Comparaisons d'enregistrements textuels

Cette première séquence est dédiée à la découverte des concepts fondamentaux du couplage d'enregistrements et l'expérimentation de comparaisons entre chaînes de caractères.

Objectifs

- se familiariser avec la manipulation de chaînes de caractères en Python ;
- apprendre les concepts fondamentaux du couplage d'enregistrement ;
- comprendre les conséquences des imperfections de données numériques "réelles" ;
- comprendre le fonctionnement d'une mesure classique de distance entre chaînes de caractères : la distance de Levenshtein.

> [!IMPORTANT] Les différents codes de cette première séquence sont à réaliser en complétant le fichier de script Python `sequence_1.py`.

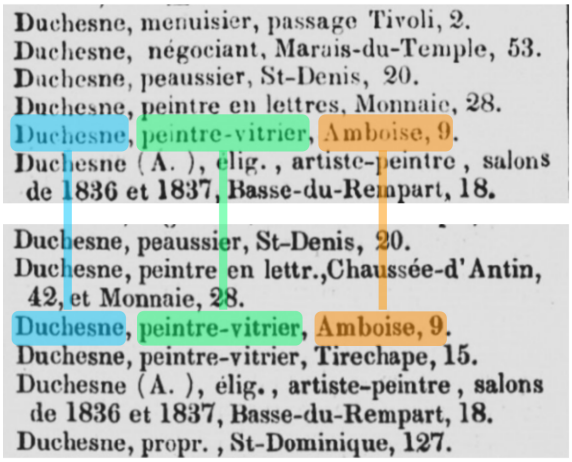
> [!TIP] Pour exécuter l'interpréteur Python sur un fichier de script depuis un terminal :
>
> ```
> python /chemin/vers/le/script.py
> ```

Le couplage d'enregistrements (*Record linkage*) en quelques mots

Le *record linkage* ou *data matching* , en français "couplage d'enregistrements", désigne un ensemble de technique pour reconnaître dans deux bases de données les entrées qui correspondent à la même *entité*, qu'il s'agisse de personnes, d'objets, d'événements, etc. Le processus s'appuie sur deux opérations essentielles: (1) identifier des liens entre les éléments de deux ensembles de données A et B, puis (2) décider pour chaque lien s'il s'agit d'une correspondance véritable (*match*) ou non (*non-match*), voir éventuellement d'une correspondance plausible (*possible match*).

Formalisée dans le champ des technologies de l'information, cette tâche est à l'ère des humanités numériques un classique du traitement de données en sciences sociales. Un cas typique en histoire consiste à relier des ensembles de données extraites dans des sources d'archives pour identifier, par exemple, les références aux mêmes entités du monde réel, qu'il s'agisse de personnes, de lieux, etc.

Voici par exemple deux extraits des entrées extraites dans les annuaires du commerce de Didot-Bottin en 1842 et 1843. Intuitivement, on a envie de considérer qu'il s'agit de deux enregistrements successifs du même commerce[^1] tenu par M. Duchesne car les champs des deux entrées sont rigoureusement les mêmes. Notre critère de décision est ici très simple: dans une paire quelconque d'entrées prises dans chaque jeu de données, on considère qu'il s'agit de la même entité si les chaînes de caractères des trois champs PER, ACT et LOC sont exactement égales deux à deux. On dit alors que ces champs constituent des **identifiants** (partiels), et qu'il y a couplage (*match*) car notre méthode décision **s'accorde** sur la correspondance de chacune des valeurs de ces identifiants.



Sources

PER	ACT	LOC
Duchesne	menuisier	passage Tivoli, 2
Duchesne	négociant	Marais-du-Temple, 53
Duchesne	peaussier	St-Denis, 20
Duchesne	peintre en lettres	Monnaie, 28
Duchesne	peintre-vitrier	Amboise, 9
Duchesne (A.)	artiste-peintre, [...]	Basse-du-Rempart, 18

Annuaire général du commerce, de l'industrie, de la magistrature et de l'administration : ou almanach des 500.000 adresses de Paris, des départements et des pays étrangers, ed. Firmin-Didot frères (Paris), **1842**. gallica.bnf.fr/ark:/12148/bpt6k6286466w

PER	ACT	LOC
Duchesne	peaussier	St-Denis, 20
Duchesne	peintre en lettres	Chaussée d'Antin, 42
Duchesne	peintre en lettres	Monnaie, 28
Duchesne	peintre-vitrier	Amboise, 9
Duchesne (A.)	artiste-peintre, [...]	Basse-du-Rempart, 18
Duchesne	propr.	St-Dominique, 127

Annuaire général du commerce, de l'industrie, de la magistrature et de l'administration : ou almanach des 500.000 adresses de Paris, des départements et des pays étrangers, ed. Firmin-Didot frères (Paris), **1843**. gallica.bnf.fr/ark:/12148/bpt6k628646

Entrées d'annuaires extraites

Cet exemple trivial donne l'intuition du principe général du couplage d'enregistrement, mais la réalité est inévitablement plus complexes. Les erreurs des sources, les doublons, le bruit OCR et les erreurs de détection s'accumulent et se combinent pour compliquer le problème.

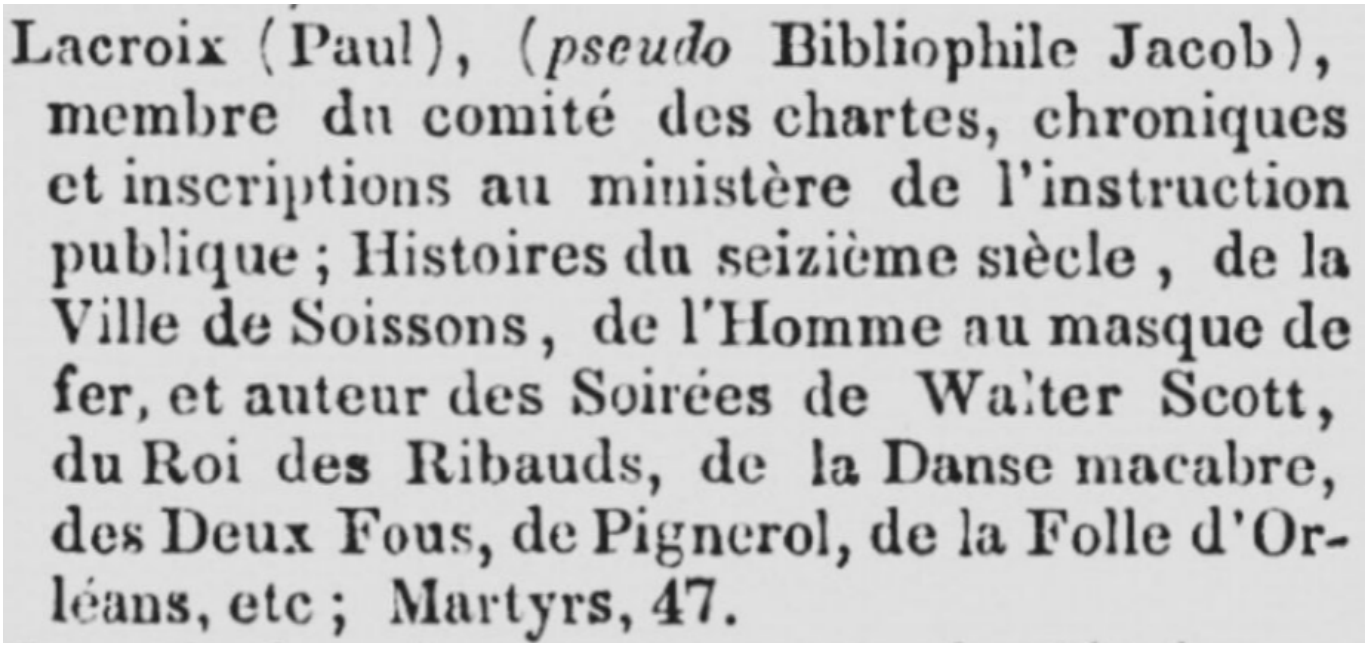
On distingue deux grands catégories d'algorithmes de couplage : celles **déterministes** , et celles **probabilistes**. Les premières déterminent si les paires d'enregistrements sont en accord ou en désaccord sur un ensemble donné d'identifiants, l'accord sur un identifiant donné étant évalué comme un résultat discret - "tout ou rien". Ces méthodes sont simples mais on tendance à créer de nombreux faux négatifs lorsque les données sont imparfaites, bruitées ou incomplètes. Pour surmonter ce problème, on peut quitter le mode binaire et chercher plutôt à associer les paires d'enregistrement à un nombre représentant la probabilité que cette paire soit un véritable *match* ou non, étant donné les informations à disposition. C'est ce que font les méthodes probabilistes traditionnelles. Aujourd'hui, cette tache est également réalisé avec des méthodes par apprentissage profond.

> [!NOTE] Pour aller plus loin Foster, Ian, et al., eds. Big data and social science: data science methods and tools for research and practice, chap.2. CRC Press, 2020. URL: https://textbook.coleridgeinitiative.org/chap-link.html
>
> Harron, Katie, Elaine Mackay, and Mark Elliot. "An introduction to data linkage." (2016). URL: https://eprints.ncrm.ac.uk/id/eprint/4282/

[1]: Désignés comme un "commerce" ici par simplicité, il est en fait assez difficile de connaître la réalité des éléments recensés dans les annuaires du commerce. Il peut s'agir de commerces, d'institutions, d'entrepreneurs, de notables, etc.

Exemple introductif

Dans les annuaires Didot des années 1840, on trouve par exemple M. Paul Lacroix, écrivain polygraphe, auto-désigné *bibliophile* :


Lacroix (Paul), (*pseudo* Bibliophile Jacob), membre du comité des chartes, chroniques et inscriptions au ministère de l'instruction publique ; Histoires du seizième siècle, de la Ville de Soissons, de l'Homme au masque de fer, et auteur des Soirées de Walter Scott, du Roi des Ribauds, de la Danse macabre, des Deux Fous, de Pignerol, de la Folle d'Orléans, etc ; Martyrs, 47.

https://gallica.bnf.fr/ark:/12148/bpt6k6393838j/f364

Ces annuaires sont transformés, par OCRisation et traitements automatiques, en bases de données numériques sérielles. On obtient ainsi de grands tableaux de données où chaque enregistrement d'annuaire est une ligne, et les colonnes stockent les différentes parties d'un enregistrement selon s'il s'agit du **nom** (PER) de la personne/institution/commerce , son **activité** (ACT) ou son **adresse** (LOC).

Cas trivial : couplage exact

Voici par exemple deux mentions de Paul Lacroix dans les annuaires de 1841 et 1844 :

Édition	PER	ACT	LOC
1841	Lacroix (Paul), (pseudo Bibliophile Jacob)	membre du comité des chartes[...]	Martyrs, 47
Édition	**PER**	**ACT**	**LOC**
1844	Lacroix (Paul), (pseudo Bibliophile Jacob)	membre du comité des chartes[...]	Martyrs, 47

✏ **Q1.** À la lecture, il est évident qu'il s'agit de la même personne, les champs PER, ACT et LOC étant identiques. Comment reproduiriez-vous cette vérification en Python ? Implémentez votre proposition dans `sequence_1.py` en complétant le corps de la fonction `score_exact`.

> [!NOTE] 📝 **À retenir.** Coupler deux enregistrements qui représentent la même entité du monde réelle dans des sources de données différentes revient à vérifier que les champs qui permettent de l'identifier de manière **unique** sont les mêmes. Ici, on a considéré que deux enregistrements d'annuaires concernent la même personne si elles ont le même nom (PER), la même activité (ACT) et la même adresse (LOC).

Données bruitées et normalisation

En réalité, les processus d'extraction automatique, notamment l'OCR, produisent des erreurs de reconnaissance. Cela peut être dû à la qualité des documents source, à la graphie, à la présence de bruits

divers : taches, déchirures, transparence partielle des pages, etc. C'est (malheureusement) le lot de l'immense majorité des données extraites des sources historiques.

L'exemple précédent était artificiellement corrigé à la main. Voici en réalité ce qui a pu être extrait des annuaires :

Édition	PER	ACT	LOC
1841	lacroix (paul, (bibliophile jacob	membre du comite des chartes	martyrs 47

Édition	PER	ACT	LOC
1841	Lacroix Paul. (Bibliophile jacob	membre du comité des chartes	Martyrs, 47

✏ **Q2**. Quels types de différences pouvez-vous identifier ? Comment feriez-vous pour adapter ces chaînes de caractère afin que le test implémenté précédemment fonctionne à nouveau correctement ? Reportez-vous à la section Q2 de `sequence_1.py` et implémentez votre proposition en complétant la fonction `normalisation`. Testez là sur les enregistrements ci-dessus.

> [!NOTE] 📝 **À retenir.** Les données réelles sont rarement exemptes d'erreurs. C'est encore plus le cas de textes extraits automatiquement par OCR. Or, cela gêne fortement le couplage. Il est donc intéressant de faire précéder le couplage lui-même par des pré-traitement pour "nettoyer" les enregistrements à coupler.

Couplage "approximatif"

En réalité, les différences peuvent dépasser les simples erreurs de forme et toucher les mots eux-mêmes. Cela peut venir :

- des erreurs OCR : lettres mal reconnues, doublées ou manquantes
- des véritable différences de graphie, typiques des documents historiques (ex. Martyrs / Martirs), des abréviations, etc.

Voici une version particulièrement dégradée des deux mentions de M. Lacroix :

Édition	PER	ACT	LOC
1841	lacrox (paul, (bibliophile iaco	membre du coniite des chartes	martirs 4I

Édition	PER	ACT	LOC
1841	Lacroix Paul. (Bibliophile jacob	membre du com. des chartes	Martyrs, 47

Il est clair que, même normalisées, les chaînes de caractères restent différentes, le test d'égalité initial est trop stricte.

Une manière usuelle de s'en sortir consiste à mesurer un *degré de ressemblance* entre les textes, communément appelé **mesure de similarité** entre des mots. Plutôt qu'une valeur binaire (*match* / *non-match*), on va à associer un score à une comparaison entre deux mots ou deux suites de mots. Plus le score est élevé, plus les mots se ressemblent. Plutôt qu'une similarité, on peut aussi mesurer une **distance**. Le principe est le même, mais cette fois les mots semblables ont un score faible.

Il existe de nombreuses métriques de distances ou de similarité dans la littérature[^1]; nous allons tester la plus commune d'entre elles, la **distance de Levenshtein**, souvent nommée en anglais *edit distance*. Informellement, la distance de Levenshtein entre deux mots se définit comme le nombre minimal de changements unitaires (ajout ou suppression d'un caractère, substitution de 2 caractères) nécessaires pour transformer un mot en l'autre.

Par exemple, pour transformer le mot 'iaco' en 'jacob' il faut changer le caractère 'i' en 'j' et insérer un 'b'. La distance de Levenshtein est donc de 2.

✏ **Q3**. Plusieurs bibliothèques Python fournissent des métriques de similarité et de distances entre chaînes de caractères. Nous allons utiliser NLTK, une boite à outil dédiée au traitement automatique du langage naturel. Installez NLTK dans votre environnement Python courant.

```
pip install nltk
```

✏ **Q4**. Dans NLTK, la distance de Levenshtein peut être calculée avec la méthode *nltk.edit_distance*. Importez la méthode `edit_distance` du module `nltk.metrics` dans le script `sequence_1.py` et vérifiez que `edit_distance('iacob' et 'jacob')` est bien égal à 2.

✏ **Q5**. Proposez une adaptation de la fonction `score_exact` (Q1) nommée `score_approximatif`, qui imprime les distances d'édition des champs deux à deux et retourne leur moyenne arithmétique. Testez cette fonction sur les enregistrements `lacroix_1841` et `lacroix_1844`, avec ou sans normalisation préalable. Quel effet produit la normalisation préalable des champs ?

✏ **Q6**. Pouvez-vous imaginer une stratégie simple pour décider si une paire d'enregistrements est un couple valide (un *match*) ou non (*non-match*) à partir du résultat donné par `score_approximatif`? Implémentez la fonction `decision` qui prend un score en entrée (et éventuellement d'autres paramètres), et retourne un booléen : vrai si le score correspond à un match, faux sinon

> [!NOTE] 📝 **À retenir.** Lorsque les enregistrements à coupler contiennent des erreurs, ou de petites différences, le couplage exact est mis en échec et génère des **faux négatifs**, c'est à dire des couplages qu'il n'a pas réussi à identifier. Dans ce type de situation, on essaye tente plutôt de construire une mesure de la ressemblance entre deux enregistrements. Lorsque les champs sont des chaînes de caractères, on utilise des mesures de similarité ou de distance entre mots pour cela. Il faut alors se doter d'une méthode de décision pour choisir si une valeur de similarité / distance entre deux enregistrements signifie qu'ils sont couplés, ou non.

[¹]: En voici quelques unes, listées pour la bibliothèque logicielle Javascript Talisman : https://yomguithereal.github.io/talisman/metrics/

Une première chaîne de couplage minimaliste ?

Vous avez jusqu'ici :

- vu l'importance de pré-traiter les chaînes de caractère pour faciliter leur comparaison
- testé deux techniques de couplage, l'une exacte adaptée aux enregistrements sans erreurs, et l'autre approximative plus souple mais dont les résultats sont moins facilement interprétables.
- testé une méthode simple de classification d'une paire d'enregistrements en *match* ou *non-match*.

Ces étapes forment la base d'un processus de couplage d'enregistrements. Il peuvent être beaucoup plus complexes et raffinés, mais suivent en général ces étapes : (1) pré-traitements, (2) comparaison, (3) classification.

✏ **Q7**. Créez une dernière fonction `couplage`, qui prends en paramètre deux enregistrements et réalise les étapes d'un couplage approximatif. Reportez vous à `sequence_1.py` pour les instructions détaillées. Testez ensuite le processus sur les paires d'enregistrements suivants. Testez différentes valeurs de seuil pour la fonction `couplage` : trouvez-vous facile de déterminer une valeur satisfaisante pour tous les cas ?

```
lanet_a = ['Lanet (Mme J.-A', 'professeur d''harmonie', 'Beaux-Arts 6']
lanet_b =  ['Lanet (Mme)', 'professeur d''harmonic', 'Beaux Arts 6']

laplace_a = ['Laplace et Dumont (Mlles', 'institutrices', 'Lions-St-Paul
14']
laplace_b = ['Laplace et Dumont (Mlles)', 'institutrices', 'Lions-St-Paul
14']

lascols_a = ['Lascols et Souchon de la Lozère', 'fab. de draps et autres
tissus', 'boulev. Poissonnière 12']
lascols_b = ['Lascols et Souchor de la Iozere', 'fab. de draps', 'b.
Poissonnière 12']

regnault_a = ['Regnault et vve Poupinel' 'fab. d''ouates, depĕt' '47 Cha-
ronne']
regnault_b = ['Regnault et Vve Poupinel' 'fab. d''ouates, depet' 'Charonne
47']
```

> [!NOTE] 📝 **À retenir.** Le couplage est un processus composé de multiples étapes et de nombreuses approches existent. Elles reposent toutefois toutes sur les mêmes grandes étapes. Lorsqu'on utilise du couplage approximatif, fixer les paramètres du calcul n'est pas évident et demande une connaissance approfondie des données. Aujourd'hui, il existe des méthodes tentant de déterminer ce type de paramètre de manière automatique, par apprentissage. Cela dépasse toutefois le cadre de cet atelier !