# Design Pattern
# Pacman

Noé De Caestecker

2 décembre 2025

## Table des matières

**Résumé**

Le but du projet est de créer un système de recherche documentaire permettant des recherches plus ou moins avancées sur un corpus de document fourni.

# 1 Choix d'implémentation

Le projet a été réalisé intégralement en python, en utilisant le moins de librairies externes possibles (librairie graphique, gestion de fichiers et regex seulement) afin de mettre en œuvre le plus de techniques par nous-même.

Le système de recherche permet l'utilisation d'un caractère joker $*$ et fourni les réponses en les ordonnant dans un ordre décroissant de pertinence.

# 2 Index

L'index suit une structure d'arbre 2-4 où chaque mot est stocké ainsi que ses rotations (pour l'utilisation du caractère joker). Chaque feuille de l'index est une structure qui représente un mot et stocke l'ensemble des poids de ce mot pour chaque document qui le contient au moins 1 fois.

Afin d'éviter les redondances dans les structures de poids, chaque rotation d'un mot ne stocke que le mot dont il provient au lieu de ses poids. Si une recherche tombe sur l'une de ces rotations, une seconde recherche dans l'arbre est réalisée pour trouver la bonne liste de poids. Cette implémentation rallonge le temps d'exploration, mais permet de réduire grandement la taille de l'index (la taille est divisée par 4 en moyenne). De plus, le temps d'exploration est négligeable pour des parcours sans caractère joker (ce qui est le cas pour ce parcours supplémentaire).

# 3 Calcul de la pertinence

La pertinence des documents est évaluée en utilisant une distance cosine. Chaque document peut être représenté par un vecteur de dimension $n$ égale au nombre de mots différents dans la base de documents (après passage dans le stemmer) où chaque coordonnée représente le poids du document pour ce mot.

Lors d'une requête, ces vecteurs sont construits en récupérant uniquement les coordonnées dont les mots sont présents dans la requête. Ces nouveaux vecteurs partiels sont ensuite normalisés en les divisant par la norme de leur vecteur complet (calculés lors de la création de l'index). Seuls les vecteurs des documents ayant un mot en commun avec la requête sont récupérés (pour simplifier les calculs). La requête est, elle aussi, transformée en vecteur en utilisant les mêmes poids que les documents et ce vecteur est aussi normalisé.

On donne ensuite à chaque document un poids égal au produit scalaire entre son vecteur et le vecteur de la requête.

## 4 Calcul et représentation des poids

Les poids associés à chaque document pour chaque mot sont stockés dans l'arbre de l'index. Chaque feuille de l'index représente un mot $m$ (ou une rotation de $m$) et stocke soit un pointeur vers la feuille de $m$ si la feuille est une rotation, soit un dictionnaire (hash map) qui associe chaque document $d$ le nombre d'occurrences de $m$ dans $d$. La structure en dictionnaire permet de faciliter la création de l'index (dans lequel chaque mot est ajouté 1 par 1) sans augmenter la complexité de la recherche puisque la création des vecteurs de document se fait en parcourant les paires mots-documents.

Une fois tous les mots ajoutés dans l'index et le nombre de documents connu, un passage est fait dans chaque feuille de l'arbre pour calculer l'*idf* du mot et stocker (avec le nombre d'occurrences) le poids de ce mot pour chaque document.

Ces poids sont calculés en multipliant $1 + \log_{10}(w_f)$ par l'*idf* où $w_f$ est le nombre d'occurrences du mot dans le document. L'*idf* est calculé comme $\log_{10}(\frac{d_c}{d_f})$ où $d_f$ est le nombre de documents contenant le mot et $d_c$ est le nombre de documents.

## 5 Correction des mots

Afin d'aider l'utilisateur dans sa recherche, une correction des mots est proposée (mais pas imposée).

Cette correction est guidée par un fichier texte contenant les mots de la langue anglaise (langage des documents). Ce fichier est chargé dans un dictionnaire (hash map) où chaque clef est un code Soundex et chaque valeur est une liste de mots correspondants au code. Ce dictionnaire est utilisé pour guider la correction des mots vers des mots se prononçant de façon similaire.

La liste de mots similaire est ensuite parcourue pour trouver le mot le plus proche en utilisant un système de trigramme et la mesure de Jaccard.

## 6 Stop list

La stop list fournie étant trop limitée (trop de mots utiles), nous avons décidé de créer notre propre stop list en utilisant les documents fournis. Nous avons décidé de retirer tous les mots dont l'*idf* était inférieur à 0.1 puisque ces mots ont peu d'importance dans les poids des documents finaux, mais sont liés à des dictionnaires de poids très grands (leur nombre de documents lié étant supérieur à 80% du nombre de documents total).

Cette stop list n'aide pas la vitesse de requête, mais permet de réduire un peu la taille de l'index.