Rapport Technique

Moteur de Génération et Validation Morphologique Arabe

Eya Gharbi – Zaghouani Ayoub
Classe 1ING4

1. Présentation générale

Ce projet implémente un moteur de dérivation morphologique arabe capable de générer des formes dérivées à partir d'un radical trilitère et d'un schème, et de valider si un mot dérivé appartient à un radical donné.

2. Chargement des données

Au démarrage :

- `roots.txt` charge les racines dans un ABR.
- `patterns.txt` charge les schèmes dans une table de hachage.

3. Structures de données

3.1. Arbre Binaire de Recherche (ABR)

Les racines trilittérales (ex : ب-ت-ك) sont stockées sous forme compacte (كتب). La comparaison repose sur l'ordre Unicode.

Chaque nœud contient :

- la racine compacte.
- une liste chaînée de dérivés.
- deux pointeurs (gauche/droite).

Normalisation : suppression des diacritiques, unification des variantes d'alef, vérification stricte du format.

Insertion : Descente récursive selon l'ordre lexicographique. Doublons refusés.

Suppression : La suppression suit les trois cas classiques : feuille, un enfant, deux enfants avec remplacement par le successeur.

3.2. Liste chaînée

Chaque racine possède une liste de mots dérivés (mot + fréquence). Elle permet un stockage léger et évite les recalculs.

3.3. Table de hachage

Les schèmes sont stockés dans une table de hachage de taille fixe $m = 37$ (nombre premier).

Fonction de hachage polynomiale :

$$h(k) = \left(\sum_{i=0}^{|k|-1} \text{ord}(k_i) \cdot 131^i \right) \bmod 37$$

Type d'entrée. Les schèmes sont validés :

- présence des lettres ف ع ل.
- lettres arabes uniquement.
- conservation de la shadda (si présente).

Normalisation : Chaque schème est normalisé (diacritiques supprimés sauf shadda, variantes d'alef unifiées). Les schèmes doivent contenir ف ع ل et respecter une longueur minimale.

Règle de dérivation : Elle est identique au schème normalisé (`rule = pattern`). La dérivation consiste à substituer ف ع ل par les lettres de la racine.

Chaînage : Chaque case de la table pointe vers une **liste chaînée** de schèmes. Si deux schèmes ont le même hash, ils sont ajoutés à la liste du même bucket. La recherche parcourt cette liste jusqu'à trouver la clé exacte.

4. Algorithmes

4.1. Génération

1. Normalisation de la racine donnée
2. Recherche dans l'ABR
3. Recherche du schème dans la table de hachage
4. Application de la règle de dérivation
5. Stockage dans la liste chaînée de la racine

Complexité :

$$T_{\text{gen}} = O(\log n) + O(|s|) = O(\log n)$$

Pour la génération d'une famille sur les p schèmes de longueur moyenne $\bar{s}$:

$$T_{\text{family}}(n, p) = O(\log n) + O\left(\sum_{i=1}^{p} (1 + |s_i|) \right) = O(\log n + p \cdot \bar{s})$$

4.2. Validation

Validation par régénération exhaustive :

1. Recherche racine dans l'ABR
2. Parcours des m schèmes
3. Génération + comparaison

$$T_{\text{val}} = O(\log n + P)$$

5. Complexité Algorithmiquekjernvkjnqerjgn

Soient :

- n : nombre de racines
- P : nombre total de schèmes

Opération	**Cas moyen**	**Pire cas**
Recherche ABR	$O(\log n)$	$O(n)$
Insertion ABR	$O(\log n)$	$O(n)$
Suppression ABR	$O(\log n)$	$O(n)$
Recherche hash	$O(1)$	$O(P)$
Insertion hash	$O(1)$	$O(P)$
Génération (1 mot)	$O(\log n)$	$O(n)$
Validation (1 mot)	$O(\log n + P)$	$O(n + P)$
Chargement racines	$O(n \log n)$	$O(n^2)$
Chargement schèmes	$O(P)$	$O(P^2)$

6. Choix algorithmiques

Arbre Binaire de Recherche (ABR) : Les racines sont stockées dans un ABR pour permettre une recherche efficace avec un faible coût mémoire. En moyenne, les opérations de recherche et d'insertion sont en $O(\log n)$.

Table de hachage : Une table de hachage est utilisée pour assurer un accès rapide aux schèmes, avec un coût moyen en $O(1)$.

Validation exhaustive : La validation consiste à générer les formes à partir de tous les schèmes et à comparer les résultats. Cette méthode garantit la cohérence morphologique.

Normalisation : La suppression des diacritiques et l'unification des lettres assurent des comparaisons fiables et évitent les ambiguïtés Unicode.

7. Difficultés rencontrées

- Normalisation cohérente entre racines et schèmes.
- Gestion des collisions dans la table de hachage.
- Contrôle des duplicatas dans ABR et hash.

8. Point d'originalité

Le système est exploité comme un **dictionnaire morphologique** : chaque schème est associé à une définition, et chaque dérivé validé peut être affiché avec son sens linguistique. Cette exploitation donne une valeur appliquée aux structures de données. Cette exploitation démontre l'application concrète des structures algorithmiques développées.

9. Conclusion

Le système combine ABR, table de hachage et listes chaînées de manière cohérente. Les performances sont dominées par $O(\log n)$ en moyenne pour les racines et $O(1)$ amorti pour les schèmes, ce qui rend le moteur efficace et extensible.