KONTAKT

KONTAKT™ : DES RENCONTRES VRAIES, DES LIENS DURABLES.

HOUSSENALY Ali - GERARD François - DUSOLLIER Baptiste - SECHET Edouard

https://github.com/ali-hou-supaero/ETL_Project_Kontakt

I – INTRODUCTION

Aujourd'hui, malgré la multiplication des réseaux sociaux et la facilité apparente de communication, les relations humaines semblent plus distantes que jamais. Là où nos grands-parents tissaient naturellement des liens avec leurs voisins, leurs collègues ou les gens du quartier, les interactions se sont progressivement déplacées vers des plateformes virtuelles centrées sur la quantité plutôt que la proximité. Les grands réseaux sociaux, pensés pour connecter le monde entier, favorisent les échanges à longue distance mais peinent à créer du lien réel et local entre personnes pourtant physiquement proches.

C'est en partant de ce constat que nous est venue l'idée de créer un nouveau réseau social basé plutôt sur la proximité et la qualité des rencontres entre personnes.

L'objectif de ce rapport sera d'abord de présenter notre start-up, puis de montrer en quoi celle-ci crée de la valeur à partir des données collectées, et enfin d'illustrer cette approche par un exemple de pipeline ETL.

II- LA START-UP

A) CONCEPT ET MISSIONS

KONTAKT, c'est la nouvelle manière de recréer du lien social authentique. Bien plus qu'une simple application, c'est un véritable réseau de proximité qui remet l'humain au cœur des interactions.
Son ambition se décline en trois missions essentielles :

- **Créer du lien humain au quotidien**, en facilitant les rencontres spontanées entre personnes proches géographiquement ;
- **Favoriser la convivialité et l'inclusion**, en ouvrant les échanges à tous, sans barrières sociales ni culturelles ;
- **Lutter contre l'isolement social**, en reconnectant ceux qui se sentent seuls à leur environnement local ;

EXEMPLE CONCRET

Imaginez : vous êtes assis dans le métro, le regard perdu ou fixé sur votre téléphone pour faire passer le temps. Vous aimeriez discuter avec vos voisins de rame, mais franchir le pas et aborder un inconnu reste difficile pour la plupart d'entre nous. C'est précisément là que KONTAKT intervient. L'application agit comme un véritable icebreaker, en facilitant la première approche. Il suffit de sortir son smartphone et de lancer KONTAKT pour rejoindre automatiquement un chat commun réunissant tous les utilisateurs présents dans la même rame. En quelques messages, la conversation s'engage naturellement, transformant un trajet ordinaire en un moment d'échange.
KONTAKT permet aussi d'ajouter des personnes à proximité et de continuer la discussion en privé. L'objectif final est simple mais essentiel : encourager les utilisateurs à passer du virtuel au réel, pour faire naître des rencontres authentiques et durables.

B) FONCTIONNALITÉS PRINCIPALES

- **Géolocalisation intelligente** : Envoie d'une notification aux utilisateurs présents dans un cluster (N > 5 utilisateurs, Rayon = 150m)
- **Tchat commun et privé**
- **Ajout d'amis**
- **Recommandations utilisateurs** : événements à proximité, trajet recommandé pour rencontrer du monde, ...

C) BUSINESS MODEL

Le modèle économique de KONTAKT repose sur une approche freemium : une version gratuite ouverte à tous, complétée par des fonctionnalités premium offrant davantage de visibilité ou l'accès à des événements exclusifs. Par ailleurs, la plateforme mise sur une monétisation éthique des données, en proposant uniquement des analyses agrégées et anonymisées destinées à fournir des informations locales aux commerces de proximité, sans compromettre la vie privée des utilisateurs. Enfin, nous intégrerons également de la publicité, en privilégiant autant que possible une publicité locale et en lien direct avec les commerces ou événements du quartier.

III- APPROCHE DATA-DRIVEN DE KONTAKT

A) TYPES DE DONNÉES COLLECTÉES

- **Données utilisateurs** : âge, genre, centres d'intérêt, préférences.
- **Données comportementales** : temps de connexion, fréquence d'interactions, distance moyenne des contacts
- **Données de localisation** : zones de forte activité ("hotspots"), coordonnées des utilisateurs (latitude, longitude), vitesse de déplacement
- **Retour utilisateur** (notation de l'appli, signalements)

B) OBJECTIFS DE VALORISATION DES DONNÉES

- Améliorer la recommandation locale (par ex. identifier les heures où les gens se croisent le plus).
- Produire des indicateurs d'activité urbaine (ex. densité d'usagers par zone).
- Aider les commerces partenaires à comprendre les dynamiques locales (trafic, horaires, profils typiques).

IV- EXEMPLE TECHNIQUE : PIPELINE ETL

A) OBJECTIF DE LA PIPELINE

L'objectif du pipeline ETL est d'extraire les données de notre application (E), de les transformer (T) (outliers et valeurs manquantes) et ensuite de les charger dans une base de données SQL pour pouvoir les utiliser dans nos différentes fonctionnalités. Le pipeline ETL, dans notre cas, doit tourner toutes les 5 minutes, de manière à mettre à jour de manière régulière la localisation des utilisateurs actifs.

B) JEU DE DONNÉES

Dans le cadre de ce projet, nous avons choisi de **générer nous-mêmes nos données** afin qu'elles reflètent au mieux les informations réelles que nous pourrions collecter si notre service était effectivement en production.

Nous **simulons ainsi les données de 10 000 utilisateurs**, avec les champs suivants : `user_id`, `timestamp`, `latitude` et `longitude`.

Chaque ligne correspond à un utilisateur de la plateforme. Le `timestamp` représente un **« heartbeat »**, c'est-à-dire une actualisation toutes les cinq minutes. À chaque intervalle, la position géographique (latitude et longitude) de l'utilisateur est mise à jour, ce qui permettrait, dans un cas réel, de **recommander des rencontres, envoyer des notifications ou inviter à des événements** en fonction de la localisation en temps réel.

Dans le cadre de notre simulation, nous nous plaçons à **18h le 8 octobre**. En pratique, ce code pourrait être exécuté automatiquement toutes les cinq minutes sur une **machine virtuelle hébergée sur le Cloud**. Pour simuler ce fonctionnement, nous avons donc fixé une date et une heure définies.

Pour un déploiement réel du projet Kontakt, il serait également pertinent de **mesurer la vitesse des utilisateurs**, voire **leur altitude ou profondeur**, afin de détecter des situations comme des utilisateurs se trouvant dans le même train ou dans une station de métro.
Dans notre simulation, nous faisons l'hypothèse simplificatrice d'utilisateurs **immobiles et situés au même niveau d'altitude**, par exemple dans un parc, sur une place ou dans un bar.

Nos données sont générées selon la logique suivante :

- **10 000 points** sont créés au total.
- **70 %** des utilisateurs sont localisés dans de grandes villes :
  - 50 % à **Paris**,
  - 10 % à **Marseille**,
  - 10 % à **Lyon**.
- **30 %** des utilisateurs sont répartis aléatoirement sur le territoire français.
- **10 %** des enregistrements présentent un **timestamp aléatoire** à un autre moment de la journée, simulant des cas réels tels qu'une **perte de batterie**, un **mode avion activé**, ou une **impossibilité de récupérer la localisation**.

Cette génération permet de **simuler fidèlement le type de données** que nous pourrions effectivement obtenir si l'application et son **infrastructure Cloud / API** étaient mises en place.

La génération des données est présentée sous la forme d'une carte de la France dans la partie *Annexe 1*

C) LA BASE SQL

Aperçu de la table users SQL :

```
kontakt_db=# \dt
        List of relations
 Schema | Name  | Type  |  Owner
--------+-------+-------+---------
 public | users | table | fgerard
(1 row)

kontakt_db=# \d users
                        Table "public.users"
  Column   |           Type           | Collation | Nullable | Default
-----------+--------------------------+-----------+----------+---------
 user_id   | bigint                   |           |          |
 timestamp | timestamp with time zone |           |          |
 latitude  | double precision         |           |          |
 longitude | double precision         |           |          |

kontakt_db=# SELECT * FROM users LIMIT 5;
 user_id |       timestamp        |      latitude      |     longitude
---------+------------------------+--------------------+--------------------
    6253 | 2025-10-08 18:00:00+02 | 43.306915886113856 |  5.353270172485795
    4685 | 2025-10-08 18:00:00+02 |   48.8358586785286 |  2.336603866485248
    1732 | 2025-10-08 18:00:00+02 |  48.84961366407764 |  2.334522622097185
    4743 | 2025-10-08 18:00:00+02 |  48.85663238062714 | 2.3229939512958158
    4522 | 2025-10-08 18:00:00+02 | 48.878086660095406 | 2.3918494090033784
(5 rows)
```

D) ÉTAPES ETL

EXTRACT

La première étape consiste à extraire les données du jeu de données. Dans notre approche simple de génération de données pour le proof of concept, les données se trouvent déjà dans un fichier csv. Il suffit donc d'utiliser la bibliothèque pandas de python pour agencer les données sous la forme d'un Dataframe. Ce format est ensuite le plus idéal pour effectuer la partie transform.

Mais en réalité, les données seront générées dans une api et l'extraction consistera à se connecter à l'api et à extraire les données de celle-ci dans un Dataframe.

TRANSFORM

Cette étape consiste à effectuer un pré-traitement des données pour les rendre utilisables dans nos fonctions. Les erreurs dans les données viennent essentiellement d'erreurs de mesures. On effectue plusieurs traitements :

- Suppression des lignes contenant des données manquantes
- Suppression des lignes dont la localisation est erronée(latitude [-90°, +90°], longitude [-180°, +180°]
- Suppression des lignes dont l'heure du timestamp utilisateur ne correspond pas à l'heure actuelle.
- Forçage des types de données : numeric pour la localisation et datetime pour le timestamp

LOAD

Le module load a pour objectif d'automatiser l'intégration des données nettoyées dans une base de données PostgreSQL.

Pour cela, on procède à la création d'une base PostgreSQL et on charge les données transformées dans le DataFrame. Ainsi, toutes les données utilisateurs se trouvent dans la base de données users. Ce chargement est effectué via la bibliothèque SQLAlchemy, qui permet de convertir et d'insérer automatiquement les données dans les tables tout en gérant les éventuelles recréations de celles-ci.

Par ailleurs, le script assure aussi la vérification du bon déroulement du chargement en confirmant la présence des tables requises et en affichant des exemples de données chargées dans la base

Enfin, le module propose quelques requêtes exploratoires telles que le calcul du nombre d'utilisateurs à Paris, Marseille et Lyon ainsi que le pourcentage de ces utilisateurs par rapport au total. Le module permet de migrer les données vers un système de stockage structuré, facilitant par la suite l'analyse et l'exploitation des données.

E) Résultats attendus / utilisation de la base de données

Les résultats de la partie ETL sont présentés en ***Annexe 2.***

Ainsi, en mettant ses données sous la forme d'une base SQL, cela a permis de centraliser et structurer les données nettoyées. La base SQL assure une meilleure fiabilité, traçabilité et performance pour le stockage, la mise à jour et la manipulation de grands volumes de données.

Voici un exemple d'utilisation de la base de données pour l'ajout d'une fonctionnalité à l'application : l'**envoi de notifications intelligentes**

Exemple concret : envoi de notifications intelligentes

Lorsqu'un nombre conséquent (ex : N > 10) d'utilisateurs se trouve dans un périmètre restreint (R < 200m) sur une période donnée (ex : T = 10 min = 2 maj de la localisation), les utilisateurs présents dans ce cluster reçoivent une notification pour les inviter à se rencontrer ou à échanger par message.

En pratique, la fonctionnalité utilise l'algorithme DBSCAN pour clusteriser les utilisateurs avec la mesure de Haversine (calcul de la distance entre deux points sur une sphère).

Une spécificité non prise en compte dans notre exemple de pipeline est la dépendance des paramètres aux nombres d'utilisateurs de la ville. En effet, idéalement, pour chaque ville un rayon et un nombre d'utilisateurs minimal est fixé en fonction du nombre d'utilisateurs total dans cette ville. Par exemple, les périmètres de rencontre pour Paris doivent être minimisés afin de créer des clusters de taille faible incitant à une interaction physique entre les utilisateurs.

V- DÉFIS ET PERSPECTIVES

A) Défis techniques

KONTAKT fait face à plusieurs défis techniques importants. L'un des principaux consiste à **déterminer la bonne fréquence de rafraîchissement de la localisation des utilisateurs** : trop élevée, elle entraînerait une consommation excessive de batterie et une surcharge du réseau ; trop faible, elle nuirait à la précision et à la réactivité de l'application. Par ailleurs, **la scalabilité de la base de données** représente un autre enjeu clé, afin d'assurer un fonctionnement fluide même avec un grand nombre d'utilisateurs connectés simultanément. Enfin, **la sécurité et l'anonymisation des données** demeurent essentielles pour garantir la protection de la vie privée et le respect des normes de confidentialité.

B) Défis éthiques et réglementaires

Notre plateforme devra naturellement **se conformer au RGPD** et veiller au **respect strict de la vie privée des utilisateurs**. La protection des données personnelles est au cœur du projet KONTAKT : chaque information collectée devra être utilisée de manière transparente, sécurisée et proportionnée. Il sera également essentiel de **prévenir toute dérive potentielle**, qu'il s'agisse de pratiques de **traçage abusif** ou de **comportements inappropriés** entre utilisateurs, afin de garantir un environnement de confiance et de bienveillance.

C) ÉVOLUTIONS POSSIBLES

Dans l'avenir, certaines évolutions pourraient permettre à KONTAKT de franchir une nouvelle étape dans la valorisation des données.

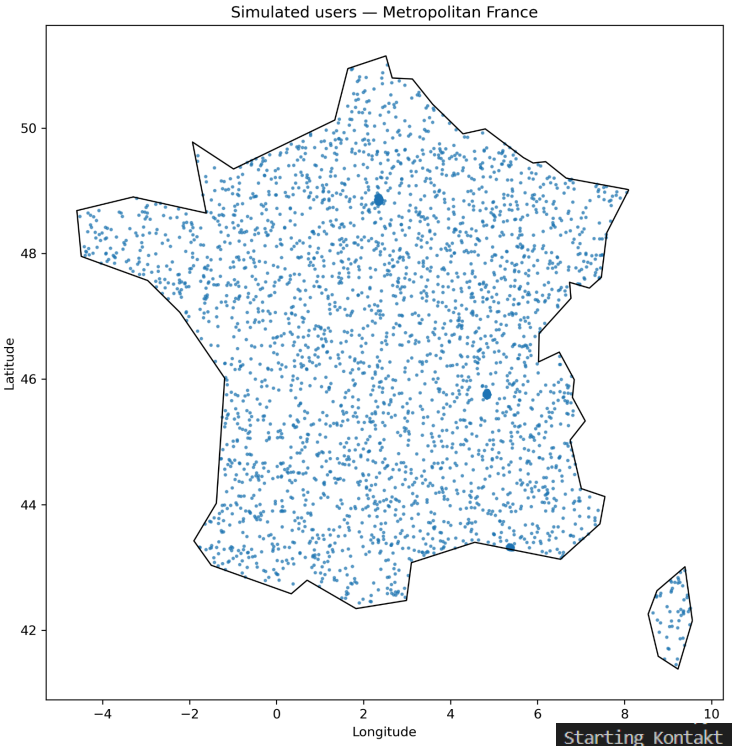
Ainsi l'ajout d'Intelligence Artificielle à travers l'usage de modèles prédictifs offrirait la possibilité **d'analyser et d'anticiper les comportements de mobilité locale**, par exemple en identifiant les zones et les moments où les interactions sont les plus susceptibles de se produire.

Ces analyses permettraient d'améliorer la pertinence des suggestions faites aux utilisateurs et d'optimiser l'expérience de rencontre en temps réel.

VI- CONCLUSION

L'application KONTAKT propose une nouvelle vision des réseaux sociaux : **plus locale, plus humaine et plus authentique**. En s'appuyant sur les données de manière éthique et intelligente, l'application facilite les **rencontres réelles entre personnes proches**, tout en respectant la vie privée de chacun. Grâce à son approche data-driven et à une base technique solide, KONTAKT ambitionne de devenir un **acteur du lien social moderne**, capable de redonner du sens aux interactions du quotidien.

VII- ANNEXES



Annexe 1 : *Aperçu de la localisation des utilisateurs à un horaire précis -*

Annexe 2 *: Visualisation de la pipeline ETL*

```
Starting Kontakt ETL Pipeline...
================================================

=== EXTRACTION ===
📥 Extracting data from sources...
📄 Reading user data from CSV...
Loaded 10000 users

=== TRANSFORMATION ===
🔄 Cleaning and transforming data...
🧹 Cleaning user data...
Starting with 10000 users
After cleaning: 9000 users remain

=== LOADING ===
💾 Loading data to database...
💾 Loading data to PostgreSQL database...
✅ Loaded 9000 users to database

=== VERIFICATION ===
✅ Verifying data was loaded correctly...
🔍 Verifying data was loaded correctly...
📊 Users in database: 9000

📋 Sample users:
 user_id                 timestamp  latitude  longitude
    6253 2025-10-08 16:00:00+00:00 43.306916   5.353270
    4685 2025-10-08 16:00:00+00:00 48.835859   2.336604
    1732 2025-10-08 16:00:00+00:00 48.849614   2.334523
📈 Running sample analysis queries...

🌍 Number of users in Marseille area:
 count
  1000

🎉 ETL Pipeline completed!
================================================
```

Annexe 3 *: Aperçu des différents clusters*

KONTAKT

```
📍 Clusters détectés :
Cluster 1: [40, 2026, 3922, 2304, 4150, 4869, 4867, 2997, 3079, 878, 4247, 232, 3368, 2399, 4288, 4370, 3576, 2380, 1012, 2673, 3569, 4110, 342, 3463]
Cluster 2: [108, 2928, 484, 4363, 4124, 711, 4342, 13, 386, 1281, 2143, 3383, 4727, 3659, 3069, 4014, 3793, 2891, 2162, 3930, 1276, 3872, 4680, 3869, 963,
 3418, 4381, 4062, 1960]
Cluster 3: [2754, 3781, 1878, 3746, 1841, 3245, 2437, 3880, 4620, 3856, 2606, 1895, 1516, 498, 871, 3642, 3169, 1662, 3736, 660]
Cluster 4: [972, 743, 1624, 4345, 749, 4025, 4635, 4040, 3164, 1989, 1392, 1217, 4411, 2225, 156, 2644, 703]
Cluster 5: [1189, 3146, 3007, 736, 361, 4338, 4476, 684, 3028, 2192, 1196, 3521, 2992, 2690, 4694, 4053, 1702, 1169, 4925, 1420, 477, 1026, 4234, 1990, 28
84, 4440, 2742, 1322, 2953, 4162, 3947, 4058, 4846, 4487, 3247, 1412, 3200, 3136, 1263, 4391, 2653, 2669, 3493, 4546, 1310, 2170, 4797, 2262, 1899, 1568,
2131, 690, 1134, 3180, 3041, 4704, 4037, 1522, 2337, 2713, 1343, 10, 1168, 785, 4637]
Cluster 6: [1964, 959, 4834, 2733, 618, 1198, 434, 4833, 1707, 1500, 1597, 4509, 4560, 3251, 4409, 2242, 202, 1448, 2528, 3918, 2296, 3070, 2212, 2427, 21
60, 2469, 3192, 3726, 4359, 3274, 208, 4474, 3931, 2915, 2480, 4200, 3891, 3557]
Cluster 7: [81, 2778, 4353, 3492, 993, 310, 3117, 3288, 1242, 4168, 4753]
Cluster 8: [89, 701, 2089, 1545, 4394, 4566, 2215, 470, 1491, 130, 4935, 3848, 821, 2087, 2504, 1238, 3057, 557, 828, 2372, 761, 729, 1529]
Cluster 9: [4218, 2030, 3707, 1673, 1480, 1627, 4690, 3861, 1018, 4292, 2269, 2730, 4792, 634, 52, 2272, 4848, 4386, 3821, 4477, 2942, 3130, 2436, 4754, 4
664, 2785, 4621, 1933, 3450, 3737, 4009, 449, 1654, 2357, 2600, 4301, 4911, 1082, 2328]
Cluster 10: [3401, 4722, 1021, 3339, 4157, 3182, 2862, 4484, 1965, 3951, 4697, 3516, 4936, 2513, 2556, 3977, 4226, 392]
Cluster 11: [2068, 305, 4774, 368, 451, 3131, 3310, 316, 1146, 2237, 4049, 2274, 1311, 3035, 2123, 915, 4346, 3129, 4824, 3657, 3317, 2132, 4615, 4825, 29
88, 1367, 3504, 3863, 2779, 3093, 2198, 1233, 3780, 2828, 2468, 3011, 2805, 3766, 43, 509, 3327, 493, 491, 4321, 2491, 2385, 903, 1830, 1890, 990, 3404, 4
137, 2925, 2823, 3738, 906, 2172, 227, 2614, 23, 4245, 1561, 3611, 4570, 1342, 3550, 1829, 3585, 2540, 4015, 4894, 4738, 2694, 2048]
Cluster 12: [2616, 4890, 4376, 2277, 2315, 4819, 160, 4677, 4244, 2455]
Cluster 13: [2674, 2485, 859, 3144, 1983, 4066, 2085, 3015, 2719, 402, 469, 802, 1633, 4367, 1472, 4277, 4571, 3955, 3632, 4350, 61, 4663, 4893, 4493, 296
8, 2850, 4870, 3328, 302]
Cluster 14: [440, 1057, 3818, 4984, 1473, 4515, 4099, 1733, 4868, 2708, 3556, 2755, 3644, 1551, 3149, 4230, 2795, 1771, 1045, 2112, 835, 2064, 1282, 3690,
 2583, 1435, 4793, 4716, 67, 2590, 4529, 2987, 1515, 2339, 663, 4865, 658, 110, 571, 4595, 3613, 2371, 3902, 4648, 188, 1768, 2047, 3416, 2395, 1782, 1092
, 4309, 1885, 633, 1267, 779, 363, 1934, 4821, 2558]
Cluster 15: [3527, 1742, 4695, 778, 4681, 1840, 574, 1555, 1371, 1333, 3958, 3347, 4086, 311, 1858, 4987, 2387, 2917, 1523, 3794, 3831, 4156, 2773]
Cluster 16: [1225, 1091, 4930, 4039, 512, 767, 1145, 1296, 1096, 320, 3434, 4993, 2062, 4405, 3716, 4020, 3540, 1328, 4547, 2972, 4959, 3719, 33, 2079, 19
97, 4963, 2332, 2811, 3230, 73, 3639, 1950, 2470, 4151, 4548, 4972, 1647, 700, 3447]
Cluster 17: [1085, 4186, 1684, 4314, 2475, 555, 2841, 69, 478, 1222, 4470, 559, 2419, 3936, 901, 957]
Cluster 18: [4203, 3205, 4398, 1370, 4520, 2302, 3523, 3884, 2534, 182, 2794, 122, 2032, 3302, 3227, 2643, 2451, 1820, 1327, 4967, 1725, 3360, 3226, 3175,
 3597, 3171]
Cluster 19: [2216, 4068, 4983, 737, 351, 3440, 3752, 175, 3787, 116, 4125]
Cluster 20: [4964, 319, 1278, 59, 4519, 2890, 4393, 4404, 2500, 2677, 1451, 4815, 3614, 4267, 2101, 728, 1705, 1291, 4147]
Cluster 21: [4081, 2108, 223, 1223, 2265, 446, 4460, 1005, 798, 425, 4645, 365]
Cluster 22: [30, 3062, 246, 1902, 4419, 4340, 3458, 4768, 433, 3287, 806, 1336, 58, 280, 1976, 3682, 3184, 4724, 1339, 691, 2799, 2786, 1280, 4969]
Cluster 23: [2679, 591, 1394, 2246, 2743, 4100, 4448, 1363, 436, 2919, 1201, 369, 2293, 1629, 281, 2461, 4215, 4854, 3757, 2137, 3473, 4720, 4469]
Cluster 24: [1632, 1766, 2589, 1604, 1503, 3056, 4610, 4862, 137, 2452, 1553, 1848, 2052, 4737]
Cluster 25: [2944, 1104, 3669, 3678, 552, 105, 3533, 548, 2880, 3099, 1894, 1757, 3916, 2251, 4822, 3600, 4957, 2016, 1228, 1716, 4294, 989]
```