UNIVERSITÉ MARIE & LOUIS PASTEUR

CMI Figure
CURSUS MASTER EN INGÉNIERIE

Complétion semi-automatique
Projet d'Initiation à la recherche

Samia Benali & Elouan BOITEUX

Année universitaire 2024 – 2025

CMI Informatique Deuxième Année

Référent : Pierre-Cyril HEAM

22 mai 2025

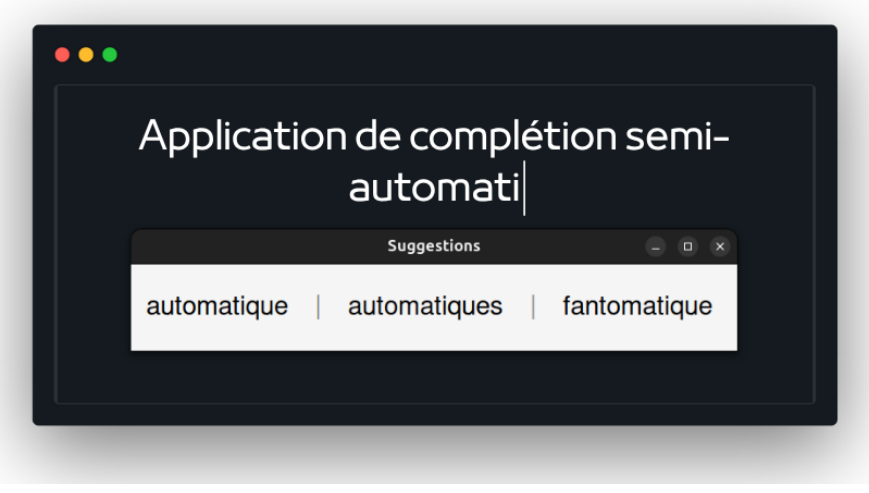
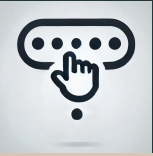
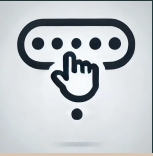


Table des matières



La complétion semi-automatique

Définition complétion semi-automatique

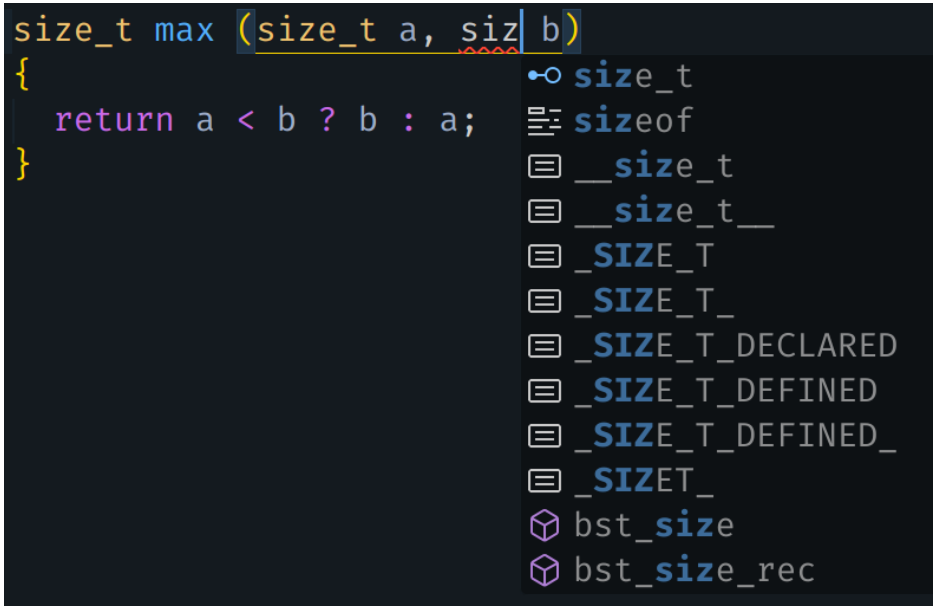


En quelques mots :

- Assistance
- Proposition des suggestions
- Choix final fait par l'utilisateur

Utilisé dans de nombreuses applications :

- Traitement de texte
- IDE
- Moteurs de recherche
- Assistance virtuelle

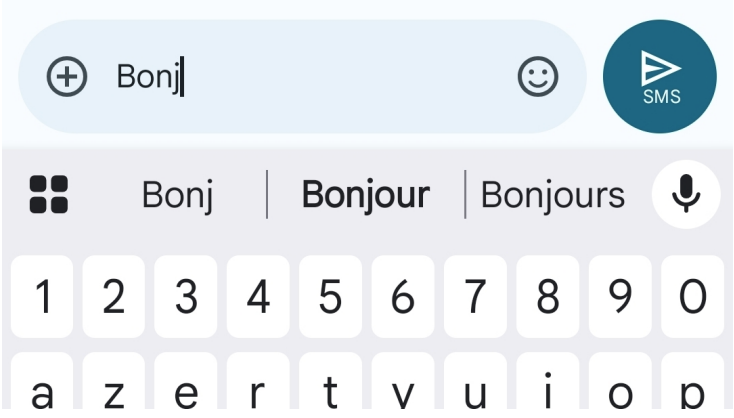


Exemple de complétion semi-automatique sur VS Code

Différence avec la complétion semi-automatique

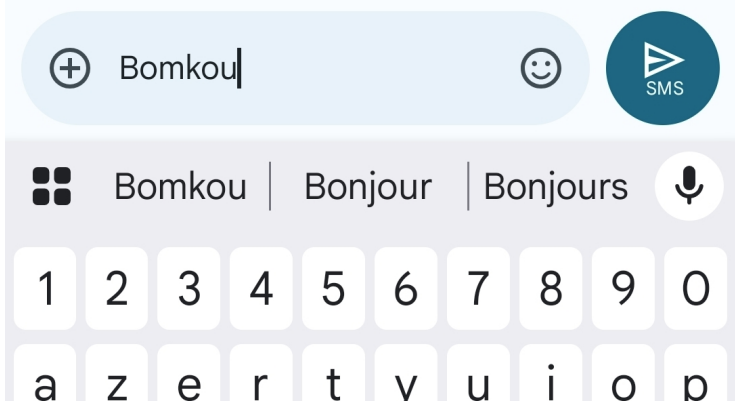
Automatique

- Pas besoin d'interaction manuelle
- Gain de temps maximal
- Peut générer des erreurs si le contexte est mal interprété



Semi-Automatique

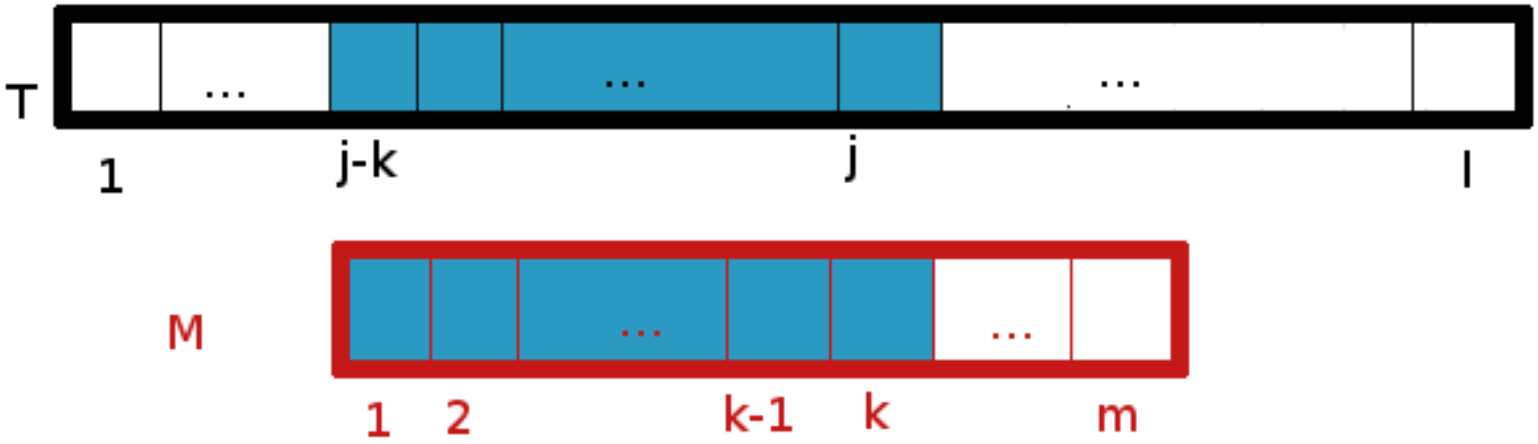
- Nécessite une validation manuelle
- Plus de contrôle pour l'utilisateur
- Moins d'erreurs



Les différentes approches utilisées

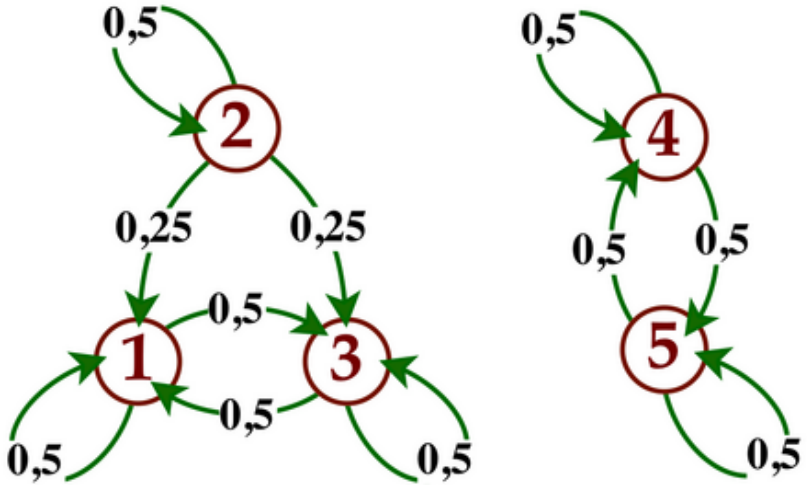
Modèle s'appuyant sur des règles

- Utilisation d'algorithmes simples s'appuyant sur des règles prédéfinies (correspondance des préfixes et/ou motifs)
- Gérer grâce à des dictionnaires statiques ou des listes.
- Avantages : Simplicité et rapidité de mise en œuvre
- Inconvénients : Rigidité, difficulté à gérer des cas complexes



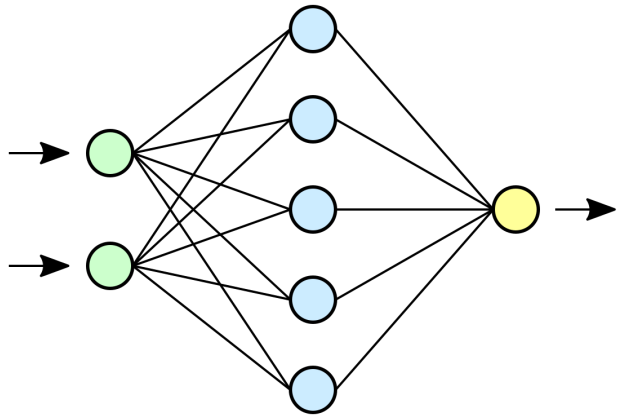
Modèle s'appuyant sur des statistiques

- Utilisation de statistiques fournies grâce aux données d'un historique
- Prédire des séquence (Markov, TF-IDF)
- Avantages : Résultats rapides et meilleure gestion des cas complexes.
- Inconvénients : Pas de compréhension sémantique et besoin d'un grand nombre de données.



Modèle s'appuyant sur l'intelligence artificielle

- Utilisation d'algorithmes s'appuyant sur l'intelligence artificielle et les réseaux neuronaux
- Apprentissage de motifs complexes à partir de données (forêt aléatoires, régressions pour plus de contexte)
- Avantages : Efficace face à des cas complexes et des demandes rares, adaptabilité
- Inconvénients : Nécessite beaucoup de temps de calcul et de ressources



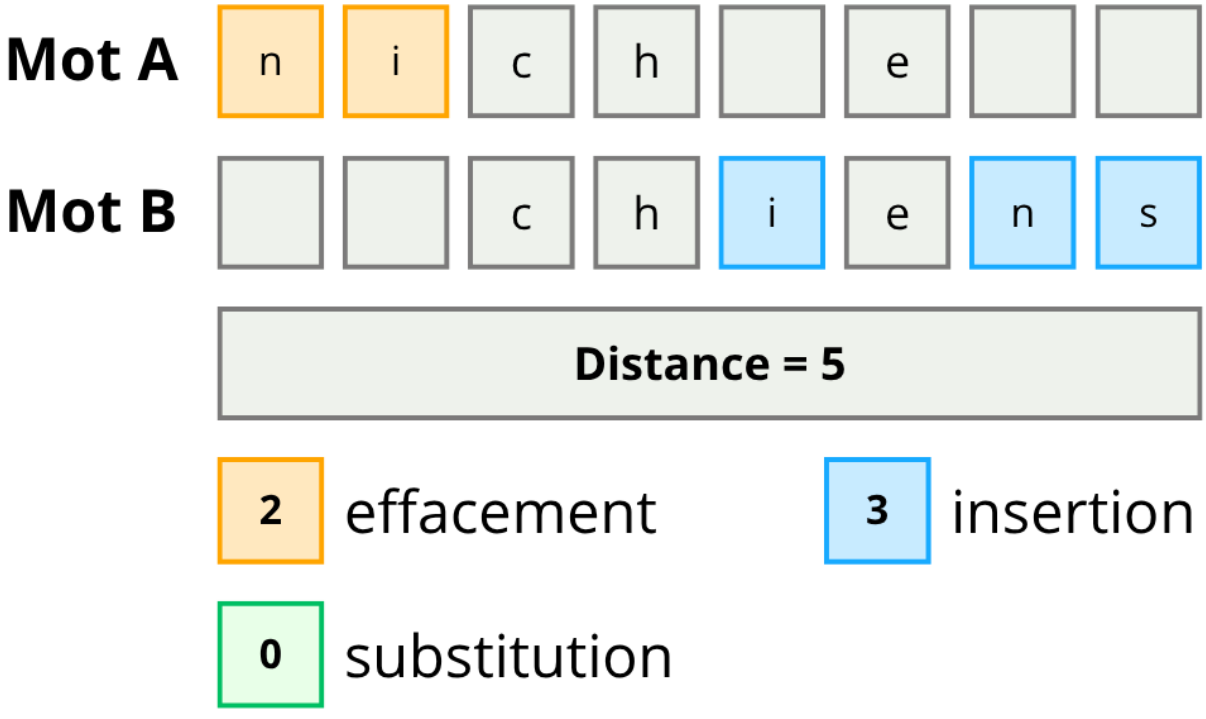
Modèle s'appuyant sur l'amélioration continue

- Utilisation d'algorithmes s'appuyant sur l'amélioration en temps réel.
- Choix fait par l'utilisateur, choix mémorisés pour une utilisation personnalisée
- Avantages : Implémentation adaptable et avec des suggestions qui ont un sens sémantique, performance très élevée.
- Inconvénients : Implémentation très complexes et longue à déployer, dépend des données collectées et des utilisateurs.

Les algorithmes de calcul de distance

Distance de Levenshtein

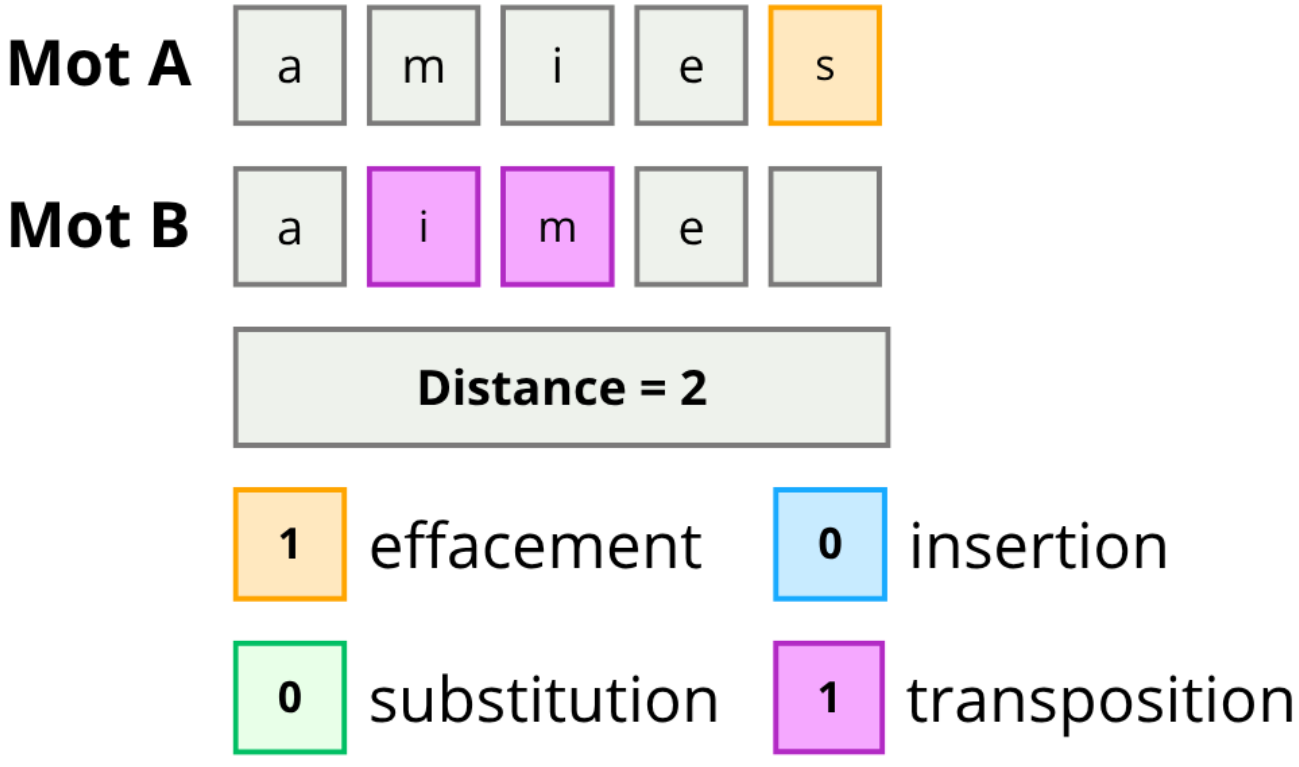
Objectif : Trouver le nombre minimal d'opérations nécessaires pour transformer une chaîne de caractères en une autre.



Complexité : $\mathcal{O}(n \times m)$ (où n et m sont les longueurs des chaînes de caractères)

Distance de Damerau-Levenshtein

Objectif : identique à la distance de Levenshtein, mais avec l'ajout de la possibilité d'échanger deux caractères adjacents.

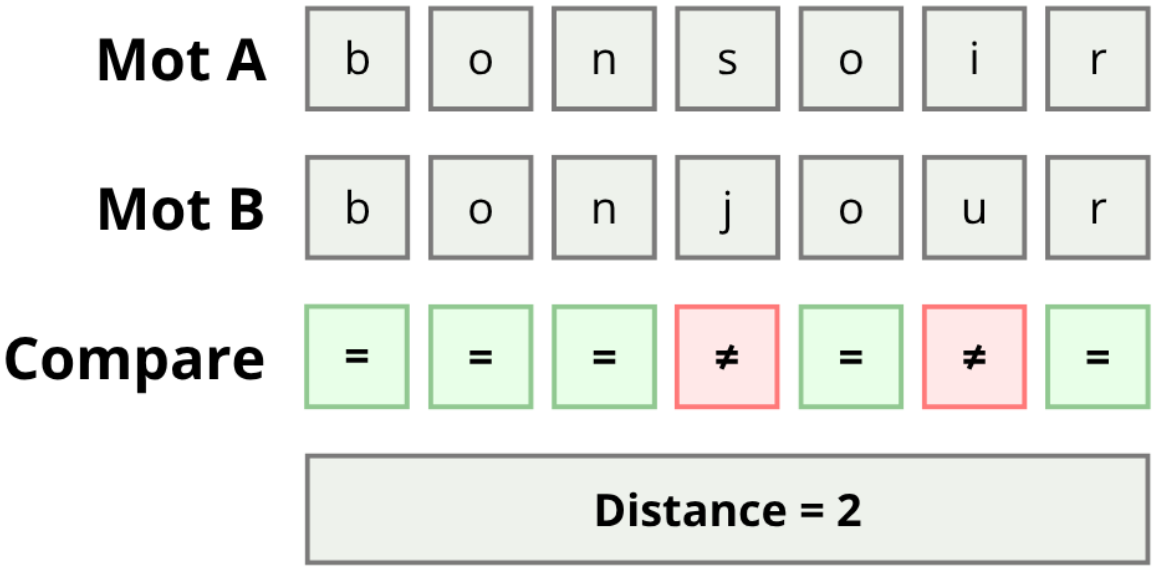


Complexité : $\mathcal{O}(n \times m)$ (où n et m sont les longueurs des chaînes de caractères)

Distance de Hamming

Compter le nombre de positions où les caractères diffèrent.

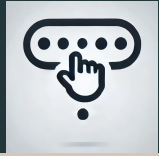
Utilisable que pour des mots de même longueur.



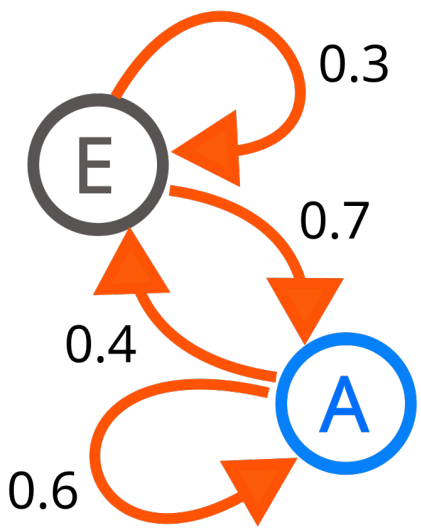
Complexité : $\mathcal{O}(n)$ (où n est la longueur des chaînes de caractères)

Les chaînes de Markov

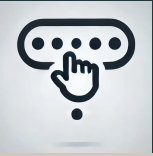
Définition d'une chaîne de Markov



- **Chaîne de Markov** : Modèle mathématique représentant un système de probabilité. Les probabilités de passer d'un état à un autre dépendent entièrement de l'état actuel.
- **Etats** : Elements du système.
- **Transition** : Action de passer d'un état à un autre.
- **Matrice transition** : Table permettant de regrouper les transitions entre tous les états.



Application



Utilisation pour modélisation de processus aléatoires, analyse de séquences, de prédictions ou d'historiques de navigation. Analyse des historiques de navigation ainsi que les actions de l'utilisateur.

Depuis	Vers	Probabilité
A	B	0.6
A	C	0.4
B	A	0.3
B	C	0.7
C	A	0.5
C	B	0.5

Exemple transitions entre trois pages web A, B et C

Matrice transition

Modélisation des transitions probables entre les étapes.

- Collecter les données
- Compter les transitions
- Calcul des probabilités
- Construction de la matrice

$$M = \begin{bmatrix} 0 & 0.6 & 0.4 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0.5 & 0 \end{bmatrix}$$

Résultat selon le tableau précédent

Exemple complet

$$A \Longrightarrow C \Longrightarrow B \Longrightarrow A \Longrightarrow B \Longrightarrow C$$

Depuis	Vers	Nombre d'observations
A	C	1
C	B	1
B	A	1
A	B	1
B	C	1

$$M = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \\ 0 & 1 & 0 \end{bmatrix}$$

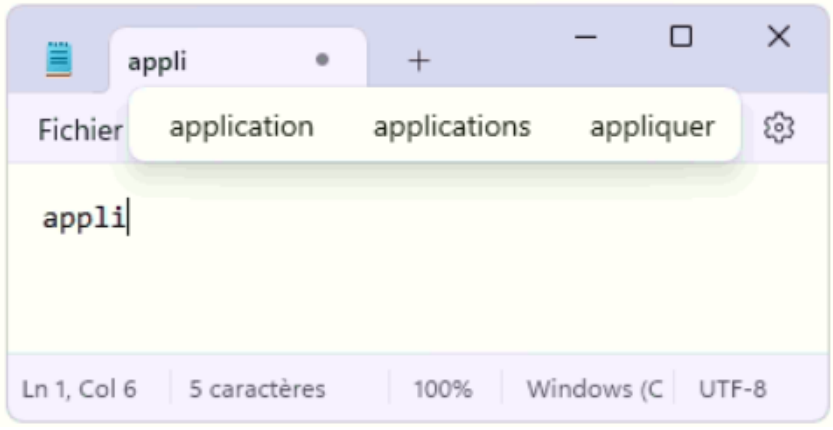
Calcul probabilités : nombre de fois où la transition X vers Y est observée, puis de diviser par le total des transitions partant de l'état X.

Notre outil de complétion semi-automatique

Choix pour l'implémentation

Notre objectif :

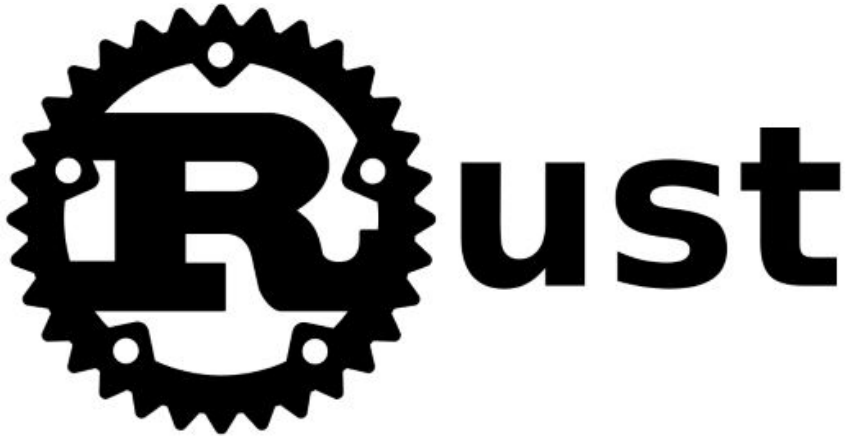
- Reproduire un outil de complétion semi-automatique comme sur téléphone mais sur ordinateur
- Utilisable dans n'importe quel application



Outil de suggestion de mots sur Windows

Choix du langage :

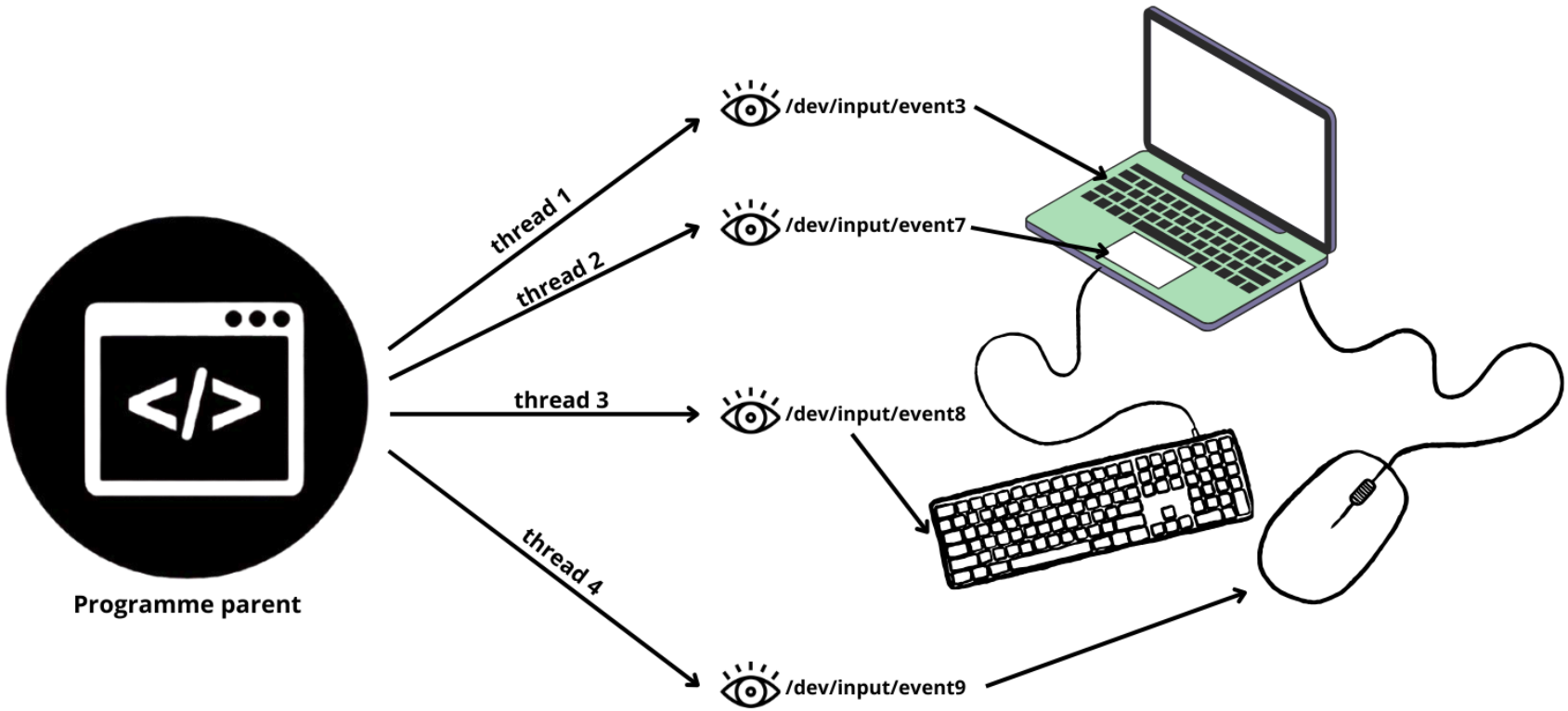
- Apprendre un nouveau langage
- Langage de programmation moderne
- Conçu pour la performance



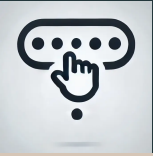
Création du keylogger & mouselogger

- Détection des péréphériques
- Lecture des évenements
- Décodage avec le fichier **input-event-codes.h**
- Sauvegarde des évènements pour récupérer le mot tapé

```
#define KEY_I                   23
#define KEY_O                   24
#define KEY_P                   25
#define KEY_LEFTBRACE           26
#define KEY_RIGHTBRACE          27
#define KEY_ENTER               28
#define KEY_LEFTCTRL            29
#define KEY_A                   30
#define KEY_S                   31
#define KEY_D                   32
```



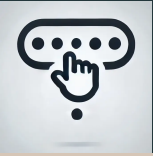
Création du clavier virtuel



Objectif : Ecrire le mot que l'utilisateur a choisi sur l'interface de l'application

- Traduction caractère $\rightarrow$ évènement clavier
- Envoi séquentiel des lettres du mot
- Suppression du mot précédent (retour arrière n fois)
- Réécriture fluide et invisible

Algorithme de suggestion



Utilisation de la **distance de Levenshtein** pour la suggestion de mots

- Comparaison avec $\simeq$ 140 000 mots
- Sélection des mots les plus proches

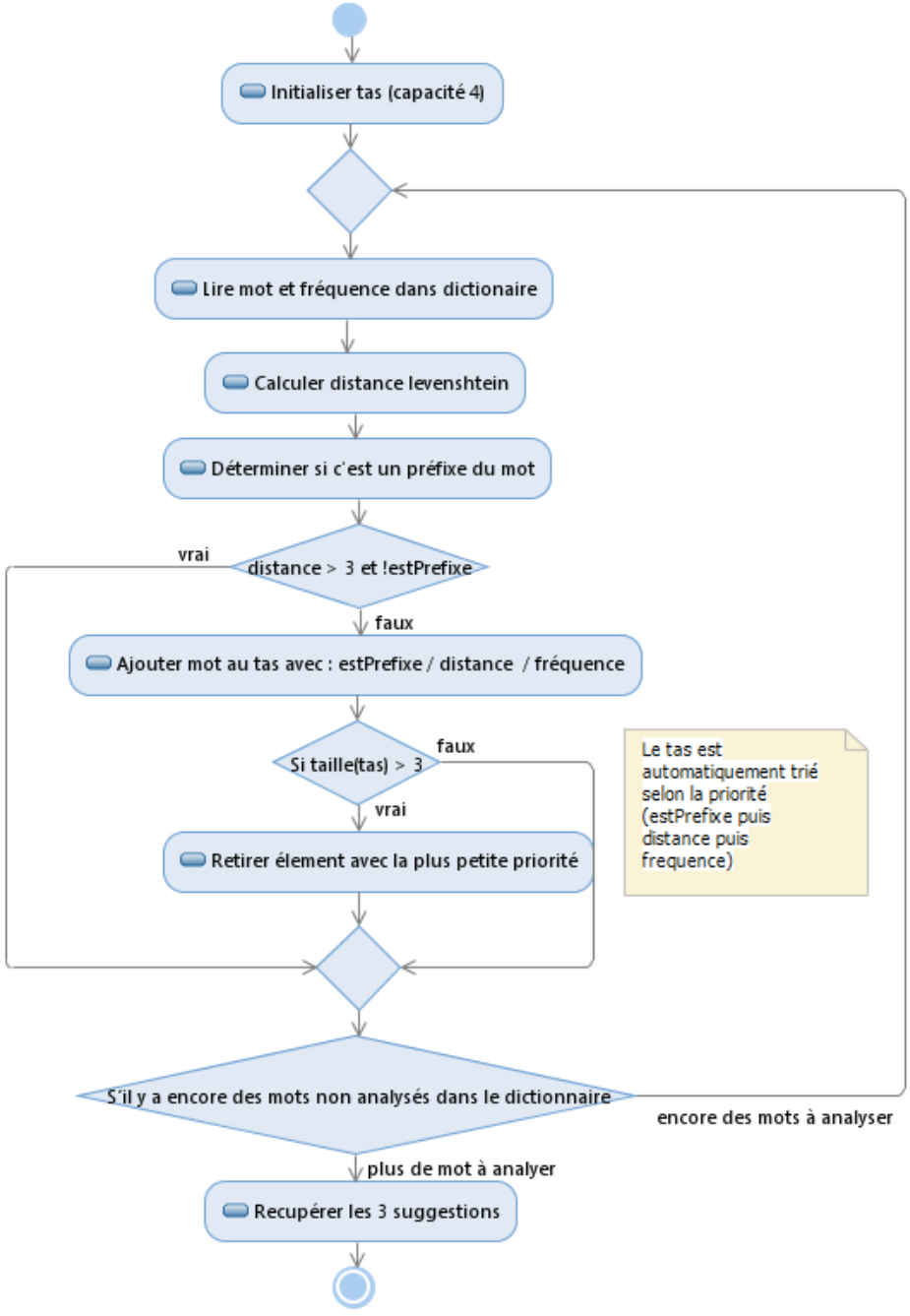
Résultats décevants

$\rightarrow$ Suggestions peu pertinentes
$\rightarrow$ Nécessité d'ajouter des critères complémentaires

Solutions apportées :

- Analyse des **préfixes**
- Pondération selon la **fréquence d'utilisation**

Algorithme de suggestion



L'interface graphique

- Initialement, interface prévue en **Rust avec GTK**
- Problème : impossible de garder la fenêtre au premier plan constamment
- Solution : interface réalisée en **Python avec Tkinter**
- Communication entre Rust et Python via leur **entrée standard**

Interface finale de notre implémentation

Installeur de l'application

- Création d'un installeur pour l'application
- Utilisation d'un **Makefile**
- Pour :
  - Compiler le code
  - Installer les dépendances
  - Attribuer les droits nécessaire
  - Créer le fichier **.desktop**

Démonstration

Conclusion

UNIVERSITÉ MARIE & LOUIS PASTEUR

CMI Figure
CURSUS MASTER EN INGÉNIERIE

Merci de votre attention !

Avez-vous des questions ?