

Exploration des résultats de l'expérience 9 - question comme document

Les graphique implémentés ne sont pas pertinent pour visualiser les données

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import json
import seaborn as sns
```

```
In [2]: with open("perts_values.json", "r") as f:
res = json.load(f)
```

```
In [3]: perts_name = [res[i]["pert"]["name"] for i in range(len(res))]
perts_name
```

```
Out[3]: ['drone -> engin autonome',
'limitation -> contrainte',
'système -> dispositif',
'technique -> spécialisé',
'drone -> vecteur',
'limitation -> bornage',
'système -> ensemble',
'technique -> difficile',
'drone->gazon',
'limitation->animation',
'système->panneau',
'technique->magique',
'drone->ver de verre',
'limitation->plastique anarchique',
'système->esperluette en vent',
'technique->feuille entubée',
'Full-replace Synonymes 1',
'Full-replace Polysémie 1',
'Full-replace Elephants 1',
'Full-replace Elephants Roses 1']
```

La perturbation de la représentation de la question

```
In [4]: baseline_score = [res[i]["results"]["baseline_perf"] for i in range(len(res))]
baseline_score
```

```
Out[4]: [[13.849666595458984],
[14.342525482177734],
[14.340707778930664],
[14.376974105834961],
[14.340707778930664],
[14.340707778930664],
[14.340707778930664],
[14.376974105834961],
[14.340707778930664],
[14.340707778930664],
[14.340707778930664],
[14.376974105834961],
[14.314400672912598],
[12.833820343017578],
[14.424174308776855],
[14.37918758392334],
[13.894460678100586],
[12.740682601928711],
[12.479087829589844],
[12.658003807067871]]
```

```
In [5]: X = [i for i in range(len(res))]

fig, axs = plt.subplots(1,2, figsize=(15,5))
```

```

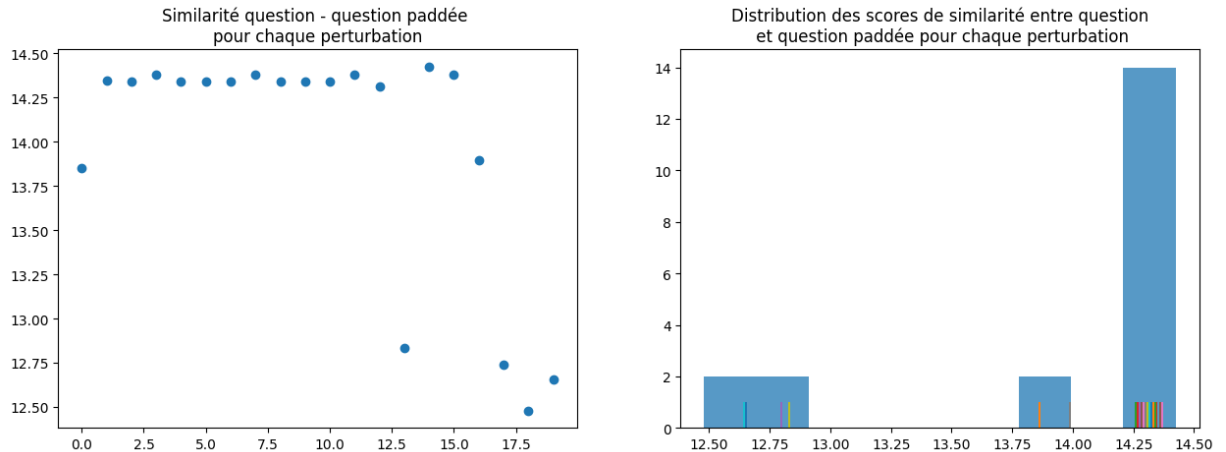
bins = np.linspace(min(baseline_score), max(baseline_score), 10).flatten()

axs[0].scatter(X, baseline_score)
axs[0].set_title("Similarité question - question paddée \npour chaque perturbation")

axs[1].hist(np.asarray(baseline_score).flatten(), bins=bins, alpha=0.75)
axs[1].hist(baseline_score)
axs[1].set_title("Distribution des scores de similarité entre question\n et question paddée pour")

plt.show()

```



On observe ici le score de similarité (produit scalaire entre les représentations) entre la question et de la question paddée.

Certaines perturbations remplacent un mot par plusieurs. Pour maintenir le même nombre de mot entre le document original et le document perturbé, le document original est rallongé avec le mot `a` (peut être modifié lors du paramettrage du dataloader). Les documents (original et perturbé) sont paddés avec le token `<pad>` pour atteindre la longueur maximale des entrées du modèle.

Dans le cas de notre expérience, la question et la question vue comme document original diffèrent. Cette différence varie aussi en fonction de la perturbation appliquée (longueur de l'expression de remplacement). C'est pour cela que le score de similarité est différent pour chaque perturbation.

On identifie bien les perturbations `full_replace` (4 derniers points du premier plot) qui necessitent plus d'ajouts de `a`

```

In [6]: perturbed_score = [res[i]["results"]["perturbed_perf"] for i in range(len(res))]
        perturbed_score

```

```

Out[6]: [[13.83007526397705],
         [14.28785514831543],
         [14.259130477905273],
         [14.309386253356934],
         [14.10728645324707],
         [14.432273864746094],
         [14.25200080871582],
         [14.263550758361816],
         [13.753355979919434],
         [14.319318771362305],
         [14.136844635009766],
         [14.160714149475098],
         [13.76951789855957],
         [14.352117538452148],
         [14.138787269592285],
         [14.246441841125488],
         [13.528707504272461],
         [13.92532730102539],
         [13.318477630615234],
         [13.583318710327148]]

```

```

In [7]: X = [i for i in range(len(res))]

fig, axs = plt.subplots(1,2, figsize=(15,5))
perturbed_min = min(perturbed_score)

```

```

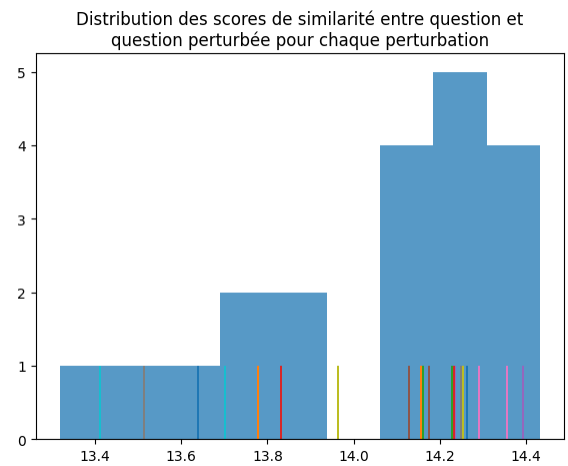
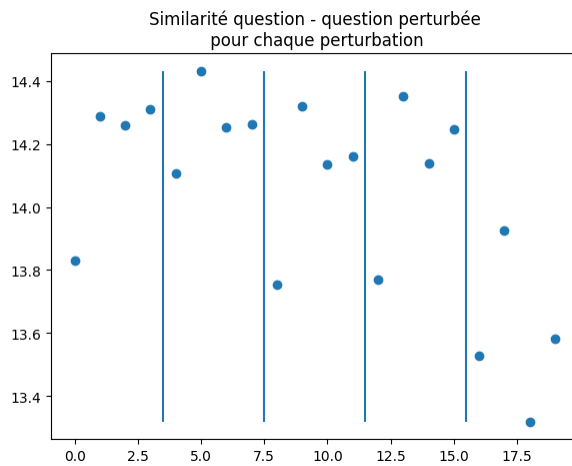
perturbed_max = max(perturbed_score)
bins = np.linspace(perturbed_min, perturbed_max, 10).flatten()

axs[0].scatter(X, perturbed_score)
axs[0].set_title("Similarité question - question perturbée\n pour chaque perturbation")
axs[0].vlines([3.5, 7.5, 11.5, 15.5], perturbed_min, perturbed_max)

axs[1].hist(np.asarray(perturbed_score).flatten(), bins=bins, alpha=0.75)
axs[1].hist(perturbed_score)
axs[1].set_title("Distribution des scores de similarité entre question et\nquestion perturbée po

plt.show()

```



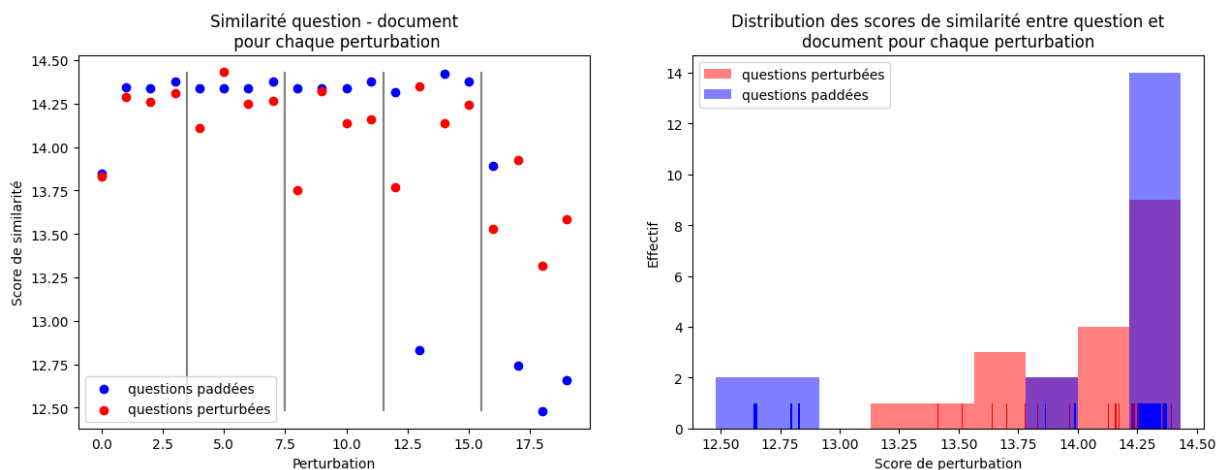
```
In [8]: X = [i for i in range(len(res))]

fig, axs = plt.subplots(1,2, figsize=(15,5))
perturbed_min = min(perturbed_score + baseline_score)
perturbed_max = max(perturbed_score + baseline_score)
bins = np.linspace(perturbed_min, perturbed_max, 10).flatten()

axs[0].scatter(X, baseline_score, label="questions paddées", color="b")
axs[0].scatter(X, perturbed_score, label="questions perturbées", color="r")
axs[0].set_title("Similarité question - document\n pour chaque perturbation")
axs[0].vlines([3.5, 7.5, 11.5, 15.5], perturbed_min, perturbed_max, color="grey")
axs[0].set_xlabel("Perturbation")
axs[0].set_ylabel("Score de similarité")
axs[0].legend()

axs[1].hist(np.asarray(perturbed_score).flatten(), bins=bins, alpha=0.5, color="r", label="questions perturbées")
axs[1].hist(perturbed_score, color=["r" for _ in range(len(perturbed_score))])
axs[1].hist(np.asarray(baseline_score).flatten(), bins=bins, alpha=0.5, color="b", label="questions paddées")
axs[1].hist(baseline_score, color=["b" for _ in range(len(baseline_score))])
axs[1].set_title("Distribution des scores de similarité entre question et\ndocument pour chaque perturbation")
axs[1].set_xlabel("Score de perturbation")
axs[1].set_ylabel("Effectif")
axs[1].legend()

plt.show()
```



Le premier graphique présente les scores de similarité (question, question paddée ou perturbée) pour chaque perturbation. Les lignes séparent les groupes de perturbations de même niveau ou type (perturbations de Niveau 1 à 4 puis perturbations de type full-replace de niveau 1 à 4). Les

On observe une plus faible similarité entre la question et la question perturbée pour les perturbations de type `replace`. Pour les perturbations `full_replace`, les scores de similarité sont plus élevés pour la question perturbée que pour la question paddée. Par ailleurs, les perturbations portant sur le mot *drone* ont un plus grand impact sur la représentation de la question que les autres. On constate aussi que le niveau des perturbation ne provoque qu'une très faible baisse de similarité.

Le second graphique montre la distribution des scores de similarité pour les question paddées (en bleu) et perturbées (en rouge) ainsi que la valeur de chaque score au sein d'une barre. On visualise bien l'étalement plus important des scores pour les questions perturbées.

Matrices de sensibilité

Visualisation d'une matrice de sensibilité

Arguments:

- `i_pert` Indice de la perturbation à visualiser dans la liste chargée dans la variable `res`
- `ax` objet Axes de Pyplot pour afficher la matrice
- `view_style = "base"` Paramètres d'affichage

```

In [9]: VIEW_STYLES = {
    "base" : {
        "cmap" : "RdBu_r",
        "vmin" : -1,
        "vmax" : 1,
    },
    "base_adapt" : {
        "cmap" : "RdBu_r",
        "vmin" : -1,
        "vmax" : 1,
    },
    "base_extr" : {
        "cmap" : "RdBu_r",
        "vmin" : -50,
        "vmax" : 50,
    },
    "base_abs" : {
        "cmap" : "Reds",
        "vmin" : 0,
        "vmax" : 1,
    },
    "base_resc" : {
        "cmap" : "RdBu_r",
        "vmin" : -1,
        "vmax" : 1,
    },
    "std" : {
        "cmap" : "magma_r",
        "vmin" : 0,
        "vmax" : 1,
    },
    "std_adapt" : {
        "cmap" : "magma_r",
        "vmin" : 0,
        "vmax" : 1,
    },
    "std_max" : {
        "cmap" : "magma_r",
        "vmin" : 0,
        "vmax" : 50,
    },
    "std_resc" : {
        "cmap" : "magma_r",
        "vmin" : 0,
        "vmax" : 1,
    },
    "norm" : {
        "cmap" : "RdBu_r",
        "vmin" : -1,
        "vmax" : 1,
    },
    "std_norm" : {
        "cmap" : "magma_r",
        "vmin" : 0,
        "vmax" : 1,
    },
    "abs" : {
        "cmap" : "magma_r",
        "vmin" : 0,
        "vmax" : 50,
    },
}

def matrice_sensibilité(i, ax, view_style="base"):
    data = np.asarray(res[i]["results"]["patching_mean"])
    data = data.astype(float)
    plt.figure(figsize=(10, 6))

    printed_data = data
    view_params = VIEW_STYLES[view_style].copy()
    if "norm" in view_style:

```

```

# printed_data = printed_data / np.linalg.norm(data)
printed_data = (printed_data - printed_data.mean()) / printed_data.std()
if "abs" in view_style:
    printed_data = printed_data.absolute()
if "extr" in view_style:
    view_params["vmax"] = (abs(printed_data.max()) + abs(printed_data.min())) / 2
    view_params["vmin"] = - view_params["vmax"]
if "max" in view_style:
    view_params["vmax"] = (abs(printed_data.max()) + abs(printed_data.min())) / 2
    view_params["vmin"] = 0
if "adapt" in view_style:
    if (printed_data > 20).any() :
        view_params["vmax"] = 50
    else:
        view_params["vmax"] = 1
        view_params["vmin"] = (VIEW_STYLES[view_style]["vmin"] * view_params["vmax"]) # si vmin
if "resc" in view_style:
    printed_data = (np.sign(printed_data) / np.absolute(printed_data).max()) * np.absolute(p

ax = sns.heatmap(
    printed_data,
    cmap=view_params["cmap"],
    vmin=view_params["vmin"],
    vmax=view_params["vmax"],
    xticklabels=True,
    yticklabels=True,
    fmt=".2f",
    ax = ax,
    annot=True,
    annot_kws={"size": 8},
)

ax.set_title(f"Matrice de sensibilité pour {perts_name[i]}")
ax.set_xlabel("Head")
ax.set_ylabel("Layer")

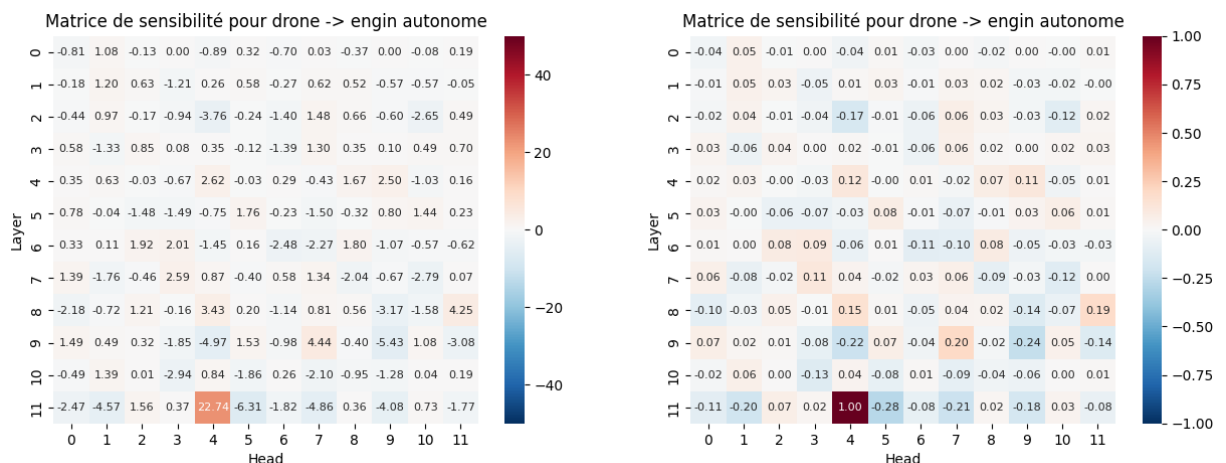
return ax

```

```

In [10]: fig, axs = plt.subplots(1,2, figsize=(15,5))
i = 0
axs[0] = matrice_sensibilité(i,axs[0], view_style="base_adapt")
axs[1] = matrice_sensibilité(i,axs[1], view_style="base_resc")
plt.show()

```



<Figure size 1000x600 with 0 Axes>
 <Figure size 1000x600 with 0 Axes>

Visualisation des noeuds sensibles

Visualisation du nombre de noeuds sensibles pour un sous ensemble défini par les critères de niveau, de type de perturbation et de mot remplacé Arguments:

- res description des résultats (otenus avec la commande CLI GenericAutomatedPerturbation.py values)
- niv Niveau des perturbations à conserver

- type type de perturbation (replace ou full_replace)
- mot Mot à remplacer dans les perturbations ("" ou None pour ne pas prendre en compte ce critère)

```
In [11]: def print_noeuds_sensibles(indices_sensibles):
    for (i_pert, noeuds) in indices_sensibles.items():
        print(perts_name[i_pert])
        mean = res[i_pert]["results"]["patching_mean"]
        resc = mean / (np.max(np.abs(mean)))
        for n in noeuds:
            print(f"\t{n[0]}, {n[1]} : {mean[n[0]][n[1]]:.2f} | {resc[n[0]][n[1]]:.2f}")

def plot_noeuds_sensibles(nd_sen, titre="Noeuds sensibles", ax = None, vmax=None ):
    mem_ax = ax
    if ax is None:
        fig, ax = plt.subplots()

    ax = sns.heatmap(
        nd_sen,
        cmap="Reds",
        vmin=0,
        vmax=nd_sen.max() if vmax is None else vmax,
        xticklabels=True,
        yticklabels=True,
        annot=True,
        annot_kws={"size": 8},
        ax=ax
    )

    ax.set_title(titre)
    ax.set_xlabel("Head")
    ax.set_ylabel("Layer")

    if mem_ax is not None:
        return ax

def noeuds_sensibles(
    data : dict,
    niveau, # 1-4 ou None
    type, # "full_replace" ou "replace"
    mot = None, # "" ou mot à remplacer
    ax = None,
    print_nodes = True
):
    perts_to_check = [i for i in range(len(data))]
    if type == "replace" and (mot is not None and mot != ""):
        perts_to_check = [i for i in perts_to_check if (data[i]["pert"]["type"] == "replace" and
match (type, niveau) :
    case "replace", 1 :
        perts_to_check = [i for i in perts_to_check if i < 4]
    case "replace", 2 :
        perts_to_check = [i for i in perts_to_check if 4 <= i < 8]
    case "replace", 3 :
        perts_to_check = [i for i in perts_to_check if 8 <= i < 12]
    case "replace", 4 :
        perts_to_check = [i for i in perts_to_check if 12 <= i < 16]
    case "replace", _:
        perts_to_check = [i for i in perts_to_check if i < 16]
    case "full_replace", 1:
        perts_to_check = [i for i in perts_to_check if i == 16]
    case "full_replace", 2:
        perts_to_check = [i for i in perts_to_check if i == 17]
    case "full_replace", 3:
        perts_to_check = [i for i in perts_to_check if i == 18]
    case "full_replace", 4:
        perts_to_check = [i for i in perts_to_check if i == 19]
    case "full_replace", _:
        perts_to_check = [i for i in perts_to_check if i >= 16]

    cumul_sensible = np.zeros(np.asarray(data[0]["results"]["patching_mean"]).shape)
    indices_sensibles = {}

    for i_pert in perts_to_check:
        mean = np.asarray(data[i_pert]["results"]["patching_mean"])
```

```

resc_mean = mean / np.max(np.abs(mean))
carte_sensible = ((np.abs(mean) > 0.1) & (np.abs(resc_mean) > 0.5))
cumul_sensible += carte_sensible
indices_sensibles[i_pert] = np.argwhere(carte_sensible)

if print_nodes:
    print_noeuds_sensibles(indices_sensibles)
plot_noeuds_sensibles(cumul_sensible, titre = f"Noeuds sensibles {type} N{niveau} {mot}", ax=

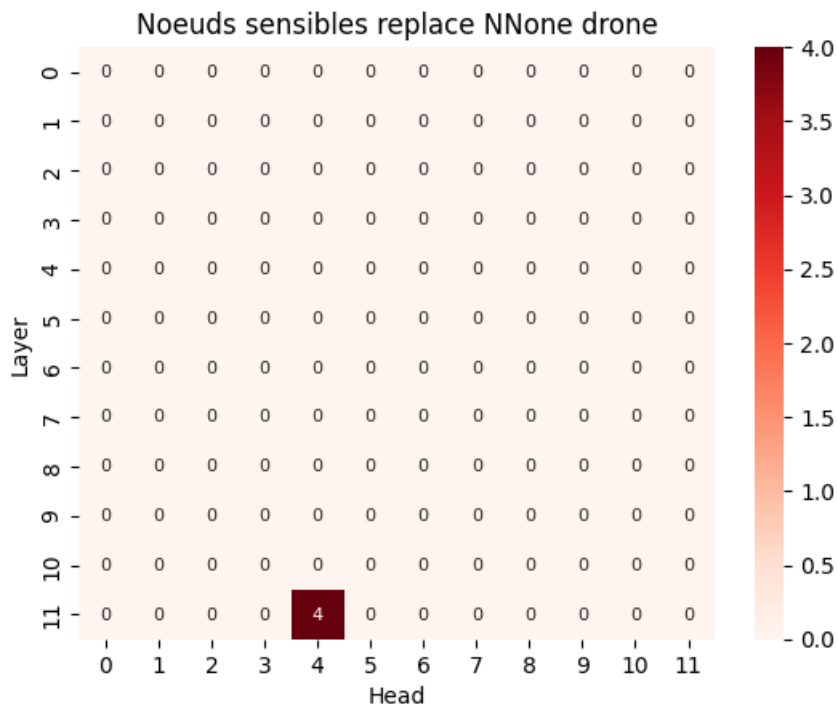
```

In [12]: noeuds_sensibles(res, None, "replace", "drone")

```

drone -> engin autonome
      11, 4 : 22.74 | 1.00
drone -> vecteur
      11, 4 : 1.63 | 1.00
drone->gazon
      11, 4 : 0.68 | 1.00
drone->ver de verre
      11, 4 : 0.80 | 1.00

```



In [13]: _, axs = plt.subplots(4,4, figsize=(25,25))

```

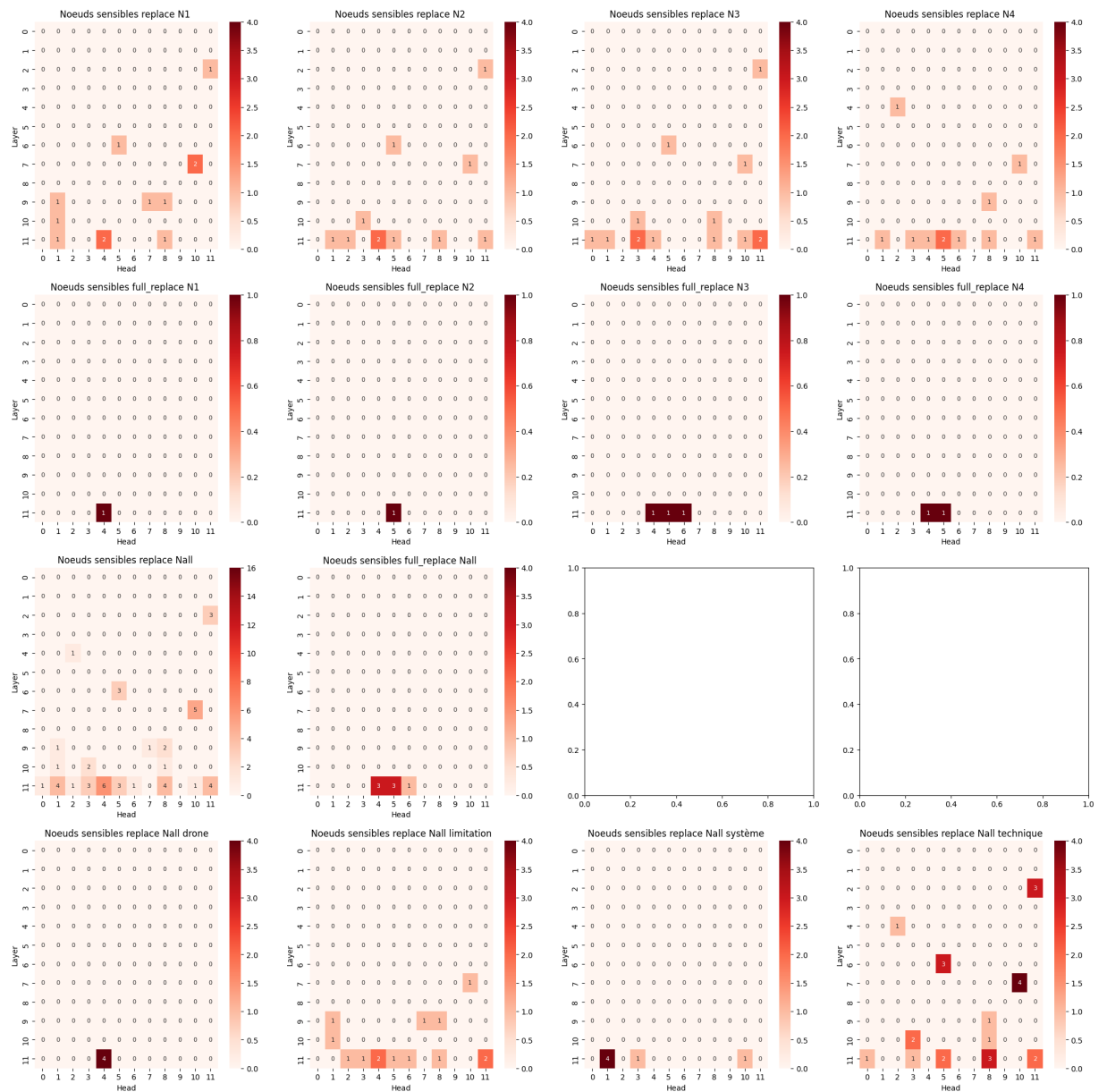
for i_t, type in enumerate(["replace", "full_replace"]):
    for i_n, niv in enumerate(range(1,5)):
        noeuds_sensibles(res, niv, type, "", ax=axs[i_t][i_n], print_nodes=False)

noeuds_sensibles(res, "all", "replace", "", ax=axs[i_t+1][0], print_nodes=False)
noeuds_sensibles(res, "all", "full_replace", "", ax=axs[i_t+1][1], print_nodes=False)

for i_m, mot in enumerate(["drone", "limitation", "système", "technique"]):
    noeuds_sensibles(res, "all", "replace", mot, ax=axs[i_t+2][i_m], print_nodes=False)

plt.show()

```



Matrices présentées:

- Ligne 1 : Remplacements simples par niveau (4 perturbation par matrices)
- Ligne 2 : Remplacements multiples par niveau (1 perturbation full_replace par graphique)
- Ligne 3 :
 - Cumul de toutes les perturbations de type `replace`
 - Cumul de toutes les perturbations de type `full_replace`
- Ligne 4 : Remplacements simples regroupés par mot remplacé dans la perturbation

Critère de sensibilité : $\text{mean} > 0.1$ et $\text{resc_mean} > 0.5$

On constate que les perturbations multiples ne rendent pas sensibles les mêmes noeuds. Il y a donc un effet d'effacement dû probablement au nombre.

Les noeuds sensibles aux perturbations `full_replace` sont tous dans la dernière couche. Et très peu de noeuds sont sensible à de multiples reprises au sein des perturbations `replace`.

Les noeuds sensibles des couches centrales semblent liés au mot remplacé plus qu'à un niveau de perturbation. On observe cela particulièrement pour le cas des perturbations qui remplacent `technique`.

Bilan : On retrouve les mêmes tendances pour les paires (question, question) que pour le reste du dataset

Mise en perspective avec les exps précédentes

```
In [14]: data = []
for filepath in ["../exp5/synonymes/perts_valuesV2.json", "../exp5/polysemie/perts_valuesV2.json"]:
    with open(filepath, 'r') as f:
        data += json.load(f)

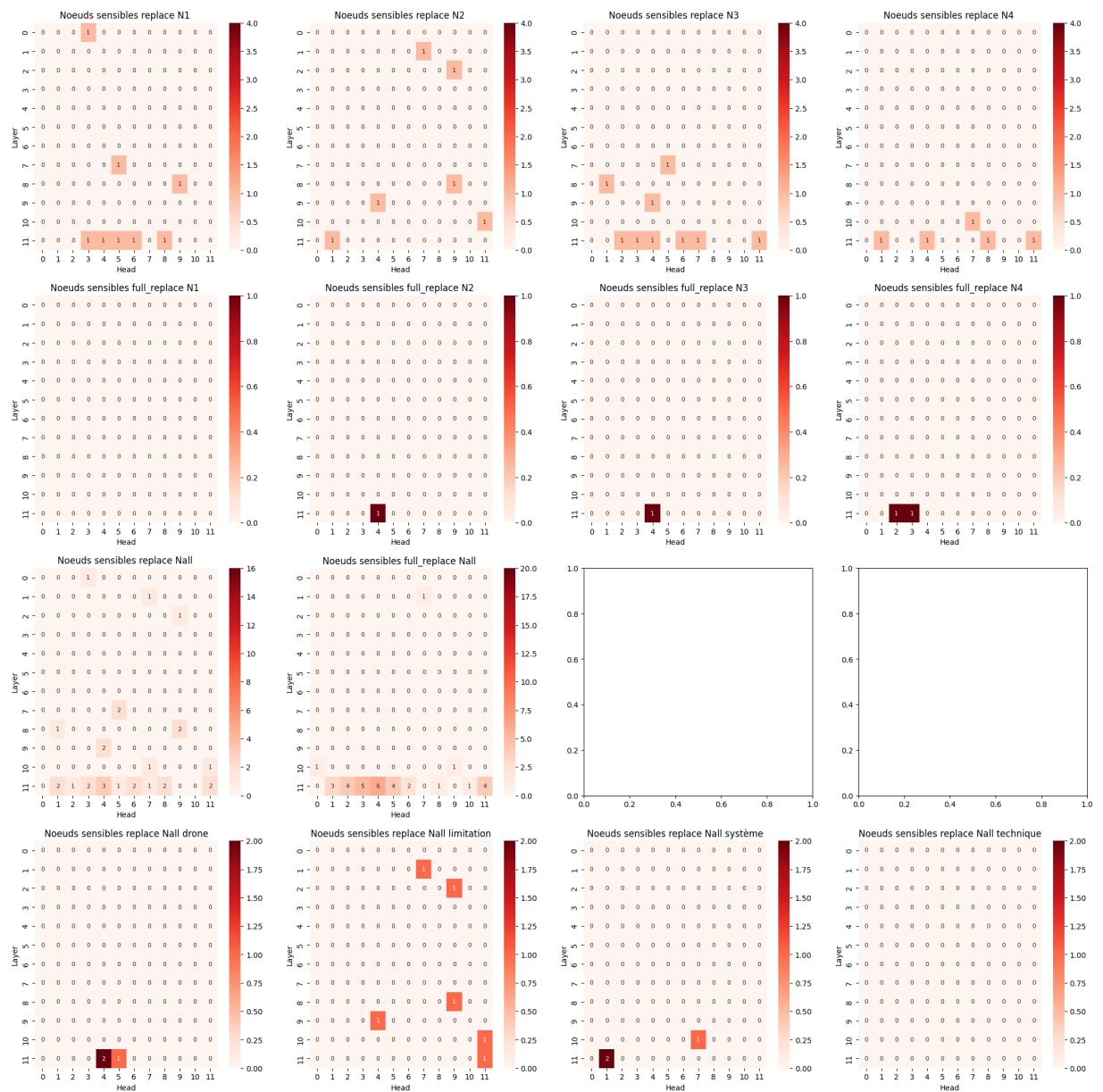
_, axs = plt.subplots(4,4, figsize=(25,25))

for i_t, type in enumerate(["replace", "full_replace"]):
    for i_n, niv in enumerate(range(1,5)):
        noeuds_sensibles(data, niv, type, "", ax=axs[i_t][i_n], print_nodes=False)

noeuds_sensibles(data, "all", "replace", "", ax=axs[i_t+1][0], print_nodes=False)
noeuds_sensibles(data, "all", "full_replace", "", ax=axs[i_t+1][1], print_nodes=False)

for i_m, mot in enumerate(["drone", "limitation", "système", "technique"]):
    noeuds_sensibles(data, "all", "replace", mot, ax=axs[i_t+2][i_m], print_nodes=False)

plt.show()
```



On observe ici les mêmes matrices que précédemment mais pour le dataset complet de la même question.

On ne constate pas ou peu de noeuds des couches centrales communs au deux lots de matrices. Seules certaines tendances persistent (noeuds concentrés dans les couches du bas, pas de noeuds central qui se dégage, des noeuds commun pour un même mot remplacé).