# Définitions des besoins :

## Besoins sur les données

### Données pour l’entrainement des modèles

#### Contexte :

Les modèles à entrainer sont des **Sentences Transformers** (une architecture siamoise à base de Transformers). Ces modèles peuvent s’entrainer sur différents type de tâches de classification de paires de textes (la plus connue étant la tâche de **similarité textuelle sémantique**). Le but après est de créer à partir des modèles, des vecteurs sémantiques pour les adresses.

Dans notre cas, nos paires de textes sont des paires d’adresses du type **(Adresse EDI reçue, Adresse destinataire correspondante)**.

#### Source des données :

Les données sont à créer à partir des tables (livraison, livraisonsbis, destinataire, typevoie, voie et commune) (voir Annexe 1) :

* **Adresse EDI reçue** : est une concaténation des champs (livraison.adresse1dest livraison.adresse2dest, commune.codepostal, commune.ville)
* **Adresse destinataire correspondante** : est une concaténation des champs (destinataire.numvoie, typevoie.nomvoie, voie.nomvoie, commune.codepostal, commune.ville)

D’autres données sont importantes à récupérer aussi pour la partie test et validation des modèles. En général, deux tables de données sont nécessaires à créer :

1. **Table LIV avec des champs associés à l’adresse EDI** : livraison.clelivraison, livraison.Destinataire livraisonbis.cledestinataire, livraisonbis.destinataireidentif, destinataire.type, livraison.date\_integration.
2. **Table DEST avec des champs associés à l’adresse destinataire correspondante**: destinataire.cle, destinataire.nom, typevoie.nomvoie, voie.nomvoie, commune.codepostal, commune.ville, destinataire.gpspositionx, destinataire.gpspositiony, commune.typelocalite, destinataire.datecrea

Note : Les requêtes pour la création des deux tables voulues sont dans l’annexe technique (voir Annexe 2).

### Données pour le système de récupération des adresses :

Le récupérateur d’adresses, en termes de données, a besoin d’un datastore d’adresses de référence (adresses normées ou adresses alias) et des adresses requêtes (Adresses EDI)

#### Datastore de récupération :

 Le datastore des adresses et selon le type d’adresses de référence choisi peut être :

* Les adresses concaténées des destinataires récupérés de la table **DEST** déjà créée**.**
* Les adresses concaténées des alias de destinataires récupérés de la table destalias.

Chaque adresse dans le datastore doit être liée à la clé ou les clés destinataire présent dans la table destinataire.

#### Encodage du datastore :

En parallèle du datastore brute, d’autres datastores d’encodages devront être créés et enregistrés. Le nombre de ces datastores dépend du nombre des modèles Transformers entraînés. Le plus important est la compatibilité entre les indices des adresses dans le datastore brute est les indices des vecteurs d’adresses dans les datastores d’encodages. Pour savoir comment encoder un datastore (voir Annexe 3)

#### Récupération et encodage des adresses EDI du jour :

* Les adresses requêtes du jour seront amenées à être créées par requête à partir de la concaténation des champs (livraison.adresse1dest livraison.adresse2dest, commune.codepostal, commune.ville) qui viennent d’être enregistrés dans la table livraison accompagnées de leur clé de livraison.
* Ces adresses de la même manière qu’avec le datastore seront encodées en différents vecteurs selon les modèles disponibles.

## Besoins sur l’environnement de test :

## Besoins pour la liaison entre Model – environnement de test – talk Messagerie:

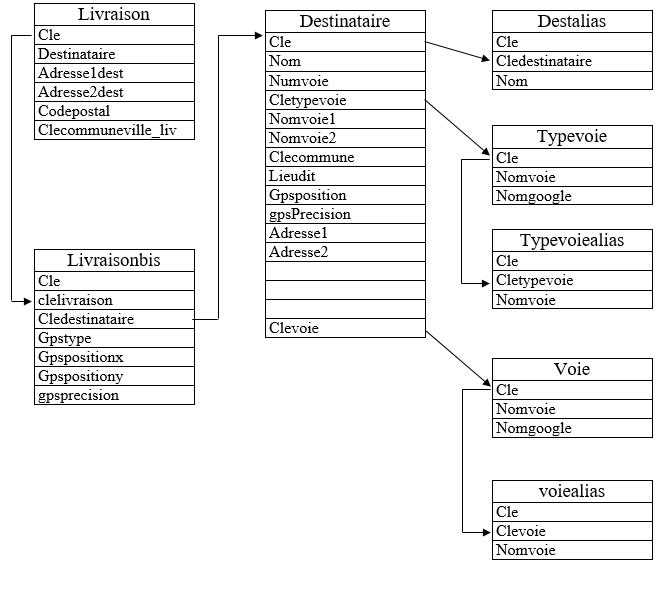
Les marges de confiance de chaque modèle : des similarités au-dessus d’un seuil avec un degré de confiance

# Mise en place d’un environnement de Test :

# Branchement du Model – Environnement de test – Talk Messagerie :

# Annexes technique

## Annexe 1



## Annexe 2

declare @datedebut as datetime

declare @datefin as datetime

set @datedebut={ts '2007-01-01 00:00:00'}

set @datefin={ts '2024-03-25 23:59:59'};

# table LIV

select rtrim(livraison.Destinataire) as Destinataire, rtrim(livraison.adresse1dest)+' '+rtrim(livraison.adresse2dest)+' '+rtrim(commune.codepostal)+' '+rtrim(commune.ville) as Sa

, rtrim(destinataire.nom) as nom, rtrim(destinataire.adresse1)+' '+rtrim(destinataire.adresse2)+' '+rtrim(destinataire.codepostal)+' '+rtrim(destinataire.ville) as SaPrim

,clelivraison, cledestinataire, destinataire.type, isnull(livraisonbis.destinataireidentif, 0) as destinataireidentif, livraison.date\_integration --, destinataire.gpspositionx, destinataire.gpspositiony, destinataire.gpsprecision, livraison.date\_integration --, destinataire.nom--, destinataire.clecommune, destinataire.type

from livraisonbis

inner join livraison on livraison.cle = livraisonbis.clelivraison

inner join destinataire on destinataire.cle = livraisonbis.cledestinataire

inner join typevoie on typevoie.cle = destinataire.cletypevoie

inner join voie on voie.cle = destinataire.clevoie

inner join commune on commune.cle = destinataire.clecommune

where destinataire.cle not in (4611, 22095, 22330, 27327) and destinataire.sommeil is null

and commune.sommeil is null and commune.cledept = 89 and commune.VerifTedies = 1

and typevoie.nomvoie is not null and voie.nomvoie is not null --and destinataire.numvoie = 0 -- and livraisonbis.destinataireidentif is null

and livraison.date\_integration >= @datedebut

and livraison.date\_integration <= @datefin

and livraison.typerecepisse='L'

order by livraison.date\_integration

# table DEST

select rtrim(destinataire.nom) as nom\_dest, rtrim(destinataire.numvoie)+' '+rtrim(typevoie.nomvoie)+' '+rtrim(voie.nomvoie)+' '+rtrim(commune.codepostal)+' '+ rtrim(commune.ville)as adresse\_dest

, rtrim(typevoie.nomvoie)+' '+rtrim(voie.nomvoie)+' '+rtrim(commune.codepostal)+' '+ rtrim(commune.ville) as noNum, rtrim(typevoie.nomvoie)+' '+rtrim(voie.nomvoie) as onlyvoie

, destinataire.cle as cledestinataire, destinataire.clecommune, destinataire.gpspositionx, destinataire.gpspositiony, typevoie.nomvoie as typevoie, voie.nomvoie as nomvoie, commune.codepostal, commune.ville, commune.typelocalite

, destinataire.datecrea

from destinataire

inner join voie on voie.cle = destinataire.clevoie

inner join typevoie on typevoie.cle = voie.cletypevoie

inner join commune on commune.cle = voie.clecommune

where destinataire.cle not in (4611, 22095, 22330, 27327) and destinataire.sommeil is null

and commune.sommeil is null and commune.cledept = 89 and commune.VerifTedies = 1

and typevoie.nomvoie is not null and voie.nomvoie is not null

order by destinataire.datecrea

## Annexe 3

!pip install sentence\_transformers

!pip install torch

from sentence\_transformers import SentenceTransformer

import torch

device = torch.device('cuda:0' if torch.cuda.is\_available() else 'cpu')

trained\_model = SentenceTransformer('Bi\_Encoders/camembert\_models/best/MNR\_GeoSCamembert\_batch\_64\_lr\_3e-5')

# datastore ici est simplement une liste de strings

datastore\_embeddings = trained\_model.encode(datastore, convert\_to\_tensor=True, show\_progress\_bar = False, device=device)