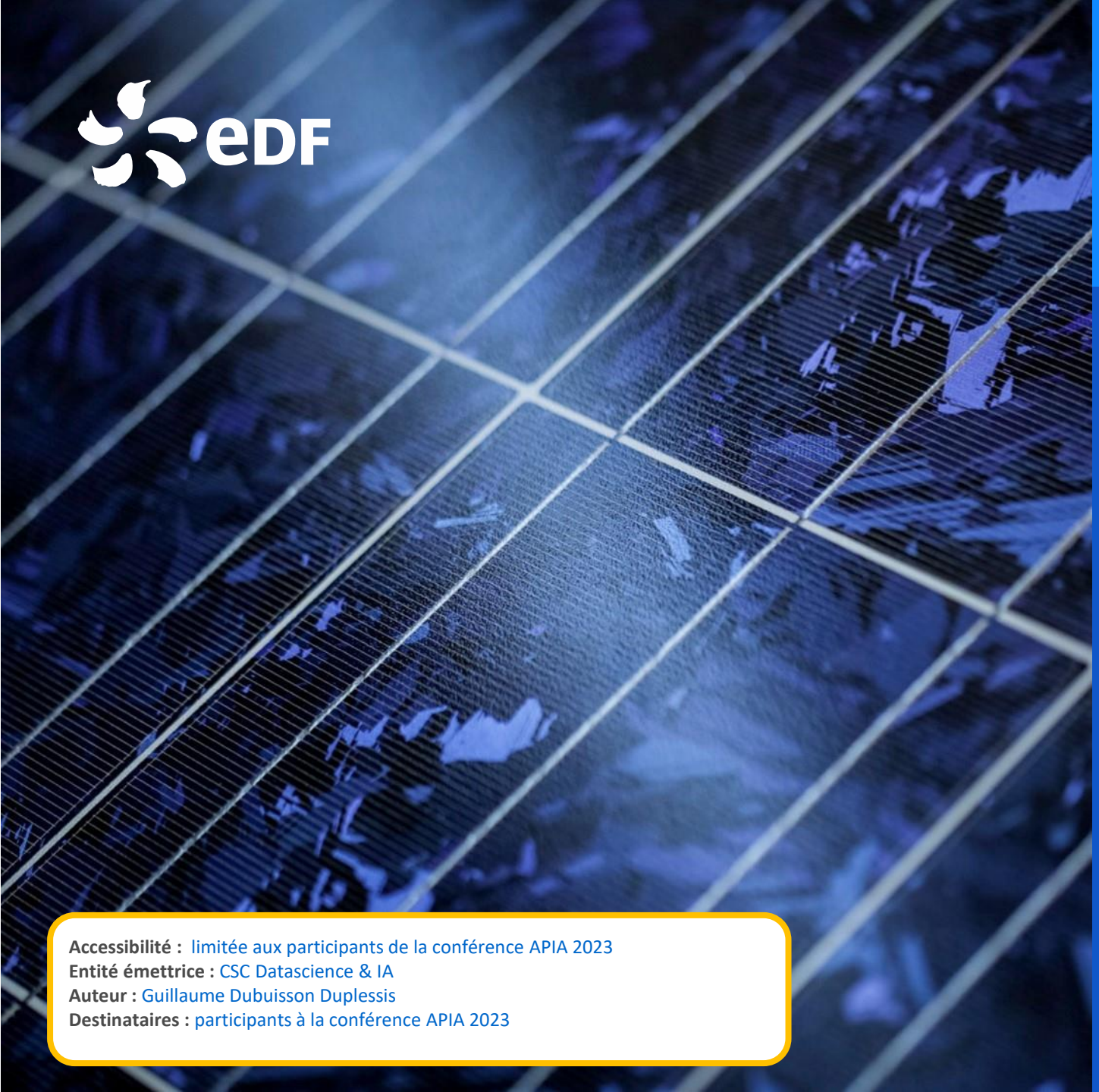


Accessibilité : limitée aux participants de la conférence APIA 2023
Entité émettrice : CSC Datascience & IA
Auteur : Guillaume Dubuisson Duplessis
Destinataires : participants à la conférence APIA 2023

EDF – Direction des Systèmes d'Information et du Numérique

Exploration sémantique de données texte de la relation client

Démonstration de « nemo »

Conférence APIA 2023

Auteurs : **Guillaume Dubuisson Duplessis, François Bullier (externe), Anne-Laure Guénet**

DSIN - CSC Datascience & IA
Responsable CSC : **Sonia Audheon**
Responsable déléguée : **Laura Rouhier**

Responsable opérationnelle équipe « données non-structurées » :
Anne-Laure Guenet

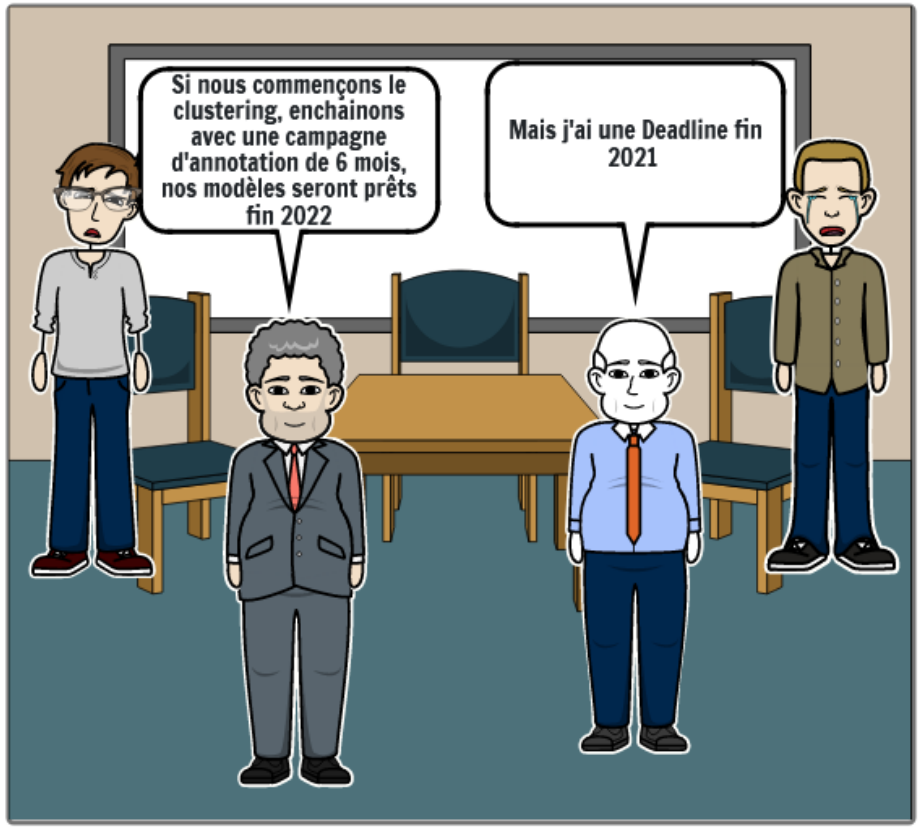
Sommaire

1

Introduction à « nemo »

- Contexte et motivations
- Objectifs
- Survol de quelques fonctionnalités

Motivation 1 : les demandes d'exploration de données texte sont les plus fréquentes (et les moins bien formalisées)

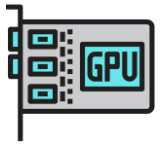


© Philippe VERY et Mourad MILED

Motivation 2 : les techniques de « deep learning » offrent des représentations plus riches et « sémantiques »

« Self-supervised learning » (cf. Yann LeCun)

Données texte « brutes »
désidentifiées



Modèles pré-entraînés
(de type « word embeddings »)

Exemple de « word embeddings »

électricité	→	EDF	insatisfait	→	réclamation	dp	→	délai
gaz	→	GDF	satisfait	→	demande	mens	→	mensualités

Analogies réalisées avec un modèle GloVe entraîné sur les données de la relation client d'EDF

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove : Global vectors for word representation. In Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, 2014.

Exploration de corpus de texte avec « nemo »

Objectif

- Explorer mieux et plus rapidement (*) des corpus de données texte EDF Commerce non-annotés (ou partiellement annotés) via des techniques opérationnelles de « deep learning »

Gains visés

- Gain en efficience comparativement aux approches existantes (LDA [1], iramuteq [2])

Principaux livrables

- Prototype « nemo » (« Neural sEMantic explOration »)

Principaux domaines techniques

- Apprentissage non-supervisé, auto-supervisé, semi-supervisé
- Apprentissage de représentation (contraste)
- Visualisation de donnée
- Réduction de dimension
- Clustering, topic modelling
- Similarité de texte

(*) « mieux » : obtenir des clusters plus informatifs grâce à des représentations plus riches
« plus rapidement » : pas ou peu (HTML) de preprocessing

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. Journal of machine Learning research, 3(Jan) :993–1022, 2003.
[2] Pierre Ratinaud and Pascal Marchand. Application de la methode alceste a de “gros” corpus et stabilité des “mondes lexicaux” : analyse du “cablegate” avec iramuteq. Actes des 11eme Journees internationales d’Analyse statistique des Donnees Textuelles, pages 835–844, 2012.

Survol des quelques fonctionnalités de nemo (1/5)


Embed


Cluster


Visualize & explore


Describe


Semantic search

In [5]:
```
from nemo import EmbeddingsExplorer
```

Encode and visualize the dataset

In [9]:
```
# Get an Embedding Explorer object
eda = EmbeddingsExplorer()
```

Next, let's encode the dataset (it may take some time and it will take some memory on the GPU):

In [10]:
```
# Transform docs to sentence embeddings (768 dims per sentence)
import torch
torch.cuda.set_device(1)
embeddings = eda.get_embeddings(docs)
```

As a good practice, let's drop duplicate documents from the dataset (this will allow to get more useful clusters):

In [11]:
```
# Drop duplicates from docs
docs, embeddings, _ = eda.extract_clusters([-1], eda.get_duplicates(docs), docs.values, embeddings)
```

Survol des quelques fonctionnalités de nemo (2/5)

Embed → Cluster → Visualize & explore → Describe → Semantic search

Hierarchical Clustering

First, let's import the clustering algorithm:

In [12]:
```
from nemo import clustering as clust
```

Let's precompute the necessary information for hierarchical clustering (this may take some time, have a break :-):

In [13]:
```
# Let's clustering with the 768-D embeddings
hc = clust.hclust(embeddings)
```

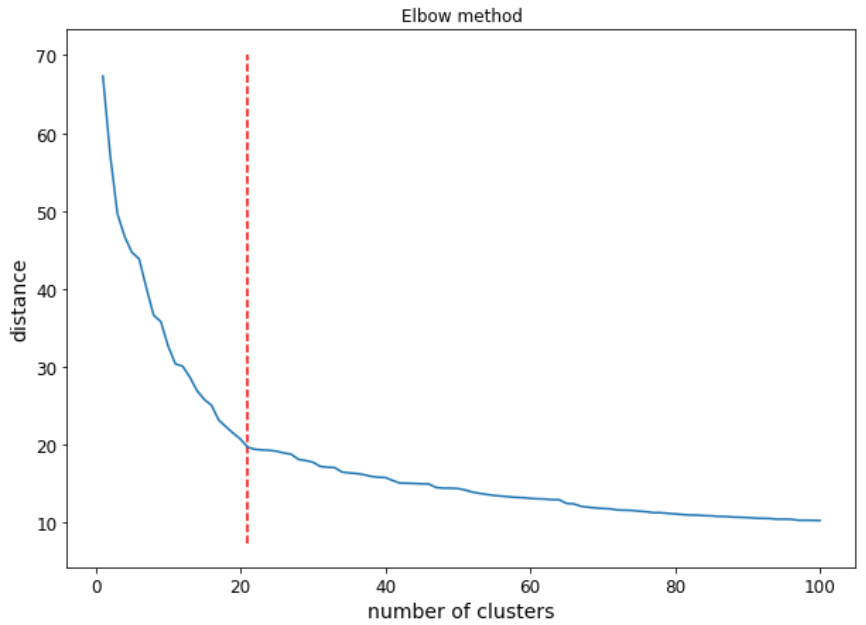
Determination of the optimal number of cluster

Two heuristics are usually used to determine the optimal number of clusters:

- the elbow method), and
- the silhouette score)

In [14]:
```
# Elbow method
nb_cluster_elbow = clust.get_number_of_clusters(hc, method="elbow", show_figure=True) # default
```

```
Nb clusters: 21
```



Computation of the clusters

Let's stick with the result of the elbow method which was crystal clear:

In [15]:
```
clusters = clust.get_clusters(hc, nb_cluster_elbow)
```

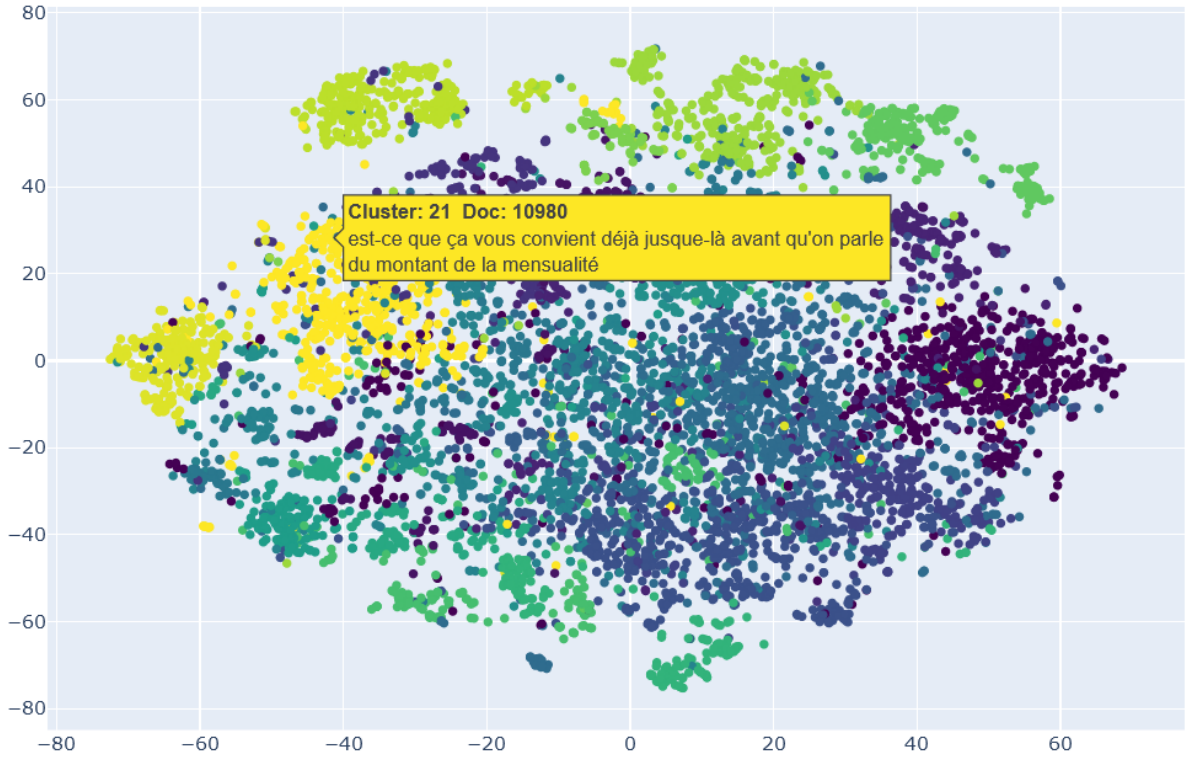
Survol des quelques fonctionnalités de nemo (3/5)

Embed → Cluster → **Visualize & explore** → Describe → Semantic search

Interactively visualize the dataset and the clusters

In [16]:
```
# Reduce embeddings in 2D
embeddings_2d = eda.get_2d_embeddings(embeddings)
```

In [17]:
```
# Display interactive 2D embeddings without clustering outliers
eda.show_2d_interactive_embeddings(embeddings_2d=embeddings_2d,
                                   clusters=clusters,
                                   docs=docs,
                                   docs_index=np.array(range(len(docs))),
                                   sample_size=8000
                                  )
```



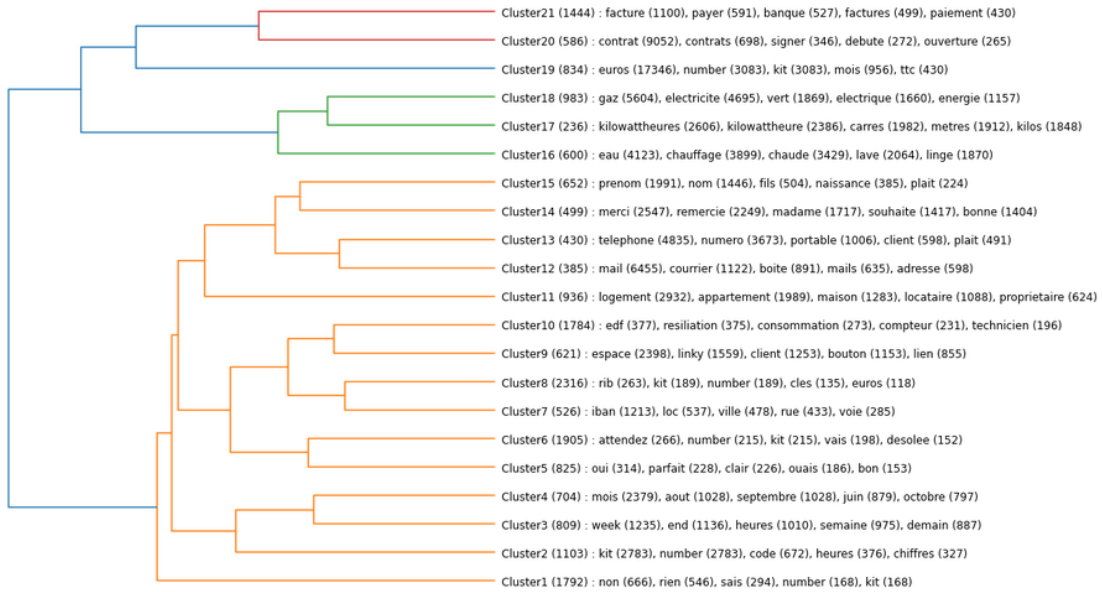
Survol des quelques fonctionnalités de nemo (4/5)

Embed → Cluster → Visualize & explore → **Describe** → Semantic search

Visualizing the hierarchy of clusters (dendrogram)

In [19]:

```
clust.show_dendogram(hc=hc,
                     nb_clusters=nb_cluster_elbow,
                     labels=labels,
                     orientation="left",
                     height = 12,
                     width=10)
```



Discover prototypical documents of a given cluster

In [21]:

```
# Search for relevant examples in the cluster (prototype)
cluster_number = 20
cluster_indexes = np.where(clusters == cluster_number)
proto_indexes, proto_perc = eda.get_prototypes(embeddings[cluster_indexes], 5)
print("Nb prototypes:",len(proto_indexes))
print("Percentage of prototypes:", proto_perc)

eda.display_docs(docs[cluster_indexes][proto_indexes])
```

```
Nb prototypes: 5
Percentage of prototypes: [0.41 0.18 0.15 0.12 0.14]

 euh faire le basculement de contrat

 des que vous le recevez vous me le dites je vous envoie votre contrat par mail j'aurais besoin donc de signature de votre part pour pouvoir valider bien sur

 alors j'accede juste a ce qui a ete historise puisque comme le contrat est je vois qu'il est en cours d'ouverture euh j'ai pas les memes capacites au service facturation alors attendez je prends connaissance de tout ca

 oui bonjour je me permets de vous appeler parce qu'on a un contrat [loc.voie]

 pas de probleme est-ce que faut aussi prevoir l'arret d'un contrat sur lequel vous etiez enfin le contrat que vous aviez avant ou pas
```

Survol des quelques fonctionnalités de nemo (5/5)

Embed


Cluster


Visualize & explore


Describe


Semantic search

Natural language similarity search

Similarity search : "je n'ai plus de courant"

In [22]:
```
similar_idx = eda.similarity_search("je n'ai plus de courant", embeddings=embeddings)
eda.display_docs(docs[similar_idx], topn=5)
```

```
on m'a on m'a coupe l'electricite donc faut que je retrouve l'electricite comment je fais

euh je vais pas avoir d'electricite pendant [kit.number] semaines

mais on est en panne euh d'electricite completement enfin le disjoncteur

a [kit.number] heures je n'ai plus du tout d'electricite

j'ai tout sauf l'electricite mais je
```

Similarity search : "je ne suis pas au courant"

In [23]:
```
similar_idx = eda.similarity_search("je ne suis pas au courant", embeddings=embeddings)
eda.display_docs(docs[similar_idx], topn=5)
```

```
euh je sais pas je pourrai pas vous dire

mais je sais pas je je j'ai pas eu d'infos la-dessus

euh non je je sais pas non je crois pas

d'accord ouais ouais oh bah je sais pas

donc je sais pas ce que ca veut dire le
```

2

Spécialisation de modèles pour l'exploration de données

- Spécialisation de modèles sur le domaine EDF Commerce
- Un exemple sur un cas concret
- Quelques astuces pour le fine-tuning

Spécialiser des modèles pour l'exploration de données texte de la relation client d'EDF

Aller plus loin

Spécialiser des modèles de « sentence embeddings » pour l'exploration de données texte de la relation client d'EDF en encodant de la connaissance du domaine de la relation client EDF

Gains

- Mieux explorer les données texte et plus rapidement
- Rapprocher exploration et modélisation (augmentation de données, apprentissage semi-supervisé)

Solution technique retenue

Fine-tuning de modèle de sentence-embeddings

Exemple : faciliter l’exploration de nouveaux datasets grâce aux modèles spécialisés (1/2)

Exploration de données du SVI en LN

« je voudrais savoir pourquoi il y a un relevé de mon compteur alors que j'ai un compteur Linky qui a été installé s'il vous plaît »

« j'ai besoin d'un certificat de un justificatif de domicile »

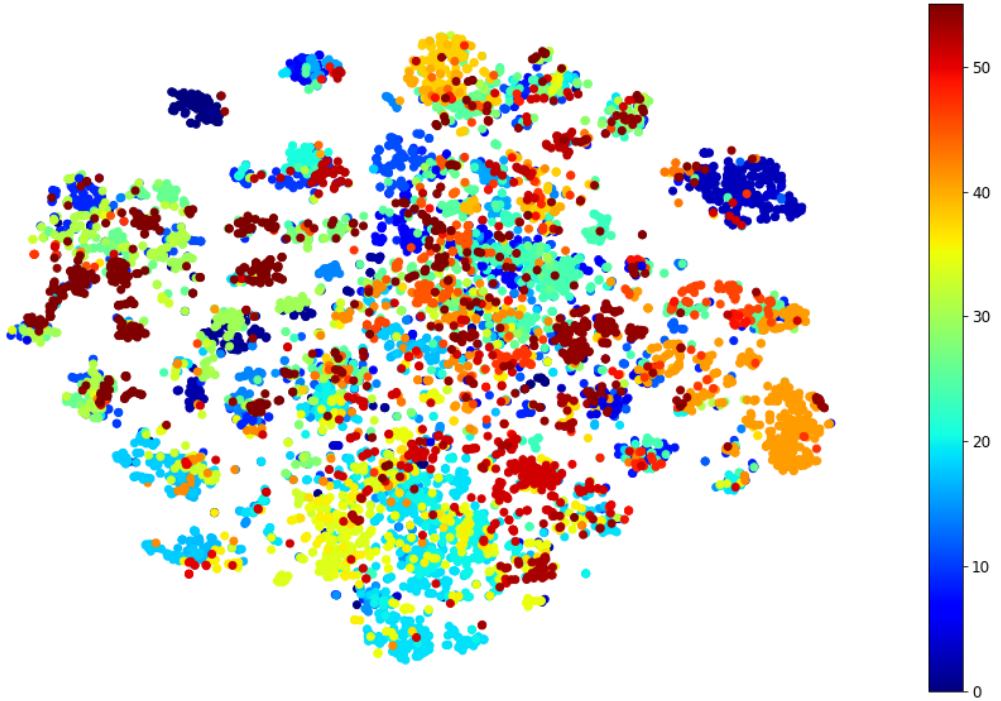
« c'est par rapport à la facture »

« il y a une erreur euh de d'adresse sur ma facture », « adresse incorrecte »

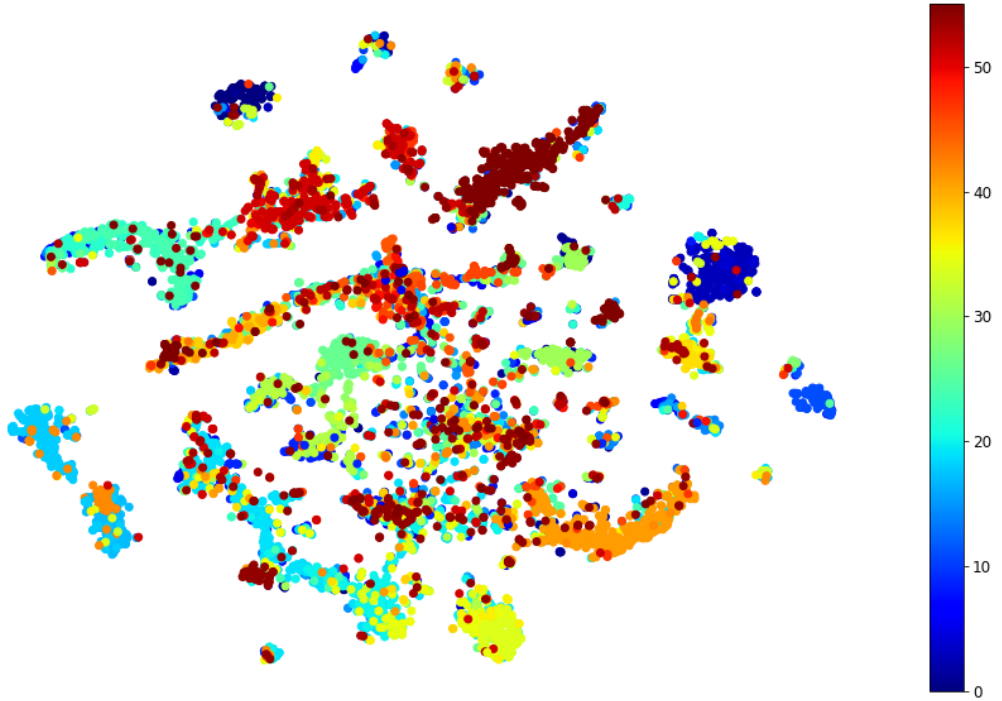
« passer de triphasé en monophasé »

« oui alors c'est tout suite on a eu un appel justement de l'E.D.F. mais c’était pas vous »

Exemple : faciliter l'exploration de nouveaux datasets grâce aux modèles spécialisés – résultats (2/2)



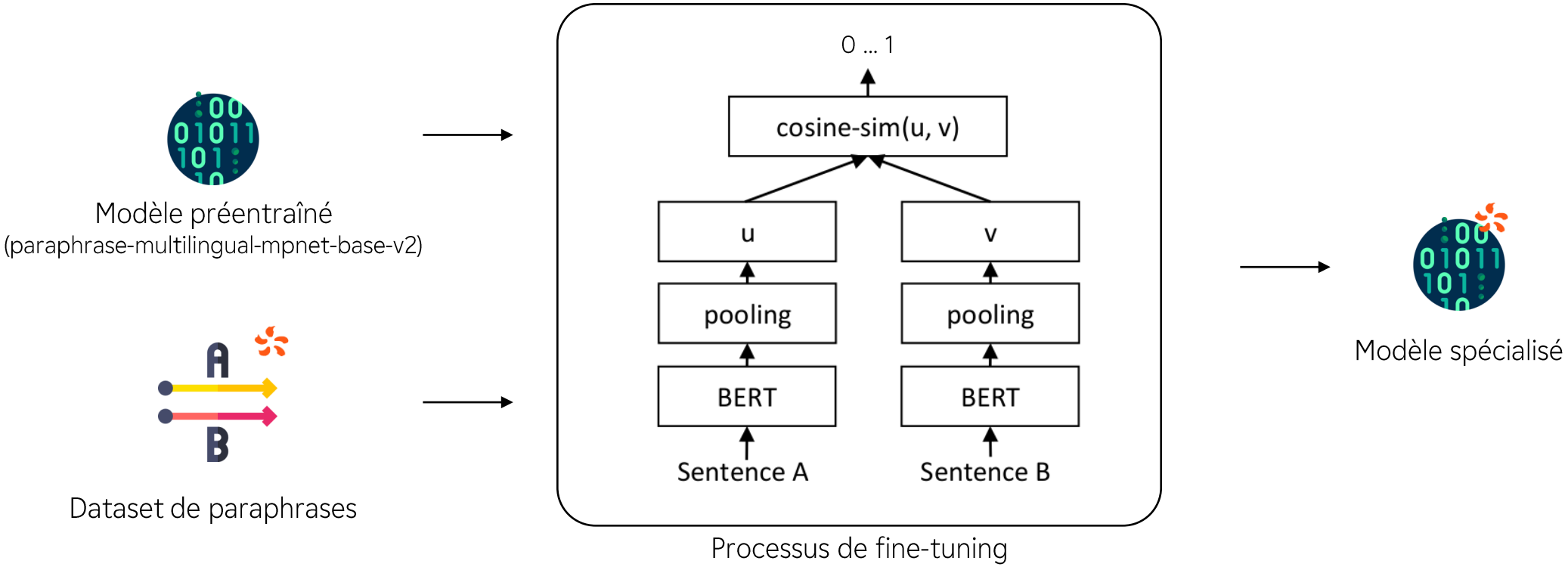
Modèle non fine-tuné
(paraphrase-multilingual-mpnet-base-v2)



Modèle fine-tuné i.e. spécialisé pour les données B2C EDF Commerce

Comparaison d'un modèle open-source et d'un modèle spécialisé pour les données EDF Commerce sur le nouveau dataset SVI en LN
(aucune donnée du SVI en LN n'a été utilisée pour la spécialisation)
(la couleur des points représente des *classes* attribuées manuellement)

Principe du fine-tuning d'un modèle de paraphrase



« je n'ai plus de courant » -> 0 <- « je ne suis pas au courant »
« obtenir une facture » -> 0.5 <- « pouvez-vous me fournir mon échéancier »
« j'ai besoin d'un justificatif de domicile » -> 1 <- « envoyez-moi une attestation de contrat »

Source : https://www.sbert.net
Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, 2019

Fine-tuning de modèles sur les données EDF Commerce

Datasets sur des tâches de classification de texte

Reconstruction d'un dataset de paraphrases spécialisé pour EDF Commerce

Fine-tuning du modèle de paraphrases open-source

Evaluation des modèles de paraphrases spécialisés

Modèles de paraphrases spécialisés prêt-à-l'emploi dans nemo

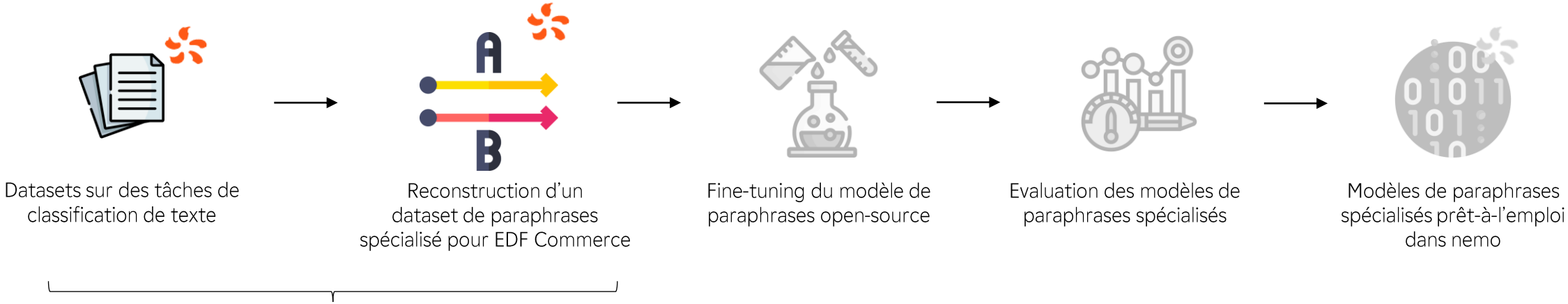
```
> embeddings = eda.get_embeddings(docs, sentence_model='/path/to/sentence_embedding_EDF_Commerce')
```

Exemple de code Python dans nemo pour charger un modèle spécialisé

Les données utilisées sont désidentifiées et décontextualisées avec notre solution de désidentification de données texte

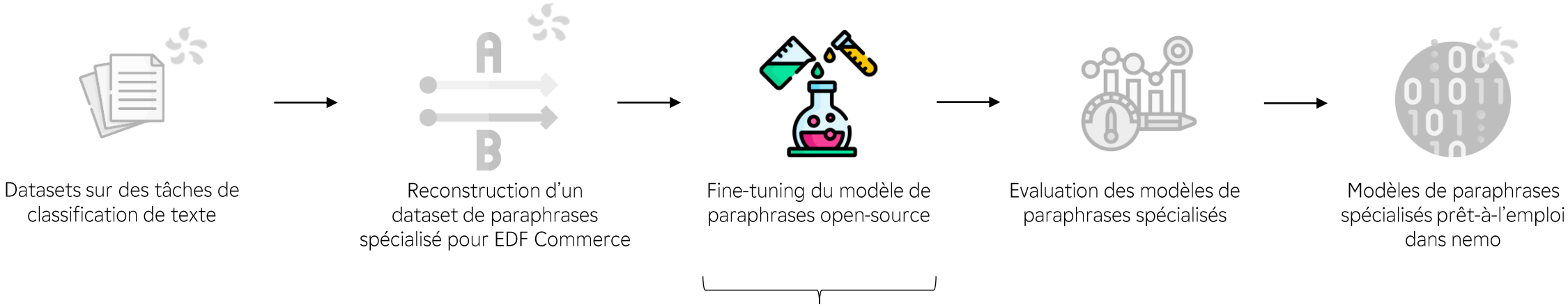
Dubuisson Duplessis, G.; Bartholme, E.; Kerroua, S.; Poulain, M.; Roulier, A.; Guénet, A.-L., **Désidentification de données texte produites dans un cadre de relation client**, 27ème Conférence sur le Traitement Automatique des Langues Naturelles, 2020

Construction d'un dataset de paraphrases EDF Commerce



- Idéalement : annotation d'un dataset de similarité sémantique spécifique à EDF Co (mais coûteux)
- Alternative à l'annotation manuelle : des classes aux paraphrases
  - Définition manuelle de règles pour faire correspondre les documents d'une classe à une autre en fonction de la similarité
  - Prise en compte de la granularité : fin (e.g., intention), moyen (e.g., contrat), gros (e.g., réclamation)
- Minage des « hard negatives »
  - Définition : mêmes mots, sens différents
  - Exemple : « je n'ai plus de courant » VS « je ne suis pas au courant »

Fine-tuning de modèles sur les données EDF Commerce



Datasets sur des tâches de classification de texte

Reconstruction d'un dataset de paraphrases spécialisé pour EDF Commerce

Fine-tuning du modèle de paraphrases open-source

Evaluation des modèles de paraphrases spécialisés

Modèles de paraphrases spécialisés prêt-à-l'emploi dans nemo

- Axes retenus
  - Plusieurs schémas de pondération en fonction de la granularité des classes/labels
  - Nombre de paires :
    - 0 à 1500 (nombre de paraphrases positives et négatives par classe/label)
    - Interprétation au niveau des modèles : pas spécialisé (0), peu spécialisé (100 à 500), très spécialisé (> 750)
  - « Hard negatives » : aléatoire ou avec ranking (BM25, top-3/20/60/100)
- Notre expérimentation de fine-tuning en chiffres
  - Nombre de modèles testés : > 100
  - Temps d'entraînement d'une gamme de modèles : 12h env. sur GPU

Fine-tuning de modèles sur les données EDF Commerce

Notre protocole d'évaluation :

- Un « triathlon » de la modélisation : clustering (silhouette), classification (F1), similarité sémantique (corrélation)
- Feedbacks utilisateur sur l'utilité (ou pas) sur des études réelles

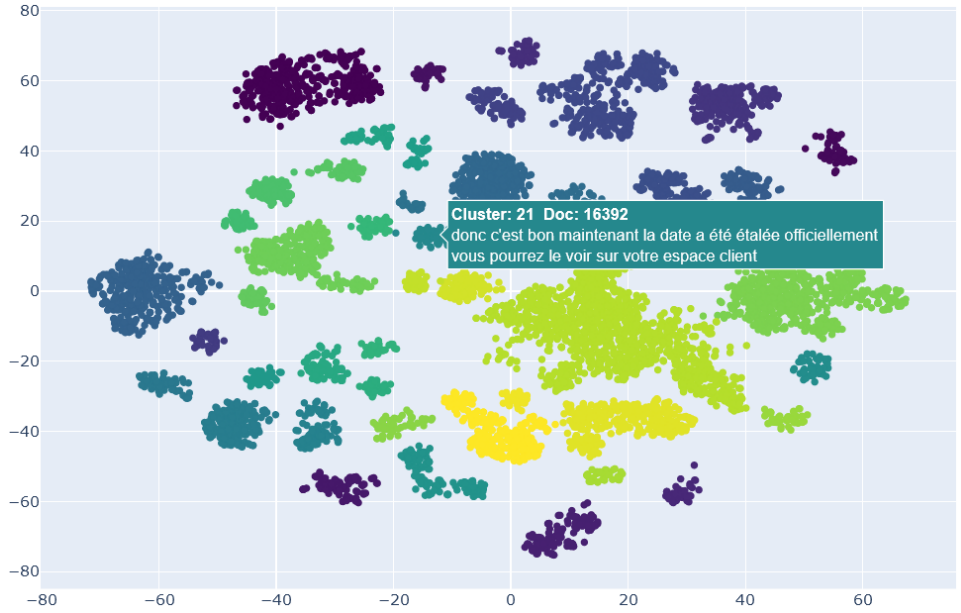
3

Conclusion

La librairie Python « nemo »

« Neural sEMantic explOration » (nemo)

- Simple d'utilisation
- Paramétrable
- Accompagnée d'une gamme de modèles prêt-à-l'emploi spécialisés et optimisés pour les données « texte » de la relation client



Objectif

Explorer mieux et plus rapidement des données texte à l'aide de techniques opérationnelles de « deep learning »

En bref

Exploration de données texte et clustering

Recherche d'information sémantique

Augmentation de données, aide à la modélisation linguistique, extraction de paraphrases

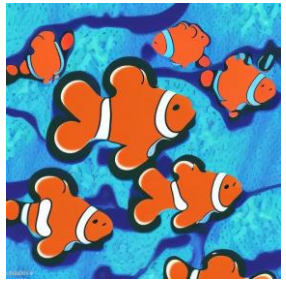
PyTorch 6 modèles de « deep learning » disponibles

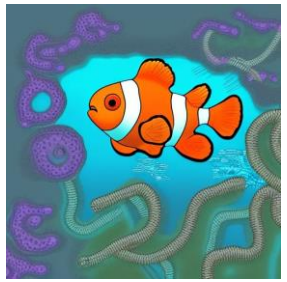
Contact

Chef de projet : Guillaume DUBUISSON DUPLESSIS

Dubuisson Duplessis, Guillaume ; Bullier, François ; Guénet, Anne-Laure ; **Démonstration: exploration sémantique de données texte de la relation client**, Conférence Nationale sur les Applications Pratiques de l'Intelligence Artificielle (APIA), 2023

Merci





Synthèse des nouveautés techniques

Revisiter l'exploration de données par le prisme des avancées du « deep learning » est prometteur pour l'opérationnel

1. Les représentations sémantiques **au niveau document** : « sentence embeddings »
2. Réduction de dimension : t-SNE et UMAP
3. Clustering : travailler par densité avec HDBSCAN et visualiser avec plotly
4. Qualification des clusters : trouver des représentants **au niveau document** avec protodash
5. Indexation et recherche sémantique : FAISS
6. Sculpture d'embeddings : « contrastive learning »

Références

De nombreux icônes sont issus du site https://www.flaticon.com/