IFT6285 - Devoir 2

Louis-Vincent Poellhuber - Youcef Barkat

26 octobre 2023

---

a. Codes et librairies

Notre code a été écrit en Python. Nous utilisons la librairie `chardet` pour détecter l'encodage d'un texte, afin de pouvoir mieux lire les ficher textes. Nous utilisons `tqdm` pour savoir à l'avance environ combien de temps notre code prendra pour rouler. Pour la distance d'édition, nous utilisons le package suggéré en classe, `editdistance`. Pour toutes les autres distances, nous utilisons le package `jellyfish`, qui sert à calculer plusieurs métriques reliés aux mots, incluant les mesures de distance, le Soundex, etc.

b. Métriques calculées

Les métriques que nous avons utilisées dans la partie d'évaluation sont la Précision Stricte (Hard Accuracy) et la Précision Contextuelle (Soft Accuracy).

1. **Précision Stricte (PS) :** La précision stricte, dans notre correcteur orthographique, mesure le pourcentage de mots dans le texte qui ont été corrigés en respectant leur orthographe correcte. Une correction est considérée comme précise uniquement si elle correspond exactement au mot attendu correct. Elle calcule le rapport de mots pour lesquels la correction suggérée correspond exactement au mot attendu, comme suit :

$$\text{PS} = \frac{\text{Nombre de Mots Correctement Corrigés}}{\text{Nombre Total de Mots}} \times 100\%$$

   La précision stricte est une mesure rigoureuse qui évalue la précision de notre correcteur orthographique. Elle nous donne des informations sur notre performance en termes de correction stricte de l'orthographe.

2. **Précision Contextuelle (PC) :** La précision contextuelle, dans notre correcteur orthographique, mesure le pourcentage de mots où le mot attendu se trouve parmi les 5 suggestions les plus proches fournies par le correcteur orthographique. Elle calcule le rapport de mots où le mot attendu figure parmi les 5 corrections suggérées les plus proches, comme suit :

$$\text{PC} = \frac{\text{Nombre de Mots avec le Mot Attendu dans les 5 Premières Suggestions}}{\text{Nombre Total de Mots}} \times 100\%$$

   La précision contextuelle prend en compte l'objectif plus large d'améliorer la lisibilité du texte et sa signification en considérant des corrections appropriées sur le plan contextuel. Elle offre une évaluation plus tolérante de la performance de notre correcteur orthographique en tenant compte de la question de savoir si le mot prévu se trouve dans la liste des suggestions proches.

Nous avons choisi de nous concentrer sur l'amélioration de la précision stricte car c'est une mesure plus complexe à améliorer. En ce qui concerne la précision contextuelle, il existe plusieurs façons d'apporter des améliorations. Par exemple, nous pouvons rechercher des corrections qui ont le même premier caractère que la faute. Par exemple, 89 % des fautes de frappe et des mots d'origine partagent la même première lettre. De plus, l'amélioration de la précision stricte peut souvent entraîner des améliorations de la précision contextuelle, conformément à une règle générale, même si nous n'avons pas de preuves concrètes pour étayer cela ; il s'agit davantage de notre opinion.

c. Distances étudiées

Voici les distances étudiées.

1. **Distance de Jaro** : Distance prenant en compte le nombre de caractères correspondants et transposés. Nous utilisons la librairie `jellyfish` : voir l'implémentation dans la section `jaro_similarity` de la page github, ligne 130.
2. **Distance de Jaro-Winkler** : Amélioration de la distance Jaro, elle accorde de l'importance à la taille du préfixe commun entre deux mots. Nous utilisons la librairie `jellyfish` : voir l'implémentation dans la section `jaro_winkler_similarity` de la page github, ligne 134.
3. **Distance d'édition** : Calcule le nombre de transformations à effectuer pour transformer un mot en un autre. Nous utilisons l'implémentation de la librairie Python suggérée en classe `editdistance`.

En plus de ces distances, nous avons aussi étudié comment on peut pondérer le résultat afin de le raffiner. Par exemple, supposons que nous notre erreur est *speling*, le mot original étant *spelling*. Avec la distance d'édition, les mots *sperling* et *spelling* sont équivalents, mais il est évident pour un humain que ce-dernier est le vrai mot. Nous avons donc essayé deux façons de pondérer les résultats afin d'améliorer nos performances.

1. **Unigramme** : Pondère la sélection des mots les plus proches de l'erreur par la fréquence relative des mots. Nous avons construit un modèle unigramme décrivant les fréquences de chaque mot dans le vocabulaire. Ensuite, pour chacun des n mots les plus semblables à une erreur, nous calculons sa fréquence relative à la fréquence des autres mots semblables. Ceci nous permet d'éviter que les mots très communs comme *the* prennent trop de place.
2. **Soundex** : Pondère la sélection des mots les plus proches de l'erreur par la distance d'édition entre les Soundex de l'erreur et de chacun des mots semblables.
3. **Composite** : Comme les deux mesures ont permi d'améliorer les résultats, nous avons défini un score composite, ajustable avec un paramètre α :

$$composite = \alpha \times unigram + (1 - \alpha) \times soundex$$

d. Description du correcteur

Le correcteur fonctionne légèrement différemment dépendamment du type de distance et de pondération utilisé.

Distances sans pondération. Pour toutes les distances sans pondération, le correcteur calcule la distance entre chaque erreur du fichier d'entrée et chaque mot du vocabulaire. Il ordonne ensuite toutes les distances en ordre croissant ou décroissant, dépendemment de la distance utilisée. Par exemple, la distance d'édition est croissante, alors que les distances de Jaro et Jaro-Winkler sont décroissantes. Finalement, il sélectionne les n mots les plus semblables. Nous avons utilisé n=5 pour ce devoir.

Distance d'édition avec pondération. Après avoir calculé la distance d'édition entre chaque erreur et chaque mot du vocabulaire, le correcteur sélectionne les n mots les plus semblables. Pour ces n mots, il calcule ensuite la pondération associée, puis ordonne par la distance d'édition en premier, et pas la pondération en deuxième.

Distances numériques avec pondération. Similairement à la distance d'édition, le correcteur calcule la distance associée (Jaro ou Jaro-Winkler dans notre cas) et sélectionne les n mots les plus semblables. Ensuite, pour ces n mots, il va calculer la pondération associée. Il va finalement multiplier la distance avec sa pondération. Il ordonne ensuite les n mots selon cette nouvelle distance pondérée.

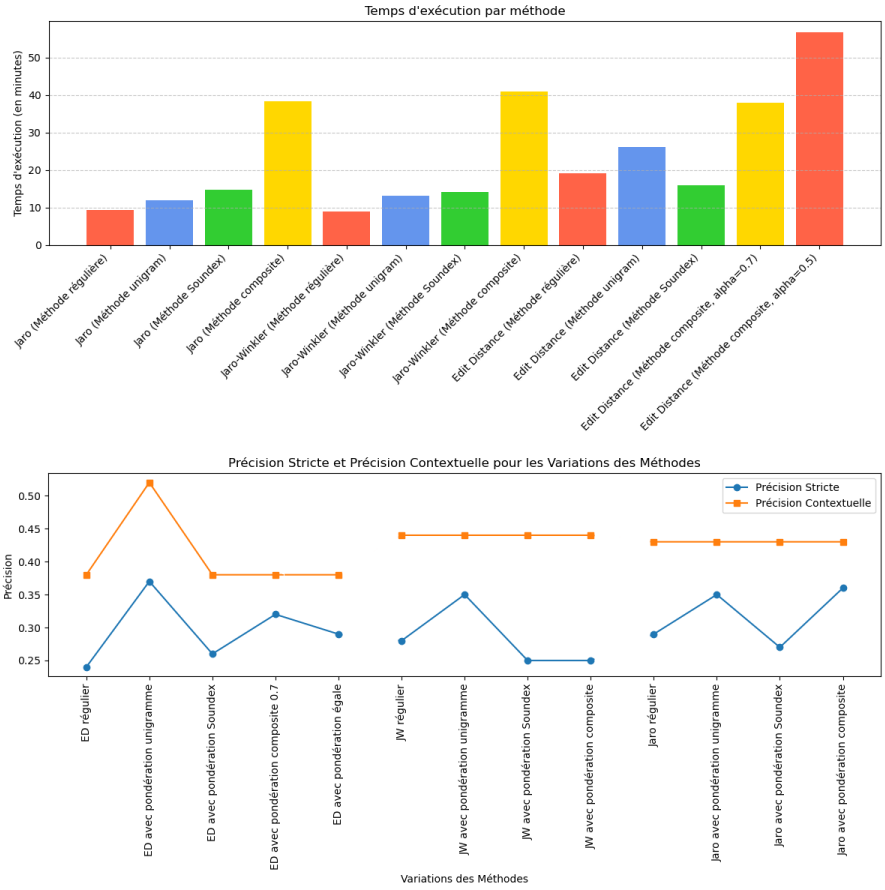

FIGURE 1 – Impact des distances étudiées sur les temps de correction et les performances

e. Impact des distances

Les données indiquent que le choix de la métrique de distance influence considérablement les temps de correction et la performance. Certaines méthodes sont plus gourmandes en calcul, ce qui se traduit par des temps d'exécution plus longs, mais elles offrent souvent une meilleure précision. En revanche, les méthodes plus rapides peuvent sacrifier la précision. Le choix de la métrique de distance devrait être en adéquation avec des besoins spécifiques, en équilibrant le temps et la précision de la correction. Une analyse et des expérimentations supplémentaires sont essentielles pour trouver la méthode optimale pour une application donnée.

f. Meilleure approche & limites

Notre meilleure approche est notre correcteur utilisant la distance d'édition et la pondération unigramme. Si on regarde la Figure 2, on constate que son *hard accuracy* est de 0.37, alors que son *soft accuracy* est de 0.52. Notre correcteur corrige donc parfaitement 37% des erreurs, et trouve 52% des erreurs parmi les 5 corrections les plus semblables à l'erreur. De plus, elle est assez facilement interprétable, étant donné la simplicité de la distance d'édition.

Une des limites majeures de notre correcteur est qu'il fonctionne essentiellement en deux étapes : trouver les 5 mots les plus semblables à l'erreur avec la distance d'édition, puis réorganiser ces 5 mots en prenant en compte leur fréquence. Ceci veut dire que si le vrai mot n'est pas dans les 5 mots, notre correcteur ne pourra jamais bien réorganiser les mots. De plus, c'est la métrique qui prends le plus de temps à exécuter.