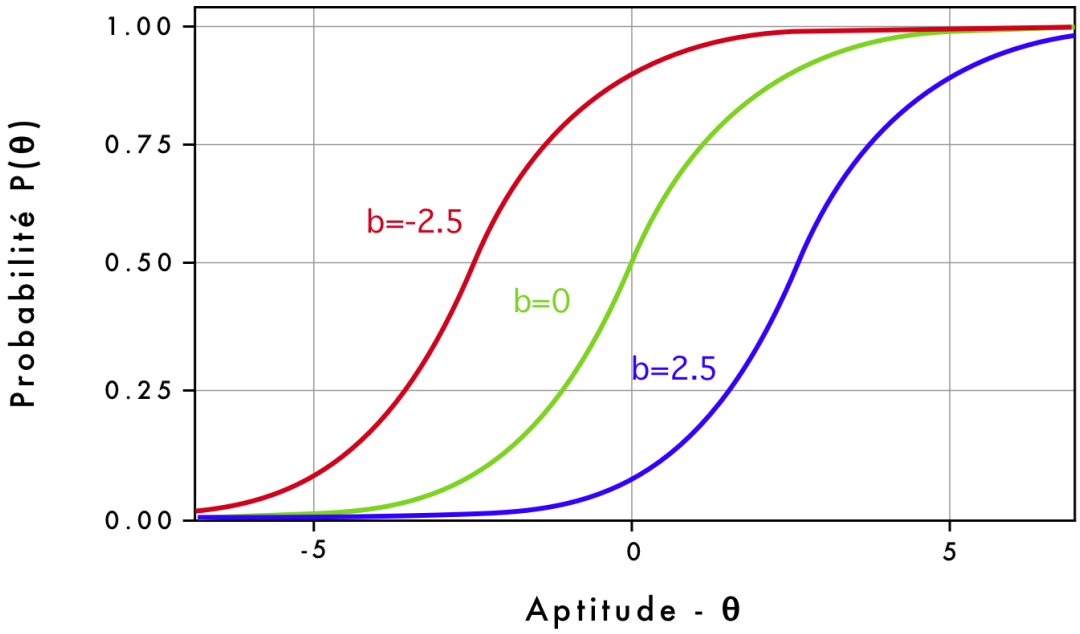


FIGURE 21 – ICC pour différentes valeurs de la difficulté

La figure 21 représente trois courbes pour trois valeurs différentes de b. La courbe rouge représente une question facile, tandis que la courbe bleue représente une question difficile. On voit bien qu'une personne ayant une aptitude de 0 a plus de chance à bien répondre à la question rouge($P(0) \simeq 0.90$) qu'à la question bleue($P(0) \simeq 0.10$).

• Ce paramètre va jouer un rôle primordial dans la mise en place du tutoriel intelligent, car il proposera, en fonction du niveau du candidat, des questions de plus en plus compliquées.

ICC à deux paramètres : Discriminant

Le modèle de l'Item Characteristic Curve à deux paramètres introduit le discriminant, noté **a**. **Le discriminant indique l'efficacité avec laquelle une question peut départager les candidats ayant une aptitude plus ou moins élevée par rapport au niveau de difficulté de la question**. Ce paramètre fait varier la pente de la courbe : Au plus la pente est raide, au mieux la question peut différencier les bons et mauvais candidats. Nous montrerons un exemple plus bas pour illustrer cette affirmation.

$$P(\theta) = \frac{1}{1 + e^{-a(\theta - b)}}$$

avec b : difficulté

a : discriminant

θ : aptitude

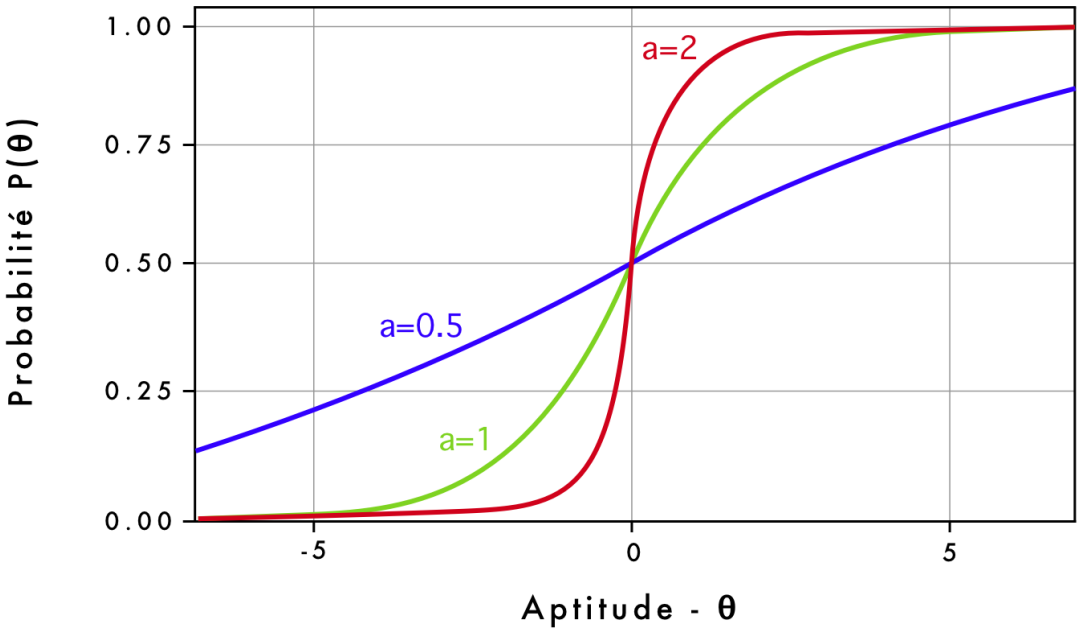


FIGURE 22 – ICC pour différentes valeurs du discriminant avec b=1

La figure 22 représente trois courbes pour trois valeurs différentes de a, avec b (la difficulté) fixe. La courbe bleue (a=0.5) discrimine très mal. Visuellement, on retrouve une pente très peu inclinée et, en observant $P(\theta)$, on voit effectivement que la probabilité de bien répondre à une question ($P(\theta)$), est presque la même pour une personne de faible aptitude ($\theta < 0$) que pour une personne ayant une aptitude élevée. En d'autres termes, plus une question a un discriminant élevé, mieux elle pourra déterminer l'aptitude d'un candidat.

• On verra dans le chapitre 5.2.7 que le discriminant joue un rôle important dans le choix des questions pour un candidat qui n'a encore répondu à aucune question et dont on voudrait, sur base de quelques questions, rapidement déterminer le niveau d'aptitude.

ICC à trois paramètres : devinabilité

Un fait récurrent dans le domaine de l'évaluation est que les candidats répondront correctement à des questions en devinant. Ceci implique que la probabilité de bonne réponse $P(\theta)$ inclut une petite partie due à la "devinabilité". Le modèle à trois paramètres établi en 1968 par Birnbaum fait intervenir le paramètre devinabilité, noté c.

$$P(\theta) = c + (1 - c)\frac{1}{1 + e^{-a(\theta - b)}}$$

avec c : devinabilité
b : difficulté
a : discriminant
θ : aptitude

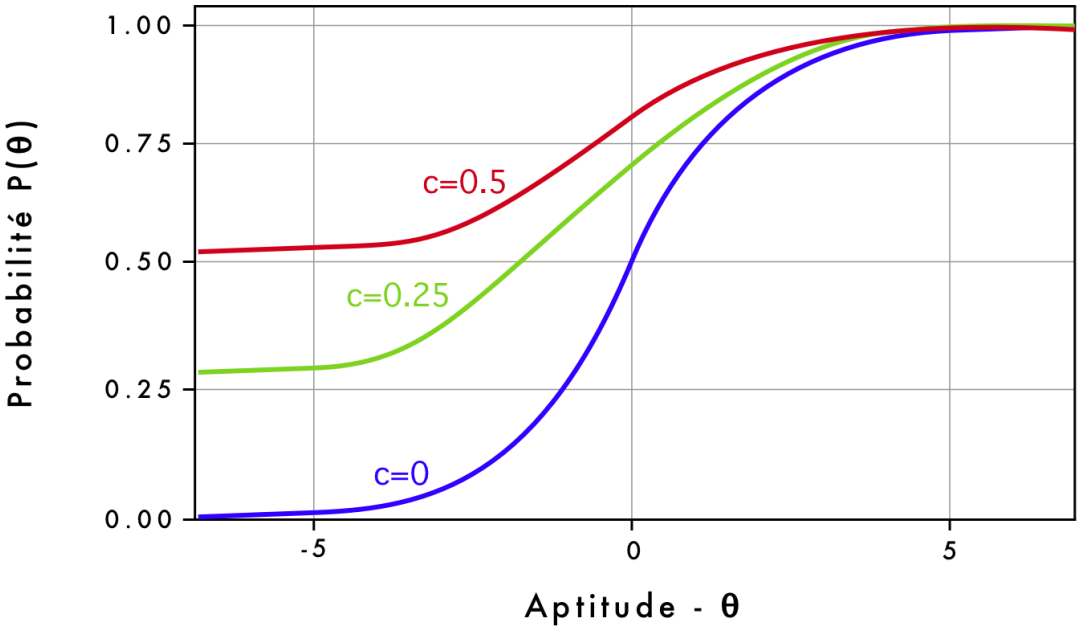


FIGURE 23 – ICC pour différentes valeurs de la devinabilité avec b=a=1

La figure 23 représente trois courbes pour trois valeurs différentes de c, avec a et b fixes. Quand le paramètre c est grand, comme par exemple pour la courbe rouge (c=0.5), on voit que même une personne avec une aptitude de -5 a plus de 50% de chance de répondre correctement à la question. La devinabilité est une caractéristique intéressante et a priori très compliquée à estimer sans outils tels que IRT.

• L'ICC à trois paramètres est le modèle le plus versatile et le plus intéressant. Pour notre travail, nous nous intéresserons à ce modèle car chacun des trois paramètres retrouve son utilité dans la création et la mise place d'un tutoriel intelligent.

5.2.2 Implémentation

Pour appliquer les modèles d'Item Response Theory, nous utilisons un package R développé par Dimitris Rizopoulos[9], disponible sur CRAN[9]. Ce package intègre les trois modèles décrits ci-dessus, mais nous utiliserons uniquement le modèle à trois paramètres, qui convient mieux à notre travail.

Dimitris Rizopoulos a implémenté le modèle par MMLE (Marginal Maximum Likelihood Estimation). En reprenant le modèle à trois paramètres, on a :

$$P(\theta) = c + (1 - c)\frac{1}{1 + e^{-a(\theta - b)}}$$

Les paramètres du modèle sont ensuite estimés en maximisant la log-vraisemblance

$$l_m(\theta) = log \int p(x_m|z_m;\theta)p(z_m)dz_m$$

avec $p()$: probabilité de densité de P(·)

x_m : vecteur des réponses pour le m-ième candidat

z_m : l'aptitude du m-ième candidat, calculée via le factor score (chapitre 5.2.4)

9. http ://CRAN.R-project.org

5.2.3 Résultats

Grâce à quelques requêtes vers notre API et un script Python, nous pouvons générer un fichier CSV, dont notre modèle aura besoin en entrée, avec les questions (n fois) en colonnes et les candidats (m fois) en rangées.
Un 1 indique une question bien répondue, un 0 indique une question mal répondue, un blanc indique que le candidat n'a pas répondu à la question.

m × n	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	...
Candidat 1	1	0			1	1	0		1	...
Candidat 2	1	1	1		0		1	1	1	...
Candidat 3	0	0	1	1	1	1	0	0	1	...
Candidat 4		0	1	0	1		0		1	...
...										...

Le fichier CSV passe ensuite par le modèle à 3 paramètres. De celui-là, on récupère une matrice (n*3) contenant les trois paramètres (a, b, c) pour chaque question.
Ensuite il est possible de tracer les questions sur notre graphique représentant $P(\theta)$ en fonction de θ. Pour les questions du topic "ITStudent", on obtient :

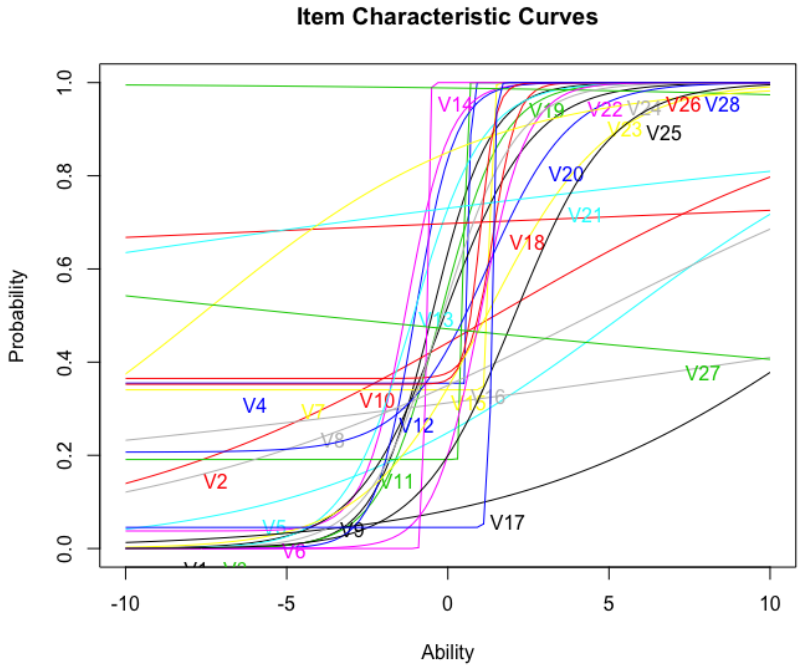


FIGURE 24 – Plot de toutes les questions ITStudent

Afin de mieux comprendre ce qui apparaît sur le graphique, subdivisons-le en plusieurs graphiques distincts.

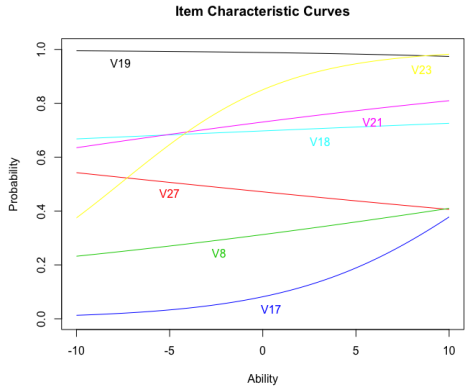


FIGURE 25 – Plot des questions à écarter

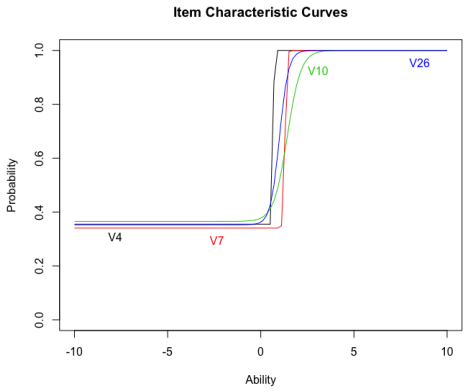


FIGURE 26 – Plot des questions devinables

Figure 25

Sur cette figure on s'aperçoit que les courbes ne forment pas le S prédit par la théorie. Ces courbes ont soit une **difficulté démesurée** ($|b| > 20$) soit un **discriminant négatif** ($a < 0$). Un discriminant négatif n'est pas normal : Cela voudrait dire que la probabilité de bien répondre à une question est décroissante pour une aptitude d'un candidat croissante. Ces questions ont souvent un problème critique. Soit elles n'ont pas de sens, soit elles sont mal formulées. Ceci est un signe pour le gestionnaire du site qu'il faut les revoir.

Figure 26

La littérature[10] suggère qu'en IRT, une question dont la devinabilité est supérieure à 0.35 n'est pas tolérable. Les questions où la **devinabilité est trop grande** ($c > 0.35$) sont représentées sur cette figure et en voici un exemple d'une question QCM avec une bonne réponse :

What is the most recent major version of the Python programming language ?

1 — 3 — 6

Il s'agit effectivement d'une question facile à deviner, car on peut trouver la bonne réponse par déduction : la dernière version de python est ni 1, ni 6 donc la bonne réponse est 3. Pour une question non devinable ($c < 0.1$), il n'est pas possible de trouver la bonne réponse simplement par déduction. Nous écarterons les questions trop devinables dans notre tutoriel car elles présentent peu d'intérêt.

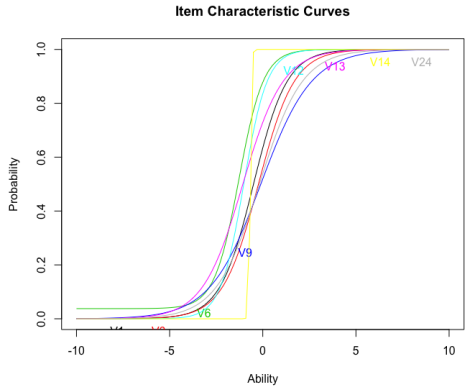


FIGURE 27 – Plot des questions faciles

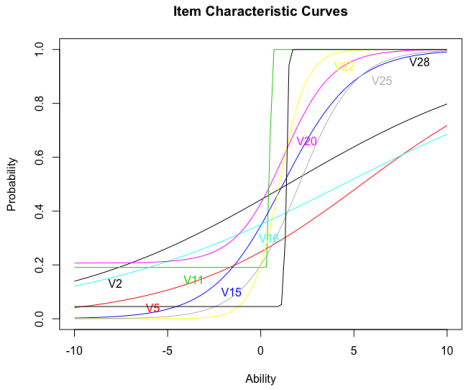


FIGURE 28 – Plot des questions difficiles

Figure 27

Ici sont représentées les questions faciles. Dans ce cas, ce sont les questions dont la difficulté (b) est arbitrairement choisie plus petite que 0. Graphiquement, il s'agit des courbes dont la probabilité pour une aptitude de 0 vaut 0.5 ou plus : $P(0) > 0.5$. Un candidat moyen($\theta = 0$) a au moins une chance sur deux de bien répondre à la question. Voici un exemple :

"Which artificial intelligence model tries to mimic the operation of the humain brain ?"

1) Neural network
2) Decision Support System
3) Machine-Brain Interface

La bonne réponse est 1).

Figure 28

À l'opposé on retrouve les questions difficiles :

"How many keys are used to exchange messages between two persons using an asymetric ciphering system ?"

1) 1
2) 2
3) 3
4) 4

La bonne réponse est 4).

La question a été classée parmi les questions difficiles, ce que nous trouvons personnellement étonnant. En analysant manuellement la question, on retrouve bien un taux de réussite net de 31% (28 bonnes réponses pour 83 candidats).
Pour analyser plus en profondeur le taux d'échec surprenant, nous avons fait une étude démographique des candidats ayant participé à ce concours ITStudent.

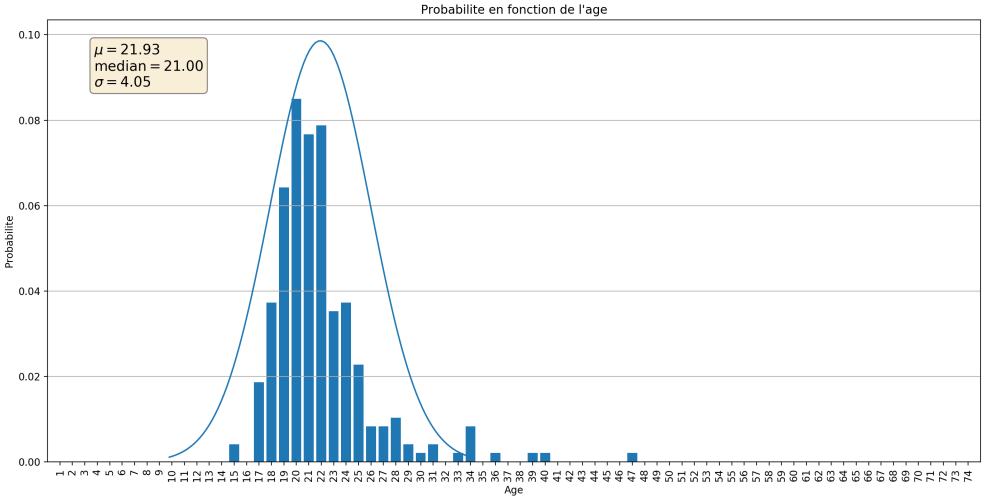


FIGURE 29 – Répartition de l'âge des candidats du concours ITStudent

La médiane d'âge se trouve à 21. Le taux d'échec paraît désormais plus évident, sachant que plus de la moitié des candidats sont encore en Bachelier, alors que les cours de sécurité informatique sont souvent enseignés en Master. Le résultat paraît dès lors moins surprenant.

Nous allons prendre compte de cette constatation dans la partie "Amélioration" (p.49) du chapitre pour ajouter une couche de personnalisation à notre tutoriel.

5.2.4 Aptitude d'une personne - Factor Score

Pour la mise en place de notre tutoriel, nous retiendrons uniquement les questions faciles et difficiles non devinables. Grâce au paramètre de difficulté établi par notre modèle, il suffit de trier les questions selon leur difficulté et nous pouvons établir une liste ordonnée selon la difficulté des questions qui serviront de tutoriel.

Il ne reste plus qu'à savoir où placer le candidat dans le tutoriel pour qu'il y trouve des questions adaptées à son niveau et puisse progresser.

Pour ce faire, nous devons **calculer l'aptitude d'un candidat, appelée Factor Score**.

Comme pour calculer les paramètres a,b et c de chaque question, Item Response Theory utilise la technique du maximum de vraisemblance pour estimer l'aptitude d'un candidat.

$$\hat{\theta}_{s+1} = \hat{\theta}_s + \frac{\sum_{i=1}^{N} -a_i[u_i - P_i(\hat{\theta}_s)]}{\sum_{i=1}^{N} a_i^2 P_i(\hat{\theta}_s) Q_i(\hat{\theta}_s)}$$

avec $\hat{\theta}_s$: aptitude estimée à l'itération s
a_i : discriminant de la question i
u_i : vaut 1 si bonne réponse à la question i sinon vaut 0
$P_i(\hat{\theta}_s)$: probabilité de bien répondre à la question i, pour une aptitude $\hat{\theta}$ à l'itération s
$Q_i(\hat{\theta}_s)$: $1 - P_i(\hat{\theta}_s)$

Cette technique itérative a besoin des informations(paramètres a,b et c) des questions auxquelles le candidat a répondu.
Initialement, on donne une valeur arbitraire à l'aptitude ($\hat{\theta}_s = 1$). On calcule la probabilité de bien répondre à la question pour chacune des N questions à cette aptitude. On peut ainsi évaluer le deuxième terme de la partie droite de l'équation, qu'on nomme "terme d'ajustement". Pour chaque nouveau recalcul de $\hat{\theta}_{s+1}$, ce terme devient de moins en moins significatif parce que l'estimation de l'aptitude devient de plus en plus précise. Quand la différence entre $\hat{\theta}_s$ et $\hat{\theta}_{s+1}$ devient insignificative, l'aptitude recherchée devient l'estimation calculée.

Le package LTM[9] permet de récupérer l'aptitude d'un candidat. La figure 30 montre les questions du challenge ITStudent (courbes en forme de S). Dessus nous avons tracé une droite verticale en $\theta = -0.33$ qui est l'aptitude calculée par IRT pour un candidat au hasard.

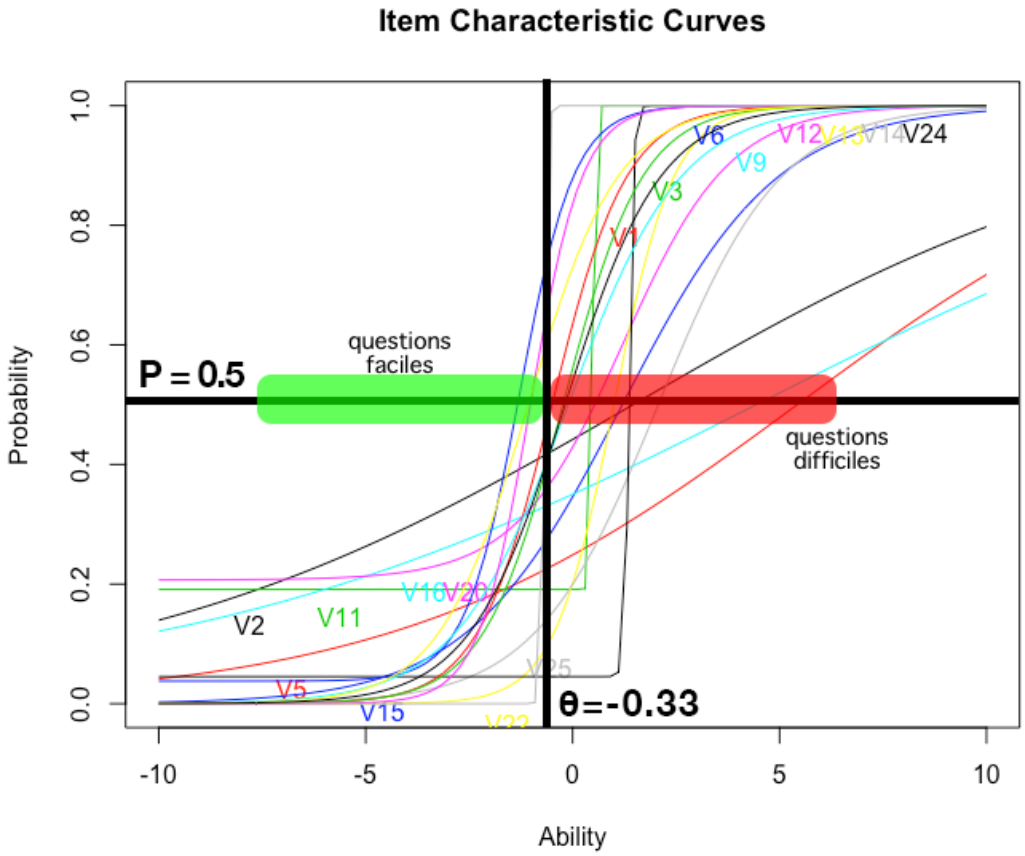


FIGURE 30 – Ségrégation des questions entre faciles et difficiles pour une aptitude de -0.33

Étant donné que lorsque $P(\theta) = 0.5$ (droite horizontale), le candidat a une chance sur deux de bien répondre à la question. On peut ainsi subdiviser les questions en deux catégories :

— Questions faciles : Courbes intersectant $P = 0.5$ et à **gauche** de $\theta = -0.33$

— Questions difficiles : Courbes intersectant $P = 0.5$ et à **droite** de $\theta = -0.33$

5.2.5 Dashboard

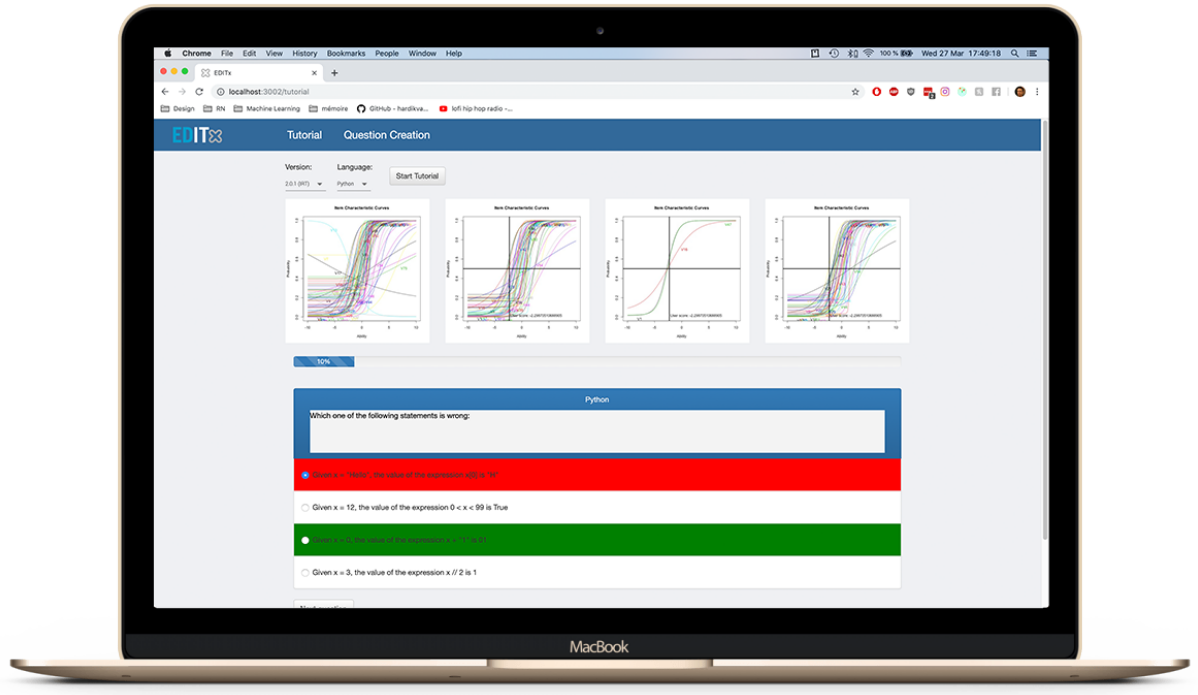


FIGURE 31 – Tutorial Dashboard

Le dashboard tutoriel se présente comme sur la figure 31. L'utilisateur connecté choisit le langage de programmation dans lequel il voudrait progresser, ensuite nos algorithmes analysent les questions, éliminent les devinables et problématiques puis calcule l'aptitude du candidat afin de lui proposer des questions adaptées à son niveau de manière à le faire progresser peu à peu. Le tutoriel est constitué d'une série de 20 questions, 5 faciles et 15 difficiles choisies au hasard dans le set de questions faciles et difficiles établi à la page précédente tenant compte de l'aptitude de l'utilisateur. Après avoir terminé le tutoriel, l'aptitude de l'utilisateur est dynamiquement mise à jour en fonction des réponses aux questions du tutoriel. L'utilisateur pourra ensuite refaire le tutoriel autant de fois qu'il veut dans le but d'augmenter son aptitude et rencontrer des questions de plus en plus compliquées.

5.2.6 Amélioration

Pour d'avantage personnaliser le tutoriel, nous passons par une autre phase de préprocessing des données. En effet, comme vu ci-dessus, l'âge joue un rôle important dans la réussite de certaines questions et cela influence considérablement la difficulté retenue par notre modèle. Pour faire face à ce problème, nous filtrons tous les candidats du fichier CSV dont la différence d'âge est supérieure à 5 ans par rapport à la personne qui suit le tutoriel.

$\rightarrow$ Comme prévu, la question difficile mentionnée plus haut au concernant la sécurité informatique est passée d'une difficulté de $b = 1.22$ pour un candidat de 18 ans à une difficulté de $b = -0.43$ pour un candidat de 26 ans.

Il faut tout de même faire attention à ne pas standardiser cette observation que les personnes âgées ont d'office un avantage par rapport aux jeunes par leur expérience, car l'informatique évolue vite et certains jeunes sont peut-être plus au courant des dernières technologies.

5.2.7 Item Information Curve (IIC)

Il est tout à fait possible qu'un nouvel utilisateur se connecte sur le site pour apprendre ou entraîner un langage de programmation. Cet utilisateur n'a encore répondu à aucune question et il est donc impossible pour notre modèle de calculer son aptitude par le factor score. Faut-il lui donner une aptitude (par défaut / catégorique) de 0 et puis l'adapter au fur et à mesure ? Cela pourrait être une façon de le faire, mais grâce à IRT, il existe une manière beaucoup plus efficace pour estimer rapidement l'aptitude d'un utilisateur.

IRT nous permet de tracer, pour chaque question, une courbe appelée Item Information Curve. Dans le domaine des statistiques, *l'information* est définie comme étant la précision avec laquelle on peut estimer un paramètre. Ainsi, si on sait déterminer un paramètre avec une grande précision, on a une meilleure information sur la valeur de ce paramètre.

Dans notre cas concrèt, plus cette information est grande à une certaine aptitude θ, mieux elle pourra déterminer si l'aptitude d'un utilisateur correspond à cette valeur ou pas.

$$I(\theta) = a^2 P(\theta) Q(\theta)$$

avec $P(\theta) = \frac{1}{1+e^{-a(\theta-b)}}$
$Q(\theta) = 1 - P(\theta)$
$\theta; a$ = l'aptitude; le discriminant

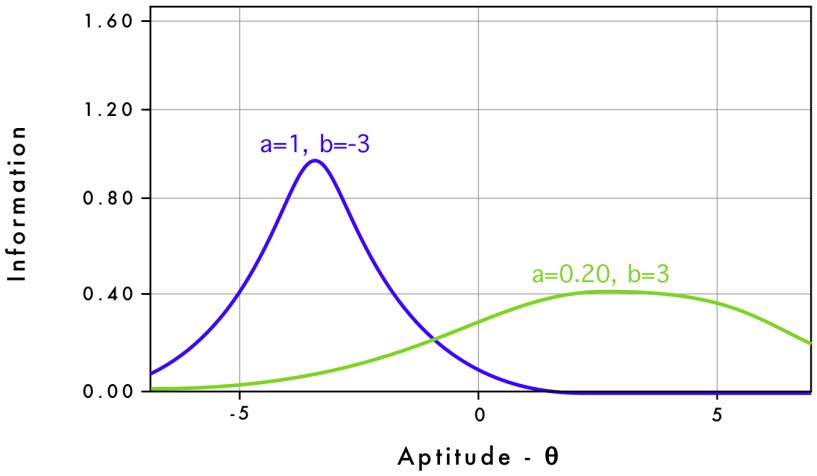


FIGURE 32 – Information de deux questions en fonction de l'aptitude

La figure 32 montre l'information de deux questions. On voit que les courbes sont maximales quand $\theta = b$.
La courbe bleue est une très bonne question pour déterminer si un utilisateur a une aptitude qui vaut environ -3, car l'information est maximale et grande en $\theta = -3$. Cependant, elle sera moins efficace que la question verte pour évaluer les aptitudes entre 0 et 5. L'avantage de la courbe verte, qui est plus étendue, par rapport à la courbe bleue est qu'elle permet de tester un plus grand intervalle d'aptitude, mais avec une moins grande précision.

• Le but de ce chapitre est trouver rapidement et précisément l'aptitude d'un utilisateur.
Pour ce faire, nous avons tracé les IIC de toutes les questions du topic ITStudent(culture générale). Parmi ces 25 questions, nous allons en choisir 5 avec beaucoup d'information et dispersées sur tout le spectre des aptitudes.

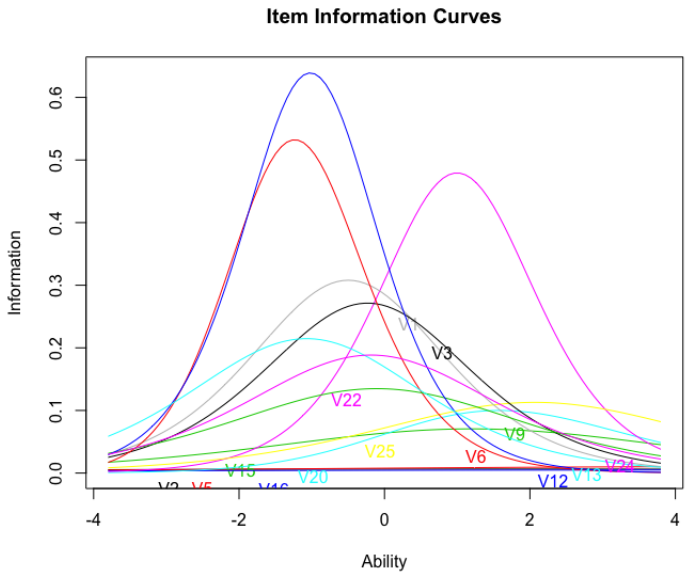


FIGURE 33 – IIC's ITStudent questions

On constate que les questions V6 et V12 forment une grande cloche à une aptitude d'environ -1.75. V22 permet de bien distinguer les utilisateurs d'une aptitude proche de 1.2. V1 et V3 ont une information raisonnable autour de 0.

Les questions V1, V3, V6, V12 et V22 forment un bon set de questions à poser à un nouvel utilisateur pour rapidement déterminer son niveau d'aptitude. Vous pouvez retrouver ces questions en annexe (page 62).

5.2.8 Validation

Pour examiner si notre modèle fonctionne comme prévu, c'est-à-dire s'il détermine correctement l'aptitude d'un utilisateur ainsi que les difficultés des questions, nous l'avons testé sur cinq candidats. Chaque candidat a dû répondre à 10 questions faciles et 10 questions difficiles, choisies par notre modèle en fonction de l'aptitude du candidat.
Voici les résultats obtenus pour deux de ces candidats, avec à chaque fois à gauche les questions faciles et à droite les questions difficiles. Si le candidat a bien répondu à une question, la couleur de la boule correspondant à cette question est verte, sinon elle est rouge.

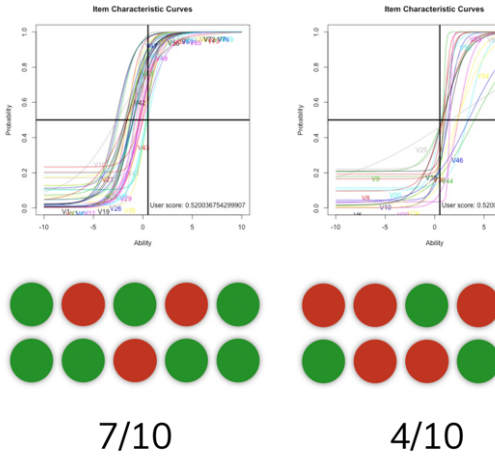


FIGURE 34 – Taux de réussite aux questions faciles et difficiles pour un candidat d'aptitude de 0.52

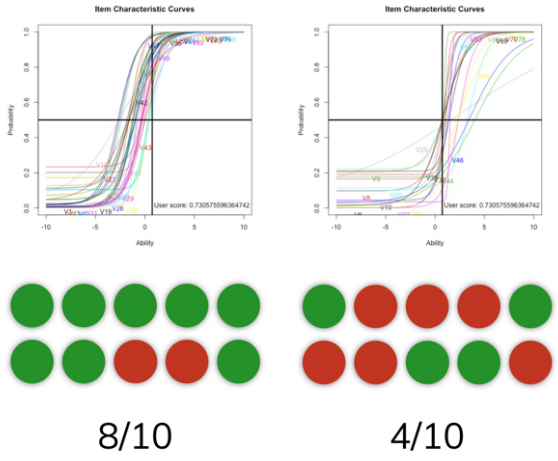


FIGURE 35 – Taux de réussite aux questions faciles et difficiles pour un candidat d'aptitude de 0.73

→ En ce qui concerne le taux de réussite, nous avons observé que les candidats répondaient en moyenne bien aux questions dont la difficulté était inférieure à l'aptitude ($b < \theta$) et inversement.

→ D'après les candidats testeurs, les questions difficiles semblaient abordables. Il s'agissait souvent de concepts qu'ils avaient appris mais oubliés. Le tutoriel était "stimulant". C'est ce que nous essayons effectivement d'atteindre : En proposant des questions adaptées au niveau d'aptitude des utilisateurs, le tutoriel a tout son sens et leur permet de progresser de manière ludique.

Ces candidats nous ont permis de valider une première fois le bon fonctionnement du tutoriel. Comme attendu, le tutoriel propose des questions dans le but de faire progresser les candidats et cela de manière agréable et précise. Dans une prochaine étape nous aimerions pouvoir le faire tester en production et le lancer officiellement sur le site d'EDITx.

6 Leaderboard et Badge utilisateur

L'IRT ne trouve pas seulement son utilité pour la mise en place d'un tutoriel d'apprentissage de langage informatique. Dans ce chapitre nous verrons qu'il peut aussi servir à améliorer des processus existants (Classement de concours, enquêtes, détermination de profils de consommation, ...) ainsi qu'à générer des badges de compétences sur les utilisateurs de la plateforme.

Réorganisation du classement des concours

À ce jour, les concours d'EDITx se basent sur la méthode d'évaluation classique dichotomique. La personne qui a le plus de bonnes réponses et le moins de mauvaises réponses sera première du concours. Or, avec IRT, nous disposons maintenant d'une méthode plus efficace pour calculer les scores des participants.
Nous avons donc repassé toutes les données du concours ITStudent et réarrangé le classement selon les aptitudes des candidats calculées avec le factor score (chapitre 5.2.4)

1	Tijl	Vandevyvere	▲ 11
2	Ayoub	Rossi	▲ 3
3	Saïkou	Barry	▲ 7
4	Dhoulkifli	Belataris	▲ 22
5	Yoan	Fath	▲ 17
6	Hamza	Mahmoudi	▲ 20
7	Pierre	Vandenhove	▼ 1
8	Mohamed	Reddahi	▼ 5
9	Konrad	Polak	▼ 7
10	Jordy	Cns	▲ 4
11	Ruben	Vermeulen	▲ 14
12	Salim	Boutlendj	▲ 19
13	Fedor	Vitkovskiy	▼ 2
14	M	V	▲ 13
15	Zakariya	Kassabeh	▲ 15
16	Andre	Jacobs	▲ 10
17	Tom	De Wulf	◀ 0
18	Jari	Van Melckebeke	▼ 1
19	William	Nagels	▲ 7
20	Alice	Goessens	▼ 5

FIGURE 36 – Classement du concours ITStudent par IRT

Le nouveau classement a bien bougé. La colonne de droite indique de combien de places le candidat a avancé ou reculé dans le classement. Un candidat a par exemple avancé de 22 places ! Les données du concours montrent effectivement que ce candidat est le seul à bien avoir répondu à une certaine question, mais un peu moins bien aux autres. Cette question donc difficile lui aurait donné, selon IRT, plus d'un point.
L'avantage d'avoir un leaderboard classé selon l'aptitude est qu'avec cette technique on mesure la compétence réelle d'un candidat et pas seulement un score basé sur le nombre de questions bien répondues sans tenir compte des différents niveaux de difficulté des questions.

L'inconvénient de la technique IRT est que c'est une méthode complexe. Lors de concours, il faut souvent une méthode transparente pour l'évaluation du classement. La technique de +1 pour une bonne réponse et -1 pour une mauvaise réponse s'y prête bien et le score est facilement recalculé si un candidat veut des justifications. Avec IRT, il sera plus compliqué d'expliquer clairement aux candidats comment leur score a été évalué.

Badges de compétences

Quelque chose qui n'avait pas encore été fait chez EDITx est l'attribution de "badges de compétences" à ses utilisateurs. Avec IRT on peut désormais attribuer un score précis à un utilisateur. Les cas d'utilisation de ce genre de badges sont variés. Un de ces cas sera présenté dans le prochain chapitre.
Voici, selon IRT, les trois meilleurs programmeurs en Python, avec un score relatif à un participant fictif ayant répondu correctement à toutes les questions sans exception. Attention à ne pas intégrer cette personne fictive dans le reste des calculs, cela fausserait les données.

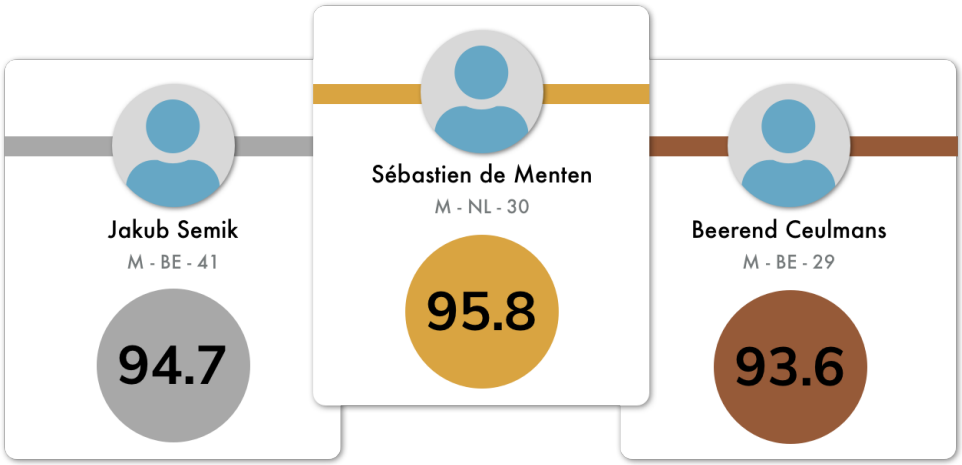


FIGURE 37 – Badges de compétence des trois meilleurs programmeurs en Python

	1	1	1	1	1	1			1	1	1	0			1			1	1			1	1	0	1	1		1	1			1		1	1
	1	1	1	1	1	1	1	1	1	1	1	0	1				1	1	0	1	1	1			1	1	1	1		0			1		1
1	1	1	1	1	1	1	1		1	1	1				1	0				1		1	1	0	1	1	1	1	1	0	0	1	1	1	

FIGURE 38 – Réponses aux questions(colonnes) par les trois utilisateurs(rangées)

La figure 38 montre une nouvelle fois que le meilleur score n'est pas obtenu par la personne avec le plus de bonnes réponses mais bien par celle qui obtient le factor score calculé par IRT le plus élevé.

On pourrait se dire qu'on peut obtenir un résultat similaire en pondérant les questions selon leur difficulté (facile=0.5, medium=1.0, difficile=1.5) mais on retomberait sur le problème initial, à savoir que la difficulté est une mesure relative.
Là réside le point fort de IRT, qui évalue la difficulté des questions par après, en fonction des résultats des participants et peut ainsi attribuer un factor score unique à chaque candidat.

7 Opportunités futures pour EDITx

En plus des résultats obtenus dans le cadre de ce travail de fin d'études qui vont permettre à EDITx d'avancer dans sa quête et de proposer de nouveaux outils à ses utilisateurs, nous avons créé de nouvelles opportunités pour EDITx.

• EDITx pourrait exploiter les badges de compétence que nous pouvons désormais générer pour chaque utilisateur en fonction d'un langage de programmation. Ces badges donnent une information précise du niveau d'un utilisateur et il est tout à fait envisageable de partager cette information avec des clients potentiels (recruteurs, entreprises, ...) ou d'étendre les questions pour des études de profils de consommation.

• EDITx pourrait également mettre en place un service d'aide au recrutement. Grâce aux Item Information Curves des questions, on pourrait choisir une série de 20 questions à haute information et recouvrant tout le spectre de l'aptitude pour mettre en place un Quiz que les recruteurs feraient passer à leurs candidats pour avoir une évaluation précise de leur niveau de compétence non seulement dans le secteur IT, mais aussi dans d'autres domaines.

• Finalement, il est évident que notre outil d'aide à la détection de doublons que nous avons développé peut être aussi utilisé pour supprimer les doublons qui étaient déjà présents dans la base de données avant son instauration.

8 Conclusion

Dans ce travail de fin d'études nous vous avons présenté la mise en place d'un tutoriel intelligent ainsi qu'un dashboard d'aide à la création de nouvelles questions de concours. Le rapport a décrit en détail les démarches de recherche, conception, implémentation et amélioration à suivre pour aboutir à un résultat fonctionnel et optimisé.

Objectifs primaires

- ✓ Outils d'aide à la création de question
  - ✓ Classification des questions
  - ✓ Estimation de la difficulté des questions
  - ✓ Détection de questions doublons
- ✓ Tutoriel d'apprentissage
  - ✓ V1 - Naïf
  - ✓ V2 - Item Response Theory

Objectifs secondaires

- ✓ Amélioration du classement des concours
- ✓ Badges de compétence
- ✓ Opportunités futures

Nous avons non seulement atteint les objectifs primaires définis dans notre cahier des charges initial, mais nous avons pu pousser notre travail un peu plus loin et ouvrir de nouvelles pistes et opportunités envisageables pour EDITx.

Dorénavant, EDITx pourra accueillir son public grandissant avec une nouvelle plateforme d'apprentissage de langage informatique. Les auteurs ne devront plus craindre de créer des questions déjà existantes, ou de mal définir le niveau de difficulté de leurs questions.

Quand la base de données franchira le cap des 5000 questions (le double de maintenant), il sera intéressant d'adapter certains algorithmes, en particulier les algorithmes de détection de doublons. Nous avons indiqué dans l'état de l'art qu'il existe d'autres algorithmes, comme Random Forest, qui sont très puissants et très efficaces dans la détection de questions semblables, mais nécessitent beaucoup plus de données pour s'entraîner correctement.

De nos jours, ce serait une erreur de la part des entreprises de ne pas se tourner vers le Big Data et d'exploiter leurs données. Comme l'a montré ce travail, par une analyse approfondie des données, on peut rapidement en ressortir des applications utiles. Toutes les grandes entreprises utilisent les outils de data analytics pour enrichir leur business.

Merci à EDITx de m'avoir donnée l'opportunité de travailler avec eux et d'utiliser leur données pour réaliser ce travail. J'ai ainsi eu l'occasion de bien progresser dans le domaine du data science et de me rendre compte de l'impact que le Big Data peut avoir dans une entreprise. Je pense que l'Item Response Theory a énormément de potentiel et qu'il faut sensibiliser les écoles à utiliser cette technique qui est bien plus efficace pour l'évaluation d'examens et plus juste envers les étudiants.

En fin de compte, ce travail de fin d'études nous a menés à faire preuve de créativité et d'initiative. Il nous a permis d'appliquer nos connaissances acquises à un projet de grande envergure et de démontrer notre esprit d'ingénieur pour fournir un travail de qualité du début à la fin.

A Annexes

A.1 IRT Graphs - Python

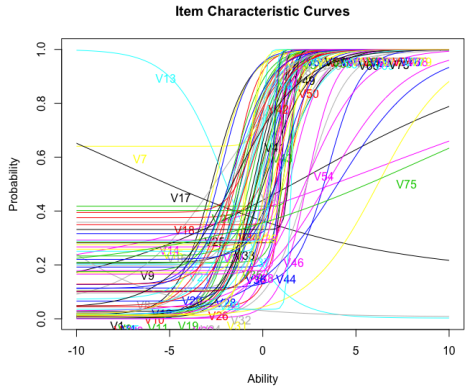


FIGURE 39 – Plot de toutes les questions (n=79)

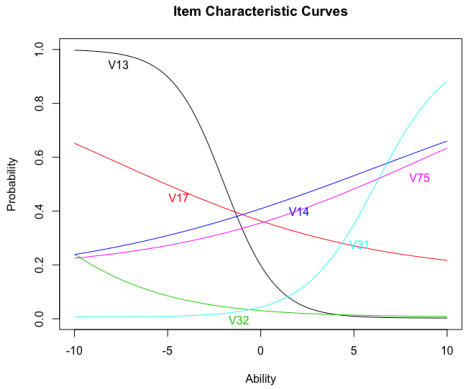


FIGURE 40 – Plot des questions à écarter (n=6)

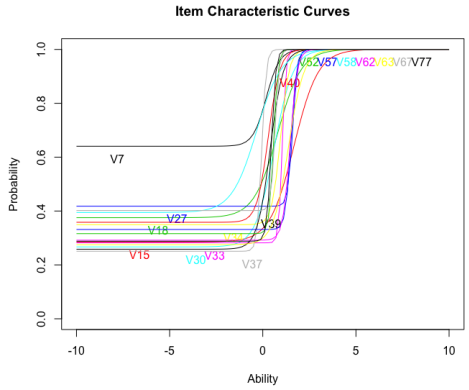


FIGURE 41 – Plot des questions devinables (n=17)

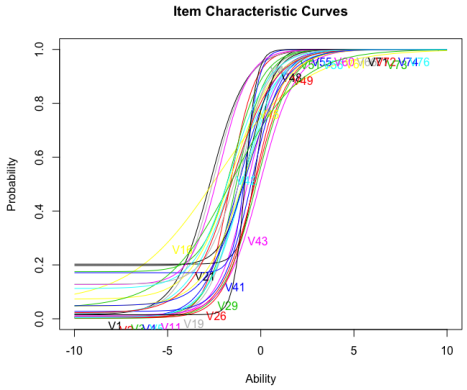


FIGURE 42 – Plot des questions faciles (n=29)

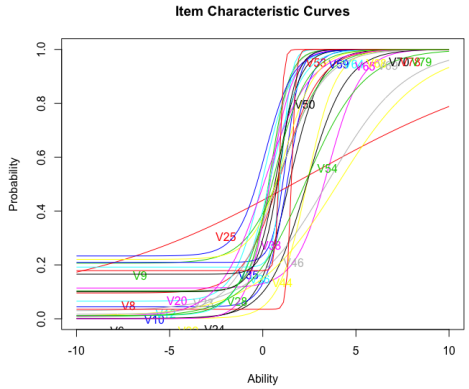


FIGURE 43 – Plot des questions difficiles (n=27)

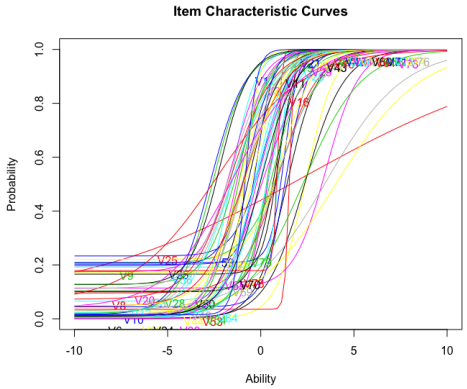


FIGURE 44 – Plot des questions faciles et difficiles (n=56)

A.2 IRT Graphs - .NET

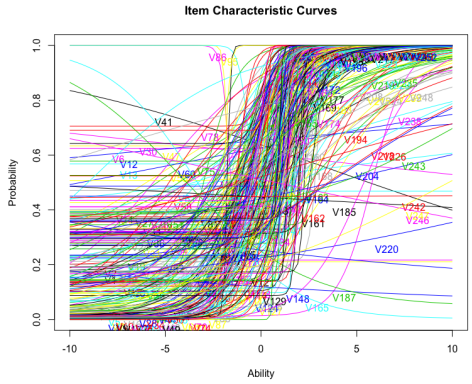


FIGURE 45 – Plot de toutes les questions (n=252)

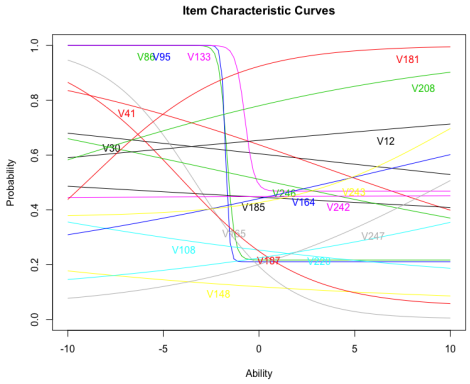


FIGURE 46 – Plot des questions à écarter (n=19)

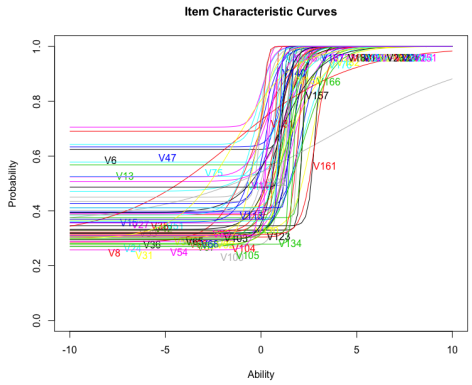


FIGURE 47 – Plot des questions devinables (n=62)

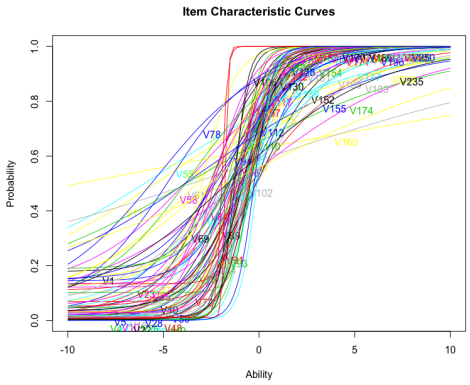


FIGURE 48 – Plot des questions faciles (n=84)

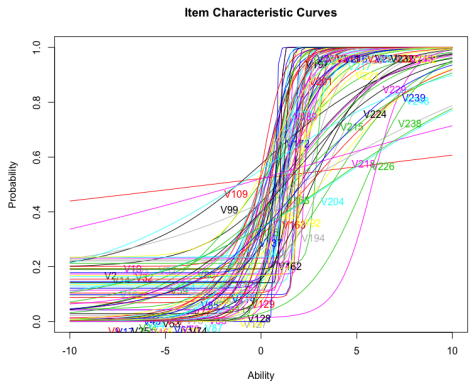


FIGURE 49 – Plot des questions difficiles (n=87)

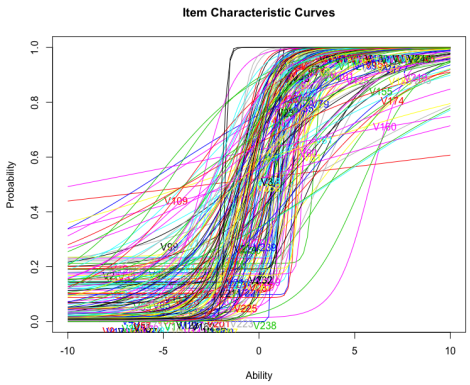


FIGURE 50 – Plot des questions faciles et difficiles (n=171)

A.3 Exemples de questions pour rapidement calculer le niveau d'aptitude d'un utilisateur

— **V1** : What does the D mean in the ACID acronym for distributed systems ?

•1) Durability

2) Distributed

3) Decoupling

4) Discoverability

— **V3** : What does XSS refer to ?

•1) A vulnerability that allows an attacker to inject code in a web page so that it gets executed on the client side

2) A compression algorithm specifically designed for XML files

3) DA project management methodology, inspired by XP

4) The codename of the future version 4 of CSS

— **V6** : What programming paradigm are you using when you write Map/Reduce algorithm for Big Data ecosystems ?

1) Event-Driven programming

•2) Functional programming

3) Object-Oriented programming

4) Imperative programming

— **V12** : Regarding distributed databases, which assertion is not true about sharding ?

1) it increases the performance of the database

2) it is a way to horizontally scale the database

3) it can be combined with data replication

•4) it adds redundancy to be more robust w.r.t. node crash

— **V22** : What can SHA-1 be used for ?

•1) to ensure data integrity

2) to ensure data confidentiality

3) to ensure data availability

4) none of the above

Table des figures

B Glossaire

aptitude Niveau de compétence d'une personne. 12, 13, 37–43, 45, 46, 48–50, 52–55, 57, 64, 65

DB Data Base; base de données. 11

devinabilité Le fait qu'une question soit devinable. 13, 41, 42, 45, 64

IIC Item Information Curve. 53

IRT Item Response Theory. 3, 13, 37, 42, 45, 49, 52, 55–57, 65

K-Means Algorithme de clustering qui permet de réaliser des analyses non supervisées. 9, 10

KNN K-Nearest Neigbors. 10

n-gram Sous-séquence de n éléments construite à partir d'une séquence donnée. 9

R Langage de programmation largement répandu en data science. 13

Random Forest Algorithme effectuant un apprentissage sur de multiples arbres de décision entraînés sur des sous-ensembles de données légèrement différents. 11

Tensorflow Librairie open-source pour la modélisation de modèles de machine learning développée par Google. 3

C Références

[1] Yew Choong Chew, Yoshiki Mikami, Robin Lee Nagano, *Language Identification of Web Pages Based on Improved N-gram Algorithm*, Nagaoka University of Technology, 2011.

[2] Clive Souter, Gavin Churcher, Judith Hayes, John Hughes, Stephen Johnson, *Natural Language Identification using Corpus Based Models*, Nagaoka University of Technology, 1994.

[3] M R Prathima, H R Divakar *Automatic Extractive Text Summarization Using K-Means Clustering*, PES College of Engineering, Mandya, Karnataka, India, 2018.

[4] Quan Wang, Jing Liu, Bin Wang, Li Guo *A Regularized Competition Model for Question Difficulty Estimation in Community Question Answering Services*, Harbin Institute of Technology, 2014.

[5] Lei Guo, Chong Li, Bin Wang, Haiming Tian *Duplicate Quora Questions Detection*, New York University, 2017.

[6] B. Martins, H. Galhardas, N. Goncalves *Using Random Forest classifiers to detect duplicate gazetteer records*, Univ. of Lisbon, 2012.

[7] Dong Liu and Chenghua Lin, *Sherlock : a Semi-Automatic Quiz Generation System using Linked Data*, University of Aberdeen, 2015.

[8] Xinming An and Yiu-Fai Yung, *Item Response Theory : What It Is and How You Can Use the IRT Procedure to Apply It*, SAS Institute Inc., 2014.

[9] Dimitris Rizopoulos, *ltm : An R Package for Latent Variable Modeling and Item Response Theory Analyses*, Université Catholique de Louvain, 2006.

[10] Frank B. Baker, *The basics of Item Response Theory*, University of Wisconsin, 2001.

[11] Justyna Brzezińska, *Item Response Theory models in the measurement theory with the use of ltm package in R*, University of Economics in Katowice, 2018.

[12] Nadia Kasto and Jacqueline Whalley, *Measuring the difficulty of code comprehension tasks using software metrics*, AUT University, 2013.

[13] Fernando Martinez-Plumed and Ricardo B. C. Prudencio, *Making Sense of Item Response Theory in Machine Learning*, AUT University, 2016 .

[14] Sarah L. Thomas, Shelby, NC, *Combining Rasch and Item Response Theory Model Estimates with Support Vector Machines to Detect Test Fraud*, University of North Carolina, 2016 .

[15] Neo4j Documentation, *https ://neo4j.com/docs/*