RÉPUBLIQUE FRANÇAISE

Liberté
Égalité
Fraternité

Dares

Déchiffrer le monde du travail
pour **éclairer** le débat public

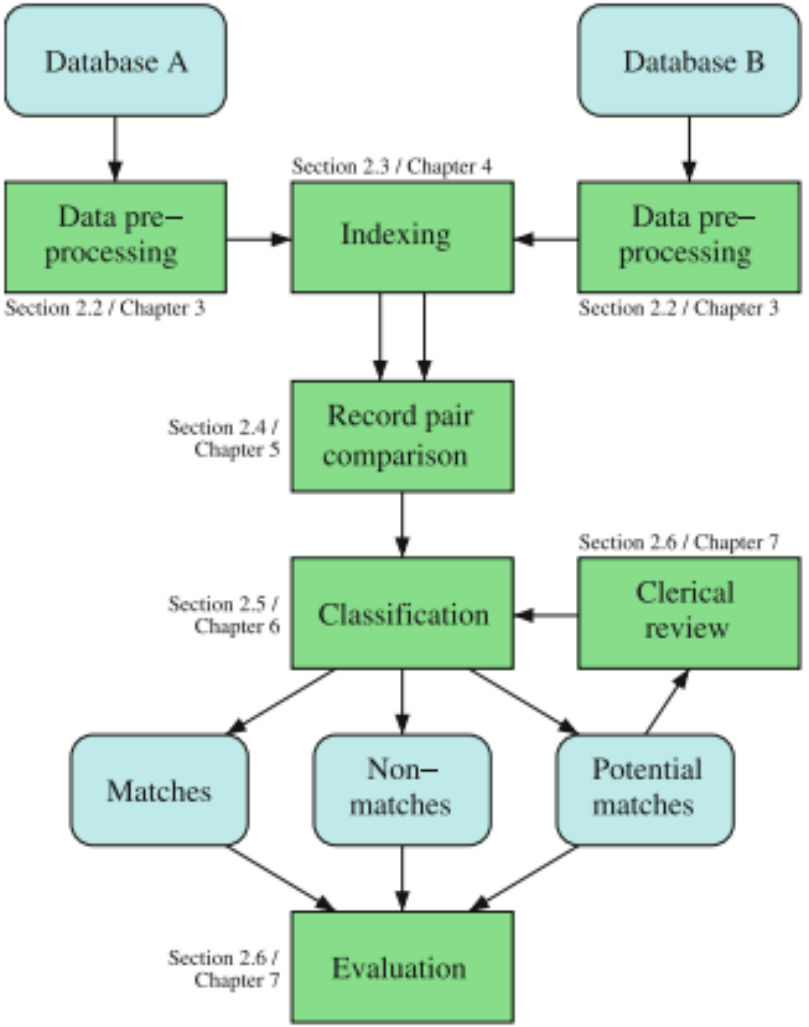
depp Direction de l'évaluation, de la prospective et de la performance

Les appariements sur identifiants indirects dans InserJeunes

InserJeunes

- **Objectif premier** : mesurer l'insertion des jeunes sortants d'apprentissage ou de voie pro scolaire.
- **Comment?** : InserJeunes mobilise de nombreuses bases (élèves, examen, DSN, contrats d'apprentissage) et les apparie :
  - Parfois sur identifiant direct (l'INE RNIE identifiant unique de chaque élève)
  - Parfois sur identifiants indirects (noms, prénoms, date et commune de naissance, sexe)
- La mise en place d'un outil d'appariement sur identifiants indirects est donc un **enjeu central d'InserJeunes** :
  - Rapide
  - Générique
  - De bonne qualité d'un point de vue statistique
  - Basé sur des outils open source et lui-même en open source
- Illustration sur l'appariement « qualité » (315 000 apprentis appariés avec 7,5 millions de salariés)

Le processus général



L'indexation des données

- **Objectif :** établir une liste de paires à analyser dans la suite du processus

- **Approche naïve** : faire le produit cartésien des 2 tables. 2,3 billions de paires sur l'appariement qualité => trop de paires, pas opérant.

- **Approche usuelle par clé de blocage** : ne conserver que les paires qui partagent la même modalité d'une ou plusieurs variables indirectement identifiantes qu'on appelle la ou les clés de blocage
  - Produit tout de même trop de paires.
  - Problématique sur les noms et prénoms même en utilisant un algorithme de phonétisation

L'indexation des données : l'approche InserJeunes

- **Étape 1 : appariement exact** => plus que 50 000 apprentis à apparier avec 7,2 millions de salarié

- **Étape 2 : indexation via l'union de requêtes SQL (*"fuzzy matching"* en SQL)**

create table paires as select * from tableA, tableB

where Levenshtein (tableA.nom, tableB.nom)<**3**

and Levenshtein (tableA.prenom, tableB.prenom)<**3**

and tableA.idDepartementNaissance=tableB.idDepartementNaissance

and tableA.anneeNaissance=tableB.anneeNaissance

=> 1 million de paires sur l'appariement qualité.

- Remarques
  - Possibilités d'améliorer encore les performances via du **partitionnement** (*« partition wise joins »* par exemple à partir de la V11 de postgresql)
  - A questionner sur de très gros volumes : une indexation via elastic search est alors sans doute plus adaptée.

distance de Levenshtein

nb insertions, suppressions et remplacement des caractères entre mot A et mot B.

Mot A	Mot B	distance de Levenshtein	explications
examen	examan	1	1 remplacement (changement du *e* en *a* en bleu)
niche	chiens	5	2 suppressions (n et i en rouge) et 3 insertions (i,n,s en vert)
méchant	machine	3	3 remplacements (en bleu)

Calcul de similarités dans InserJeunes

- Pour les noms et prénoms : similarité de Jaro-Winckler
- Similarité ad hoc InserJeunes pour les dates de naissance
- Similarité binaire pour le sexe
- Pour le code COG de la commune de naissance (0,1 ou 0,5)

- Travail de « feature engineering » InserJeunes sur les noms, prénoms et les dates de naissance (ex cas d'inversion).

Classification dans InserJeunes

- Score global établi à la main (fonction des similarités individuelles) et seuil établi à la main.

- Autre option possible testée dans InserJeunes : machine learning comme SVM ou forets aléatoires. Dans notre cas résultats équivalents au score manuel.

- Dans la littérature : classification probabiliste développée originellement par Fellegi et Sunter => nous n'avons pas trouvé d'implémentation rapide de cette méthode dès que le volume de données est assez important.

Evaluation dans InserJeunes

- Appariement qualité (taux d'appariement de 97%)

- Mesure de précision, rappel et f-score sur un échantillon de paires annotées à la main : développement d'une IHM d'annotation pour faciliter le travail. Pour une évaluation donnée nous annotons environ 1 000 paires (sondage stratifié selon le score).

- Dans le cas de l'appariement qualité :
  - la précision est de 95%
  - le rappel est de 99%.

L’outil d’appariement développé par InserJeunes

- Développé en Python (utilise postgresql dont module fuzzystrmatch, librairies Python jellyfish, Pandas et scikit learn)

- Rapide sur nos cas d’usage : l’appariement qualité dure 15 mn.

- Générique :
  - Spécification de chaque appariement en XML (domain specific langage)
  - traçabilité

RÉPUBLIQUE FRANÇAISE

Liberté
Égalité
Fraternité

Dares

Déchiffrer le monde du travail
pour **éclairer** le débat public

depp Direction de l'évaluation, de la prospective et de la performance

ANNEXES

Phonétisation

String	Soundex	Phonex	Phonix	NYSIIS	Double Metaphone	Fuzzy Soundex
peter	p360	b360	p300	pata	ptr	p360
pete	p300	b300	p300	pat	pt	p300
pedro	p360	b360	p360	padr	ptr	p360
stephen	s315	s315	s375	staf	stfn	s315
steve	s310	s310	s370	staf	stf	s310
smith	s530	s530	s530	snat	sm0, xmt	s530
smythe	s530	s530	s530	snat	sm0, xmt	s530
gail	g400	g400	g400	gal	kl	g400
gayle	g400	g400	g400	gal	kl	g400
christine	c623	c623	k683	chra	krst	k693
christina	c623	c623	k683	chra	krst	k693
kristina	k623	c623	k683	cras	krst	k693

Exemple : le Soundex

Supprimer les espaces et mettre le mot en majuscule

Garder la première lettre

Supprimer toutes les occurrences des lettres : a, e, h, i, o, u, w, y (à moins que ce ne soit la première lettre du nom)

Attribuer une valeur numérique aux lettres restantes de la manière suivante (Version pour l'anglais) :

- 1 = B, F, P, V
- 2 = C, G, J, K, Q, S, X, Z
- 3 = D, T
- 4 = L
- 5 = M, N
- 6 = R

Si deux lettres (ou plus) avec le même nombre sont adjacentes dans le nom d'origine, ou s'il n'y a qu'un h ou un w entre elles, alors on ne retient que la première de ces lettres.

Renvoyer les quatre premiers éléments complétés par des zéros si nécessaire.

Similarité de Jaro : exemples

Mots	Paramètres éloignement, m et t	Similarité de Jaro
MARTHA MARHTA	**Eloignement** =6/2-1=2 **m**=6 car pour toute lettre de mot 1 on trouve la même lettre dans mot 2 avec pas plus de 2 positions d'écart. **t**=2/2=1	0,944
DWAYNE DUANE	**Eloignement** =6/2-1=2 **m**=4 : on retrouve D,A,N et E dans les 2 mots à pas plus de 2 positions d'écart correspondants de s1= correspondants de s2 = DANE donc **t**=0	0,822
DIXON DICKSONX	**Eloignement** =8/2-1=3 **m**=4 : on retrouve D,I,O et N dans les 2 mots à pas plus de 3 positions d'écart **t**=0	0,767

Similarité de Jaro-Winckler

$$d_w = d_j + (\ell p(1 - d_j))$$

où :

- d_j est la distance de Jaro entre s_1 et s_2
- ℓ est la longueur du préfixe commun (maximum 4 caractères)
- p est un coefficient qui permet de favoriser les chaînes avec un préfixe commun. Winkler propose pour valeur $p = 0.1$

Mots	Similarité de Jaro (rappel)	Paramètres l et calcul	Similarité de Jaro-Wincler
MARTHA **MAR**HTA	0,944	3 premiers caractères communs donc l=3 donc : 0,944 + [3*0.1 * (1-0,944)]	0,961
DWAYNE **D**UANE	0,822	0,822+ [1*0.1 * (1-0,822)]	0,84
DIXON **DI**CKSONX	0,767	0,767+ [2*0.1 * (1-0,767)]	0,813