Document De Compétences

(format ESN)

Téo Orthlieb - Data Engineer

Téo Orthlieb

TOR - Data Engineer

~3 ans d'expérience

ENVIRONNEMENT TECHNIQUE

OS: Windows, Linux

Langages: Python, Java, C#, C/C++, SQL, Rust, TypeScript, JavaScript, HTML/CSS

Frameworks: PyTorch, TensorFlow, Jest, Vue, React, Sequelize

Outils: Git, Docker, SonarQube, Redmine

COMPÉTENCES LINGUISTIQUES

Français: Natif • Anglais: Courant

FORMATION

2022 • Master Informatique Opt. Machine Learning • UdeM

2018 • Licence Informatique Opt. Jeux-Vidéo • UQAC

2017 • DUT Informatique • UGA IUT2

2015 • Bac Scientifique Spé. SI • Lycée Arago

QUALITÉS

Communication • Attention au détail • Initiative

EXPÉRIENCES PROFESSIONNELLE

Développeur full stack — Solutec — 08/2022 - 05/2023

Stack: TypeScript • SQL • Docker • Node.js • Sequelize • Express • Vue • Jest • Python

Contexte

J'ai repris une application interne en cours de développement qui fait interface avec GitLab pour avoir plus de contrôle sur ce que voit le client et leur permettre d'interagir avec GitLab sans avoir de licence.

Réalisations

Maintenance et développement du back en Node.js TypeScript

- Calcul des heures ouvrées depuis création du ticket pour SLA (avec week-end, jours fériés, pauses)
- Refonte de la gestion de pièces jointes (stockage sur GCP), pour qu'elles soient intégrées aux commentaires et affichées sur GitLab
- Refacto de la gestion d'erreur (- de code + cohérence)
- Maintenance des tests unitaires Jest

Maintenance et développement du Front en Vue TypeScript

- Refonte des filtres à tickets (ergonomie)
- Reprise du DashBoard pour afficher les statistiques des projets
- Refonte des commentaires pour ajouter des pièces jointes intégrées
- Options de paramétrages des temps du SLA stockés avec Sequelize

Scripts Python pour pouvoir manipuler les tickets en batch sur GitLab avec des règles sur les labels

Documentation d'usage pour les utilisateurs et pour les développeurs sur les systèmes non triviaux, expliquant les choix effectués (ex: gestion de pièces jointes)

Data Engineer — BMU — 09/2018 - 07/2019

Stack: Python • NumPy • Matplotlib • Kivy • JavaScript

Contexte

Beam Me Up évalue des contenus commerciaux audiovisuels clients avec des casques EEG qui permettent de lire des états mentaux (confusion, calme, agacement, satisfaction, etc.) du public. Les expériences sont réalisées dans les locaux avec un panel varié de participants. Mes missions étaient d'étendre le service à la navigation de site web clients et d'enrichir les rapports d'analyse.

Réalisations

Extension du service aux sites web

- Add-on navigateur pour récolter les données de navigation (JavaScript)
- Détection des étapes de navigation et labellisation automatique de la capture vidéo de la navigation
- Script Python pour synchroniser les données de navigations aux données EEG

Production automatique de figures pour le rapport (Python, NumPy, SciPy)

- Sankey Diagrams pour visualiser le flot des utilisateurs sur le site
- Heatmaps des états mentaux moyens divisés par étapes de navigation
- Box plots des temps passés sur les étapes de navigation

Portage de l'application EEG de Python 2.7 à Python 3.7 (interface Kivy)

- Réduction du code par un facteur x10
- Modularisation du code en plusieurs API POO
- Amélioration de l'ergonomie et de la vitesse du programme

Data Engineer (stage) — LIG — 04/2017 - 07/2017

Stack: C • HTML/CSS • JavaScript • jQuery • SQL • PhP

Contexte

Dans le cadre d'un projet académique sur un moteur de recherche expérimental, j'ai réalisé un *parser* pour nourrir ce moteur avec les données de Wikipedia, ainsi qu'une page web pour tester et évaluer le moteur.

Réalisations

Parser multi-thread en C capable de convertir l'entièreté du Wikipédia anglais (~60 Go de texte) en données d'entraînement en 1 min 30. Le parser fait usage du pattern Producer Consumer avec 2 buffers pour que les threads soient toujours actifs.

Page Web (HTML/CSS JS) pour interagir avec le moteur de recherche, recevoir des suggestions de mots clés, prévisualiser les résultats.

Système pour recueillir les statistiques d'utilisations (PHP, SQL) et les visualiser avec un dashboard (HTML/CSS JS)

MÉMOIRE DE MASTER

Stack: Python • PyTorch • HuggingFace • NumPy • SciPy • Matplotlib

Contexte

Pour conclure mon master en Informatique option Machine Learning à l'Université de Montréal, j'ai réalisé en 2021-2022 un mémoire sur le Fact Checking automatique disponible ici: https://papyrus.bib.umontreal.ca/xmlui/handle/1866/27060

Réalisations

Bibliographie, résumé de l'état de l'art appuyé sur ~30 articles et catégorisation en 3 types de modèles.

Modèle sans connaissances externes

- Modèle de Langue Transformer (architecture Bert) fine-tuned sur un dataset de fake news
- Expériences et conclusion sur leur utilisation adaptée comme premiers modèles de tri pour classifier les fake news

Modèle avec Knowledge Graph (connaissance structurée)

- Réalisation d'un nouveau modèle utilisant le Transformer pour inférer des raisonnements de fact checking dans un graphe de connaissance (basé sur Wikidata)
- Expériences et conclusion sur leur utilisation adaptée comme modèle de débunkage de fake news, permettant de tracer des liens entre logique entre des entités

Modèle avec Knowledge Base (connaissance non structurée)

- Réalisation d'un modèle multitâche permettant de citer des passages de bases de connaissances (comme Wikipédia) pour débunker des fake news.
- Expériences et conclusion sur leur utilisation adaptée comme modèle de citations de sources pour donner des articles contraires aux fake news.

PROJETS PERSONNELS

J'ai réalisé beaucoup de projets personnels sur mon temps libre. Il y en a de toutes sortes (petites pages web, outils utiles, jeux, etc.) mais j'inclus ci-dessous une sélection pertinente au poste de Data Engineer.
Je vous invite à les consulter sur mon site web dans lequel j'écris aussi des billets de blog (en anglais) sur les problèmes que j'ai traités https://teo-orthlieb.github.io/

Bot Discord de nuage de mots

Stack: Python (puis réécrit en Rust)

Contexte

Discord est une plateforme de chat — similaire à Slack mais orientée vers le gaming — que j'utilise avec des amis. Comme pour Slack, il est possible de créer des Bots capables d'interagir avec les utilisateurs via l'API Discord. Par curiosité, j'ai réalisé un bot qui fait un nuage de mots pour chaque utilisateur du serveur.
Billet de blog: https://teo-orthlieb.github.io/blog/user-word-cloud/

Réalisations

Bot de nuage de mot

- Lit et tokenise les messages utilisateurs pour calculer leur vocabulaire
- Compare leur vocabulaire avec le vocabulaire global pour distinguer les mots spécifiques à chaque utilisateur
- Agence les mots dans une image pour faire le nuage de mots
- Statistiques sur les utilisations des émojis

Librairie Rust pour faire des images de nuage de mots (https://crates.io/crates/wordcloud-rs)

Outil d'assignation de postes

Stack: Python

Contexte

Un ami m'a fait part de la problématique suivante.
Au cours de leur cursus à l'Enjmin (école de jeux-vidéo) les étudiants doivent pitcher des idées de jeux à un jury et en réaliser le prototype en équipe. Cependant, les pitchs sont sélectionnés (seule une fraction de pitchs sont choisis) et il convient alors de répartir les élèves sur les projets pour les réaliser. Pour répondre au problème, j'ai réalisé un programme d'assignement des élèves sur les projets qui prend en compte les vœux des élèves et les contraintes des projets.
Code: https://github.com/Inspirateur/PitchAssignment

Réalisations

Programme d'assignement des élèves sur des pitchs (Python)

- Fonction de score flexible qui prend une solution en entrée (une répartition des élèves) et renvoie un score en fonction de la satisfaction des vœux et des contraintes du projet
- Algorithme évolutif qui commence avec des solutions aléatoires puis sélectionne les solutions qui ont obtenu le meilleur score pour les modifier
- Paramètres permettant de privilégier plus ou moins les vœux des élèves contre les contraintes des projets

Outil de renommage en batch

Stack: Python (puis réécrit en Rust)

Contexte

Je rencontre souvent le besoin de renommer un batch de fichiers qui ont le même format de nom, et les outils existant (à base de RegExp) ne me satisfaisaient pas. J'ai donc réalisé sur mon temps libre un renommeur de fichier intelligent avec lequel il suffit de renommer 1 seul fichier pour que les autres du même format soient automatiquement renommés.

Code avec démo: https://github.com/Inspirateur/SimpleRenamer

Réalisations

Renommeur Intelligent (Python, puis Rust)

- Fonction qui scanne le dossier pour trouver les fichiers au format similaire à celui sélectionné et déduire les parties variables.
- Interface qui présente ces fichiers et un champ texte pour renommer le fichier sélectionné.
- Fonction qui identifie les parties qui ont changé dans le renommage du fichier sélectionné pour déduire le renommage à appliquer sur les autres fichiers
- Prévisualisation du renommage en temps réel (il est appliqué seulement après validation)