NLP: Prédiction du sexe à partir des données personnelles

Pierre-Louis Barbarant

April 30, 2024

Le code est disponible en suivant ce lien: `https://github.com/pbarbarant/nlp-ensae-2024`

1. Présentation de la tâche

L'objectif de ce travail consiste à prédire le sexe d'un individu à partir de données des données de retranscriptions issues du projet *Socface*[1]. Cette information pourtant essentielle à l'étude des registres vient parfois à manquer suite à un oubli sur le registre original ou lors d'une erreur lors de la reconnaissance optique de caractères. Il est donc important de pouvoir l'estimer de manière automatique afin de compléter les registres dans l'optique de futures études.

2. Présentation et description des données

Afin de mener à bien notre étude, nous disposons de deux fichiers:

- `transcriptions_with_sex.csv` contenant les retranscriptions manuelles et automatiques relatives à 242 individus distincts avec notamment leur prénom et leur sexe.
- `firstname_with_sex.csv` contenant une base de 6047 prénoms avec la fréquence homme/femme associée.

Notre tâche consistera donc à mettre en place un système d'estimation du sexe de chacun des 242 individus du fichier de retranscription à partir du prénom en utilisant la base de prénoms associée et d'évaluer ses performances face à l'information réelle contenue dans la colonne `sex`.

Préalablement à la conception d'un modèle de classification, il est nécessaire d'extraire les informations utiles présentes au sein du fichier `transcriptions_with_sex.csv`. En effet les transcriptions manuelles et automatiques sont stockées sous formes de chaînes de caractères dont on donne un exemple ci-dessous:

```
surname: Chardon firstname: Marie occupation: idem link: fille age: 30
```

Le fichier `preprocessing.py` implémente donc une tokenization permettant de récupérer la valeur associée à chaque mot-clef puis stocke l'intégralité des informations au sein d'un grand dataframe nommé `preprocessed_dataset.csv`.

De même, on prétraite le fichier `firstname_with_sex.csv` en sélectionnant le sexe le plus fréquent afin d'obtenir le fichier `gender_table.csv` qui contient le sexe le plus probable associé à chaque

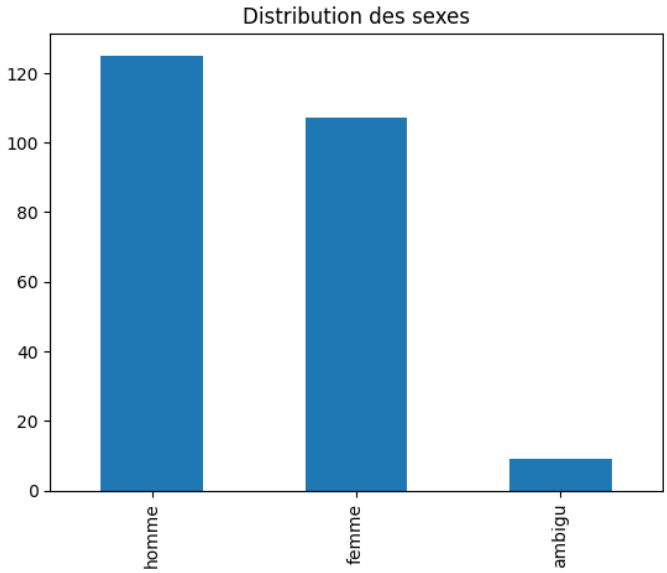

Figure 1: Distribution des sexes au sein du fichier de retranscription. La répartition homme/femme est équilibrée mais neuf individus sont marqués comme “ambigus”.

prénom.

La figure 1 présente l’histogramme de répartition des sexes telle qu’apparaissant au sein du fichier de retranscription. On notera l’apparition du label “ambigu” qui pourra être source de confusion notre modèle. En observant notre base de donnée, on se rend compte que ce label apparaît lors de discordances entre le prénom et le lien de parenté (ex: prénom féminin mais rôle de chef de famille) ou lorsque le prénom est unisexe (ex: Claude).

3. Analyse des modèles

Une vaste palette de classifieurs nous est offerte, néanmoins nous avons à disposition un dataset de transcriptions de très faible taille. Il nous est donc impossible d’entraîner un modèle de bonne qualité en partant de zéro, d’autant plus qu’il serait nécessaire de partitionner le dataset à entre données d’entraînement/test, ce qui ne laisserait quelques centaines d’exemples au modèle pour apprendre.

De manière alternative, on pourrait envisager de télécharger un modèle pré-entraîné sur des données francophone à l’instar de `camenBERT` [Mar+20], puis d’insérer au sein d’un prompt bien choisi les informations que nous avons pré-traité dans la partie précédente :

```
Détermine le sexe (homme ou femme) de l’individu ayant pour prénom: <PRENOM>,
            occupation: <OCCUPATION> et rôle familial: <ROLE>
```

Néanmoins, si cette approche peut s’avérer fonctionnelle, elle n’en reste pas moins très coûteuse en puissance de calcul et ne tire pas parti du fichier `gender_table.csv` que nous avons construit.

La méthode la plus efficace est aussi la plus naturelle et consiste à simplement regarder les correspondances entre le prénom issu de la retranscription et la proportion homme/femme donnée au

[1] https://socface.site.ined.fr

sein de `firstname_with_sex.csv`.

La difficulté majeure provient du fait que l'appariement entre les prénoms retranscrit dans les deux tables n'est pas nécessairement exact suite à la présence de majuscules, fautes d'orthographes ou traits d'union. Notre problème se réduit donc à trouver la correspondance la plus "proche" entre un prénom issu de la transcription et notre *gender table* qui contient le sexe le plus fréquent pour chaque prénom.

Pour ce faire nous utilisons le *fuzzy matching* entre chaînes de caractères dont une des mesures les plus couramment utilisées est la distance de Levenshtein.

En effet la distance de Levenshtein mesure le nombre minimum d'opérations nécessaires pour transformer une chaîne de caractères en une autre. Ces opérations comprennent l'insertion, la suppression ou la substitution d'un caractère. Plus la distance de Levenshtein entre deux chaînes est petite, plus elles sont similaires.

Par exemple, la distance de Levenshtein entre les prénoms `Marie` et `marion` est de 3 puisque nécessite la substitution du 'M' en 'm', du 'e' en 'o' et l'insertion d'un 'n'.

Cette approche est particulièrement adaptée à notre application puisqu'elle est relativement simple à calculer et un grand nombre d'implémentations très performantes sont disponibles, ce qui nous permet de traiter un large volume de données avec peu de ressources. Cette efficacité a fait le succès du *fuzzy matching* dans les domaines de la correction automatique et des moteurs de recherches où l'utilisateur doit avoir un retour avec un minimum de latence.

4. Expérimentation

Notre mise en œuvre du *fuzzy matching* repose sur le package `rapidfuzz`[2], accessible sur PyPI. Nous commençons par convertir toute la colonne des prénoms disponibles dans le fichier `gender_table.csv` en une liste. Ensuite, pour chaque prénom apparaissant dans la transcription, nous calculons un score de similarité, basé sur la distance de Levenshtein, compris entre 0 et 1, pour chaque prénom de la liste. Enfin, nous retournons le sexe associé au prénom présentant le score de correspondance le plus élevé.

Nous évaluons le modèle initialement sous l'angle de son *accuracy* qui s'élève à 94.61% (90.04% sur les données estimées par reconnaissance de caractères), ce qui nous montre que notre algorithme fondé sur le *fuzzy matching* offre des résultats largement satisfaisant avec un temps moyen d'exécution par prénom d'environ $3ms$.

En metttant à part les individus considérés comme "ambigus" nous pouvons tracer la courbe Vrai/-Faux Positif, couramment nommée courbe ROC (Fig. 2). Nous obtenons une aire sous la courbe de 0.98, ce qui nous assure une excellente capacité de notre méthode à distinguer les prénoms féminins et masculins.

Enfin, une analyse plus fine des erreurs de classification (Fig. 3) révèle que ce sont principalement les prénoms "ambigus", tels que `Claude`, qui sont à l'origine des erreurs, comme prévu, étant donné que notre modèle ne peut prédire qu'un des deux sexes. De plus, il est intéressant de noter que la distance de Levenshtein peut poser des problèmes dans le cas des prénoms composés, tels que `Jean-Marie`, qui, du point de vue du nombre de substitutions/suppressions, est plus proche du prénom `Marie` que `Jean`.

[2] https://pypi.org/project/rapidfuzz/

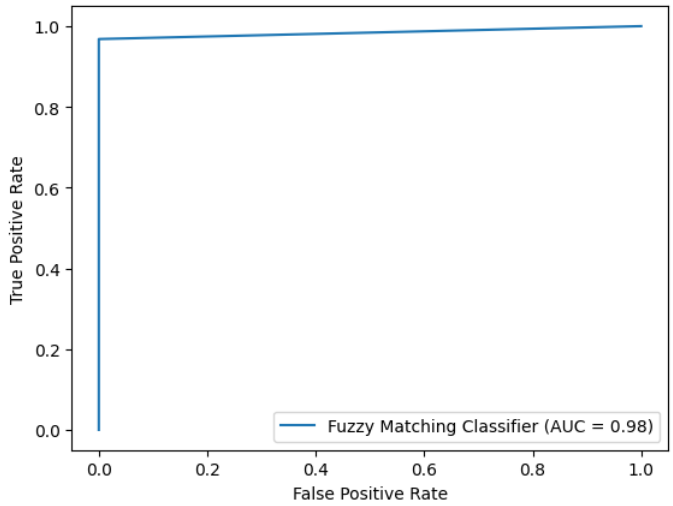

Figure 2: Courbe ROC (Receiver Operating Characteristic). $AUC = 0.98$. Le modèle distingue avec une précision presque parfaite les prénoms masculins des prénoms féminins.

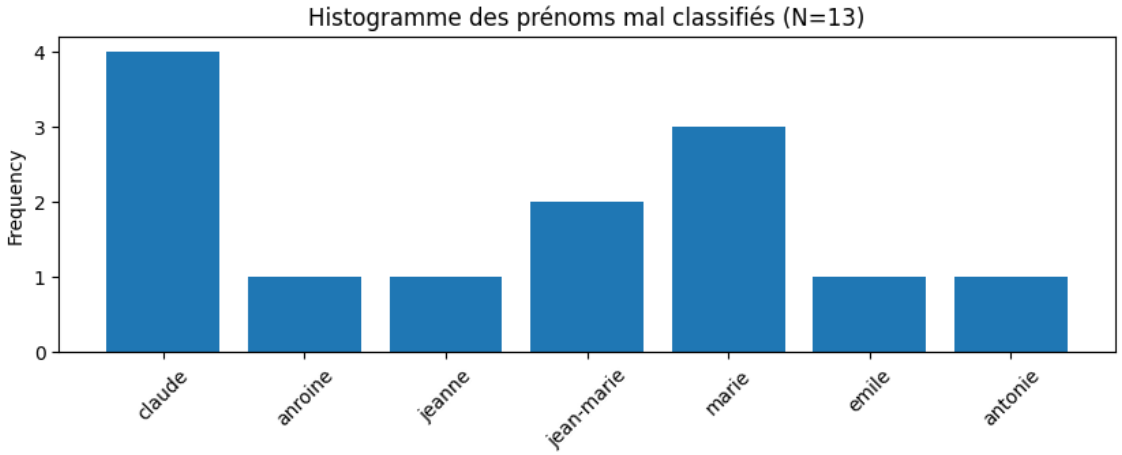

Figure 3: Répartition des prénoms mal classifiés par notre méthode. On remarque que ce sont principalement les individus notés comme “ambigus” ou portant un prénom unisexe pour lesquels la prédiction échoue.

5. Conclusion

Nous enrichissons le projet *Socface* en introduisant une méthode d'estimation du sexe d'un individu à partir des informations extraites de son registre. Notre approche repose sur le *fuzzy matching* des prénoms tels qu'ils sont enregistrés dans le registre, avec une base de données préexistante associant les prénoms au sexe le plus probable. Notre algorithme délivre des résultats satisfaisants sur un jeu de données de 242 prénoms tests, tout en nécessitant un minimum de ressources en termes de mémoire et de temps de calcul. De plus, notre algorithme est conçu pour être facilement parallélisable sur différentes unités de calcul, ce qui le rend adaptable à des bases de données contenant des millions d'individus. Cependant, notre analyse révèle que la principale limitation du modèle actuel réside dans son incapacité à prédire l'ambiguïté du sexe d'un individu. Pour pallier cette limitation, nous suggérons l'intégration des données relatives à l'occupation et au rôle dans la famille, en plus du prénom, afin d'établir le sexe le plus probable à l'aide d'un algorithme de vote.

References

[Mar+20] Louis Martin et al. "CamemBERT: a Tasty French Language Model". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020.