Datalchemy

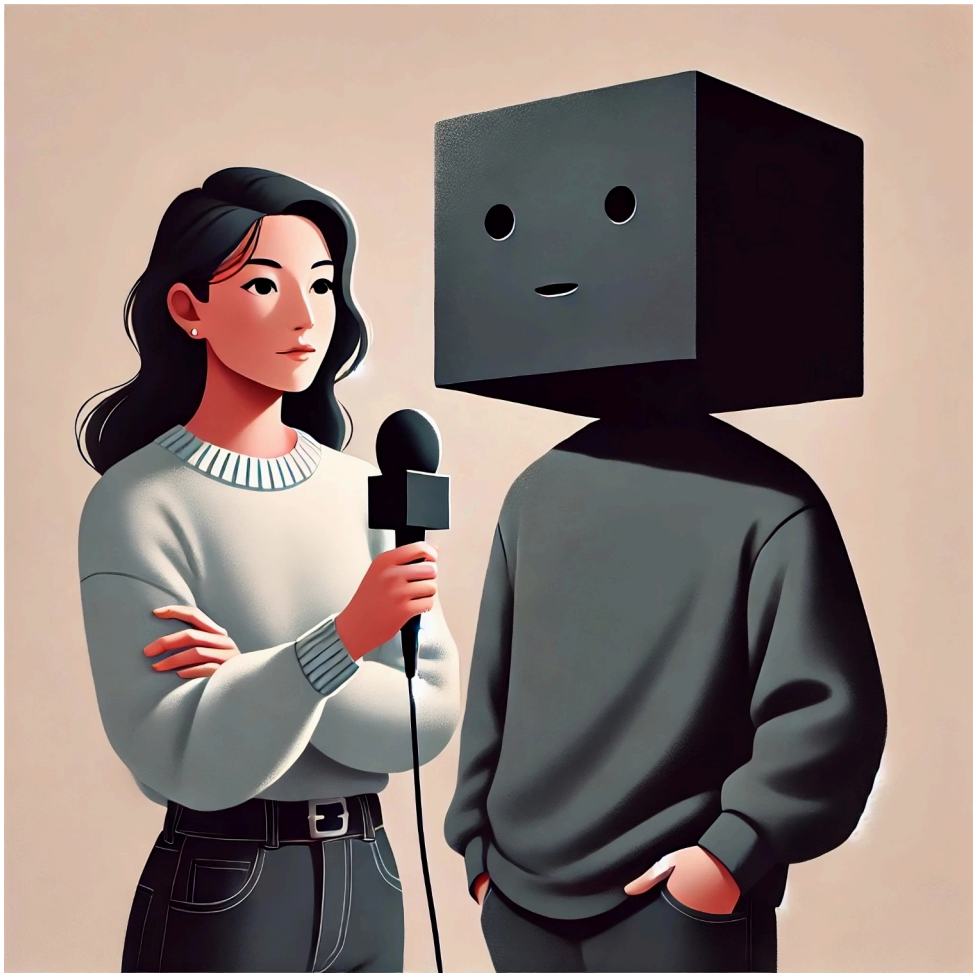
ECHOS

DE LA RECHERCHE #17

SEPTEMBRE 2024

LE RAG SOUS CONTRÔLE, LA BOÎTE NOIRE OPTIMISÉE ET LA MODÉLISATION 3D DU FUTUR

TL;DR ?



Six mot-clés de ces échos

#RAG, #évaluation, #boîte noire, #optimisation, #gaussian splatting, #NeRF

Pourquoi lire cette publi peut vous être utile concrètement ?

Si vous utilisez un RAG et vous vous demandez désespérément comment évaluer la qualité de ses réponses d'une manière objective, vous trouverez ici un travail récent et prometteur pour déjà mieux mesurer les choses. Si vous voulez chercher les solutions d'un problème complexe le plus vite possible, Google nous offre maintenant un de ses meilleurs outils . Et si vous travaillez avec 3 dimensions et suivez l'évolution du domaine avec l'intelligence artificielle (Nerfs, Gaussian Splatting), NVIDIA a poussé l'effort sur une application simple en proposant de nombreuses innovations.

Quels process métier seront probablement modifiés sur la base de ces recherches ?

Les RAGs sont partout, mais leurs limites sont trop souvent sous-estimées par rapport à leur exposition. L'apparition d'un nouveau framework et de nouvelles métriques pour évaluer ces outils facilitera grandement leur mise en place et leur qualification. D'autre part, l'optimisation de boîtes noires est un sujet très générique, que l'on retrouve en étude d'impact d'actions publiques, en optimisation de ressources disponibles, ou en recherche par simulation ou jumeau numérique. L'arrivée de Google Vizier va donner un avantage certain aux acteurs qui s'en saisiront. Enfin, les NeRFs sont une nouvelle technique de modélisation 3D issue du Deep Learning qui impactera à terme de nombreux domaines : jeu vidéo, cinéma, interaction en ligne, etc.

Si vous n’avez qu’une minute à consacrer à la lecture maintenant, voici le contenu essentiel en 7 phrases

1. Amazon propose un nouvel outil et de nombreuses métriques pour mieux évaluer la qualité d’un RAG.

2. Ces métriques permettent de sortir d’une vision trop globale de ces outils pour, enfin, en analyser les avantages et défauts. On ne peut pas imaginer l’industrialisation d’un outil sans moyen sain d’évaluer ses performances

3. Plusieurs approches sont testées, avec des résultats intéressants exposant notamment le « savoir interne » du LLM ou sa capacité à halluciner

4. L’optimisation de « black boxes » permet de régler de très nombreux sujets au-delà de l’intelligence artificielle. Google propose aujourd’hui, en open source, une version python de son outil maison : Google Vizier

5. Cet outil est particulièrement générique et demande très peu d’adaptation au problème cible. Il est aussi redoutablement bien optimisé pour accélérer la recherche de solutions.

6. Le monde de la 3D connaît depuis quatre ans un bouleversement avec les NeRFs, méthode Deep Learning pour représenter cette information avec une qualité incroyable.

7. NVIDIA propose un travail pour utiliser correctement et efficacement ces nouveaux outils, via la génération et l’animation d’avatars : GAvatar.

OPTIMISATION DE BOÎTES NOIRES (GOOGLE VIZIER), EVALUATION COMPLÈTE D'UN RAG & GAVATAR PAR NVIDIA

RAG Checker : Amazon tente d'évaluer correctement un RAG

Et le sujet reste, évidemment, très/trop complexe.

Un peu de contexte : si vous travaillez de près ou de loin en intelligence artificielle et n'avez jamais entendu parler de RAG (retrieval augmented generation), vous êtes soit isolés de l'actualité avec talent, soit particulièrement robuste pour ignorer les « modes » techniques. Derrière ces termes se cache une application théoriquement facile des Large Language Models pour aller rechercher, dans une base documentaire plus ou moins gigantesque, des éléments pertinents à partir de la requête d'un utilisateur. Nous avions notamment beaucoup parlé de ces approches et (surtout) de leurs limites lors d'un webinaire en 2024.

Car s'il y a, depuis l'arrivée de la pseudo-AGI d'OpenAI (chatGPT), une effervescence énorme autour de ces sujets, force est de reconnaître que dans une majorité de cas, l'outil RAG constitue un excellent Proof of Concept qui illuminera des slides de présentation, mais n'arrive pas à être concrétisé comme un outil industrialisable et contrôlable. De nombreuses défaillances peuvent se produire dans le cadre d'une application naïve (éléments remontés insuffisants ou erronés, hallucinations dans la génération finale…). Ce n'est d'ailleurs pas un hasard si, à date, nous proposons des approches moins « révolutionnaires » en apparence, mais beaucoup plus stables dans leur implémentation, notamment en exploitant les Large Language Models en extraction et en structuration, et en refusant un échange par le langage naturel en restitution pour préférer un outil de navigation efficace et testable (nos excuses pour cet aparté à la limite du commercial, mais n'hésitez pas à nous contacter si vous voulez des approches fonctionnelles et industrialisables 😊)

Le RAG est donc un champ de ruine, et se reporter à la recherche fondamentale permet de s'extraire de la hype pour retrouver la terre ferme. Nous avions dans un webinaire notamment cité les travaux SAFE de Google Deepmind qui valident un résultat de RAG par des recherches Google unitaires. Ici, les équipes d'Amazon nous proposent RAGChecker: A Fine-grained Framework for Diagnosing Retrieval-Augmented Generation, de Ru et al[1]. L'objectif de ce travail très récent est de proposer un moyen sain et contrôlé d'évaluer correctement la qualité d'un RAG.

À ce stade, prenons un peu de recul. Pendant que de nombreux acteurs vendent leur RAG optimal et fonctionnel, les chercheurs, eux, tentent déjà d'estimer la qualité d'une solution RAG d'une manière correcte. Ce grand écart est révélateur de la hype en cours, et soyons lucides : si nous ne savons pas réellement comment évaluer un outil de ce type à date, nous avons peu de chance de pouvoir prétendre le faire fonctionner d'une manière industrielle. Ces travaux de recherche sont donc fondamentaux pour nous, en nous donnant de nouveaux outils pour évaluer les RAG, identifier des sources d'erreur et ainsi pouvoir se projeter dans une utilisation future dépourvue de mauvaises surprises…

RAGChecker propose donc de nouvelles métriques pour évaluer un RAG, et nous allons voir qu'aucune d'entre elles n'est superflue.

[1] https://arxiv.org/pdf/2408.08067v2

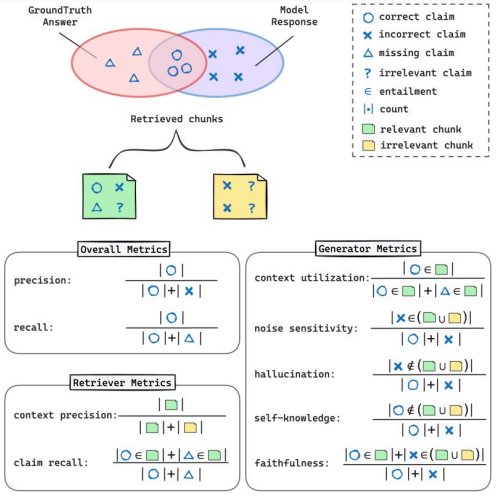


Figure 1: Illustration of the proposed metrics in RAGCHECKER . The upper Venn diagram depicts the comparison between a model response and the ground truth answer, showing possible correct(○), incorrect(×), and missing claims(△). The retrieved chunks are classified into two categories based on the type of claims they contain. Below, we define the overall, retriever, and generator metrics, illustrating how each component of the RAG system is evaluated for its performance.

Dans le schéma ci-dessus, nous avons, représentés en vert et jaune, respectivement les chunks (morceaux de document, une autre névrose des RAGs) récupérés par la recherche initiale qui correspondent à la recherche de l'utilisateur (en vert) et ceux remontés à tord car sans lien avec cette recherche (en jaune). Chaque ensemble de chunk peut contenir des affirmations vraies (correct claims, ronds), des affirmations fausses (incorrect claims, croix) et des affirmations hors sujets (points d'interrogation).

Une fois les chunks remontés, le RAG va tenter de les exploiter pour générer une réponse finale. Dans le schéma tout en haut, nous trouvons à gauche (dans l'ellipse rouge) les bonnes réponses qui auraient dû être données, et à droite en bleu la réponse du modèle. RAGChecker propose trois types de métriques :

- Overall metrics : Précision et rappel, deux métriques très classiques en classification. Notons que ces métriques prennent en compte autant la phase de récupération des chunks que la phase de génération de la réponse. Un élément pertinent peut ne pas être remonté en récupération, comme il peut être remonté mais être totalement ignoré par le LLM.

- Retriever metrics : Nous nous intéressons ici exclusivement à la récupération de candidats (première phase du RAG). Le claim recall permet de voir à quel point cette récupération n'a pas raté d'information importante. Le context precision, lui, vérifie la proportion d'éléments pertinents remontés globalement.

- Generator metrics : C'est ici que le RAGChecker propose les choses les plus intéressantes. La Context Utilization permet de vérifier si le générateur n'a pas ignoré trop d'éléments valables remontés par le Retriever. La noise sensitivity, elle, observe l'impact de la présence d'éléments hors sujet remontés par le Retriever. Deux métriques portent sur les impacts du LLM en génération : hallucination permet d'observer le nombre d'affirmations fausses générées par le modèle sans qu'elles soient présentes dans les chunks remontés. Self-knowledge, lui, permet d'observer si le modèle a rajouté des informations valables qui n'étaient pas présentes dans la documentation.

Des tests sont ensuite réalisés en comparant plusieurs solutions disponibles à date. Il est ici très important de regarder en quoi correspondent ces tests. Cela permettra de se projeter face à une implémentation sur un problème spécifique. Notamment, nous attirons l'attention du lecteur sur le fait que ces datasets d'évaluation portent très majoritairement sur des problèmes de connaissance générale. Dès lors, il est un peu risqué de se projeter, depuis ces résultats, vers l'utilisation d'une base documentaire très spécifique comme nous en rencontrons régulièrement chez nos clients :

Table 1: Statistics of the RAG benchmark. This benchmark is repurposed from public dataset across 10 domains, containing 4,162 questions. For the domains of Finance, Lifestyle, Recreation Technology, Science and Novel, the short answers are extended to long-form answers with GPT-4.

Dataset	Domain	# Query	# Doc.	Source	Example Query
ClapNQ	Wikipedia	300	4,293	ClapNQ	Difference between russian blue and british blue cat
NovelQA	Novel	280	19	NovelQA	When do the Ewell kids go to school?
RobustQA Writing	Writing	500	199,994	LoTTE, RobustQA	What is the difference between online and internet?
RobustQA BioASQ	Biomedical	511	197,816	BioASQ	What hand deformities do patients with Apert syndrome present with?
RobustQA Finance	Finance	500	57,638	FiQA, RobustQA	Is it safer to send credit card number via unsecured website form or by e-mail? What safer options are there?
RobustQA Lifestyle	Lifestyle	500	119,461	LoTTE, RobustQA	Can i eat a day old peanut butter sandwich?
RobustQA Recreation	Recreation	500	166,975	LoTTE, RobustQA	Why are so many american (spy) movies set in europe?
RobustQA Science	Science	500	125,368	LoTTE, RobustQA	Where is the flaw in this proof that 1=2? (derivative of repeated addition)
RobustQA Technology	Technology	500	638,509	LoTTE, RobustQA	Why not use larger cipher keys?
KIWI	AI Science	71	429	KIWI	What are the prior approaches proposed to improve faithfulness of the reasoning steps generated by LLMs and what tasks are they applied on?

Et donc, les résultats :

Table 3: The averaged evaluation results for different RAG systems across 10 datasets. The overall performance of the RAG system is quantified using precision (Prec.), recall (Rec.), and F1 scores. The retriever component is evaluated based on claim recall (CR) and context precision (CP), while the generator component is diagnosed through context utilization (CU), relevant noise sensitivity (NS(I)), irrelevant noise sensitivity (NS(II)), hallucination (Hallu.), self-knowledge (SK), and faithfulness (Faith.). Additionally, the average number of response claims for each RAG system is provided.

RAG systems	Overall			Retriever		Generator						#Claim
	Prec.↑	Rec.↑	F1↑	CR↑	CP↑	CU↑	NS(I)↓	NS(II)↓	Hallu.↓	SK↓	Faith.↑	
BM25_GPT-4	61.0	49.7	50.3	74.0	52.3	61.4	26.2	4.1	8.7	3.4	87.9	12
BM25_Llama3-8b	52.1	43.9	42.1	74.0	52.3	54.9	31.3	6.1	9.8	1.8	88.4	11
BM25_Llama3-70b	59.1	44.9	46.3	74.0	52.3	56.2	30.4	5.3	5.1	1.7	93.2	9
BM25_Mixtral-8x7b	52.5	44.3	42.9	74.0	52.3	54.9	34.3	5.8	6.2	1.8	92.0	9
E5-Mistral_GPT-4	62.0	53.0	52.7	83.5	61.8	60.4	28.9	3.5	5.7	1.4	92.9	12
E5-Mistral_Llama3-8b	53.8	48.3	45.0	83.5	61.8	55.0	33.5	5.5	6.6	0.8	92.7	11
E5-Mistral_Llama3-70b	60.6	50.4	50.2	83.5	61.8	57.6	31.7	4.3	3.3	0.8	95.9	10
E5-Mistral_Mixtral-8x7b	53.1	48.6	45.7	83.5	61.8	55.2	36.5	5.1	4.0	0.8	95.2	10

- Nous voyons déjà ici l'intérêt d'avoir plusieurs métriques spécifiques. Typiquement, en considérant uniquement les scores de précisions et de rappel, GPT4 remporte le combat haut la main. Néanmoins, nous voyons par exemple qu'il aura tendance à beaucoup plus halluciner que d'autres approches basées sur LLama3

- GPT4 l'emporte aussi grâce à son savoir interne (Self Knowledge). C'est un bon et un mauvais point pour GPT4. Plus le sujet sera spécifique/technique, moins ce savoir interne aura de chance d'aider les résultats.

- Sans surprise, l'approche BM25, souvent choisie par défaut car plus simple et plus courante, est nettement en dessous de l'approche E5 basée sur un Mistral.

Plusieurs observations intéressantes sont proposées par les auteurs :

- La qualité du Retriever est, sans surprise, fondamentale. La qualité de la réponse finale est très dépendante des chunks documentaires remontés.

- Plus le modèle d'un générateur est gros, meilleur il sera (LLama 70b VS Llama 8b). Nous retrouvons le classique et déprimant Bigger is better du Deep Learning

- Une utilisation stable du contexte fourni est clé de bons résultats dans l'outil.

De nombreuses autres observations sont proposées dans cette publication que nous vous suggérons de lire si le sujet vous intéresse. Ce type de travail est fondamental : mieux évaluer la qualité d'un outil est toujours un axe pertinent pour améliorer cet outil. Dit autrement, l'aveuglement actuel qui entoure la capacité à qualifier un RAG est source de très nombreux problèmes dans l'implémentation de ces outils en entreprise ou auprès du grand public.

Black box optimization : Google propose son outil phare, Google Vizier, en open source

« Optimisation de boîtes noires » ? Derrière ce terme quelque peu barbare se cache une collection d'outils fondamentaux qui, certes, intéressent tout pratiquant en intelligence artificielle, mais qui va bien au-delà en termes d'intérêts et d'applications. Et avant de plonger (avec joie) dans la technique, il peut être pertinent de rappeler l'étendue du sujet.

Une « boîte noire » en informatique est un système entrée/sortie trop complexe pour que l'on puisse analyser son fonctionnement interne et en tirer des enseignements effectifs. Optimiser une telle « boîte noire » revient à chercher le meilleur moyen de l'exploiter malgré l'aveuglement intrinsèque quant à son fonctionnement. Ce type d'approche est très générique, et concerne un nombre important d'applications. Nous avons l'habitude, en Deep Learning, d'utiliser ce type d'outils pour optimiser les hyper-paramètres qui définissent fortement la convergence d'un réseau de neurones, mais au-delà, ils peuvent permettre d'étudier l'impact d'une politique publique comme d'une campagne publicitaire ou d'optimiser une simulation (comme un jumeau numérique) pour résoudre un problème spécifique. On procèdera de la même manière pour chercher une meilleure solution, par exemple en assignation de ressources, même si ces cas sont d'ordinaire mieux gérés par un bon vieil optimiseur par contraintes (hello Timefold, notre ami de toujours).

Évidemment, un tel outil d'optimisation représente un challenge technique d'envergure. Le fait d'avoir autant d'applications différentes empêche de spécialiser la recherche de solution. Mais de nombreux outils sont apparus ces dernières années, et si vous êtes passés devant l'un de nos formateurs, vous aurez probablement entendu parler du bon vieil Optuna

Aujourd'hui, nous accueillons un nouveau candidat, et clairement pas des moindres, comme nouvelle corde à notre arc. Google Vizier est un outil qui, depuis plus de dix ans, est utilisé en interne chez Google pour régler des problèmes extrêmement différents avec succès. Une version open source est sortie récemment, et la publication The Vizier Gaussian Process Bandit Algorithm de Song et al est l'occasion de parcourir les approches et les résultats de cet outil d'optimisation. On estime à plusieurs dizaines de millions le nombre de problèmes gérés en interne chez Google par Vizier, et si la version initiale était implémentée en C++ (une technologie formidable si vous avez déjà perdu votre jeunesse, mais assez complexe et demandant des compétences et névroses plus rares sur le marché du travail), une nouvelle version est arrivée en Python basée sur TF-probability, qui exploite les GPUs pour accélérer fortement la recherche de solution. Et nous allons voir que les résultats sont assez impressionnants.

*** Attention, début d'immersion technique, sautez les prochains paragraphes si vous voulez rester à haut niveau. ***

L'approche globale est basée sur l'optimisation Bayésienne de processus gaussiens, un classique dans ce domaine. Cette approche est présentée ci-dessous :

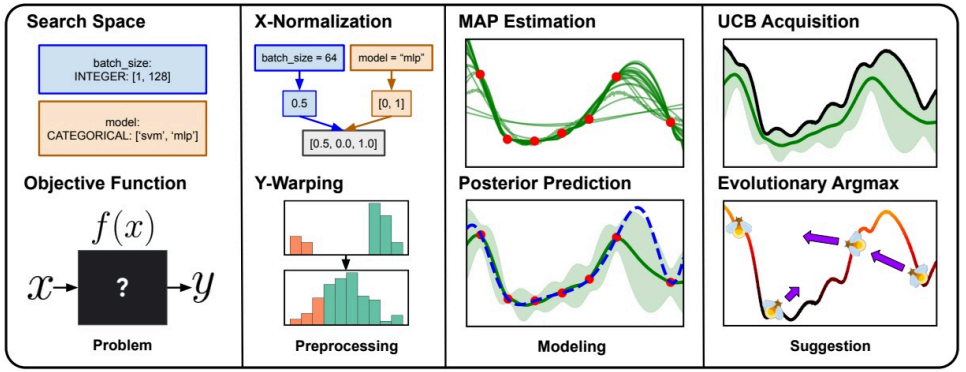


Figure 1 | Key components of the Google Vizier Bayesian optimization algorithm.

De gauche à droite :

- Search space : nous parlons ici des différents paramètres modifiant le comportement de la boîte noire que nous désirons utiliser. Ces paramètres peuvent être de formes très différentes : variables continues, variables discrètes, ou même valeurs issues d'une énumération.

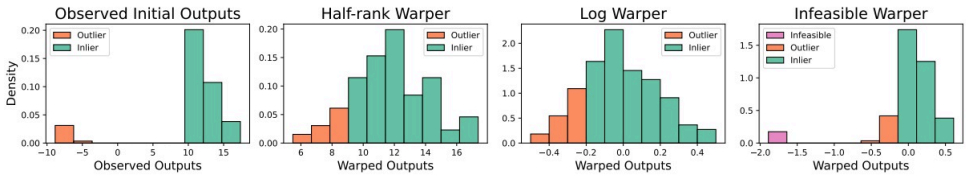
- Objective function : Nous voulons optimiser un résultat, soit, obtenir le meilleur score possible de cette fonction en jouant sur les paramètres du search space.

- X-normalization & Y-warping : Nous en reparlons juste après 😊

- MAP estimation & Posterior prediction : application d'un UCB Bayésien classique. Chaque tentative (choix de paramètres pour observer la valeur résultante en objectif) permet d'affiner une estimation de la fonction objectif que nous voulons optimiser. Cette estimation permettra de disposer de nouvelles valeurs à tester pour la prochaine fois

- UCB Acquisition & Evolutionary argmax : Technique de bandits classique (si si), et application ensuite d'une recherche de solution par optimisation intéressante, que nous détaillons plus bas.

Nous nous proposons juste d'éclairer deux points techniques pertinents dans cette approche. Le premier porte sur le Y warping qui est ici particulièrement travaillé. Rappelons que nous sommes face à un outil générique, supposé capable de s'atteler à de nombreux problèmes différents. Les valeurs retournées par la fonction objectif n'ont donc aucune chance d'avoir un comportement « appréciable » permettant une recherche de solutions. L'approche ici est donc séquentielle, représentée par le schéma ci-dessous :



Le Half-rank wrapping va permettre d'agir sur les résultats obtenus décevants (valeurs ici inférieures à 10) en normalisant la distribution de valeurs. Ces résultats décevants seront déplacés pour occuper la moyenne inférieure des valeurs cibles. Le Log warping, lui, améliore la distribution des valeurs pour les résultats intéressants. Enfin, le Infeasible Warper va lui identifier des objectifs inutiles et les déplacer dans une région à part afin d'éviter que dans sa recherche de solutions, l'algorithme continue d'explorer cet espace.

Le second point technique intéressant est ici l'algorithme permettant de mettre à jour l'ensemble des valeurs détectées. Le Firefly Algorithm (algorithme des lucioles) est un algorithme d'optimisation massive proposé en 2009, et inspiré du comportement des lucioles. Chaque « luciole » représente ici une expérimentation avec sa solution. La « lumière émise » par la « luciole » est la qualité de la solution trouvée. Dans une passe d'update, les lucioles les plus lumineuses vont attirer celles moins lumineuses, et un vecteur de mouvement moyen va permettre de mettre à jour ces solutions pour chercher de meilleurs candidats. Nous sommes dans un algorithme de « swarm optimization » où on multiplie le nombre d'élément pour obtenir, globalement, une itération pertinente sur les solutions, d'une manière assez proche des algorithmes à évolution.

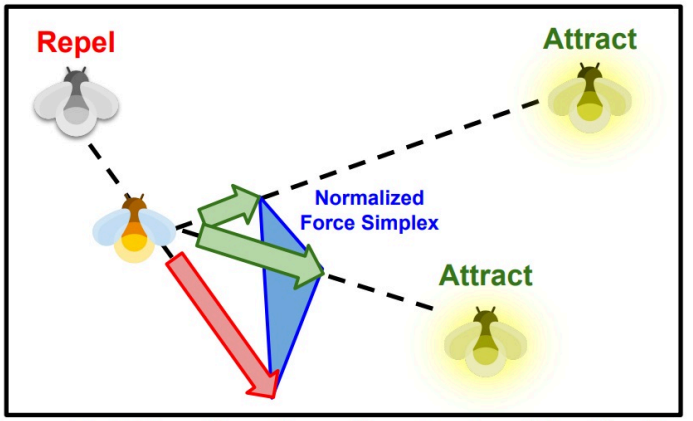


Figure 3 | Visualization of vectorized firefly force normalization and summation. The resulting firefly will move into the simplex.

*** Attention, fin d'immersion technique, bienvenue à ceux qui nous avaient quittés. ***

Passons aux résultats. Google Vizier est-il aussi performant qu'il le promet ? Les résultats sont effectivement assez pertinents. Il convient de souligner que dans les résultats donnés, les auteurs partent de méthodes par défaut sans recherche spécifique propre au problème ou à l'algorithme. L'enjeu est de se mettre dans la situation d'un acteur qui désire optimiser son problème avec le moins possible d'adaptations spécialisées. Cette approche est intéressante pour nous, mais peut desservir certains algorithmes concurrents. Ces concurrents sont ici Ray Tune (un ancêtre de 2018 toujours vaillant), Ax (outil de MetaAI très performant), HEBO (un travail d'optimisation bayésienne datant de 2020 et très efficace), HyperOpt, Optuna et Sci-kit Optimize (vous je ne sais pas, mais je fatigue sur les parenthèses). Plusieurs expérimentations sont proposées.

La première comparaison se fait sur un problème strictement continu (variables pouvant prendre n'importe quelle valeur dans un segment donné, par exemple, la position d'un élément ou la force appliquée sur un élément mécanique). Les auteurs affichent l'efficacité (donnée prenant en compte la qualité des résultats, mais aussi la puissance de calcul nécessaire pour obtenir ces résultats). Nous observons que Vizier n'est pas nécessairement le meilleur :

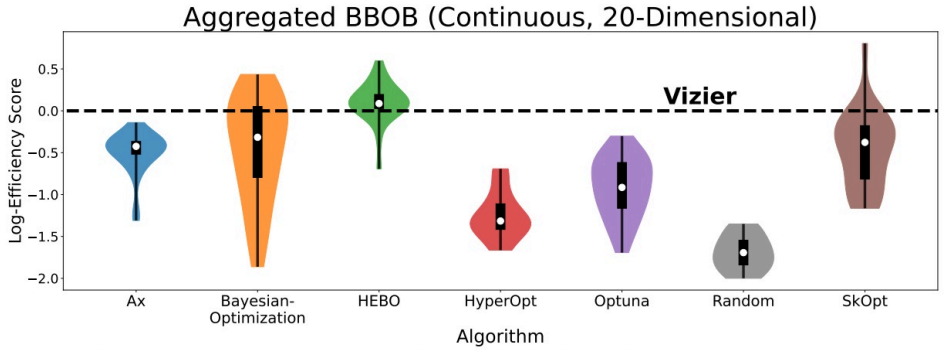


Figure 6 | Higher is better (↑). Violin plots displaying distributions of log-efficiency scores across all BB functions. Similar plot for noisy objectives in Appendix A.2.

En effet, HEBO et SkOpt peuvent donner des résultats assez supérieurs. Mais nous sommes ici sur un problème à 20 dimensions (20 valeurs différentes pour modéliser une solution), et tout data-scientist qui se respecte sait que ce nombre peut facilement augmenter. Vizier prend alors son avantage :

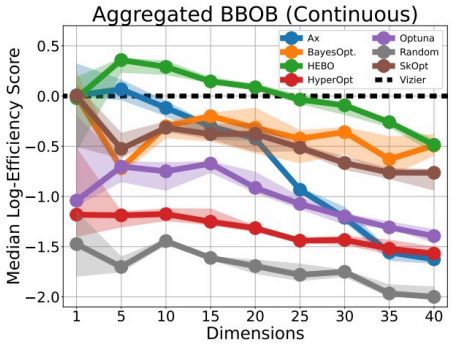


Figure 7 | Higher is better (↑). Median log-efficiency scores across algorithms over continuous BBOB functions, while varying search space dimensionality.

Notons aussi que dès que nous passons dans des problèmes où les paramètres ne sont pas continus, Google Vizier prend un avantage très net. Or, la majorité des problèmes ne sont pas continus dans leurs modélisations :

Enfin, sachez que Vizier brille particulièrement dans le cas où la recherche est parallélisée (Batched cases) et où l'objectif est multidimensionnel (plusieurs valeurs pour modéliser une solution).

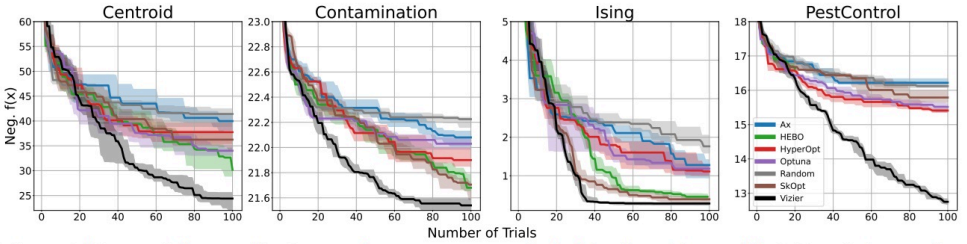


Figure 9 | Lower is better (↓). Best-so-far across categorical objectives (**Centroid:** 24D × 3 Categories, **Contamination:** 25 Booleans, **Ising:** 20 Booleans, **PestControl:** 25D × 5 Categories). Note: HEBO halted early in PestControl.

Google Vizier est donc un outil à mettre de toute urgence dans la boîte à outils d'un data scientist. Très efficace et particulièrement générique face à différents problèmes, il peut permettre de produire des solutions pertinentes pour un coût acceptable (selon la complexité du processus à optimiser, évidemment).

GAVATAR

L'AVENIR DE LA MODÉLISATION 3D EN IA GAGNE EN SOLIDITÉ

Et sans surprise, NVIDIA est aux premières loges. Mais un peu de contexte avant de présenter ce travail ne sera pas superflu.

Ce sera le sujet de notre webinaire d'octobre, la modélisation en 3 dimensions révolutionnée par le Deep learning. Nous avions aussi publié une revue de recherche fin 2023 spécialisée sur ce sujet. Originellement, une forme en 3 dimensions était modélisée soit par un nuage de points flottant dans l'espace, soit par des formes très simples (meshes) joints entre eux. Ces modélisations fonctionnaient très bien, mais souffraient d'une limite forte de complexité, notamment pour générer de nouvelles formes en 3 dimensions. Or, le NeRF, apparu en 2020, a radicalement changé la donne, en proposant de modéliser une forme en 3 dimensions par un réseau de neurones spécifique. L'application était la suivante : un utilisateur prend une série de photographies d'une scène ou d'un objet en conservant l'information de position de l'appareil photo. Le NeRF était ensuite entraîné pour apprendre à modéliser la scène, et pouvait alors être utilisé pour générer de nouvelles vues (sous de nouveaux angles/positions) de la scène avec succès.

Comme à l'accoutumée en intelligence artificielle, cette approche portait autant de qualités que de défauts. En effet, si l'application de génération et la qualité des résultats sont sidérantes, la génération de la modélisation était très lourde et très lente (initialement jusqu'à une heure). Au-delà, utiliser correctement ces nouveaux outils de modélisation pour des applications pratiques était un sujet totalement en friche. Deux travaux sont ensuite arrivés :

- Le InstantNGP de NVIDIA qui permettait de contrôler le niveau de précision de la scène en 3D modélisée, et qui optimisait déjà la manipulation du résultat en 3D

- Surtout, le Gaussian Splatting de l'INRIA (cocorico de rigueur) qui, en 2022, a énormément optimisé la modélisation et a fortement démocratisé ces approches. Suite à ces travaux, on a vu apparaître des outils de visualisation 3D basés sur cette technique, encapsulés dans un navigateur web.

Dès lors, c'était une simple question de temps avant de voir apparaître des travaux plus appliqués. Notons que l'industrie du jeu vidéo est aux premières loges pour se saisir de ces technologies et les employer. Aussi la sortie du GAvatar de Yuan et al (NVIDIA)[1] est-elle un événement à ne pas rater

Ici, l'approche vise directement à générer et à animer des avatars en 3 dimensions. Les auteurs expliquent que leur approche permet d'animer un modèle en allant jusqu'à 100 images par seconde (Attention : les auteurs sont ici des coquins qui ne donnent pas le GPU utilisé pour obtenir ce nombre, nous forçant à un minimum de précaution).

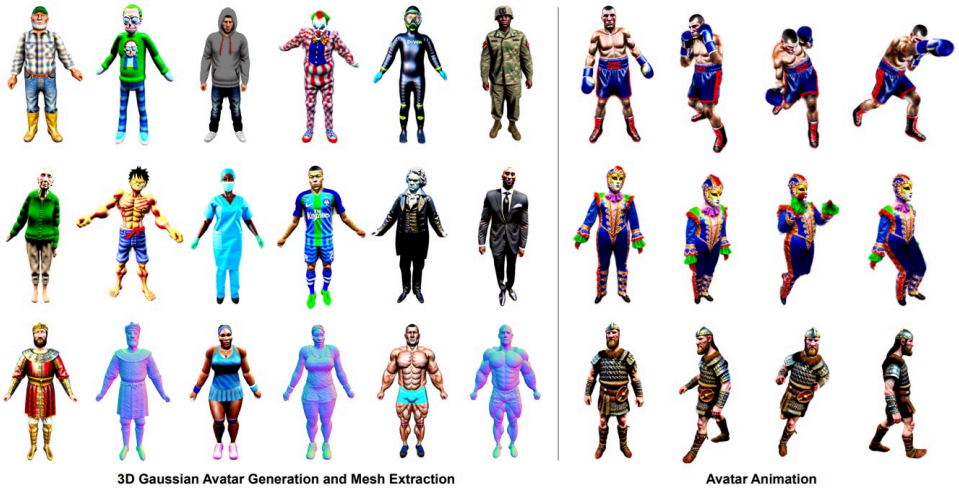


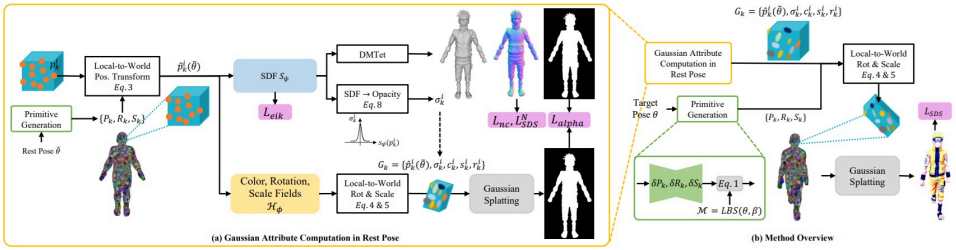
Figure 1. GAvatar synthesizes high-fidelity 3D animatable avatars from text prompts. Our novel primitive-based implicit Gaussian representation enables efficient avatar animation (100 fps, 1K resolution) and also extracts a highly detailed mesh from learned 3D Gaussians.

[1] https://nvlabs.github.io/GAvatar/

La méthode employée mérite ici d'être analysée :

*** Attention : plongée en environnement technique, les lecteurs ayant souffert face à la géométrie sont priés d'enjamber cette section jusqu'à la présentation des résultats ***

Nous partons du schéma ci-dessous pour ensuite isoler certains points à relever :



- Modélisation de l'avatar : Ici, le corps humain est représenté par une collection de formes primitives (des cubes) dont la position et la déformation va évoluer au cours de l'optimisation des modèles. Chaque primitive « contient » un ensemble de gaussiennes en 3D similairement au Gaussian Splatting original. Cette approche permettra notamment d'animer l'avatar, mais aussi de regrouper les gaussiennes générées géométriquement d'une manière efficace et d'éviter certains écueils du Gaussian Splatting

- Paramétrisation des gaussiennes : le rectangle jaune en bas à gauche permet de retrouver la caractérisation de chaque gaussienne. On parle ici de Neural Implicit Fields. C'est donc un réseau de neurones unique qui sera entraîné à effectuer ces prédictions d'apparence et de déformation. Cette approche stabilise l'entraînement par rapport à la version initiale où les auteurs optimisaient ces valeurs comme des attributs directes des gaussiennes.

- SDF (Signed Distance Field) : le rectangle bleu à gauche (SDF) joue un double rôle : évaluer l'opacité des gaussiennes, mais aussi modéliser le résultat comme un Signed Distance Field. Cette modélisation est importante, car elle permet beaucoup plus facilement de générer un mesh exploitable que les NERFs qui, eux, sont peu exploitables ensuite dans une chaîne graphique. La partie de droite modélise l'utilisation.

Avec une pose cible (position de l'avatar), les primitives vont être déformées en entraînant avec elles les gaussiennes liées. On utilisera ensuite le splatting (projection sur une surface en deux dimensions pour affichage) classique pour afficher le résultat.

*** Attention :fin de plongée en environnement technique, veuillez laisser les gaussiennes dans l'état où vous les avez trouvé à votre arrivée ***

Les résultats sont (seraient ?) très efficace (100 fps), mais aussi visuellement bien au-dessus de la concurrence :

Figure 3. **Generated avatars by our method and their mesh normals and texture meshes.**

Pourquoi ce travail est-il si important ? Parce que nous commençons enfin à avoir des applications concrètes des travaux en *NeRF*. Ces applications permettent de confirmer que ce qui était une révolution technique théorique et esthétique va devenir demain une modélisation centrale. Et les sujets d'application ne manquent pas : *SLAM* (capacité d'un agent robotique à parcourir un espace, à modéliser cet espace en 3D et à se situer), segmentation/ analyse d'une scène, robotique (modélisation d'interactions)… Ici, la génération d'avatar peut paraître un peu décevante comme application, mais c'est un pas important vers une démocratisation et une industrialisation de ces approches. *NVIDIA* se permet ici de générer facilement de nouveaux avatars, de pouvoir exporter les résultats en *meshes* exploitables par les outils traditionnels, mais aussi de les animer en temps réel. Certaines limites restent présentes : les couleurs sont ici souvent trop saturées, l'animation reste approximative par rapport à ce que l'on observe en technique classique. Pour ce dernier point, les auteurs imaginent que l'emploi de règles physiques ou de modèles de diffusion vidéo pourrait améliorer les choses. A suivre, donc.

Datalchemy

contact@datalchemy.net

Images extraites des articles respectifs ou générées par IA