Deuxième partie

Partie Informatique

Chapitre 6

Robustesse des LLMs dans le roleplay

Lors de ce stage j'ai changé à plusieurs reprise de sujet de recherche. Malheuresement cela n'a pas aboutit à de résultat publiables. Ce qui suit retrace donc mes recherches de cet été dans leur ordre chronologique.

Le sujet originel de mon stage porte sur la robustesse des LLM en situation de roleplay i.e. améliorer la capacité des LLM à jouer un rôle et à ne pas en diverger. Par exemple si l'on demande à un LLM de jouer le rôle d'un marchand dans une ville médiévale ce dernier ne devrait pas pouvoir répondre à des questions portant sur l'algorithme Quicksort en python.

Afin de résoudre ce problème j'ai d'abord passé la majeure partie de mon temps à lire la littérature existante autour de ce sujet. Le nombre modéré d'approches qui existent actuellement peuvent être divisées en deux catégories : celles qui modifient les poids du modèle et celles qui vont créer une "infrastructure" autour des LLMs pour améliorer leur robustesse. Ces "infrastructures" peuvent être des systèmes de mémoire externes, un autre LLM rajoutant du contexte ou simplement demander au LLM de décrire le ton de la voix ou les expressions faciales de son personage avant de répondre.

Après m'être immergé dans la recherche existante j'ai aussi commencé à expérimenter avec des IA génératives disponible en ligne afin de gagner un sens pratique du problème. De ces expériences j'ai pu tout d'abord confirmer que les capacités de roleplay des LLMs et leur robustesse sont corrélés positivement avec le nombre de paramètres. De plus sans modification spéciale de la

prompt ou fine tuning la robustesse des LLMs de taille moyenne (8 à 12B) est très limitée. Il est par exemple assez simple de faire sortir de leur rôle les IA conversationelle gratuite de CharacterAI qui est le leader du marché pour le roleplay avec des IA génératives. De même des modèles comme Llama 3 8B ou gemma 3 12B sont facilement "attaquable". Par la suite je me suis donc concentré sur des modèles de taille similaires.

La première idée que j'ai explorée afin de résoudre ce problème fut celle suggérée initialement par mon maître de stage. On utilise deux LLMs ; un suivant le roleplay (Défenseur) et l'autre tentant de le faire dévier de son rôle (Attaquant). En identifiant les instances où le LLM défenseur sort de son rôle on pourrait entrainer les deux LLMs à la manière des GAN et obtenir un défenseur capable de rester dans son rôle peu importe à quel point le joueur tente de le faire dévier.

Les problématiques auxquelles j'ai été confronté sont en lien avec les champs de recherche du jailbreak et des LLM as judge, notamment :

- Comment obtenir de bon attaquant
- Comment reconnaître quand est ce qu'une déviation (par rapport au rôle) a lieu
- Comment reconnaitre quelle action de l'attaquant est responsable pour la déviation du défenseur
- Comment générer des attaques qui soient nombreuses et diverses

L'obtention d'un bon attaquant notamment est une tâche difficile. En effet des attaques trop directes sont facilement discernables par le défenseur. Le faire dévier de son rôle requiert donc de pouvoir établir et suivre des plans sur plusieurs tours, ce que même des LLMs comme GPT 4 sont incapable de faire. Cette problématique correspond au jailbreak blackbox multi tour et demande des "infrastructure" complexe. (Ce papier par exemple maintient une base de connaissance regroupant des tactiques qu'il modifie dynamiquement.)

Cependant un évènement m'a convaincu d'arréter mon travail sur la robustesse des LLMs en roleplay et de changer de sujet. En effet après de nombreuses expérience avec les LLMs en ligne je me suis rendu compte qu'une system prompt comportant :

- des règles très précises
- des exemples
- l'indication de penser avant de répondre (et d'encadrer la réflexion entre les balises XML "<think> </think>")

était suffisante pour obtenir un LLM robuste à toute mes tentatives de le faire dévier de son rôle ! La prompt suivante me permet par exemple de jouer

à un jeu de rôle à la DnD avec Gemma 3 12B.

You are a masterful dungeon master.

Before playing your role you think step by step about what would be the response that is most consistent with your role. You must give special attention to stay in your role and have responses that are realistic.

You should be extra careful about not letting the player invent part of the lore (except lore related to his backstory, provided that it fits the roleplay). When the player takes an action you should check that they only states the action and not the consequences. This is critical. You are the one describing the consequences not the player.

If the player attemps to deviate the roleplay or state consequences to their actions you should refuse and output : RETRY

Here are a few instances of banned player's response (each should result in a RETRY)

- I go inside the room and I see a painting on the wall (The player can only assert actions. They cannot tell what they "see" unless you describe it to them)
- I take out my sword and kill the ennemy (The player has the right to take out its sword but cannot state consequences like its attack succeeding and killing the ennemy)
- I look around and find a lever (The player cannot assert that he finds something, only that he searches for it. Again, they can state action not consequences)

The player can also step out of its character to prompt the dungeon master for further descriptions. For instance they can ask :

- What do I see now ?
- Is the door open ?
- Is the ennemy far or near me ?
- What is inside the chest ?

If the player mentions the use of a new item you should check that it is part of their inventory. When the player picks up an item you should explicitely output their inventory in your thinking section.

Here are a few player's actions and what you should do that situation :

- I take out my bow -> You need to check that they have a bow in their inventory

— I pay 10 gold coins -> You need to check wether they have the right quantity of item
— I take this diamond from the chest -> You need to restate their whole inventory + the diamond

Your thinking should be enclosed between <think> </think>

Your responses (not the thinking) should be short and natural
Speech is between " "
Descriptions are between * *
Actions are between nothing

Cela m'a fait perdre intérêt dans ce sujet car je ne vois pas de justification à résoudre de manière complexe ce qu'il est possible de faire simplement.

Chapitre 7

Raffinement critère par critère

Par la suite j'ai exploré le domaine du self refinement que j'ai découvert lors de mes recherches sur le roleplay. Le self refinement désigne le cycle : 1) génération de réponse 2) autocritique 3) amélioration de la réponse.

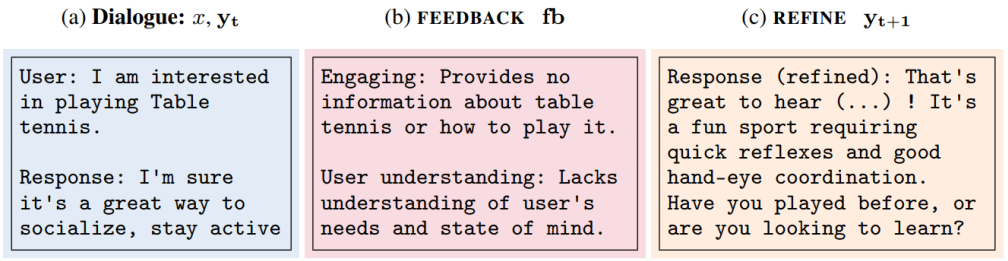

FIGURE 7.1 – Exemple de raffinement extrait du papier SELF-REFINE

La plupart des papiers que j'ai lus portant sur le self refinement s'attaque à la problématique de suivi d'instruction complexes. Ils évaluaient donc explicitement ou implicitement les réponses sur plusieurs critères (la réponse satisfait elle la contrainte 1, 2, 3, ... ? est elle dans le bon format ? est elle claire ?...) Ma première idée fut donc de décomposer l'étapes de raffinement selon les différents critères. C'est à dire que si l'on a K critères utilisés pour juger la qualité d'une réponse on va générer K réponses cherchant chacune à améliorer la qualité de la réponse originelle sur la dimension d'un des critère. On va ensuite fine tuner le modèle en utilisant la méthode DPO (Direct Preference Optimization) (papier originel)

Pour rappel la formule d'update du gradient avec DPO pour la paire de préférence (x, y_w, y_l) où x est la prompt et y_w la réponse préférées est :

$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = \\ -\beta \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\sigma(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w)) \left[\nabla_\theta \log \pi(y_w \mid x) - \nabla_\theta \log \pi(y_l \mid x)\right]\right],$$

où $\hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ est la récompense implicite

Sous des hypothèses modérées j'ai montré que cette approche permettaient de mieux ajuster le poids du gradient dans l'update de DPO. Içi $\nabla_\theta \mathcal{L}_{\text{DPOfullfeedback}}$ dénote l'update par DPO classique que l'on aurait si on raffine une réponse sur tous les critères à la fois et qu'on utilise la paire (x, y^*, y) où y est la réponse originelle et y^* la réponse raffinée qui est préférée. $\nabla_\theta \mathcal{L}_{\text{DPOmultifeedback}}$ dénote la somme des K updates sur chacune des réponses $\{y_i\}_{1\leq i\leq K}$ où la réponse y_i est raffinée uniquement par rapport au critère i.

$$\nabla_\theta \mathcal{L}_{\text{DPOfullfeedback}}(\pi_\theta; \pi_{\text{ref}}) \\ = -\beta \mathbb{E}_{(x,y_f,y)\sim\mathcal{D}} \left[\sigma\left(\sum_{i=1}^{K} \hat{r}_\theta(x, y) - \hat{r}_\theta(x, y_i)\right) \left[\sum_{i=1}^{K} (\nabla_\theta \log \pi_\theta(y_i \mid x)) - \nabla_\theta \log\ \pi_\theta(y \mid x))\right]\right]$$

$$\nabla_\theta \mathcal{L}_{\text{DPOmultifeedback}}(\pi_\theta; \pi_{\text{ref}}) \\ = \sum_{i=1}^{K} -\beta \mathbb{E}_{(x,y_f,y)\sim\mathcal{D}} \left[\sigma\left(\hat{r}_\theta(x, y) - \hat{r}_\theta(x, y_i)\right) \left[\nabla_\theta \log \pi_\theta(y_i \mid x)) - \nabla_\theta \log\ \pi_\theta(y \mid x)\right]\right]$$

La différence entre les deux formules est donc au niveau du poids dans la sigmoide. La version multifeedback permet d'avoir des poids séparés. Cela peut être avantageux. En effet imaginons que π_θ assigne une mauvaise reward implicite sur le critère i i.e. $\hat{r}_\theta(x, y) - \hat{r}_\theta(x, y_i) > 0$ mais que $\hat{r}_\theta(x, y) - \hat{r}_\theta(x, y_j) < 0$ alors la somme des deux va à la fois diminuer la magnitude du gradient dans la direction qui améliore le critère i et l'augmenter dans la direction j.

Cependant ces résultats ne semblaient pas assez fort pour justifier de multiplier par K les coûts d'entrainement des modèles. J'ai donc poursuivi sur un autre sujet lié au self refinement.

Chapitre 8

Etude du papier SPaR

Cette fois çi j'ai décidé de me concentrer sur le papier suivant SPaR: Self-Play with Tree-Search Refinement to Improve Instruction-Following in Large Language Models

L'idée générale de SPaR est de générer plusieurs raffinement possible d'une mauvaise réponse et de traiter le problème comme une recherche dans un arbre. Le score de chaque réponse est donnée par un ensemble de juge et l'on continue d'explorer l'arbre tant que le score n'atteint pas un certain seuil.

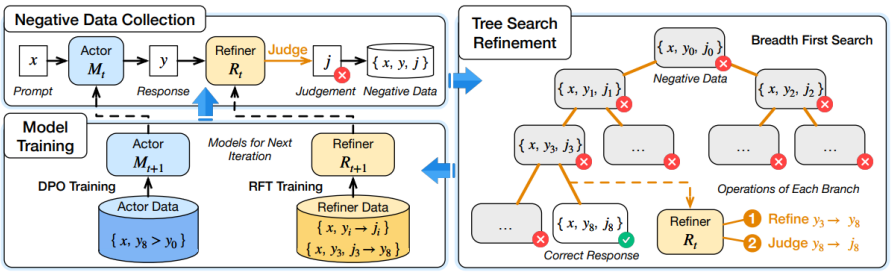

Figure 2: SPAR iterative training framework. At iteration t, the refiner R_t first judges the generated responses from the actor M_t to collect negative data. Next, a tree-search algorithm is employed to refine these imperfect responses. Finally, using the data from the above steps, we can optimize the actor and refiner for the next iteration, aiming for continuous self-improvement.

J'ai d'abord dû prendre le code en main afin de pouvoir lancer des expériences. Pour ce faire j'ai utilisé le supercalculateur Jean Zay et au bout d'une semaine de résolution de bugs j'ai pu exécuter le code du papier.

J'ai cependant remarqué plusieurs choses étranges à propos de l'article. Leur dataset notamment comportait un nombre important de questions sans queue ni tête. J'ai donc nettoyé le dataset en demandant à GPT 4.1 de juger si les questions étaient pertinentes / répondables.

J'ai ensuite essayé de renforcer la condition de sortie. Originellement le code de SPaR utilise 5 juges et considère une réponse comme satisfaisante dès lors que 3 juges la considère comme tel. J'ai élevé ce seuil à 5 juges, tous les juges doivent donc trouver la réponse satisfaisante. Pour détecter si les réponses générées avec cette plus stricte condition étaient de meilleure qualité j'ai calculé le taux de préférence entre réponse originale et réponse finale en utilisant DeepSeek. Surprenament renforcer la condition de sortie dégrade les performances. J'ai donc par la suite poursuivi mes recherches sur le thème du reward hacking avec l'espoir de pouvoir en appliquer les résultats à SPaR.

Chapitre 9

Analyse du reward hacking

Tout d'abord j'ai cherché à étudier la simulation suivante censé approximer la situation de SPaR. On a :

- Une variable aléatoire Q dont les réalisations représentent des questions
- Un modèle générateur de réponse π où $\pi(q)$ est une distribution sur les réponses à la question q
- Une fonction $X : q, r \to \{0, 1\}$ qui vaut 1 si la réponse r satisfait la question q et 0 sinon
- Une fonction $Y : q, r \to \{0, 1\}$ qui approxime la fonction X (le juge proxy)

En pratique Q représente un tirage aléatoire de question à partir d'un dataset. π et Y représentent un LLM de taille moyenne (j'ai utilisé Llama 3.1 8B et Llama 3 70B pour mes expériences) tandis que X représente un large LLM (J'ai utilisé DeepSeek v3) censé représenter une vérité objective.

Pour une question $q \sim Q$ donnée on va tirer des réponses r jusqu'à ce que $Y(q, r) = 1$ (on s'arrête dès que le juge proxy déclare que l'on a une bonne réponse).

Pour chaque question on peut modéliser X et Y par :

- p le paramètre de la loi de Bernoulli suivi par X
- $p_0 = \mathbb{P}(Y = 1 \mid X = 0)$ et $p_1 = \mathbb{P}(Y = 1 \mid X = 1)$ les deux termes caractérisant la matrice de confusion du juge proxy Y

Le but est alors de trouver une relation entre le nombre de tirage N effectué avant d'avoir trouvé $Y = 1$ et le taux de vrai positif $P(X = 1 \mid Y = 1)$

en utilisant un minimum de query à X. Dit autrement, comment utiliser optimalement un petit LLM en se basant sur des informations obtenues avec un nombre d'appel minimal à un large LLM plus couteux en temps/argent. Mon objectif alors était de trouver le nombre de tirage limite à ne pas dépasser afin que le taux de vrai positif reste au dessus d'un certain seuil.

Ma première approche fut de supposer que l'on connaissait les quantités p_0, p_1 . Au N-ème tirage si l'on n'a pas déjà obtenue $Y = 1$ on va estimer la probabilité de vrai positif à N+1 si l'on a $Y_{N+1} = 1$. Pour cela on va estimer le paramètre p en se basant sur les observation $Y_1 = Y_2 = ... = Y_N = 0$ et l'hypothèse $Y_{N+1} = 1$ (par souci de clarté on notera $O = (Y_1, Y_2, ..., Y_N, Y_{N+1}) = (0, 0, ..., 0, 1), p_0, p_1$ les observations). On a donc :

$$\mathbb{P}(X_{N+1} = 1 \mid O) = \int \mathbb{P}(X_{N+1} = 1 \mid O, p) f(p \mid O) dp$$

Où $f(p)$ est le prior sur p. En utilisant un beta prior ou un prior uniforme j'ai réussi à obtenir une expression fermée. Cependant les calculs sont assez long et pas particulièrement intéressants donc je me permets de ne pas les inclure içi.

J'ai ensuite essayé de prévoir les quantités p_0, p_1 à partir de la dernière couche d'activation du LLM π sur le dernier token de la question q . J'ai testé cette approche sur le modèle Llama 3 8B et n'ai pas obtenu de résultat concluant car la dimension de sa dernière couche étant grande (de dimension 4096) il eu fallu un nombre élevé d'appels à X pour contrer l'overfit.

Par la suite j'ai tenté de déterminer empiriquement le taux de vrai positif selon N le nombre de tirage qu'il a fallu effectuer. Pour cela on va procéder à un large nombre de tirages tels que décrit plus haut. On va stocker les réponses selon le nombre de tirage qu'il a fallu effectuer avant de les obtenir. On va donc pouvoir choisir quelles réponses et questions nous allons "soumettre" à X afin d'obtenir le taux de vrai positif selon N. Cette estimation étant empirique il nous faut calculer des intervals de confiance. J'ai choisi d'utiliser l'inégalité d'Hoeffding car simple et non asymptotique. L'objectif se transforme alors en : "Quel est le N^* limite tel qu'à $p\%$ de chance le taux de vrai positif se situe au dessus du seuil T pour $N < N^*$"

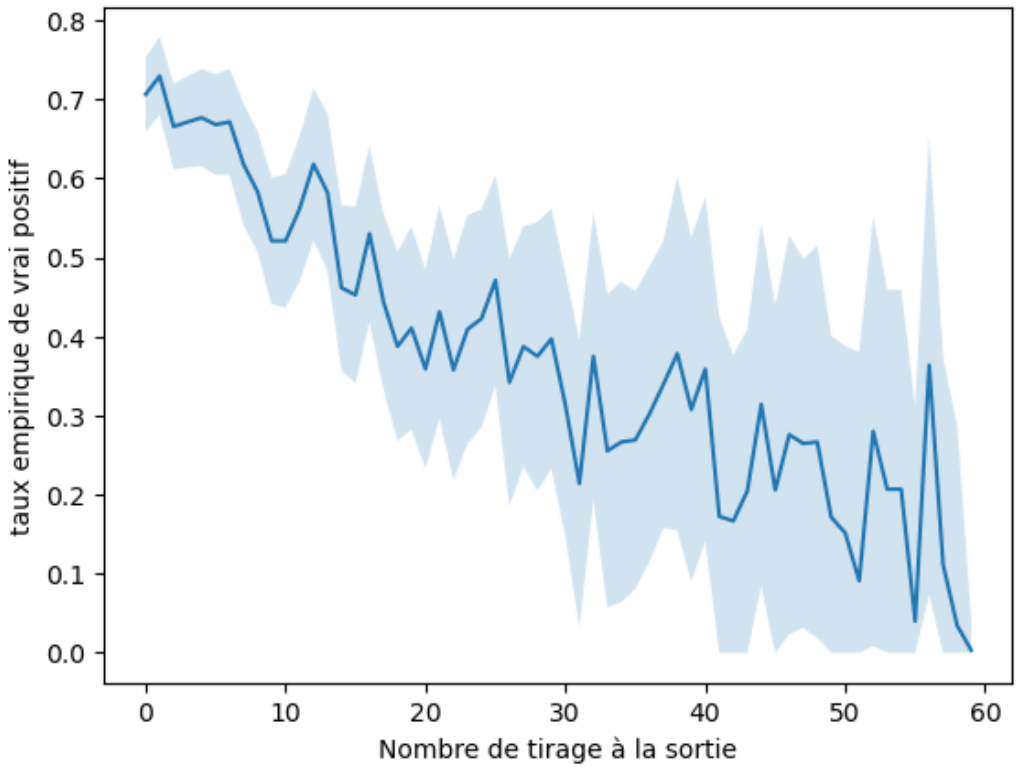

FIGURE 9.1 – Taux empirique de vrai positif selon le nombre de tirage effectué, la zone bleu clair correspond aux intervals de confiance à 95%. On observe une diminution du taux empirique ce qui est consistent avec nos observations sur SPaR, si l'on cherche plus on se soumet à davantage de risque de faux positifs.

Si l'on suppose que la véritable courbe de taux de vrai positif est décroissante selon N alors on peut adapter notre algorithme pour diminuer le nombre d'appels à X. En effet si par exemple nous cherchons le N^* limite pour un seuil à 0.8 et que pour $N = 5$ nous obtenons un interval de confiance avec un minimum au dessus de 0.8 alors nous pouvons considérer que $N^* > 5$ et restreindre nos recherches. On peut bien sûr appliquer la même idée pour les intervals de confiance dont le max est en dessous du seuil.

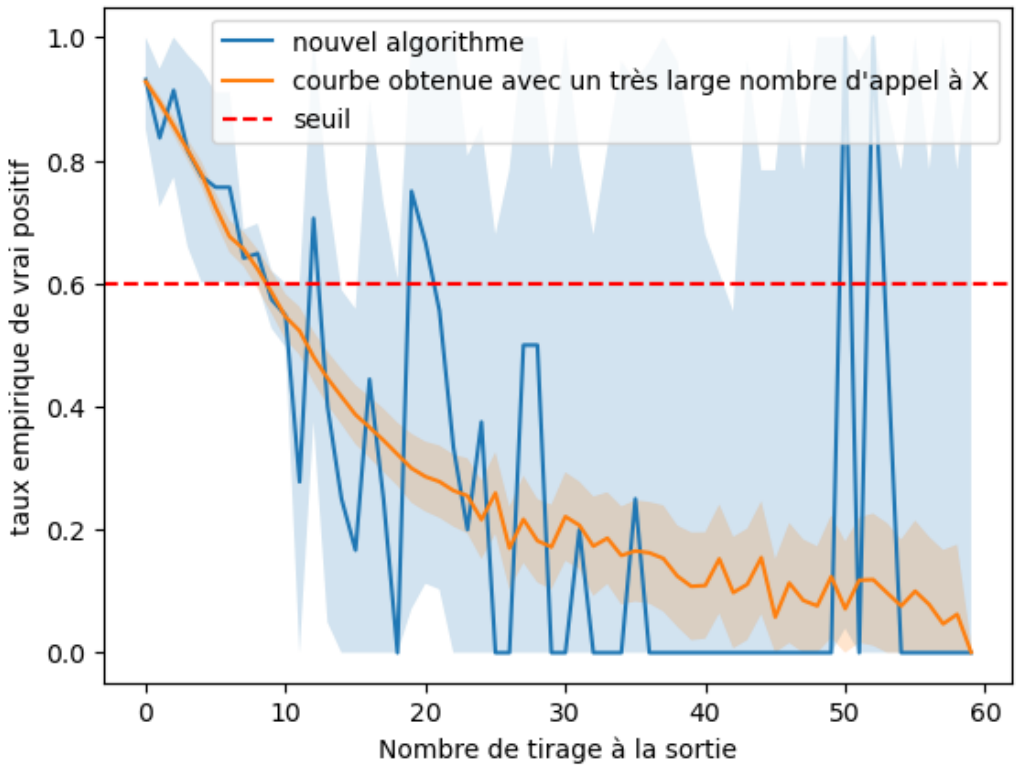

FIGURE 9.2 – Nouvel algorithme. Remarquez que les intervals de confiance deviennent plus précis autour de l'intersection de la courbe avec le seuil ce qui témoigne d'une allocation plus intelligente du budget d'appels à X

La prochaine étape de mes recherches était alors de prouver que l'hypothèse d'une courbe décroissante selon N pouvaient être faite en supposant des conditions réalistes sur le juge (par exemple que le taux de vrai positif soit supérieur au taux de faux négatif). Malheuresement je n'ai pas su résoudre ce problème. A posteriori peut être aurais je du passer plus de temps à essayer de résoudre cette difficulté mais à cette période j'hésitai déjà entre cette approche et une autre qui me paraissait plus générale, cet obstacle m'a convaincu de changer.

Cette nouvelle approche reprends exactement le même vocabulaire que la précédente $(X, Y, Q, p_0, p_1, ...)$ mais diffèrent dans la manière de tirer des réponses.

On se place dans un contexte de Best of N (BoN) c'est à dire que l'on choisit N le nombre de réponses à une question q générées par π, le juge proxy Y

assigne des labels 0 ou 1 à chacune des réponses et nous allons sélectionner une réponse au hasard parmis celles avec le label 1. Cette simulation de tirage a l'avantage d'être plus générale que la précédente car BoN est une méthode de génération de réponse largement utilisée dans le domaine des LLMs.

Mon objectif était d'optimiser le nombre de réponse et le nombre de juge utilisés pour labelliser chaque réponse afin de maximiser à la fois le taux de vrai positif et la probabilité de trouver une réponse jugée correcte (i.e. maximiser $Y = 1$) tout en limitant le nombre de query à X. L'important pour moi était notamment d'obtenir des garanties sur la qualité des résultats observés ce qui m'a ammenée à utiliser des inégalités de concentrations telles que l'inégalité de Bernstein empirique ou l'inégalité de McDiarmid. Cependant après un peu plus d'une semaine de travail sur cette voie je me suis aperçue d'une erreur bête sur une formule fondamentale à mon approche. N'ayant pas de perspective de résolution de cette erreur et la date limite de rendu de rapport approchant j'ai choisi d'arrêter mon stage içi.

Chapitre 10

Conclusion de la partie informatique

En conclusion malgrès l'absence de résultats j'ai pu à travers les changements fréquents de sujets explorer un grand nombre de domaines de recherche sur les LLMs. J'ai donc gagné beaucoup d'expérience avec les LLMs aussi bien sur le plan théorique que pratique. Maitriser l'utilisation du supercalculateur Jean Zay par exemple me sera très utile dans le futur. Cependant si je devais répéter cette expérience je ferai attention à mieux définir mon sujet de recherche au préalable.

Chapitre 11

Conclusion Générale

Ce stage au LAMSADE a constitué une expérience formatrice, tant sur le plan humain que scientifique. Sur le plan organisationnel, il m'a permis d'observer le fonctionnement concret d'un laboratoire de recherche : équilibre entre rigueur académique, ouverture internationale et convivialité. J'ai ainsi pu mieux comprendre les dynamiques de communication et de collaboration propres à un environnement de recherche de haut niveau.

Sur le plan informatique, même si mes travaux n'ont pas débouché sur des résultats publiables, ils m'ont permis d'explorer un large spectre de problématiques liées aux LLMs : robustesse en roleplay, raffinements automatiques des réponses, méthodes d'évaluation et risques de reward hacking. Ces explorations m'ont apporté une maîtrise accrue des environnements de calcul distribués et des protocoles expérimentaux, compétences précieuses pour la suite de mon parcours.

Enfin, cette expérience m'a amené à prendre du recul sur ma manière d'aborder un sujet de recherche. Si je devais la reproduire, je veillerais à circonscrire plus précisément la problématique initiale afin de mieux canaliser mes efforts. Ce stage a néanmoins renforcé mon intérêt pour la recherche en intelligence artificielle et constitue une étape importante dans la construction de mon projet professionnel.

Bibliographie

[1] LAMSADE. *Rapport HCERES*. 2022. Disponible en ligne : https://www.lamsade.dauphine.fr/fileadmin/mediatheque/lamsade/documents/Rapport_HCERES/DAE_LAMSAEDE_11avril.pdf