Université Dauphine-PSL

Master 1 - MIAGE

2023 - 2024

Comment améliorer la qualité d'extraction de données non structurées par des LLM dans le cadre de notations ESG ?

Paul MALET

Sous la supervision de Florian Yger.

Alternance effectuée chez Iceberg Data Lab, sous la supervision de Pierre de Sainte Agathe.

Table des matières

Introduction

Fondée en 2019, Iceberg Data Lab (IDL) est une start-up spécialisée dans le calcul d'empreinte environnementale pour la finance grâce à des équipes de modélisateurs et d'analystes sectoriels. IDL calcule les empreintes environnementales d'entreprises, puis propose ces données à des institutions financières qui constituent ensuite leurs portefeuilles d'investissement avec une considération pour l'environnement. Les empreintes environnementales sont calculées à partir d'un modèle environnemental conçu en compilant des études scientifiques et en utilisant de nombreuses données collectées en continu. Les données utilisées sont à la fois financières et extra-financières, et proviennent de fournisseurs externes (payant ou opensource) et d'une collecte réalisée par les analystes sectoriels. L'extraction de données extra-financières effectuée par les analystes est cependant très chronophage, les rapports financiers et ESG étant longs et non-structurés, comprenant en moyenne entre 200 et 500 pages, et rendant l'extraction des données fastidieuse.

C'est donc avec l'objectif d'améliorer l'efficacité du travail des analystes, tout en gardant un contrôle sur les sources de données, que le projet Barbatus a débuté. A ce jour, Barbatus est un outil basé sur l'intelligence artificielle générative ayant deux applications : tout d'abord un outil conversationnel répondant à des questions au sujet des ESG, et ensuite un outil d'extraction de données structurées depuis des pdf.

A présent, seul un nombre limité de types d'informations peuvent être extraites de manière automatique et avec une certaine fiabilité. L'enjeux est donc d'étendre le domaine de compétence de Barbatus, et surtout d'assurer que les données qu'il extrait et qu'il renvoie soient de qualité.

Comment améliorer la qualité d'extraction de données non structurées par des LLM dans le cadre de notations ESG ?

Nous présenterons tout d'abord ce que sont les données ESG, puis nous établirons un état de l'art des LLM et leur application pour la notation ESG, ensuite nous étudierons des solutions pour améliorer la performance de l'extraction de données, et enfin nous analyserons ces solutions et nous formulerons des préconisation pour notre cas d'usage.

1. Les données ESG

1.1. Finance ESG : aperçu et acteurs

Qu'est ce que sont les critères ESG ? Acronyme de Environnement, sociale et gouvernance, ils sont définis par l'Autorité des marchés financiers (AMF) comme des critères "permettant d'évaluer la prise en compte du développement durable et des enjeux de long terme dans la stratégie des acteurs économiques (entreprises, collectivités, etc.)" (Autorité des marchés financiers, 2021). Différents indicateurs sont compris dans chaques critères, tels que par exemple :

- L'empreinte carbone, la gestion des déchets ou utilisation de l'eau pour le pilier E
- La qualité du dialogue social, l'insertion de personnes handicapés ou la mobilité interne pour le pilier S
- La transparence sur les hautes rémunérations, la féminisation des conseils d'administration et de direction ou la lutte contre la corruption pour le pilier G

Pour mesurer les indicateurs, différentes sources de données sont nécessaires, comme les rapports d'entreprises, les audits de tiers ou des registres publics, pour récupérer les données extra-financières nécessaires. De ces indicateurs sont créées des méthodologies de notation pour agréger l'information. Dans la suite de ce mémoire, nous ne traiterons que du pilier environnemental, le terme ESG fera donc référence à ce pilier là.

La finance ESG, aussi connue sous le nom de finance durable ou responsable, correspond à l'intégration des facteurs ESG dans la prise de décision d'investissement financiers. Ce terme a été inventé par le Programme des Nations unies pour l'environnement en 1997, mais la pratique ne s'est développée qu'au cours de la dernière décennie avec les premières obligations écologiques émises par la Banque Européenne d'Investissement (BEI) et la Banque Mondiale à la fin des années 2000 (*Critères ESG : Définition, Exemple, Enjeux Pour Les Entreprises*, 2023). Les premières obligations n'étaient cependant que peu contrôlées car aucun label ou instrument de mesure ne validait scientifiquement qu'un projet était vert.

En parallèle de ces règlementation, une augmentation des déclarations extra-financiers par les entreprises s'est développée, c'est-à-dire des critères "permettant d'évaluer un acteur économique en dehors des critères financiers habituels que sont la rentabilité, le prix de l'action et les perspectives de croissance" (Autorité des marchés financiers, 2021). Ces données pouvant être exploitées, des labels de validation de qualité environnementale émergent et des méthodes de calcul d'impacts environnementaux sont conçues par des agences de notation existantes et par de nouveaux acteurs.

Le marché se retrouve alors dominé par une poignée de gros acteurs, souvent des filiales de grandes agences de notation ou de fournisseurs de données, tels que S&P, Moody's ou MSCI. Cela s'explique notamment par la réputation établie de ces entreprises dans le marché de la donnée financière et extra-financière, mais également par le coût à l'entrée liée à la collecte et le traitement de la donnée et à la confiance des clients à gagner. Des fintech spécialisées sur un domaine voient également le jour et se positionnent alors sur des aspects précis des impacts environnementaux (Fournier, 2023).

L'importance de la lutte contre le réchauffement climatique se faisant de plus en plus évidente et les moyens à mettre en œuvre de plus en plus acceptés, les réglementations sont très croissantes. En 2014 a été mise en place la directive NFRD (Non-Financial Reporting Directive) qui oblige les entreprises européennes de plus de 500 employés à publier des rapport détaillés, suivant une certaine nomenclature, sur les sujets environnementaux, sociaux et de gouvernance. Cette directive sera remplacée en 2025 par la CSRD (Corporate Sustainability Reporting Directive) afin d'accroître la fiabilité des données publiées et de rendre plus intelligibles les performances ESG des entreprises auprès des investisseurs et du grand public (Fournier, 2023).

A chaque mise en place d'une nouvelle directive, les données disponibles aux agences de calculs d'impacts environnementaux et de notation évoluent, leur volume devenant toujours plus conséquent. Les méthodes de collectes et de traitement se doivent d'être modifiées et de nouvelles crées. Un des enjeux devient d'arriver à créer ces méthodes de la manière la plus rapide et efficace pour pouvoir mettre à profit ces nouvelles informations disponibles et arriver à mieux évaluer les impacts environnementaux des entreprises.

Ces nouvelles données existantes, il s'agit de se les procurer auprès de différentes sources.

1.2. Les sources et formes des donnés ESG

1.2.1. Diversité des sources et des formats

Les agences de notation et les fournisseurs de données ESG utilisent différentes méthodologies pour évaluer la performance environnementale d'une entreprise. Certaines méthodes sont qualitatives, basées sur des entretiens et des revues de littérature, et d'autres méthodes sont quantitatives, basées sur des modèles statistiques construits avec des revues de littérature. Dans les deux cas, il est nécessaire de posséder des données relatives aux entreprises analysées.

Les données utilisées pour l'évaluation et la notation ESG sont complexes, multiformes et multisources. Pour ce qui est des sources, nous pouvons citer notamment :

- Les rapports et déclarations des entreprises : de nombreuses données ESG sont présentes dans les rapports annuels que les entreprises publient. Ces données sont publiques et destinées aux investisseurs. Ces données peuvent également être complétées dans des rapports sur le développement durable ou dans des déclarations réglementaires. La difficulté dans l'analyse de ces rapports est que l'information est diluée et présentée sous des formats propres à chaque entreprise, ce qui rend la comparaison des données entre entreprises difficile. Cela est dû aux faibles normes de standardisation existantes.
- Les bases de données gouvernementales : ces bases de données sont cependant de taille, de format et de qualitée variée. On peut citer la base de données EDGAR mise en place par la U.S. Securities and Exchange Commission qui permet d'accéder à de nombreux documents d'entreprises.
- Des rapports d'ONG et autres associations : certaines d'entre elles sont spécialisées dans les données ESG, parfois même spécialisées sur des secteurs d'industrie. Une

fois de plus, ces sources sont de taille, format et qualité variées. Une de ces sources importantes est CDP (Carbon Disclosure Project).
- Des données achetées à des fournisseurs de données, tels que S&P, Bloomberg ou FactSet.

Au sein de chacune de ces sources, les données prennent différentes formes, présentes dans différents éléments, c'est-à-dire des textes, des tableaux ou des images. Dans la suite de ce mémoire nous nous concentrerons sur les données se trouvant dans des textes ou dans des tableaux. Les données pertinentes dans des textes sont diluées dans de grands volumes d'informations et exprimées avec des termes et des vocabulaires variants selon la source. Les données dans des tableaux sont quant à elles plus concentrées mais également présentées dans des formes et avec du vocabulaire dépendant de la source. Il n'y a pas de standardisation dans les documents et ces données sont donc non-structurées.

Ayant alors à disposition des sources et connaissant leur format, il revient à présent de s'assurer de leur qualité et de leur fiabilité.

1.2.2. La question de la qualité et de la fiabilité

Après la collecte de données, vient leur analyse et leur utilisation pour la notation. Les données collectées se doivent donc d'être fiables et de qualité car elles impactent directement l'exactitude et la crédibilité des évaluations ESG. Ces évaluations sont utilisées par des investisseurs et des régulateurs, qui ont en plus des exigences de stabilité dans la durée.

Tout d'abord, la qualité des données ESG peut être compromise par des lacunes dans la collecte initiale. Comme expliqué précédemment, les sources sont disparates, multiformes et peuvent donc contenir des incohérences, des erreurs de saisie et être incomplètes, ne couvrant pas toutes les activités de l'entreprise regardée ou tout le secteur d'activité. En plus de cela, seul une quantité limitée de données est disponible car de nombreuses entreprises ne divulguent pas toutes les informations pertinentes, que ça soit volontairement ou car ne souhaitent pas faire l'effort de le faire.

De plus, l'absence de standardisation des rapports d'activitées en matière d'impacts environnementaux fait que beaucoup de données sont agrégées et peu granulaires. L'enjeux est alors de trouver des données plus granulaires ou de mettre au point des méthodologies permettant de consolider les données existantes et d'en déduire de nouvelles à partir des existantes.

Par ailleurs, la fréquence de mise à jour des données ESG varie considérablement d'une entreprise et d'une source à l'autre. Certains indicateurs sont mis à jour régulièrement tandis que d'autres peuvent rester inchangés pendant plusieurs années. Cela rend difficile le suivi des tendances à court terme et la détection rapide des changements significatifs de la performance ESG d'une entreprise.

En conclusion, la qualité et la fiabilité des données utilisées pour la notation ESG représente un défi majeur pour les analystes et les investisseurs. Améliorer ces aspects présente des défis techniques en matière de collecte et de traitement de la donnée. Il est par ailleurs nécessaire de normaliser les méthodes de déclarations, accroître la transparence des

données et encourager des acteurs du secteur financier à collecter ces données la et à les vérifier de manière rigoureuse. Un effort concerté pour faire cela permettrait d'améliorer la fiabilité des évaluation ESG et de renforcer la confiance en elles, à fin d'intégrer les considérations ESG dans les décisions d'investissement et de gestion des risques.

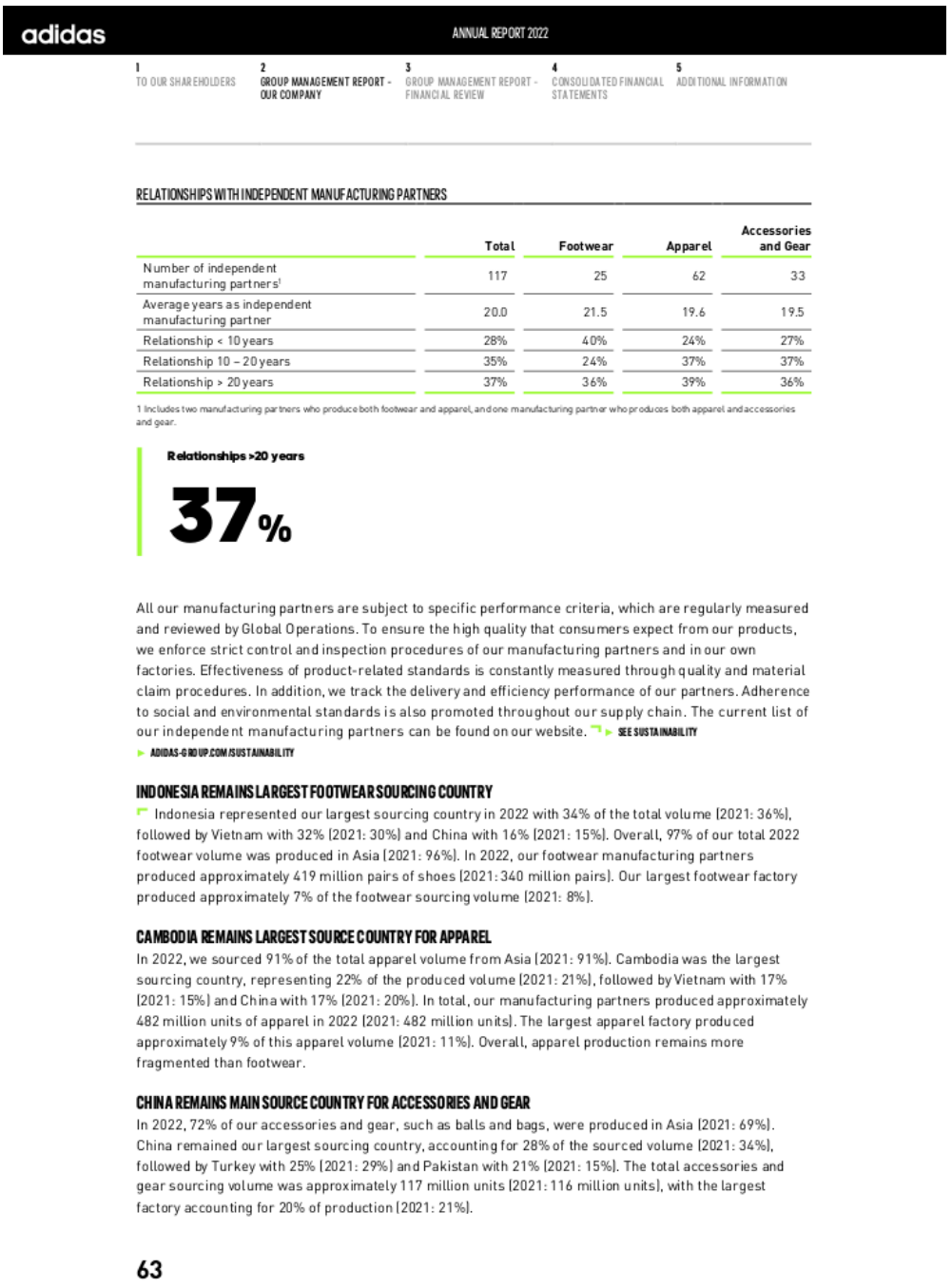
1.2.3. Exemple de formats de données non structurés

Voyons à présent des exemples de documents contenant des données non structurées à extraire, c'est-à-dire des données présentes dans des textes libres, en opposition aux données structurées qui sont organisées avec des formats précis. Voyons également sous quelles formes nous souhaitons organiser les données une fois extraites.

La source principale de données utilisée sont les rapports annuels d'entreprises. Ces rapports sont destinés aux investisseurs et contiennent de très nombreuses informations financières mais aussi des données chiffrées sur les activités, telles que le volume de pétrole extrait et vendu par une entreprise pétrolière, le nombre de chaussures vendues par une marque de vêtement ou les localisations de sites de production. Ces documents font généralement plusieurs centaines de pages, sont très complets, mais contiennent des informations redondantes et sont exprimés dans des termes, des structures et des mises en page spécifiques à chaque entreprise. Ils sont généralement en anglais, mais peuvent aussi être bilingues, une partie en anglais et une dans la langue du pays d'origine de l'entreprise.

Les rapport annuels contiennent du texte, des images et des tableaux. Lorsque l'on cherche à extraire de manière automatique des données, les tableaux sont en réalité des données non structurées. Cela est dû aux mises en pages et aux conversions en pdf des documents qui est très souvent déstructurante. Voyons à présent des exemples.

La figure suivante (fig. 1) est issue du rapport annuel 2022 d'adidas (adidas, 2023) et comporte comme information le nombre de chaussures, de vêtements et d'accessoires qu'ils produisent. Le document fait plus de trois cent pages, avec de nombreuses images, et l'information désirée est présente dans les trois derniers paragraphes de la page. On observe que les titres des paragraphes ne correspondent pas vraiment à ce qu'on cherche comme information et que de nombreuses autres informations sont présentes. L'enjeu est d'isoler les données précises que l'on cherche, ici le nombre de chaussures, de vêtements et d'accessoires.



adidas ANNUAL REPORT 2022

1 TO OUR SHAREHOLDERS | 2 GROUP MANAGEMENT REPORT - OUR COMPANY | 3 GROUP MANAGEMENT REPORT - FINANCIAL REVIEW | 4 CONSOLIDATED FINANCIAL STATEMENTS | 5 ADDITIONAL INFORMATION

RELATIONSHIPS WITH INDEPENDENT MANUFACTURING PARTNERS

	Total	Footwear	Apparel	Accessories and Gear
Number of independent manufacturing partners[1]	117	25	62	33
Average years as independent manufacturing partner	20.0	21.5	19.6	19.5
Relationship < 10 years	28%	40%	24%	27%
Relationship 10 – 20 years	35%	24%	37%	37%
Relationship > 20 years	37%	36%	39%	36%

1 Includes two manufacturing partners who produce both footwear and apparel, and one manufacturing partner who produces both apparel and accessories and gear.

Relationships >20 years

37%

All our manufacturing partners are subject to specific performance criteria, which are regularly measured and reviewed by Global Operations. To ensure the high quality that consumers expect from our products, we enforce strict control and inspection procedures of our manufacturing partners and in our own factories. Effectiveness of product-related standards is constantly measured through quality and material claim procedures. In addition, we track the delivery and efficiency performance of our partners. Adherence to social and environmental standards is also promoted throughout our supply chain. The current list of our independent manufacturing partners can be found on our website. ▸ SEE SUSTAINABILITY

▸ ADIDAS-GROUP.COM/SUSTAINABILITY

INDONESIA REMAINS LARGEST FOOTWEAR SOURCING COUNTRY

Indonesia represented our largest sourcing country in 2022 with 34% of the total volume (2021: 36%), followed by Vietnam with 32% (2021: 30%) and China with 16% (2021: 15%). Overall, 97% of our total 2022 footwear volume was produced in Asia (2021: 96%). In 2022, our footwear manufacturing partners produced approximately 419 million pairs of shoes (2021: 340 million pairs). Our largest footwear factory produced approximately 7% of the footwear sourcing volume (2021: 8%).

CAMBODIA REMAINS LARGEST SOURCE COUNTRY FOR APPAREL

In 2022, we sourced 91% of the total apparel volume from Asia (2021: 91%). Cambodia was the largest sourcing country, representing 22% of the produced volume (2021: 21%), followed by Vietnam with 17% (2021: 15%) and China with 17% (2021: 20%). In total, our manufacturing partners produced approximately 482 million units of apparel in 2022 (2021: 482 million units). The largest apparel factory produced approximately 9% of this apparel volume (2021: 11%). Overall, apparel production remains more fragmented than footwear.

CHINA REMAINS MAIN SOURCE COUNTRY FOR ACCESSORIES AND GEAR

In 2022, 72% of our accessories and gear, such as balls and bags, were produced in Asia (2021: 69%). China remained our largest sourcing country, accounting for 28% of the sourced volume (2021: 34%), followed by Turkey with 25% (2021: 29%) and Pakistan with 21% (2021: 15%). The total accessories and gear sourcing volume was approximately 117 million units (2021: 116 million units), with the largest factory accounting for 20% of production (2021: 21%).

63

Figure 1 : Extrait du rapport annuel 2022 d'adidas (adidas, 2023).

Dans la page suivante du document, des diagrammes répètent l'information, en donnant également des informations sur l'activité des années précédentes.

Regardons à présent un tableau (fig. 2) dans le rapport de Glencore PLC (Glencore, 2024), une entreprise d'extraction de matières premières, et intéressons-nous aux données concernant le charbon. Le tableau est grand, porte sur de nombreuses autres choses que le charbon, et lorsque l'on récupère le contenu de la page, la structure du tableau n'est pas récupérée : le nom des colonnes et des lignes ne sont pas toujours bien alignées. La présence d'informations sur plusieurs années complique également la tâche.

Industrial activities *continued*

Carbon intensity of Industrial activities

We show the carbon intensity of our operations as Scope 1 and 2 emissions compared to production from those operations. We have shown metals mining, coal mining, metals smelting and oil refining separately. Emissions data is collected on a site-by-site rather than activity-by-activity basis. Integrated sites with mining and smelting capability have therefore been allocated to the most appropriate category.

From 2022, we use the market-based approach to emissions recording as the primary method for our target-setting and progress measurement. The 2019 baseline has been restated to account for such. The 2019 baseline has also been restated to reflect industrial asset portfolio changes from acquisitions and divestments, most materially the acquisition of Cerrejón. 2019-21 emissions have also been restated to account for implementation of the organisational boundary of operational control and the change in emissions factors sources.

1. Includes integrated mine / smelter operations: Mount Isa, Kazzinc, INO, Murrin Murrin, Koniambo, Mopani (disposed 2021).
2. Includes integrated mine / smelter operations: Ferroalloys.
3. Converted to Cu-equivalents on the basis of 2019 average prices.
4. Also includes by-products such as platinum, palladium and rhodium.
5. Astron Energy's refining operations have been suspended since early 2020. While the refinery is being rebuilt and upgraded, Astron Energy has imported refined products for distribution in South Africa and Botswana.

Metals mining[1]

		2022	2021	2020	2019 baseline
Reported own sourced metals production					
Copper	kt	1,058.1	1,195.7	1,258.1	1,371.2
Zinc	kt	938.5	1,117.8	1,170.4	1,077.5
Cobalt	kt	43.8	31.3	27.4	46.3
Nickel	kt	107.5	102.3	110.2	120.6
Lead	kt	191.6	222.3	259.4	280.0
Gold	koz	661	809	916	886
Silver	koz	23,750	31,519	32,766	32,018
Converted to copper equivalents[3,4]	kt	2,271	2,465	2,592	2,803
Less: attributable Cu-equivalent production from JVs	kt	(496)	(530)	(503)	(474)
Add: Cu-equivalent production from Volcan	kt	156	157	120	164
Less: Cu-equivalent production of assets disposed since 2019	kt	(25)	(140)	(173)	(181)
Relevant Cu-equivalent production	**kt**	**1,907**	**1,952**	**2,036**	**2,312**
CO_2e emissions of operated assets (Scope 1)	mt	5.8	5.1	5.1	5.5
CO_2e emissions of operated assets (Scope 2)	mt	2.1	2.1	2.4	2.7
CO_2e emissions of operated assets (Scope 1 & 2)	**mt**	**7.9**	**7.2**	**7.5**	**8.2**
Carbon intensity of metals mining	t CO_2e/t Cu-equiv	4.2	3.7	3.7	3.5

Metals smelting[2]

		2022	2021	2020	2019 baseline
Reported smelter production					
Copper anode	kt	474.9	454.0	490.1	510.7
Copper cathode	kt	456.9	490.6	482.6	432.9
Lead	kt	273.4	244.9	198.0	190.5
Zinc	kt	683.0	800.6	787.2	805.7
Ferroalloys	kt	1,487.8	1,468.3	1,028.8	1,438.4
Converted to copper equivalents	kt	1,523	1,573	1,518	1,552
Add: minority interests share of operated JVs	kt	54	54	37	52
Relevant Cu-equivalent production	**kt**	**1,577**	**1,627**	**1,556**	**1,605**
CO_2e emissions of operated assets (Scope 1)	mt	5.0	4.9	3.8	5.1
CO_2e emissions of operated assets (Scope 2)	mt	8.1	8.2	6.5	8.2
CO_2e emissions of operated assets (Scope 1 & 2)	mt	**13.1**	**13.1**	**10.3**	**13.3**
Carbon intensity of metals smelting	t CO_2e/t Cu-equiv	8.3	8.0	6.6	8.3

Coal mining

		2022	2021	2020	2019 baseline
Reported coal production	mt	110.0	103.3	106.2	139.5
Add: minority interests' share of operated JVs	mt	16.8	17.9	18.7	23.0
Add: two-thirds of Cerrejón JV not previously reported	mt	–	15.6	8.3	17.2
Less: other non-operated JVs	mt	(4.3)	(5.6)	(7.5)	(8.5)
Less: coal production of assets disposed since 2019	mt	(0.1)	(1.4)	(1.9)	(2.7)
Relevant coal production	mt	**122.4**	**129.8**	**123.7**	**168.5**
Converted to copper equivalents	mt	1,407	1,492	1,421	1,936
CO_2e emissions of operated assets (Scope 1)	mt	5.8	5.9	6.1	7.6
CO_2e emissions of operated assets (Scope 2)	mt	1.1	1.2	1.2	1.2
CO_2e emissions of operated assets (Scope 1 & 2)	mt	**6.9**	**7.1**	**7.3**	**8.8**
Carbon intensity of coal mining	t CO_2e/t coal	0.056	0.054	0.059	0.052
Carbon intensity of coal mining	t CO_2e/t Cu-equiv	4.9	4.7	5.1	4.5

Oil refining

		2022	2021	2020	2019 baseline
Astron Energy - energy content of refined products	billion Btu	–	–	24,445	140,468
CO_2e emissions of Astron Energy (Scope 1)	mt	0.0	0.0	0.1	0.7
CO_2e emissions of Astron Energy (Scope 2)	mt	0.0	0.0	0.0	0.2
CO_2e emissions of Astron Energy (Scope 1 & 2)	**mt**	**0.0**	**0.0**	**0.1**	**0.9**
Carbon intensity of Astron Energy[5]	t CO_2e/ billion Btu	–	–	6.0	6.4

CO_2e emissions of operated assets (Scope 1 & 2)

		2022	2021	2020	2019 baseline
CO_2e emissions of operated assets (Scope 1 & 2)					
Metals	mt	7.9	7.2	7.5	8.2
Coal	mt	6.9	7.1	7.3	8.8
Smelters	mt	13.1	13.1	10.3	13.3
Astron Energy	mt	0.0	0.0	0.1	0.9
Add: other assets	mt	0.1	0.0	0.0	0.0
Reported CO_2e emissions (Scope 1 & 2)	mt	**28.0**	**27.4**	**25.2**	**31.2**
Change vs 2019 baseline		-10%	-12%	-19%	

Figure 2 : Extrait du rapport annuel 2023 de Glencore PLC (Glencore, 2024).

L'enjeu d'extraire de la donnée est de structurer une information en la prenant d'une source pour la rendre utilisable ailleurs de manière facile et automatique. Il ne faut donc pas seulement extraire la valeur souhaitée, mais tout un ensemble de métadonnées, et créer une structure adaptée. La structure usuellement choisie est un tableau associatif car elle donne plus de liberté qu'une structure de simple tableau et permet de faire des recherches de valeur plus efficaces.

2. Etat de l'art des LLM et leurs applications aux données ESG

2.1. Traitement automatique du langage naturel (NLP) et grand modèle de langage (LLM)

2.1.1. Débuts et évolution du domaine

Le traitement automatique du langage naturel, natural language processing ou NLP en anglais, est une discipline qui met à profit de nombreuses connaissances et techniques de linguistique, d'informatique et d'intelligence artificielle pour créer des outils de traitement du langage naturel, c'est-à-dire des langues humaines.

Les premiers travaux de recherche dans ce domaine datent des années 1950 avec comme événement fondateur la publication par Alan Turing de l'article "Computing machinery and intelligence" dans lequel il théorise une méthode pour mesurer le degré d'intelligence d'une machine, connu aujourd'hui sous le nom de test de Turing. Dans les deux décennies suivantes, les travaux en NLP étaient principalement orientés vers la traduction automatique, notamment à cause du contexte de la guerre froide car la capacité de traduire rapidement et efficacement d'une langue à une autre était d'une grande importance stratégique (Ghofrane, 2021).

Ces années-là voient également l'élaboration de théories linguistiques permettant de mieux comprendre la grammaire et la construction des langages. L'un des contributeurs majeurs dans ce domaine est Noam Chomsky avec ses travaux sur la grammaire générative dans lesquels sont introduits de nouveaux concepts sur les structures des phrases. Ces publications ont eu une grande influence sur la linguistique théorique et les axes de recherche en NLP, permettant de créer des approches basées sur des mécanismes de règles qui retranscrivent des règles syntaxiques et grammaticales conçues par des linguistes de ce courant de pensée (Wikipedia, 2024 a).

Dans les années 1980 et 1990, l'avènement de l'ordinateur personnel et les progrès en puissance calculatoire des ordinateurs ont permis des avancées significatives dans le NLP. Les modèles basés sur des règles, bien qu'efficaces pour certaines tâches, se révélaient limités face à la complexité et à la variabilité des langages naturels. Des approches statistiques sont alors adoptées en utilisant de larges corpus de textes pour entraîner des algorithmes capables de reconnaître des patterns et d'inférer des significations.

L'essor de l'apprentissage automatique (machine learning en anglais), et encore plus récemment de l'apprentissage profond (deep learning en anglais) ont permis une nouvelle révolution dans le NLP en décuplant les performances et les applications réelles.

Nous allons illustrer la différence entre les méthodes d'apprentissage automatique et les méthodes d'apprentissage profond en présentant le TF-IDF. Le TF-IDF, de l'anglais term frequency-inverse document frequency et qui se traduit par fréquence de terme-fréquence de document inverse, consiste à évaluer l'importance d'un terme contenu dans un document au sein d'un corpus en effectuant une mesure statistique (Mei, 2019). Le terme analysé est généralement un mot, mais peut aussi être un ensemble de mots, ou même une partie d'un mot. Le poids du terme augmente proportionnellement à son nombre d'occurrences dans le

document et diminue proportionnellement au nombre d'autres documents du corpus dans lequel il est présent. Cela retranscrit la force de signification qu'a un mot : si il est présent dans de nombreux documents alors il est courant et n'a pas un sens fort, alors qu'au contraire si il est présent que dans un seul document alors il a un sens fort. La première formule de ce type a été mise au point en 1972 par Karen Spärck Jones, puis des variantes en ont découlées, permettant d'avoir des résultats de meilleure qualité dans certains cas. Voilà là formule de Karen Spärck Jones pour mieux comprendre la logique derrière tf-idf et les méthodes d'apprentissage automatique.

$$TF(t,\ d)\ =\ \frac{Nombre\ de\ fois\ que\ le\ terme\ t\ apparaît\ dans\ le\ document\ d}{Nombre\ total\ de\ termes\ dans\ le\ document\ d}$$

$$IDF(t,\ D)\ =\ \log(\frac{Nombre\ total\ de\ documents}{Nombre\ de\ documents\ où\ le\ terme\ t\ apparaît})$$

$$TF\ -\ IDF(t,\ d,\ D)\ =\ TF(t,\ d)\ \times\ IDF(t,\ D)$$

A l'opposé, les grands modèles de langage, ou LLM pour large language models en anglais, sont basés sur des méthodes d'apprentissage profonds bien plus complexes dont nous ne détaillerons peu le fonctionnement. Leur apprentissage s'effectue sur de très grands volumes de données en apprenant des relations statistiques entre unités textuelles, appelés tokens. La différence principale avec les méthodes statistiques classiques est qu'une forme de compréhension sémantique et contextuelle en découle, ce qui permet une compréhension de phrases et une génération de texte.

2.1.2. Applications générales du NLP

Un très grand nombre d'applications sont possibles au NLP, et ce de natures très différentes.

Tout d'abord, le NLP permet de traiter des photographies de textes et des enregistrements audios. Cela correspond tout d'abord à la reconnaissance d'écritures manuscrites et de caractères dactylographiés, appelé optical character recognition ou OCR, et permettant de les transcrire et ensuite de les interpréter si souhaité. Les OCR sont par exemple utilisés pour noter les copies d'examens, pour scanner les plaques d'immatriculations de voitures mal stationnées ou en excès de vitesse, pour la lecture des passeports aux douanes ou encore pour la vérification de documents administratifs par les banques. Le traitement et la reconnaissance d'enregistrements audios de paroles permet de retranscrire des paroles en texte (Marshall, 2020).

Le NLP a des capacités sémantiques, c'est-à-dire de comprendre le sens d'un énoncé, et des capacités de génération de texte. Cela est appliqué pour la traduction automatique depuis les premiers modèles de NLP, et les performances se sont immensément améliorées en un peu plus de 70 ans. La compréhension sémantique permet la correction orthographique, application dont nous profitons au quotidien sur nos téléphones et ordinateurs. Le résumé et les systèmes conversationnels sont quant à eux des combinaisons de compréhension sémantique et de génération de texte syntaxiquement et sémantiquement corrects.

Enfin, le NLP permet l'extraction d'informations. Cela correspond entre autres à la thématique à laquelle nous nous intéressons. Plus largement, cela est utilisé pour la

recherche d'information, dans les moteurs de recherche par exemple ou encore pour l'analyse de sentiment, chose employée pour des algorithmes de recommandations présents dans (Searle, 2024). Voyons en détail quels sont les intérêts des LLM pour l'extraction de données non structurées.

2.1.3. Intérêt des LLM pour l'extraction de données non structurées

Les LLM présentent un intérêt majeur pour l'extraction de données non structurées grâce à leurs capacités de compréhension de texte, du contexte et des nuances du langage humain. Ils sont entraînés sur de vastes ensembles de données ce qui leur permet d'être compétent sur un très grand nombre de domaines, étant ainsi utilisables pour de multiples choses sans avoir à les entraîner. Leur utilisation pour l'extraction de données extra financières depuis des rapports d'entreprises à des fins d'évaluation environnementale est donc possible et adaptée.

L'utilisation de LLM pour de l'extraction de données est en soit basique, il suffit de fournir un texte et de poser une question pour obtenir une réponse. Atteindre un objectif précis, des performances élevées et à grande échelle est cependant plus complexe. Cela relève de la mise en place de pipelines complets.

Il s'agit donc d'intégrer les LLM dans des pipelines de données parfois déjà existants en permettant d'extraire les informations souhaitées dans des données textuelles non structurées et de les transformer en données structurées exploitables dans d'autres algorithmes, et ce de manière automatique. La mise en place de tels systèmes permet de réduire considérablement la charge de travail manuel et d'automatiser des tâches.

On vise donc d'avoir un pipeline du type :

- Ingestion de données à partir de sources variées comme les fichiers texte ou les flux de réseaux sociaux
- Ensuite, un LLM est utilisé en lui donnant une question et le document dans lequel chercher. Une réponse est retournée sous forme structurée ou non au choix.
- Une phase de post processing a lieu pour valider les données et les organiser dans la forme souhaitée
- Les données sont enfin stockées dans des bases de données relationnelles ou NoSQL, prêtes à être intégrées dans d'autres algorithmes pour des tâches diverses et variées.

Il est ensuite possible d'ajouter différentes étapes dans ce pipeline afin de gagner en performance, que ce soit du temps d'exécution ou de précision dans les résultats.

En résumé, le NLP a évolué des premières approches basées sur des règles à des modèles d'apprentissage profond tels que les LLM qui permettent une compréhension de nos langages humains. Ces progrès ont transformé le NLP et rendent à présent possible des applications toujours plus nombreuses allant de la traduction automatique à l'extraction d'informations.

Dans la suite nous aborderons ce que sont les embeddings, des représentations numériques fondamentales pour le traitement du texte par les LLM.

2.2. Embeddings et similarité sémantiques

2.2.1. Qu'est-ce qu'un embedding ?

Un embedding, ou prolongement sémantique en français, est une représentation numérique et mathématique d'objets du monde réel, que ce soit du texte, une image ou encore un son, et est utilisé par des systèmes d'intelligences artificielles comme source d'information. Pour apprendre, un humain lit un texte, alors qu'un LLM ingère un embedding. Les embeddings sont des vecteurs dans un espace vectoriel de dimension finie. Les embeddings capturent les relations sémantiques entre les éléments en les projetant dans un espace vectoriel où les éléments similaires sont proches les uns des autres.

Pour les LLM, les embeddings transforment des groupes de lettres, des mots ou des phrases en vecteurs de nombres réels de dimensions fixes. Les modèles Word2Vec, GloVe, et BERT sont des modèles de création d'embeddings de mots qui prennent compte des contextes d'utilisation de ces derniers, capturant ainsi les significations et relations contextuelles. L'analyse complexe de textes devient possible car la sémantique est capturée.

Par exemple, le mot "plus" en français peut à la fois être un comparatif de supériorité ou une négation. Dans le cas où la sémantique n'est pas prise en compte, ils seraient mis sous la même forme vectorielle, mais en prenant compte du contexte d'utilisation, l'embedding associé n'est pas le même, et même très éloigné.

Les embeddings permettent aux LLM d'effectuer des tâches complexes car ils permettent aux algorithmes de manipuler des mots et des phrases sous une forme numérique plus compréhensible et exploitable par les machines, mais aussi de capturer les sens grâce à l'analyse des contextes.

Voilà par exemple une représentation des mots "king", "queen", "man" et "woman" dans un espace de dimension 3 (fig. 3). Les embeddings sont donc de taille 3 (Jumelle, 2023).

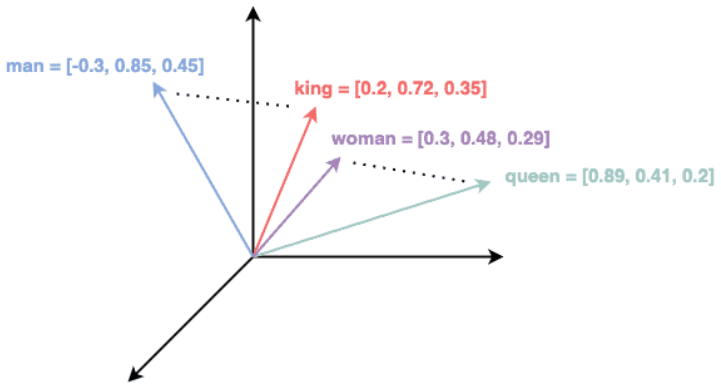


Figure 3 : Représentation des mots en exemple dans un espace (Jumelle, 2023).

2.2.2. Similarité et similarity search

Les propriétés des espaces vectoriels permettent de calculer les distances entre vecteurs. Etant donné que les embedding sont formés en prenant en compte le contexte et le sens

des mots, les distances entre les vecteurs représentent les proximités sémantiques. Deux mots de sens proche auront une faible distance dans l'espace vectoriel de représentation, alors que deux mots de sens très différents seront éloignés dans cet espace vectoriel.

Pour chercher une information dans un texte, on peut utiliser un modèle d'embedding pour représenter dans un même espace sémantique la question que l'on pose et les textes dans lesquels on cherche. En mesurant la distance entre l'embedding d'une question et chacun des embeddings des textes, on trouve la similarité de la question avec chacun d'eux, et donc quels textes sont les proches de la question. Cela donne ainsi une recherche par similarité. Nous utiliserons le terme anglais de similarity search pour la suite car il n'y a pas de traduction en français qui fasse l'unanimité dans le milieu.

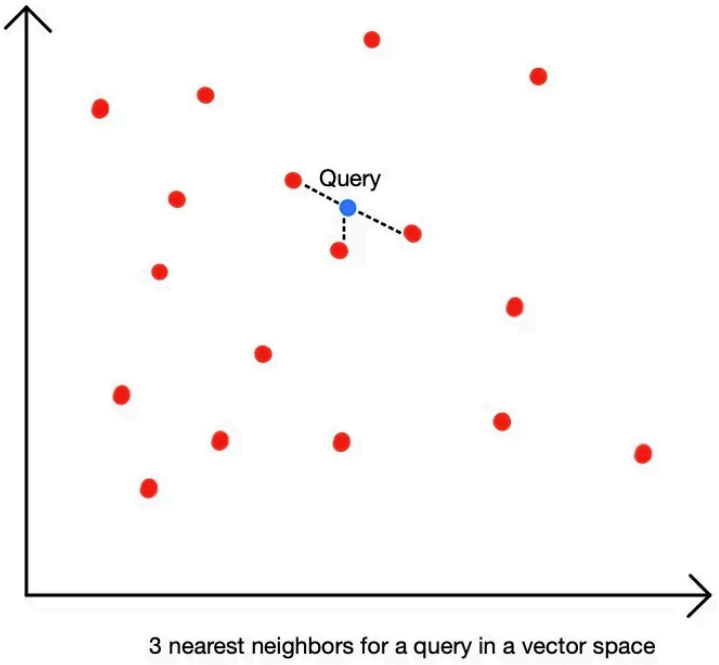


Figure 4 : Représentation graphique d'une similarity search (Tripathi, 2023).

La similarity search présente un grand avantage par rapport aux algorithmes et méthodes de recherche traditionnels. Les méthodes traditionnelles ne font que regarder quels mots sont présents dans les documents et analysent leurs occurrences statistiques sans capturer le sens des mots et des phrases. Avec l'exemple du mot "plus" cité plus haut, les méthodes traditionnelles ne peuvent pas faire la différence entre les deux significations qu'il peut avoir et donneraient la même "similarité" à deux documents ayant ce mot là mais de sens différents.

Le calcul de similarité entre des embeddings peut être effectué en utilisant différentes métriques, parmi lesquelles la similarité cosinus est la plus couramment utilisée (Mansurova, 2024).

La similarité cosinus, cosine similarity en anglais, mesure l'angle entre deux vecteurs dans un espace vectoriel. Elle est définie comme le produit scalaire des vecteurs normalisés :

$$Similarité\ cosinus(A,\ B)\ =\ \frac{A\cdot B}{|A|\ |B|}\ =\ \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^{\ 2}} \times \sqrt{\sum_{i=1}^{n} B_i^{\ 2}}}\ =\ \cos(\theta)$$

où A et B sont des embeddings. Cette mesure varie entre -1 et 1, où 1 signifie que les vecteurs sont identiques (les embeddings sont identiques), 0 qu'ils sont orthogonaux (aucune similitude entre les embeddings), et -1 qu'ils sont de sens opposés.

La cosine similarity est particulièrement utile dans le NLP pour comparer des mots, des phrases ou des documents, car elle prend en compte l'orientation des vecteurs plutôt que leur longueur, ce qui est crucial pour capturer la similarité sémantique.

La distance Euclidienne, également appelée norme 2, est la racine carrée de la somme des différences aux carrées des coordonnées de deux vecteurs. Elle va donc de 0 à l'infini, avec 0 pour deux vecteurs identiques et croît avec l'éloignement des deux vecteurs.

$$Distance\ euclidienne\ (A,\ B)\ =\ \sqrt{\sum_{i=1}^{n} (A_i\ -\ B_i)^2}$$

La distance de Manhattan, également appelée norme 1, est la somme des différences absolues des coordonnées de deux vecteurs. Tout comme la distance euclidienne, elle va de 0 à l'infini et croît avec l'éloignement des vecteurs.

$$Distance\ de\ Manhattan\ (A,\ B)\ =\ \sum_{i=1}^{n} |A_i\ -\ B_i|$$

2.2.3. Le choix de la taille des embeddings

La taille de l'embedding est la dimension de l'espace vectoriel dans lequel il est défini. C'est une valeur fixée avant la création de l'embedding. Cette dimension représente le nombre total de features qui sont encodés dans la représentation vectorielle du mot.

Comme déjà évoqué, il existe différentes méthodes et services permettant de générer des embeddings, la plupart ont des dimensions différentes, allant de 300 à plus de 8000 (mteb, 2024).

Il est aisé de penser que plus la taille d'un embedding est grande, mieux c'est. Effectivement, si la dimension de l'espace sémantique est trop petite, les vecteurs seraient tous très proches et les similarités calculées seraient trop peu précises et fines. Des mots différents se retrouveraient avec des coordonnées très proches. A l'opposé, un espace trop grand conduit à une chute de la densité des points, rendant la détection de similarités significatives entre les vecteurs difficiles. Cela conduit également à une forme de sur-apprentissage, le modèle capturant des détails superflus sur les données d'apprentissage et ne performant alors pas bien sur les données nouvelles (Aibin, 2024).

En situation réelle, les modèles ne sont jamais trop grands car il est recherché de limiter la taille des embeddings pour accélérer la vitesse calculatoire. Des vecteurs de plusieurs

milliers ou dizaines de milliers de dimensions sont beaucoup plus complexes et longs à manipuler que des vecteurs de plus petite taille.

La taille des embeddings créés et utilisés par les algorithmes d'OpenAI ou d'autres entreprises a considérablement augmenté au cours des dernières années, passant d'environ 300 à plus de 3000 en moins de 4 ans. Cela a été possible car les modèles de LLM ont gagné en efficacité dans leur exécution et une plus grande taille permet une meilleure précision. L'enjeu de la taille des embeddings est cependant toujours de taille.

2.2.4. Créer des embeddings avec différents services

Un moyen simple de créer des embeddings de mots ou phrases souhaitées est d'utiliser une API, notamment celle d'OpenAI, de Google ou de Mistral. Ce service payant reste très abordable.

Les prix se comptent par token. Un token est l'unité textuelle utilisée par les LLM. Ils sont issus d'un découpage du texte en morceaux de différentes tailles, souvent correspondant à un mot mais pas nécessairement. Les tokens peuvent contenir des espaces si le découpage en contient. La taille des textes que prennent en entrée ou retournent en sortie les LLM est donc comptée en tokens pour déterminer le prix du service.

Le modèle le moins cher d'OpenAI, appelé text-embedding-3-small, coûte 0.02 dollars pour 1 million de tokens, soit environ 750 000 de mots, 1500 pages, ou presque l'intégralité du Seigneur des Anneaux (Weaviate, n.d.). Pour le même volume de texte, leur modèle le plus cher coûte 0,130 dollars, soit 6 fois plus, ce qui reste encore très peu cher (OpenAI, n.d.). Le service proposé par Mistral est également dans le même ordre de grandeur.

Les prix de ces services ont dès le début été très raisonnables et ont même fortement baissé avec le temps, étant à présent plus de 30 fois moins cher qu'il y a un an.

Grâce aux librairies d'implémentation de solutions LLM comme LangChain, il est très aisé de changer de fournisseur et de méthode d'embedding. Nous présenterons un exemple juste après pour illustrer cela. Il est également possible de créer soit même ses embeddings, sans faire appel à un service payant, en utilisant des modèles open source. Nombre d'entre eux existent sur la plateforme Hugging Face. Cette plateforme comporte également un classement des modèles existants, mesurant leurs performances respectives pour différentes tâches (mteb, 2024).

Montrons a présent un exemple :

```python
from openai import OpenAI
import pandas as pd
from api_key import API_KEY
from typing import List
```
Executed at 2024.07.24 19:10:18 in 47ms

```python
client = OpenAI(max_retries=5, api_key=API_KEY)
def get_embedding(text: str, model="text-embedding-3-small") -> List[float]:
    return client.embeddings.create(input=[text], model=model).data[0].embedding
```
Executed at 2024.07.24 19:10:18 in 21ms

```python
words = [
    "king", "queen", "male", "female", "apple"
]

embeddings = [{"word": word, "embedding": get_embedding(word)} for word in words]
```
Executed at 2024.07.24 19:10:20 in 1s 495ms

```python
df = pd.DataFrame(embeddings)
df.head()
```
Executed at 2024.07.24 19:10:20 in 11ms

5 rows × 2 columns pd.DataFrame

	word	embedding
0	king	[0.037228088825941086, -0.022083600983023643, 0.05191672593355…
1	queen	[0.043808333575725555, -0.03978610783815384, 0.044801775366067…
2	male	[0.0762191042304039, -0.0034149582497775555, 0.014211745001375…
3	female	[0.06657705456018448, -0.021973706781864166, -0.00098645954858…
4	apple	[0.017625167965888977, -0.016837788745760918, -0.0418885424733…

Figure 5 : Création d'embeddings à l'aide de l'API OpenAI.

Ce notebook (fig. 5) utilise l'API d'OpenAI pour générer des embeddings pour une liste de mots donnée. La liste est très simple et courte et a seulement vocation à montrer la facilité qu'il y a pour créer des embeddings. Il est effectué :

- L'import des modules nécessaires
- L'initialisation de la clé client OpenIA. Cette clé n'est pas affichée ici car c'est la clé d'authentification du client, à laquelle est relié un moyen de paiement.
- La définition de la méthode get_embedding : elle prend en argument un texte et un modèle d'embedding qui sera celui utilisé, et elle retourne l'embedding du mot. Ici "text-embedding-3-small" est à ce jour le modèle le moins cher d'OpenIA
- La liste de mots à vectoriser est ensuite donnée et les mots sont passés au service. Le formatage du résultat permet d'avoir côte à côte le mot initial et son vecteur.

Avec quelques lignes de code en plus, on peut calculer la similarité cosinus des embeddings entre eux et faire une matrice de similarité :

```python
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

embeddings_matrix = np.array(df['embedding'].tolist())
similarity_matrix = cosine_similarity(embeddings_matrix)

similarity_df = pd.DataFrame(similarity_matrix, index=df['word'], columns=df['word'])

plt.figure(figsize=(6, 4))
sns.heatmap(similarity_df, annot=True, fmt=".2f", cmap='coolwarm', linewidths=0.5)
plt.title('Cosine Similarity Heatmap')
plt.show()
```

Executed at 2024.07.24 19:25:05 in 69ms

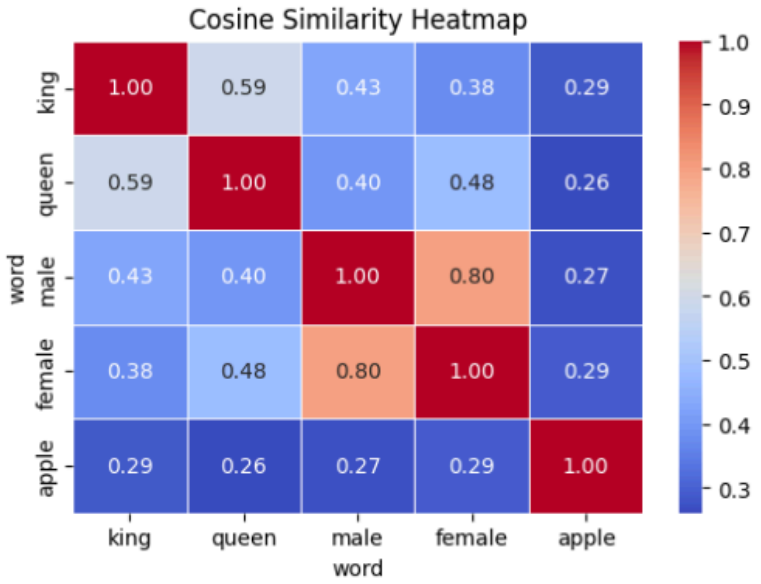


Figure 6 : Représentation de la similarité entre différents embeddings de mots.

On voit dans la figure 6 qu'effectivement "king" est plus proche de "queen" que de "apple", et est plus proche de "male" que de "female". Les sens des mots sont retranscrits dans ces distances.

Ces exemples sont sur de simples mots, mais la même chose peut être réalisée sur des phrases ou des documents entiers. En voilà un exemple avec les phrases "I am happy", "I am very happy", "There is a cat in the room", "There is a dog in the room", "To be or not to be".

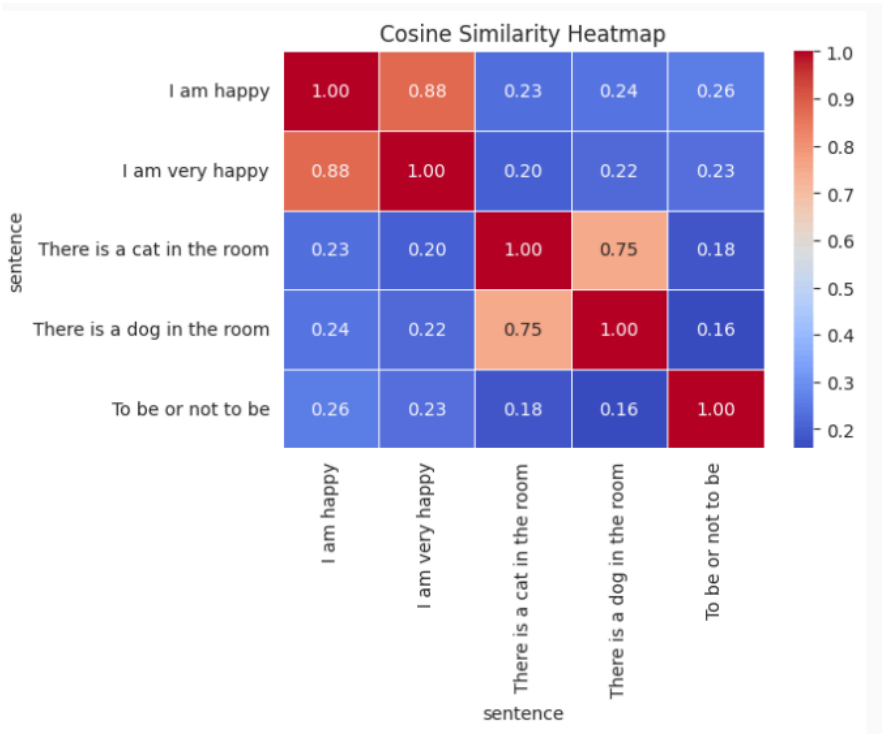


Figure 7 : Représentation de la similarité entre différents embeddings de phrases.

On voit dans la figure 7 que les phrases “I am happy” et “I am very happy” sont les deux plus proches, ce qui fait sens car leur signification est très similaire. Les phrases “There is a cat in the room” et “There is a dog in the room” sont également proches, ce qui fait sens car la seule différence est l’animal dont il est question. Ensuite, “To be or not to be” est plus proche des phrases comportant “I am…” que des phrases “There is a…”, ce qui peut être expliqué par la notion d’être évoqué par le “I” et qui n’est pas présent dans les autres phrases.

2.2.5. Vector store : présentation du concept

Comment créer de manière efficace les embeddings de documents entiers ?

La première difficulté est de créer des embeddings des morceaux de texte de tailles optimales. On appelle ce découpage un chunk. Un chunk trop grand a l'inconvénient de contenir des informations très variées et de ne pas capturer précisément le sens du texte. Prendre des chunks plus petits permet d'être plus précis car le texte correspondra à une portion plus cohérente et thématiquement homogène de texte. Mais prendre des chunks trop petits présente aussi des désavantages : la donnees trop granulaire perd de la richesse et du contexte, contexte qui peut devenir insuffisant pour permettre une interprétation significative.

La taille des chunks est donc un paramètre important dans l’utilisation des LLM. Il n’y a pas de méthode universelle pour déterminer la taille optimale, elle se détermine généralement de manière empirique pour chaque type d’utilisation. Une taille communément acceptée est entre 250 et 500 mots par chunk. Selon le profil des données utilisées ou l’application développée, il peut alors être judicieux d’augmenter cette taille ou de la diminuer.

Les vecteurs store sont des bases de données spécialement conçues pour stocker et gérer les embeddings de manière efficace. Ces vector stores permettent également de gérer facilement le découpage en chunk des documents, ou encore d'effectuer des similarity search dans la base de manière simple et optimisée.

De nombreux services de vectors store existent. Certains ont été conçus par des acteurs proposant déjà d'autres types de bases de données, comme ElasticSearch, Redis, Mongodb ou Postgres ; alors que d'autres proviennent de nouveaux acteurs spécialisés dans ce domaine, comme Weaviate, Croma ou Pinecone.

Voilà un exemple de création de vector store Croma ou est ingéré le rapport annuel d'adidas (fig. 8). Le document est récupéré dans l'espace de stockage de l'ordinateur, découpé en chunks, puis chaque l'embedding de chaque chunk est créé (LangChain, 2023 a).

```python
import os
from api_key import OPENAI_API_KEY
from langchain_openai import ChatOpenAI

os.environ["OPENAI_API_KEY"] = OPENAI_API_KEY
llm = ChatOpenAI(model="gpt-4o-mini")
```

Executed at 2024.08.14 15:03:53 in 1s 570ms

```python
from langchain_chroma import Chroma
from langchain_openai import OpenAIEmbeddings
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain_community.document_loaders import PyPDFLoader

# Load report
loader = PyPDFLoader(
    "files/annual-report-adidas-ar22.pdf",
)
docs = loader.load()

# Split the documents in chunks, create their embeddings and store them in a vector database
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
splits = text_splitter.split_documents(docs)
vectorstore = Chroma.from_documents(documents=splits, embedding=OpenAIEmbeddings())
```

Executed at 2024.08.14 15:05:01 in 1m 7s 843ms

Figure 8 : Création d'un vector store et indexation d'un document.

2.3. Comment utiliser des LLM

2.3.1. LLM et embeddings.

Les LLM sont des modèles de traitement du langage naturel NLP de grande envergure, entraînés sur d'énormes ensembles de données textuelles. Ces modèles sont conçus pour comprendre et générer du texte humain avec un haut degré de cohérence et de pertinence. Ils sont basés sur des architectures de deep learning appelées Transformers, dont nous ne rentrerons pas dans le détail de fonctionnement. Ces modèles sont caractérisés par leur grande taille, mesurée en nombre de paramètres, allant de centaines de millions à des centaines de milliards.

Grâce à leur taille, les LLM peuvent capturer une vaste gamme de connaissances et de nuances linguistiques, leur permettant de comprendre des contextes complexes et de générer du texte cohérent.

Les modèles les plus connus sont GPT (pour Generative Pre-trained Transformer) de OpenAI, BERT (Bidirectional Encoder Representations from Transformers) de Google, LLaMA de Meta et Mistral Laster de Mistral AI.

Comme vu précédemment, les LLM sont utilisés dans une multitude d'applications, incluant la génération de texte, la traduction automatique, les chatbots ou la recherche d'informations.

Tout comme pour la création d'embeddings, les entreprises créant ces modèles les mettent à disposition à travers l'utilisation d'une d'API et/ou d'une librairie. En simplifiant, pour les utiliser il suffit d'envoyer du texte à l'API, que ce soit une question avec ou sans un contexte (un contexte est un texte dans lequel le modèle peut puiser des informations additionnelles à sa base de connaissance) et une réponse est retournée. La taille des textes envoyés et retournés est comptés en tokens.

Les modèles sont pour la plupart payant à utiliser, coûtant pour le modèle gpt-4o d'OpenAI 5 dollars pour 1 million de tokens envoyés et 15 dollars pour 1 million de tokens retournés. Par exemple, si on demande de résumer la Déclaration des Droits de l'Homme et du Citoyen, cela coûtera environ 0,006 centimes pour la question et 0,005 centimes pour une réponse de 200 mots (OpenAI, n.d.).

2.3.2. LangChain Expression Language

LangChain Expression Language (LCEL) est un langage de requête conçu par la librairie LangChain permettant d'interagir de manière efficace avec les LLM, facilitant l'extraction et la manipulation de données textuelles. LCEL permet aux utilisateurs de formuler des requêtes complexes de manière simple et expressive, en tirant parti des capacités des LLM pour interpréter et traiter le langage naturel.

LCEL permet non seulement d'effectuer des requêtes aux API des modèles LLM mais également de créer des pipeline de traitement de données pour structurer les réponses des LLM. Des réponses structurées comme souhaitées sont alors intégrables dans des workflows déjà existants (LangChain, n.d. a).

Les outils de structuration de ce qui est en sortie des LLM s'appellent des OutputParsers. Il en existe de nombreux types et accomplissant différentes tâches de transformation des données : en un objet de type Python *string*, en format JSON ou CSV, ou encore de typer des données en des types personnalisés ce qui permet de mieux maîtriser les objets manipulés par la suite.

Voilà à présent un exemple simple d'utilisation de l'API d'OpenAI pour une tâche de résumer, ici de la Déclaration des Droits de l'Homme et du Citoyen (fig. 9). Le résultat complet obtenu de l'API est affiché et contient le texte mais aussi des informations additionnelles telles que la taille en token du texte en entrée et la taille du texte en sortie.

C’est sur cet objet de sortie que l’on peut utiliser des OutputParser, ce qui sera fait juste après.

In 2
```python
import os
from api_key import OPENAI_API_KEY
from langchain_openai import ChatOpenAI

os.environ["OPENAI_API_KEY"] = OPENAI_API_KEY
model = ChatOpenAI(model="gpt-4o")
```
Executed at 2024.07.28 13:16:09 in 41ms

In 3
```python
from langchain_core.messages import HumanMessage, SystemMessage

with open('DDHC_1789.txt') as fp:
    ddhc = fp.read()

messages = [
    SystemMessage(content="Fait un résumé de ce texte."),
    HumanMessage(content=ddhc),
]

model.invoke(messages)
```
Executed at 2024.07.28 13:16:17 in 5s 345ms

Out 3
```
nationale française, proclame les droits naturels, inaliénables et sacrés de l'homme. Elle vise à rappeler
constamment ces droits et devoirs aux membres de la société pour garantir le respect des pouvoirs législatif
 et exécutif et assurer le bonheur général.\n\nPrincipaux articles :\n\n1. Les hommes naissent libres et
égaux en droits; les distinctions sociales ne sont justifiées que par l'utilité commune.\n2. Les droits
naturels et imprescriptibles de l'homme sont la liberté, la propriété, la sûreté et la résistance à
l'oppression.\n3. La souveraineté réside essentiellement dans la nation.\n4. La liberté consiste à faire
tout ce qui ne nuit pas à autrui, avec des limites définies par la loi.\n5. La loi ne peut interdire que les
 actions nuisibles à la société.\n6. Tous les citoyens participent à la formation de la loi, qui doit être
la même pour tous.\n7. Nul ne peut être arrêté ou détenu sans une procédure légale.\n8. Les peines doivent
être strictement nécessaires et basées sur des lois établies antérieurement.\n9. Tout homme est présumé
innocent jusqu'à preuve du contraire.\n10. La liberté d'opinion, y compris religieuse, est protégée.\n11. La
 libre communication des pensées et des opinions est un droit précieux.\n12. Une force publique est
nécessaire pour garantir les droits de l'homme.\n13. Une contribution commune est nécessaire pour
l'entretien de la force publique et des dépenses administratives.\n14. Les citoyens peuvent contrôler
l'emploi de la contribution publique.\n15. La société a le droit de demander des comptes à tout agent
public.\n16. Une société sans garantie des droits et séparation des pouvoirs n'a pas de Constitution.\n17.
```

Figure 9 : Exemple d’utilisation d’un LLM à l’aide de LangChain.

LangChain fonctionne par composants qu’il est possible de chaîner, ce qui permet de rendre le code plus compréhensible et surtout réutilisable. Chaque étape d’une chaîne a à sa disposition les données d’entrée et de sortie du chaînon précédent.

Voilà un exemple de code qui effectue le parsing de la réponse du LLM en plus de son appel, toujours dans le cadre du résumé de la Déclaration des Droits de l’Homme et du Citoyen (fig. 10). L’OutputParser utilisé là est *StrOutputParser*, un parser disponible dans la librairie LangChain, qui convertit seulement le résultat sorti du LLM sans métadonnées et en type *string*. D’autres types d’OutputParser seront utilisés dans des exemples suivants afin de montrer la force de ces outils.

In 6

```python
from langchain_core.output_parsers import StrOutputParser
from langchain_core.prompts import ChatPromptTemplate

system_template = "Fait un résumé de {length} mots"

prompt_template = ChatPromptTemplate.from_messages(
    [("system", system_template), ("user", "{text}")]
)

parser = StrOutputParser()

chain = prompt_template | model | parser
chain.invoke({"length": "200", "text": ddhc})


```

Executed at 2024.07.28 13:25:57 in 4s 679ms

Out 6

```
"La Déclaration des Droits de l'Homme et du Citoyen de 1789, rédigée par les représentants du peuple français
en Assemblée nationale, vise à rappeler et établir les droits naturels, inaliénables et sacrés de l'homme.
Cette déclaration, conçue pour être constamment présente à l'esprit des membres de la société, a pour but de
garantir que les actes des pouvoirs législatif et exécutif respectent les principes fondamentaux de toute
institution politique. Elle énonce que les hommes naissent libres et égaux en droits, et que les distinctions
sociales ne se justifient que par l'utilité commune. Les droits naturels et imprescriptibles incluent la
liberté, la propriété, la sûreté, et la résistance à l'oppression. La souveraineté réside dans la nation, et
toute autorité doit en émaner. La liberté est définie comme la possibilité de faire tout ce qui ne nuit pas à
autrui, et la loi ne doit restreindre que les actions nuisibles à la société. La loi est l'expression de la
volonté générale, et tous les citoyens ont le droit de participer à son élaboration. La Déclaration souligne
également l'importance de la présomption d'innocence, la liberté d'opinion et d'expression, et la nécessité
d'une force publique pour garantir les droits de l'homme. Enfin, elle affirme le droit de propriété
inviolable, sauf en cas de nécessité publique avec une juste indemnité."
```

Figure 10 : Exemple d'utilisation d'un LLM et de LCEL.

La premier chaînon correspond au prompt qui est passé, le second chaînon au modèle employé, qui a été défini plus haut, et enfin le troisième chaînon définit le type de parsing souhaité, ici ne récupérer que la réponse et pas les métadonnées.

2.4. La méthode RAG pour l'extraction d'information

2.4.1. Principe de fonctionnement et architecture

L'une des applications les plus puissantes permises par les LLM est celle des chatbots sophistiqués de réponse aux questions, c'est-à-dire de répondre à des questions sur des informations spécifiques. Notre cas étudié d'extraction de données est une forme de question-réponse, où un texte dont il faut se servir est donné au LLM et une question précise est posée. Ces applications LLM de question-réponse utilisent une technique connue sous le nom de Retrieval Augmented Generation (RAG). Cette technique a été introduite pour la première fois dans la publication "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" par Patrick Lewis et al. (Lewis et al., 2020).

La RAG est une technique permettant d'augmenter les connaissances des LLM avec des données supplémentaires. Les LLM peuvent raisonner sur des sujets très variés mais leurs connaissances sont limitées aux données publiques sur lesquelles ils ont été formés, qui s'arrêtent alors à un moment passé plus ou moins lointain. Par exemple, les connaissances du modèle le plus avancé d'OpenAi, gtp-4o a des connaissances qui s'arrêtent à décembre 2023. Il devient donc essentiel de pouvoir créer des applications d'IA capables de raisonner

sur des données privées ou des données introduites après la date de fin d'entraînement du modèle. Il faut l'enrichir avec des connaissances en intégrant des informations spécifiques dont il a besoin pour accomplir la tâche qu'on attend de lui (Mansurova, 2024).

Un LLM est limité par le nombre maximal de tokens qu'il peut traiter en une seule fois, appelé taille du contexte. La taille du contexte est généralement autour d'une centaine de milliers de tokens pour les modèles récents. Cela signifie que pour des contextes et questions complexes, il n'est pas possible de fournir directement au modèle toutes les informations disponibles et qu'on souhaiterait lui donner. Cela est encore plus vrai dans le cadre de l'extraction et de la recherche d'informations où les informations cherchées peuvent se trouver dans de très grands documents. Pour utiliser la technique RAG, il faut alors sélectionner les informations les plus pertinentes de la base de données ou de documents que l'on possède pour que la contrainte de taille de contexte soit respectée.

Par ailleurs, comme pour un humain, donner un contexte plus petit à un modèle lui permet de mieux se concentrer sur l'essentiel et de ne pas se perdre. Il faut alors s'assurer que le modèle reçoit les éléments essentiels pour maximiser ainsi la qualité des réponses. La RAG répond en plus à ce critère car on peut sélectionner ce que l'on donne comme informations au modèle.

Une application RAG typique comporte deux composantes principales :

- L'indexation : c'est un pipeline pour ingérer les données d'une source et les indexer. Cela signifie le découpage des données en chunks et leur mise sous forme d'embeddings dans un vector store. Le vector store permettra de faire des recherches sémantiques et de donner un accès rapide aux données dont le LLM aura besoin.

- Récupération et génération de la réponse : c'est le cœur de la RAG proprement dite. Tout d'abord, la phase de récupération des documents prend une requête de l'utilisateur au moment de l'exécution et récupère les données les plus pertinentes du vector store grâce à la similarity search. Ensuite, ces données pertinentes sont transmises au modèle LLM auquel est posée en plus une question. C'est lui qui génère la réponse (LangChain, n.d. b).

C'est donc là un des grands intérêt de la mise sous forme d'embeddings de documents et l'utilisation de vector store : il est possible de rechercher les morceaux de documents les plus utiles et susceptibles de contenir les informations nécessaires à la formulation d'une bonne réponse par un modèle LLM.

LangChain et son LangChain Expression Language disposent de composants conçus pour faciliter la création d'applications utilisant la technique de RAG.

2.4.2. Schéma illustrant la méthode RAG

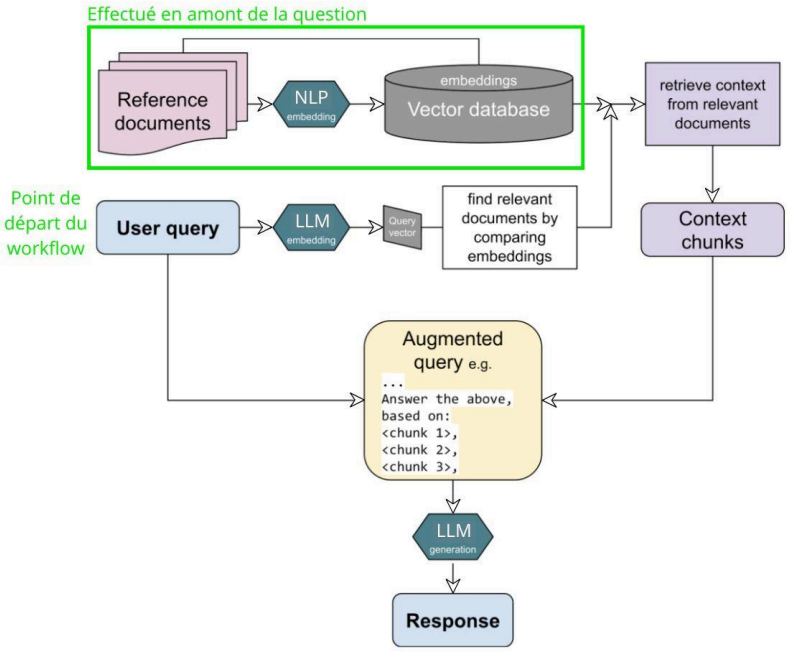


Figure 11 : Schéma de RAG (Wikipedia, 2024 b)

Ce schéma (fig. 11) illustre comment la RAG fonctionne.

Avant de pouvoir lancer une requête, il faut ingérer dans un vector store les documents que l'on souhaite utiliser pour enrichir le LLM qui formulera la réponse. Des documents peuvent être également rajoutés à n'importe quel moment et être utilisés pour les questions suivantes.

Une fois que les documents souhaités sont ajoutés à la base vectorielle, une question peut être posée. Cette question sera mise sous forme d'embedding, puis une similarity search sera effectuée par le vecteur store pour trouver les documents ou les chunks les plus similaires, c'est-à-dire ceux qui seront les plus utiles pour aider le LLM à répondre à la question.

Après cette phrase de récupération, retrieval en anglais, les chunks ou documents sont passés au LLM avec la question, sous forme de texte. Le LLM formule la réponse.

2.4.3. Exemple de RAG

Voilà un exemple d'implémentation de la RAG avec LangChain (LangChain, n.d. b).

Ce premier bloc de code (fig. 12) sert à importer la clé API d'OpenAI qui permet d'accéder à ce service payant, et d'indiquer pour la suite quelle modèle nous voulons utiliser, ici gpt-4o-mini.

```python
import os
from api_key import OPENAI_API_KEY
from langchain_openai import ChatOpenAI

os.environ["OPENAI_API_KEY"] = OPENAI_API_KEY
llm = ChatOpenAI(model="gpt-4o-mini")
```

Figure 12 : Liaison à l'API OpenAI pour la RAG.

Dans ce second bloc de code (fig. 13), nous définissons les composants de l'OutputParser, élément de la librairie LangChain évoqué plus tôt permettant d'imposer un format de réponse au modèle. Le document sur lequel nous allons extraire de l'information est le rapport annuel de l'entreprise adidas de de l'année 2022, et la question posée est "Quels sont les revenus annuels d'adidas par année ?".

```python
from langchain_core.output_parsers import PydanticOutputParser
from pydantic import BaseModel, Field
import enum

# Units class, to force the units to be of a certain value
class Units(str, enum.Enum):
    DOLLAR = "dollar"
    EURO = "euro"
    YEN = "yen"

# Pydantic class for formatting the LLM answer for one year
class RevenuePerYear(BaseModel):
    """Parsing information about revenue of the company for a given year."""
    company_name: str = Field(..., description="name of the company")
    year: int = Field(..., description="year of the revenue")
    revenue: float = Field(..., description="amount of revenue")
    unit: Units = Field(..., description="unit of revenue")

# Pydantic class that takes the revenue for all the years
class AnnualRevenues(BaseModel):
    """Parsing information about all available revenues of the company."""
    data: list[RevenuePerYear] = Field(..., description="Revenue for a given year", )

parser = PydanticOutputParser(pydantic_object=AnnualRevenues)
```

Figure 13 : Création de l'OutputParser pour la RAG.

La première étape pour avoir la réponse à cette question est d'indexer le document dans la vector database, ici Chroma (fig. 14). Il faut alors récupérer le document dans notre espace de stockage, puis découper le document en chunks, ici de 1000 tokens avec 200 tokens de recoupement pour garder plus de contexte, et enfin créer les embeddings de chacuns de ces chunks.

Le prompt (fig. 14) que nous écrivons sert au modèle à comprendre ce qui est attendu de lui et qu'il doit chercher dans le contexte qui lui a été donné la réponse à la question posée.

La *rag_chain* (fig. 14) qui est définie, puis appelée, fait la similarity search sur le vector store avec la question posée, puis donne ce contexte au modèle dans le prompt, et ensuite la forme attendue est donnée à la réponse.

```python
from langchain_core.prompts import PromptTemplate
from langchain_chroma import Chroma
from langchain_core.runnables import RunnablePassthrough
from langchain_openai import OpenAIEmbeddings
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain_community.document_loaders import PyPDFLoader

# Load report
loader = PyPDFLoader(
    "annual-report-adidas-ar22.pdf",
)
docs = loader.load()

# Split the documents in chunks, create their embeddings and store them in a vector database
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
splits = text_splitter.split_documents(docs)
vectorstore = Chroma.from_documents(documents=splits, embedding=OpenAIEmbeddings())

# Prompt that will be given to the model. It explains to use the retrieved chunks to answer the given question
prompt = PromptTemplate(
    template="You are an assistant for question-answering tasks. Use the following pieces of retrieved context
      to answer the question. If you don't know the answer, just say that you don't know.
     .\n{format_instructions}\nQuestion: {question} \nContext: {context} \nAnswer:",
    input_variables=["query"],
    partial_variables={"format_instructions": parser.get_format_instructions()},
)

def format_docs(docs):
    return "\n\n".join(doc.page_content for doc in docs)

# Create the retrieval and generation chain to answer the question using the relevant chunks of the report.
retriever = vectorstore.as_retriever()
rag_chain = (
    {"context": retriever | format_docs, "question": RunnablePassthrough()}
    | prompt
    | llm
    | parser
)

# Call the chain : retrieval and answer generation will happen there
rag_chain.invoke("What adidas annual revenue per years?")
```

Figure 14 : Retrieval et génération de la réponse de la RAG.

Le résultat de ce code est le suivant (fig. 15). On a alors un objet de classe AnnualRevenues, avec deux éléments de classe RevenuePerYear dans la liste *data*, un pour l'année 2022 et un pour l'année 2021

```
Out 4   AnnualRevenues(data=[RevenuePerYear(company_name='adidas AG', year=2022, revenue=4814.0, unit=<Units.EURO: 'euro'>),
        RevenuePerYear(company_name='adidas AG', year=2021, revenue=4475.0, unit=<Units.EURO: 'euro'>)])
```

Figure 15 : Résultat de la RAG.

La page du rapport annuel d'adidas (adidas, 2023) dans laquelle ces informations ont été trouvées est la suivante (fig. 16), en haut du premier tableau on trouve bien les valeurs 4814 et 4475 pour les années 2022 et 2021.

INCOME STATEMENT

STATEMENT OF INCOME IN ACCORDANCE WITH HGB (CONDENSED) € IN MILLIONS

	2022	2021
Net sales	4,814	4,475
Change in inventory	2	–
Total output	**4,816**	**4,475**
Other operating income	1,226	649
Cost of materials	(1,878)	(1,744)
Personnel expenses	(726)	(769)
Depreciation and amortization	(140)	(117)
Other operating expenses	(3,415)	(2,462)
Operating profit	**(117)**	**32**
Financial result	2,237	1,916
Taxes	(63)	(98)
Net income	**2,057**	**1,850**
Retained earnings brought forward	724	580
Allocation to other revenue reserves	(500)	(925)
Allocation to capital reserves	(12)	(8)
Utilization for the repurchase/issuance of adidas AG shares	(1,546)	(163)
Retained earnings	**723**	**1,334**

ADIDAS AG NET SALES € IN MILLIONS

	2022	2021
Royalty and commission income	2,394	2,237
adidas Germany	1,511	1,436
Foreign subsidiaries	80	39
Central distribution	118	120
Other revenues	711	643
Total	**4,814**	**4,475**

NET SALES INCREASE 8%

Sales of adidas AG comprise external revenues generated by adidas Germany with products of the adidas and Reebok (until February 2022) brands as well as revenues from foreign subsidiaries. Revenues of adidas AG also include royalty and commission income, mainly from affiliated companies, revenues from central distribution, and other revenues. From March 2022 onwards the commission income for the Reebok sales is shown in other revenues. In 2022, adidas AG net sales increased 8% to € 4,814 million compared to € 4,475 million in the prior year.

OTHER OPERATING INCOME UP 89%

In 2022, other operating income of adidas AG increased 89% to € 1,226 million (2021: € 649 million). This development was primarily due to higher positive currency effects.

OTHER OPERATING EXPENSES INCREASE 39%

In 2022, other operating expenses for adidas AG increased 39% to € 3,415 million (2021: € 2,462 million). This was largely attributable to higher currency losses.

Figure 16 : Page du rapport annuel où a eu lieu l'extraction (adidas, 2023).

3. Solutions pour améliorer la performance de l'extraction de données

3.1. Amélioration de la recherche de documents

3.1.1. De nouvelles méthodes de recherche : search query, hybrid search et reranking

Comme vu précédemment, la similarity search permettant de trouver les meilleurs chunks est critique dans la formulation d'une meilleure réponse et de l'extraction des données voulues. Il est alors pertinent de chercher à améliorer cette étape.

L'approche la plus directe présentée précédemment est de chercher les k embeddings les plus proches de la question avec comme mesure de distance la similarité cosinus. On pourrait chercher à trouver une autre mesure de distance plus efficace de manière empirique, mais il n'y a pas de garantie de succès et puis il faudrait refaire cette analyse à chaque nouveau cas d'usage. D'autres techniques sont bien plus efficaces, en voici certaines (Poudel, 2024).

Search query

Dans le cadre de la RAG, l'utilisation d'une "search query", ou requête de recherche, en plus de la requête initiale de l'utilisateur permet d'améliorer la qualité des réponses générées en améliorant la récupération des documents pertinents. La séparation de la requête initiale et de la requête de recherche permet de spécifier chacune d'elle. La requête initiale peut se concentrer sur l'information précisément voulue et formuler des instructions sur la forme des réponses attendues, tandis que la requête de recherche peut se concentrer sur le contexte dans lequel sera certainement contenu l'information souhaitée.

Par exemple, prenons comme requête initiale “Quels sont les impacts économiques de la pandémie de COVID-19 sur les petites entreprises aux États-Unis ? Répond en français même si les sources sont en anglais et n'écris pas plus de 300 mots.” et comme ensemble de recherche un corpus d'articles de presse provenant de différents types de journaux. La requête de recherche pourrait contenir en plus de la question initiale les noms des revues économiques nord-américaines car ces dernières sont plus susceptibles de contenir des informations justes, détaillées et précises, à contrario de revues françaises ou européennes qui entreront moins en profondeur dans le sujet. Enlever les instructions sur la forme de la réponse permet de réduire le bruit car ce n'est pas quelque chose de pertinent à ajouter à la recherche de documents.

Une requête de recherche efficace améliore la récupération d'informations en rendant la requête plus spécifique et contextuelle, et permettant de trouver des documents directement liés à l'information qu'on souhaite extraire. En désambiguïsant les requêtes et en réduisant le bruit des documents non pertinents, la requête de recherche garantit que le contenu récupéré est de haute qualité, ce qui se traduit par des réponses générées plus précises et pertinentes.

Hybrid search

L'hybrid search est la combinaison de deux méthodes de recherche, une recherche par mot clé et une recherche vectorielle : BM25 (une variante de la recherche TF-IDF précédemment vue) et la similarity search. La recherche hybrid effectue tout d'abord ces deux techniques de recherche, chacune sortant un ensemble des k chunks les plus proches de la question, soit un total de k à 2*k chunks suivant la proportion de chunks en commun. Puis selon une pondération sur chacune des deux méthodes de recherche, généralement 0.5 chacune, les k meilleurs chunks sont choisis.

Cette méthode permet d'avoir des résultats plus moyennés et de limiter les biais que peuvent avoir les modèles d'embeddings. Cela permet d'éviter de trop grandes erreurs qui peuvent avoir lieu à cause de ces biais.

Re-ranking, ou re-classification

Les re-ranking models, ou modèles de re-classification, sont des modèles utilisés pour améliorer les résultats initiaux produits par les méthodes de recherche par mots clés, de recherche vectorielle ou de recherche hybride. Ces modèles de re-classification sont des LLM spécialisés dans la classification de documents.

L'utilisation de modèles de reranking consiste tout d'abord à effectuer une recherche sémantique, par mot cles ou hybride, sur un corpus de documents à partir d'une requête et de récupérer une liste classée des k documents les plus proches. Un modèle de re-classification est ensuite utilisé en lui fournissant cette liste initiale de k documents et la question qui a été posée, il génère une liste classée des m documents les plus pertinents. Le modèle de reranking produit ainsi une nouvelle liste de documents généralement plus pertinente et mieux alignée avec l'intention de la requête initiale de l'utilisateur.

Les reranking modèles permettent d'affiner les résultats en prenant en compte des interactions complexes et des caractéristiques supplémentaires qui ne sont pas considérées par les modèles de recherche initiale. Cela conduit à une meilleure précision et pertinence des résultats.

3.1.2. Le modèle du Parent Document Retriever

Comme évoqué précédemment, le choix de la taille des chunks dont on crée les embeddings est primordial dans la qualité de la récupération des documents. Des chunks plus petits sont plus précis mais porteurs de trop peu de contexte, et des chunks plus grands ont plus de contexte mais sont trop peu précis et ont trop de bruit. Une solution à ce problème permettant de trouver un équilibre entre le besoin d'informations détaillées et spécifiques et le besoin d'une compréhension contextuelle plus large est le modèle de Parent Document Retriever (Nayak, 2023).

Le modèle de Parent Document Retriever utilise une stratégie de recherche en deux temps. Le premier temps consiste à extraire des éléments plus petits et plus précis qui répondent directement aux spécificités d'une requête, ce qui garantit des réponses pertinentes. Le second temps consiste à récupérer les documents parents de plus grandes tailles desquels ces éléments sont dérivés. Cette étape fournit un contexte et une profondeur

supplémentaires, enrichissant la réponse d'une perspective plus large qui soutient les informations détaillées.

En pratique, cela s'effectue en découpant les documents dont on dispose en gros chunks qui seront les parents, puis ces chunks sont redécoupés en chunks plus petits. Ce sont ces petits chunks qui sont mis sous forme d'embeddings dans un vector store, en stockant bien l'information de quels chunks parents ils proviennent. Lorsqu'une recherche est effectuée, c'est donc sur ces petits chunks qu'a lieu la recherche sémantique, mais ce sont les chunks parents des meilleurs petits chunks qui sont renvoyés (Poudel, 2024).

Ainsi, on fournit au LLM qui génère la réponse à la requête un contexte plus grand et plus clair, tout en s'assurant une recherche efficace des chunks les plus pertinents.

Par exemple, dans le cas de l'extraction de données ESG de rapports annuels d'entreprises, on peut prendre comme taille de chunk parents une page entière, puis la découper en trois. Ainsi le LLM répondant à la requête aura plus de contexte quand un chunk lui sera fourni comme contexte.

Voilà un schéma sur le Parent Document Retriever (fig. 17).

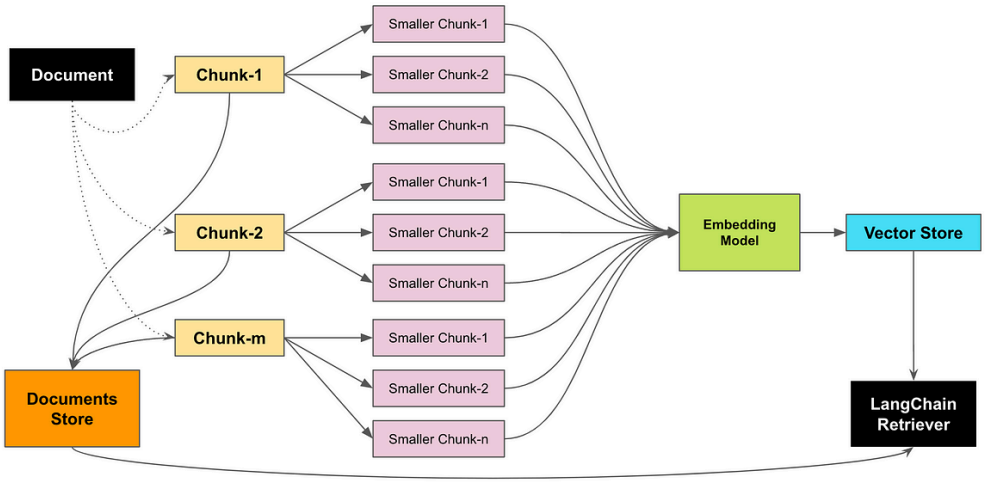


Figure 17 : Schéma de la méthode Parent Document Retriever (Rutecki, 2024).

3.1.3. Recherche par table et multi-vector retriever

Les données ESG recherchées dans les rapports annuels d'entreprises sont souvent dans des tableaux comme on a pu le voir dans des exemples précédents, et contiennent beaucoup d'informations. Ces tableaux ne sont pas pour autant des données structurées car ils sont du texte sans les marqueurs de colonnes et de lignes que les fichiers CSV peuvent par exemple avoir. La mise au point d'une méthode spécialement pour l'extraction de données de tableau peut alors sembler pertinente.

Extraire directement les tableaux des documents est tentant. Pour faire cela, il faut utiliser une des différentes méthodes de détection de tableaux déjà existantes. Cependant, ces méthodes ne sont pas parfaites et ont tendance à perdre l'entête des tableaux, ce qui est un contexte précieux pour l'interprétation d'un tableau.

Une solution plus efficace consiste à ne pas passer par l'extraction ciblée des tableaux qui a tendance à mal les isoler, mais à effectuer un découpage du document initial en chunks puis à chercher si un tableau est présent ou non dans ce chunk. Etant donné que les documents que nous utilisons sont écrits par et pour des humains, les tableaux sont généralement conçus pour tenir sur une seule page afin de permettre une meilleure lisibilité et lecture de ces derniers. Il est alors logique de faire le découpage du document pour les parents en morceaux d'une page (LangChain, 2023 b).

On a à présent découpé le document initial et repéré quelles pages contiennent un ou des tableaux à l'aide d'une méthode déjà existante dans la librairie LangChain. Il alors à mettre en place un mécanisme de recherche et de récupération des tables en fonction de la question posée. La solution est alors de faire un retriever qui se concentre sur ces tables. Pour ce faire, on va utiliser un LLM en lui donnant les pages contenant au moins un tableau et lui demander de résumer chacun de ces tableaux. Le LLM est capable de faire un résumé de bonne qualité car il possède toute la page, et donc le contexte du tableau. Ensuite, nous créons les embeddings de ces résumés que nous mettons dans un vector store qui sera utilisé par la RAG, tout en ayant un mécanisme permettant de remonter d'un embedding à la page qui le contient. Cette technique consistant à faire le résumé d'un chunk et de faire la récupération de document dessus dans le cadre de la RAG s'appelle plus généralement le multi-vector retriever et peut s'effectuer avec du texte, des tableaux, des images ou encore des enregistrements audios.

Lorsqu'une requête est effectuée, la recherche du retriever s'effectue sur les embeddings des chunks de texte mais aussi sur les embeddings des tableaux, ces derniers ayant été résumés avec leur contexte et donc leur sens (LangChain, 2023 b). Un certain nombre de chunks est récupéré de chacun des vector stores et ils peuvent être reclassés si souhaité (fig. 18). Ce sont, au choix, les pages originales des tableaux de forte similarité ou les résumés de ces derniers qui sont ensuite envoyés au LLM qui répondra à la requête initiale.

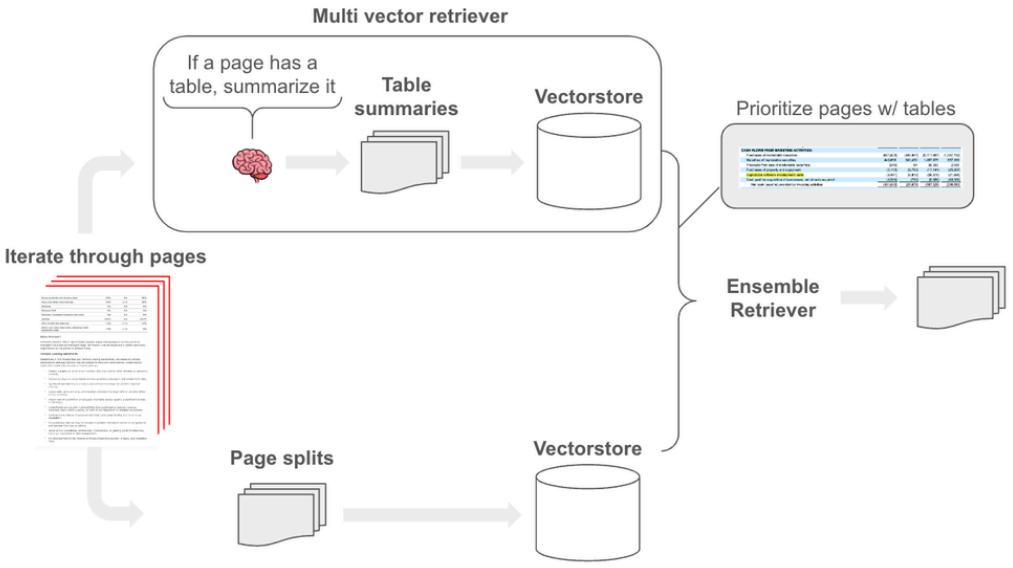


Figure 18 : Schéma de multi vector retriever (LangChain, 2023 b).

En conclusion, diverses techniques permettent d'améliorer la recherche sémantique et la récupération d'informations pertinentes, telles que l'utilisation de requêtes de recherche spécifiques, la recherche hybride, le re-ranking, le Parent Document Retriever et la recherche par table. Ces méthodes améliorent la récupération des textes contenant les informations recherchées et permettent de ne donner que des choses utiles au modèle répondant à la question posée, ce qui lui permet de bien accomplir la tâche attendue de lui.

Nous allons maintenant explorer des solutions de prompt engineering qui visent non plus à améliorer la récupération de documents mais à améliorer le travail que fait le LLM en formulant de meilleures requêtes.

3.2. Amélioration des prompt : prompt engineering

Alors que la phase de retrieval dans les architectures RAG se concentre sur l'identification des informations pertinentes au sein d'un vaste corpus, le Prompt Engineering est le moyen d'améliorer le résultat du LLM générant la réponse en l'orientant vers des réponses plus précises, pertinentes et contextualisées.

Le Prompt Engineering consiste à concevoir, affiner, et adapter les instructions données aux modèles de langage pour qu'ils produisent des résultats optimaux en fonction de la tâche spécifique. Cette démarche devient essentielle lorsqu'il s'agit d'extraire des données structurées car ce sont des données complexes qu'il faut chercher dans des documents non structurés et qu'il faut ensuite structurer d'une façon précise. La formulation du prompt peut significativement influencer la performance du modèle en lui donnant un cadre clair ou encore des indications sur le raisonnement à suivre.

3.2.1. Chain-of-thought

La chain-of-thought est la première technique de prompt engineering que nous allons présenter. Cette technique est particulièrement intéressante pour les tâches complexes car elle incite le modèle à appliquer un raisonnement logique et à décomposer le problème posé en plusieurs étapes plutôt que de fournir une réponse immédiate. Le prompt soumis au modèle contient alors les informations qui le guident à expliciter son processus de réflexion, en explicitant chaque phase du raisonnement qui le mène à la solution finale. Cette technique repose sur l'idée que, tout comme un être humain qui résout un problème, un LLM peut améliorer la qualité de ses réponses s'il suit une approche plus structurée de raisonnement. Cette technique a été pour la première fois présentée dans "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models" par Jason Wei et al. (Wei et al., 2022).

Pour appliquer cette technique, deux méthodes existent. Tout d'abord le few-shot chain-of-thought qui consiste à mettre dans le prompt, en plus du problème qu'on veut résoudre, un ou plusieurs exemples de problèmes similaires avec la question et la réponse qui déroule le raisonnement à faire pour y répondre. Ces exemples montrent explicitement comment appliquer la technique de raisonnement en chaîne à des cas particuliers, et donc d'avoir une meilleure compréhension du déroulement de chaque étape intermédiaire.

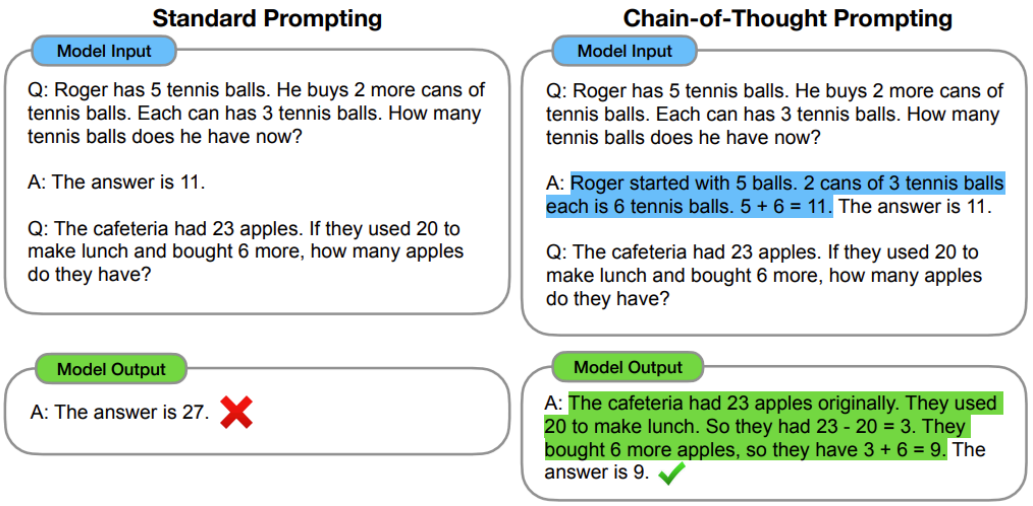


Figure 19 : Schéma de la chain-of-thought prompting (Wei et al., 2022).

La seconde méthode est appelée zero-shot chain-of-thought et consiste à rajouter dans le prompt en plus du problème souhaité la phrase “Let’s think step by step”, ou en français “Réfléchissons étape par étape”. Cela a pour effet de pousser le modèle à appliquer un raisonnement logique étape par étape. Cette méthode peut se peut par ailleurs se combiner avec le few-shot chain-of-thought (Badhan, n.d.).

Testée avec un même modèle PaLM 540B sur l’ensemble de problèmes mathématiques GSM8K, standard de test d’intelligences artificielles, la chain-of-thought permet de grandement améliorer les résultats par rapport à un prompt classique. Le taux de résolution des problèmes passe de 18% avec des méthodes de prompt classiques à 51% de taux de résolution (Wei et al. 2023)

L'application de la chain-of-thought est pertinente dans les tâches d'extraction d'informations car elle force le modèle à tout d’abord identifier les sections de texte intéressantes, puis à extraire les informations clés, et enfin à synthétiser ces éléments pour répondre précisément au problème qui lui a été posé. Ce processus en plusieurs étapes limite l’oublis de détails importants en incitant le modèle à considérer l'ensemble de ce qu’il a à sa disposition avant de formuler sa réponse.

Un autre avantage de la chain-of-thought est qu’elle améliore la transparence du modèle en rendant visible le cheminement logique emprunté pour arriver à la conclusion effectuée. Cela permet de mieux comprendre la réponse fournie mais aussi de repérer plus facilement les erreurs ou incohérences dans le raisonnement du modèle. En voyant une erreur de raisonnement du modèle on peut alors rajouter des indications dans le prompt afin de mieux le guider et qu’il ne reproduise pas les mêmes erreurs.

3.2.2. Step-back prompting

La seconde technique de prompt engineering que nous allons présenter est le step-back prompting. Le step-back prompting peut sembler similaire à la chain-of-thought mais repose sur un principe différent. Le principe est que pour les humains, dans le cadre d’une prise de décision complexe, faire preuve d’abstraction aide à avoir une vision d’ensemble, plus large,

et d'ainsi mieux juger quelle décision est la bonne à prendre. Appliquer ce principe aux LLM a été démontré comme étant juste dans la publication "Take a Step Back: Evoking Reasoning via Abstraction in Large Language Models" par Huaixiu Steven Zheng et al. (Zheng et al., 2023)

Cela se fait pour les LLM en ne posant pas directement la question souhaitée, mais en posant tout d'abord une question plus large ou plus abstraite. Ensuite, la question souhaitée est posée au LLM en ajoutant dans le contexte la question abstraite et la réponse qui a été formulée. Ainsi, l'attention du LLM se pose sur la notion abstraite qui lui sert de base et de guide pour formuler une réponse à la question.

Tout comme pour la chain-of-thought, la méthode few-shot peut être utilisée en donnant un exemple qui sert au LLM à mieux comprendre ce qui est attendu de lui. La publication à l'origine de la méthode step-back prompting explique qu'il y a un gain de performance à donner un seul exemple, puis que donner plus d'exemples ne permet aucun autre gain.

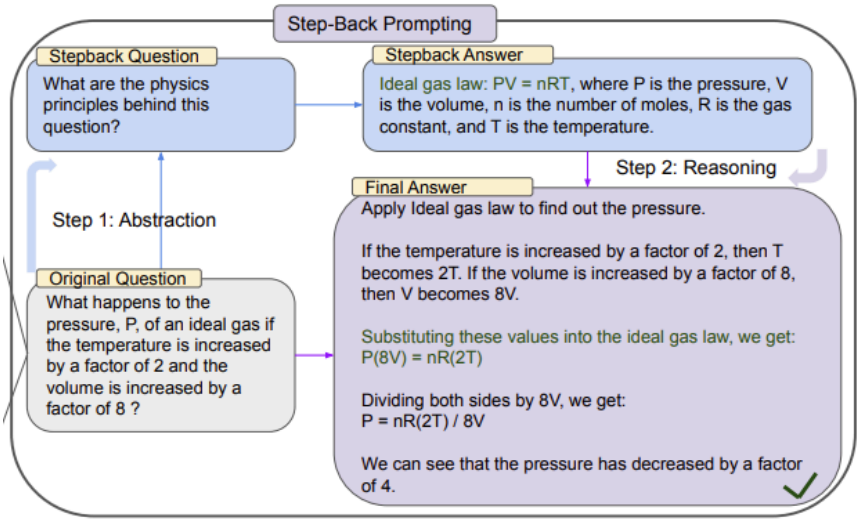


Figure 20 : Schéma du step-back prompting (Zheng et al., 2023).

En conclusion, la chain-of-thought et le step back prompting sont des techniques de prompt qui permettent d'améliorer les performances des modèles de langage dans des tâches complexes. Évaluons à présent l'intérêt de ces méthodes et des méthodes d'amélioration de recherche de documents dans le cas de l'extraction de données ESG afin de proposer des solutions plus précises.

4. Analyse des solutions proposées

Comme présenté précédemment, dans le cadre de notations ESG les données proviennent de documents non structurés de grande taille et présentant des spécificités à prendre en compte pour leur extraction. Analysons les pistes d'améliorations de recherche de documents et de prompt sous le prisme de notre problématique.

4.1. L'enjeux de la recherche de l'information dans des documents

Les sources de données sont des documents de grande taille et de densité d'information non homogène, et cela rend nécessaires d'isoler les sections susceptibles de contenir les données cherchées pour respecter les contraintes de taille de contexte des modèles. C'est dans cette optique que nous avons présenté diverses solutions possibles pour améliorer la recherche d'informations dans des documents. Voyons lesquelles sont les plus adéquates.

Avant toute chose, soulignons que pour déterminer l'intérêt d'une méthode plutôt qu'une autre, il faut établir un ensemble de données et un protocole de test qui mesure les performances d'une méthode. Dans notre cas d'extraction de données ESG depuis des documents, cela peut passer par un tableau avec chaque ligne indiquant quel document est analysé, quelle est l'information attendue et où elle se situe dans le document. La composante de positionnement dans le document est importante car c'est justement les performances de localisation de la donnée que nous cherchons à améliorer.

La première solution présentée a été la création d'une search query. Cela semble être une bonne piste car les questions posées au modèle expliquent quelles informations sont précisément voulues et sous quelles formes les renvoyer, ce qui représente un bruit qui risque de perturber la recherche des sections intéressantes du texte. L'intérêt serait alors de faire une search query contenant une question la plus directe possible et/ou des mots clés portant sur ce qui est cherché et qui pourraient alors se trouver proche dans le texte de l'information recherchée.

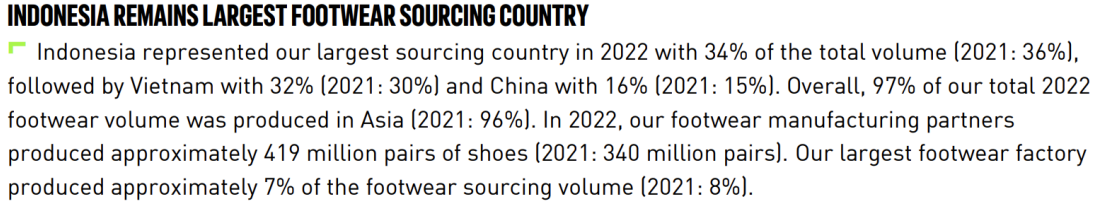
Une autre méthode que l'on peut considérer comme très certainement bénéfique est celle du modèle du Parent Document Retriever. Cette méthode permet de replacer les extraits de texte dans leur contexte global, une nécessité dans les rapports ESG où le contexte peut radicalement modifier l'interprétation des données. Cette méthode augmente grandement les chances de fournir un contexte suffisamment grand au modèle pour permettre une bonne extraction de l'information, tout en permettant une recherche performante des segments de texte pertinents contenant l'information recherchée. Cela permet également de faire un découpage page par page du document analysé et d'ainsi posséder une métrique simple et efficace pour mesurer la position des chunks dans les documents.

Ensuite, l'hybrid search et le reranking sont deux méthodes dont nous ne pouvons pas anticiper si elles ont un fort intérêt dans notre cadre d'application. Leur mise en place ne représente pas de difficulté majeure grâce à la librairie LangChain, en quelques lignes de code cela peut être intégré. Il est donc encore plus nécessaire de tester leur implémentation.

Enfin, la dernière méthode proposée, la recherche par table, convient très bien à notre cas d'étude. Comme présenté plus tôt, dans les documents que nous utilisons, rapport annuels ou rapport ESG d'entreprises, les données chiffrées sont très souvent présentées dans des tableaux par souci de clarté pour le lecteur. Cependant ce format n'est pas forcément optimal pour effectuer des similarity search de qualité car il n'y a que peu de mots - titre, nom des lignes et des colonnes - pour un grand nombre de caractères numériques, et donc pas de sens évident. Mettre en place un vector store hybride et plus adapté à ces tableaux peut permettre une recherche plus fine et précise et est donc pertinent. Mettre en place cela est plus lourd que la mise en place des autres méthodes proposées et devra également être testé après sur un ensemble de données pour évaluer le gain que cela apporte.

4.2. Amélioration des prompts par de nouvelles techniques.

L'autre axe d'amélioration présenté était celui des prompts. Ces questions et les contextes passés au LLM sont clé car c'est cela qui dirige son raisonnement, lui fait comprendre ce qu'on souhaite et lui fournit des informations additionnelles. L'intérêt d'une technique de prompt dépend cependant beaucoup plus du contexte d'usage du modèle et du résultat attendu, en particulier s'il est attendu une simple extraction d'une valeur d'un document ou si un travail de calcul, de logique ou de raisonnement doit avoir lieu pour renvoyer ce qui est attendu. Il faut alors faire une distinction entre ces deux cas pour la suite. Nous nous appuierons sur la figure 1, un extrait du rapport annuel d'adidas portant sur leur production annuelle de vêtements, en particulier de chaussures (adidas, 2023).



INDONESIA REMAINS LARGEST FOOTWEAR SOURCING COUNTRY

Indonesia represented our largest sourcing country in 2022 with 34% of the total volume (2021: 36%), followed by Vietnam with 32% (2021: 30%) and China with 16% (2021: 15%). Overall, 97% of our total 2022 footwear volume was produced in Asia (2021: 96%). In 2022, our footwear manufacturing partners produced approximately 419 million pairs of shoes (2021: 340 million pairs). Our largest footwear factory produced approximately 7% of the footwear sourcing volume (2021: 8%).

Figure 21 : Extrait du rapport annuel où doit avoir lien l'extraction (adidas, 2023).

Dans le cas de questions simples et directes, comme par exemple "Quelle est la production totale de chaussures par adidas pour l'année 2022 ?", la réponse à extraire est simplement "419 million de paires de chaussures", qui est littéralement dans le texte. Il n'y a pas de calcul ou de logique à appliquer. La chain-of-thought ou le step-back prompting n'apportent rien à cette question.

A l'opposé, pour une question plus complexe comme "Quelle est la production de chaussures par pays par adidas pour l'année 2022 ?", un raisonnement doit être appliqué. Tout d'abord, le modèle doit voir et comprendre que c'est la production totale, à l'échelle de l'entreprise, qui est donnée en unités de production, et que la production par pays est donnée en pourcentage de cette production totale. Si cette compréhension a eu lieu, le modèle doit encore calculer quel est le nombre de paires de chaussures produites par pays à partir de ces deux informations :

- Pour l'Indonésie : 0.34 x 419 = 142.46 millions de paires,
- Pour le Vietnam : 0.32 x 419 = 134.08 million de paires,
- Pour la Chine : 0.16 x 419 = 67.04 millions de paires,

Additionnellement, le modèle doit appliquer un raisonnement lui permettant de comprendre que ces trois pays ne forment pas l'intégralité des pays producteurs - les pourcentages de production de chacun ne somment pas à 100 - et alors calculer ce que les autres pays représentent comme production : (1 - 0.34 - 0.32 - 0.16) x 419 = 75.42 millions de paires.

La méthode de la chain-of-thought semble alors particulièrement appropriée à cet exemple.

La méthode de step-back prompting est peut être moins adaptée à notre cas d'application car ce ne sont pas des compétences d'abstraction dont il faut faire preuve au premier abord. Cela pourrait être peut être utile dans le cadre de données plus scientifiques où une compréhension théorique est nécessaire. Il pourrait y avoir un intérêt pour l'extraction de données environnementales présentes dans les rapports d'entreprises, comme les émissions de CO_2 ou de substances toxiques. Là encore, utiliser un ensemble de données de test serait nécessaire pour prouver que cette méthode a ou non un intérêt dans notre cas d'application.

Conclusion

L'extraction de données non structurées dans le cadre des notations ESG représente un défi de taille compte tenu de la diversité des sources, des formats et de la complexité des informations en elles-mêmes. Ces informations sont présentes dans des textes longs et présentées de manière non standardisées, ce qui rend leur extraction à la fois difficile et coûteuse, d'où l'intérêt d'automatiser ce processus.

En premier lieu il convenait d'exposer les caractéristiques des données ESG en insistant sur leur diversité et en précisant les formats spécifiques sur lesquels nous nous sommes concentrés. Nous avons ensuite présenté le NLP, les LLM, leurs spécificités et les moyens de les utiliser. Cette revue a permis de mettre en évidence le potentiel qu'ont les LLM dans le domaine ESG.

En troisième partie, nous avons proposé plusieurs solutions pour améliorer la performance de l'extraction de données à travers deux axes, la recherche dans les documents et les techniques de prompt engineering. Parmi les solutions présentées de recherche, la search query, le Parent Document Retriever et la recherche par table sont les approches les plus prometteuses. Ces techniques améliorent la précision de la recherche de présence d'information dans un texte et sont très adaptées aux documents sur lesquels nous travaillons. La solution de prompt engineering la plus intéressante est quant à elle la méthode de la chain-of-thought car une capacité de réflexion logique est nécessaire à la bonne extraction et structuration des données non structurées.

Combiner des techniques améliorant la recherche dans des documents et une optimisation des prompts améliore grandement l'extraction d'informations de sources de données non structurées. Cela permet ainsi d'accélérer et d'améliorer le processus de notation ESG en mettant plus d'informations à la disposition des analystes.

Bibliographie

adidas. (2023). *Annual report 2022*. https://report.adidas-group.com/2022/en/_assets/downloads/annual-report-adidas-ar22.pdf

Aibin, M. (2024, Mars 18). *Dimensionality of Word Embeddings*. Baeldung. https://www.baeldung.com/cs/dimensionality-word-embeddings

Autorité des marchés financiers. (2021, Mai 12). *Glossaire de la finance durable*. amf-france. https://www.amf-france.org/fr/espace-epargnants/comprendre-les-produits-financiers/finance-durable/glossaire-de-la-finance-durable

Badhan, M. (n.d.). *Comprehensive Guide to Chain-of-Thought Prompting*. Mercity AI. https://www.mercity.ai/blog-post/guide-to-chain-of-thought-prompting#what-is-chain-of-thought-prompting

Critères ESG : définition, exemple, enjeux pour les entreprises. (2023, Novembre 20). Big Média. https://bigmedia.bpifrance.fr/nos-dossiers/criteres-esg-definition-exemple-enjeux-pour-les-entreprises

Fournier, C. (2023, Février 17). *Finance durable : la jungle des labels, des réglementations et des standards*. Youmatter. https://youmatter.world/fr/categorie-economie-business/finance-durable-reglementations-revolution-flou-2023/

Ghofrane, T. (2021, Février 16). *L'Historique de L'NLP*. LinkedIn. https://www.linkedin.com/pulse/lhistorique-de-lnlp-taghouti-ghofrane-/

Glencore. (2024, Mars 20). *Annual report 2023*. https://www.glencore.com/.rest/api/v1/documents/static/d09d8212-4a9f-4034-b2d4-49152e5a0aff/GLEN-2023-Annual-Report.pdf

Jouhameau, R. (2023, Septembre 12). *L'évolution du traitement des données ESG avec le développement de l'IA et des Large Language Models*. https://romainjouhameau.com/posts/esg/IPSII____L_%c3%a9volution_du_traitement_des_donn%c3%a9es_ESG_avec_le_d%c3%a9veloppement_de_l_IA_et_des_grands_mod%c3%a8les_linguistiques.pdf

Jumelle, M. (2023, Décembre 19). *Large Language Models (LLM) : tout savoir*. Blent.ai. https://blent.ai/blog/a/llm-tout-savoir

LangChain. (n.d. a). *LangChain Expression Language (LCEL)*. LangChain. https://python.langchain.com/v0.1/docs/expression_language/

LangChain. (n.d. b). *Build a Retrieval Augmented Generation (RAG) App*. LangChain. https://python.langchain.com/v0.2/docs/tutorials/rag/

LangChain. (2023 a, Octobre 20). *Multi-Vector Retriever for RAG on tables, text, and images*. LangChain Blog. https://blog.langchain.dev/semi-structured-multi-modal-rag/

LangChain. (2023 b, Décembre 13). *Benchmarking RAG on tables*. LangChain Blog. https://blog.langchain.dev/benchmarking-rag-on-tables/

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020, May 22). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. https://arxiv.org/abs/2005.11401

Mansurova, M. (2024, Février 13). *Text Embeddings: Comprehensive Guide | by Mariya Mansurova*. Medium. https://towardsdatascience.com/text-embeddings-comprehensive-guide-afd97fce8fb5

Marshall, C. (2020, Février 12). *How is OCR used in the real world?* Medium. How is OCR used in the real world?

Mei, T. (2019, Décembre 14). *Demystify TF-IDF in Indexing and Ranking*. Medium. https://ted-mei.medium.com/demystify-tf-idf-in-indexing-and-ranking-5c3ae88c3fa0

mteb. (2024, Août). *MTEB Leaderboard - a Hugging Face Space by mteb*. Hugging Face. https://huggingface.co/spaces/mteb/leaderboard

Nayak, P. (2023, Octobre 29). *Advanced RAG- Providing Broader Context to LLMs Using ParentDocumentRetriever*. Medium. https://medium.aiplanet.com/advanced-rag-providing-broader-context-to-llms-using-parentdocumentretriever-cc627762305a

OpenAI. (n.d.). *Pricing*. OpenAI. https://openai.com/api/pricing/

OpenAI. (2024, Mai 13). *ChatGPT*. Retrieved Juillet, 2024, from https://chatgpt.com/

Poudel, N. (2024, Janvier 26). *Advanced RAG Implementation using Hybrid Search and Reranking | by Nadika Poudel*. Medium. https://medium.com/@nadikapoudel16/advanced-rag-implementation-using-hybrid-search-reranking-with-zephyr-alpha-llm-4340b55fef22

Rutecki, M. (2024, Juin 12). *RAG: Parent Document Retriever in LangChain*. Kaggle. https://www.kaggle.com/code/marcinrutecki/rag-parent-document-retriever-in-langchain

Searle, J. (2024, Août 9). *Natural language processing*. Wikipedia. https://en.wikipedia.org/wiki/Natural_language_processing

Tripathi, R. (2023, Juin 30). *What is Similarity Search?* Pinecone. https://www.pinecone.io/learn/what-is-similarity-search/

Weaviate. (n.d.). *Chunking long texts*. Weaviate. https://weaviate.io/developers/academy/py/standalone/chunking

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022, Janvier 28). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. arXiv. https://arxiv.org/abs/2201.11903

Wikipedia. (2024 a, Juillet 14). *Noam Chomsky*. Wikipedia. https://en.wikipedia.org/wiki/Noam_Chomsky

Wikipedia. (2024 b, Juillet 16). *RAG diagram*. Wikipedia. https://en.m.wikipedia.org/wiki/File:RAG_diagram.svg

Zheng, H. S., Mishra, S., Chen, X., Cheng, H.-T., Chi, E. H., Le, Q. V., & Zhou, D. (2023, Octobre 9). *Take a Step Back: Evoking Reasoning via Abstraction in Large Language Models*. arXiv. https://arxiv.org/abs/2310.06117