

# Embeddings et utilisation des LMs

EMSI - Université Côte d'Azur  
Richard Grin  
Version 1.29 - 2/10/25

1

## Plan du support

- Embeddings
- Utilisation des LMs
- APIs des LMs
- Prompt engineering
- LM local
- Outils pour travailler avec les LMs et les tester
- Références

R. Grin

LM

2

# Embedding

3

## Présentation embedding

- Embedding : terme mathématique pour désigner en anglais le plongement d'un ensemble dans un espace vectoriel
- Pour ce cours, un embedding est un **vecteur de nombres réels** qui représente un token, un mot, un texte ou, plus généralement, un objet (image, audio, graphe,...)
- Dans ce support, pour simplifier, on se limite aux embeddings de textes

R. Grin

LM

4

## Embedding et sens

- Un embedding capture le sens d'un texte sous une forme compactée, en intégrant des informations variées dans les coordonnées du vecteur
- Par exemple une des coordonnées peut représenter le caractère « animal » : 2 embeddings de mots désignant des animaux auront des valeurs proches pour cette coordonnée
- Les vecteurs sont souvent de grande dimension (plusieurs centaines de coordonnées) pour intégrer du sens selon de nombreux axes

5

## Similarité

- Principe de base pour la représentation par des embeddings : des données qui ont des **sens similaires** doivent être représentés par des vecteurs similaires
- La similarité entre 2 embeddings est mesurée par leur distance : distance faible  $\Rightarrow$  données similaires
- La distance peut être le cosinus de l'angle formé par les 2 vecteurs, mais d'autres métriques peuvent être utilisées (produit scalaire des 2 vecteurs, distance euclidienne)

R. Grin

LM

6

1

## Exemple simplifié de vecteurs

- Le mot « homme » peut être représenté par le vecteur (0,69 ; 0,87 ; 1,9 ; ...), le mot « femme » par (0,70 ; 0,90 ; -1,7 ; ...)
- Les différentes dimensions représentent le fait d'être vivant, d'être un humain, le genre, ...
- Dans les embeddings produits par les modèles d'embeddings (modèles d'IA qui génèrent les embeddings), les significations des dimensions ne sont interprétables de manière aussi lisible qu'avec cet exemple simplifié

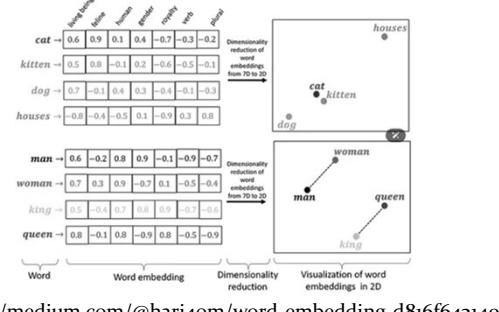
R. Grin

LM

7

7

## Autres exemples simplifiés



Image

<https://medium.com/@hari4om/word-embedding-d816f643140>

R. Grin

LM

8

8

## Exemple embeddings de phrases

- Embedding de "Le chien court dans le parc." : [0.2, 0.3, 0.7, 0.5, 0.1]
- Embedding de "Un chat joue dans le jardin." : [0.21, 0.29, 0.3, 0.51, 0.11]
- Les embeddings sont proches car les phrases parlent d'animaux qui font une action dans un espace extérieur
- Différents de celui de "Il pleut aujourd'hui." : [0.01, 0.15, 0.05, 0.9, 0.3]

R. Grin

LM

9

9

## Utilisations embeddings

- Pour représenter la signification de textes
- Pendant l'entraînement et l'utilisation des LMs, les tokens sont transformés en embeddings pour leur associer un sens et capturer les relations entre les mots et concepts
- Pour classer un texte selon certains critères : représentation de sentiments, catégories des sujets dont on parle, filtrage de spam, analyse des opinions,...
- Pour la traduction de textes
- Pour les recherches sémantiques, en particulier pour le RAG (étudié par ailleurs), pour retrouver les informations les plus pertinentes pour répondre à une question

R. Grin

LM

10

10

## Recherche sémantique

- La recherche par mots-clés se base sur la correspondance exacte (ou presque exacte) des mots présents dans la requête
- La recherche sémantique essaie de comprendre l'intention ou le sens derrière les mots et peut donner des résultats pertinents, même si les termes exacts ne sont pas présents

R. Grin

LM

11

11

## Exemple 1

- Recherche « Recette facile pour gâteau au chocolat »
- Résultats avec recherche par mots-clés : recettes qui contiennent ces mots
- Résultats avec recherche sémantique : recettes utilisant des synonymes comme « gâteau express », « fondant au chocolat », ou « dessert au chocolat simple », même si les mots « facile » et « gâteau » ne sont pas utilisés tels quels

R. Grin

LM

12

12

## Exemple 2 (RAG)

- Recherche des textes utiles pour répondre à une question
- Tous les textes sont préalablement transformés en embeddings avec un modèle d'embedding
- La question est transformée en embedding, avec le même modèle d'embedding
- Récupération des textes qui ont les embeddings les plus similaires à l'embedding de la question, donc les plus pertinents pour répondre à la question

R. Grin

LM

13

13

## Modèle d'embedding

- Modèle au sens du machine learning : nécessite un entraînement avant d'être utilisé
- Son but est de fournir des embeddings
- Entraîné en lui donnant une tâche qui force les textes similaires à avoir des vecteurs proches, et les textes différents à avoir des vecteurs éloignés
- Un embedding dépend du modèle d'embedding qui l'a créé ; pour un même texte, l'embedding généré peut être très différent selon le modèle utilisé
- Le choix du modèle d'embedding est important

R. Grin

LM



14

14

## Implémentation modèle embedding

- Les modèles d'embedding sont obtenus
  - soit avec des méthodes simples basées sur des co-occurrences de mots ou de petits réseaux de neurones
  - soit basés sur les transformateurs, principalement les encodeurs

R. Grin

LM

15

15

## Types modèles pour textes

- Basés sur des mots (Word2Vec, GloVe, FastText) ; les embeddings ne dépendent pas du contexte ; insuffisant pour phrases complexes, mais efficace, par exemple, pour reconnaître des mots qui désignent des entités ou des lieux précis
- Pour des phrases entières (SBERT, USE) ; pour recherche sémantique ; moins précis que basés sur contexte
- Basés sur les contextes (BERT, GPT-like) ; embeddings dépendent du contexte ; par exemple distinguent *banc* public ou *banc* de poissons ; pour tâches plus complexes où le contexte est important ; plus lourds en calcul et gourmands en mémoire

R. Grin

LM

16

16

## Caractéristiques des embeddings

- Dimensions des vecteurs très diverses : 300 pour Word2Vec et GloVe, 512, 768 ou 1 024 pour les modèles BERT, plus de 10 000 pour DaVinci (GPT-3)
- Le choix de la dimension est important : une grande dimension permet de capturer plus de détails, mais avec des performances moindres et des coûts plus importants
- Limite supérieure du nombre de tokens pour qu'un texte soit transformé en embeddings ; par exemple BERT limite à 512 tokens, Llama à 4 096 et GPT-3 à 32 768
- Voir <https://huggingface.co/spaces/mteb/leaderboard>

R. Grin

LM

17

17

## Autres modèles d'embedding

- Pour images : ResNet, VGGNet
- Audio : OpenAI Jukebox, Wav2Vec
- Multi-modal : CLIP (pour images et texte)
- ...

R. Grin

LM

18

18

## Entrainement modèle embedding

- Avec de très nombreux exemples d'apprentissage
- Si 2 mots peuvent se trouver dans le même contexte, ils auront des embeddings proches
- Exemple : quand de nombreux exemples du type « Le chien mange un os » ou « Le chat mange une souris » sont fournis pendant l'apprentissage, les embeddings pour « chien » et pour « chat » vont devenir plus proches car ils se trouvent dans des contextes similaires

R. Grin

LM

19

19

## Vocabulaire

- Ensemble des unités élémentaires (tokens par exemple) que le modèle peut reconnaître et coder en embeddings
- Chaque élément du vocabulaire a une représentation unique, souvent sous forme d'un indice
- Gestion des mots inconnus : un texte peut contenir des mots non répertoriés dans le vocabulaire qui sont représentés par un embedding spécial, ou par une combinaison d'embeddings quand c'est possible

R. Grin

LM

20

20

## BDs vectorielles

- Fréquemment utilisées pour enregistrer les embeddings
- BDs optimisées pour stocker et retrouver des embeddings, les textes associés (ou un lien vers ces textes), des métadonnées
- Les index de ces BDs utilisent des structures et des techniques qui permettent d'effectuer efficacement des recherches sémantiques par similarités (plutôt que de trouver des correspondances exactes comme avec les BD relationnelles)

R. Grin

LM

21

21

## Métadonnées

- Dans une BD vectorielle, un embedding peut être associé à des métadonnées, comme des dates, des types de données, des ids
- Ces métadonnées peuvent être conservées avec l'embedding et le texte source de l'embedding
- Exemples de métadonnées :
  - date de création du document
  - auteur du texte
  - type du document qui contient le texte

R. Grin

LM

22

22

## Utilisation des LMs

R. Grin

LM

23

23

## Language Model - LM

- Modèle qui permet de traiter, comprendre et générer du langage naturel (souvent basé sur un transformeur)
- Un LM est un modèle probabilistique de texte
- On a vu que la fonction de base d'un LM est la suivante :
  - en entrée, un texte En sortie ?
  - en sortie, pour chaque token de son vocabulaire, une probabilité indiquant à quel point ce token serait adapté pour continuer le texte
- Evidemment, les probabilités dépendent des textes qui ont été fournis au LM pendant son entraînement

R. Grin

LM

24

24

## Fonctionnalité de base d'un LM

- Prédire le token suivant dans une séquence de tokens
  - Le token choisi peut être le token le plus probable, mais pas nécessairement ; par exemple, le token choisi peut aussi être tiré au hasard parmi les 5 tokens les plus probables
- Chacun des 5 tokens a les mêmes chances d'être choisi ?
- A partir de cette fonctionnalité de base un LM peut générer itérativement plusieurs tokens et ainsi dialoguer avec un humain et répondre à ses questions

R. Grin

LM

25

25

## Exemple de complétion de texte

- « Quand je suis en forme, je vais ???? »
- Le LM va compléter avec un des tokens (mots pour simplifier), compte tenu de leur probabilité
  - travailler (probabilité 0.10)
  - courir (0.05)
  - voir (0.025)
  - ...
- Si le token choisi est « courir », une nouvelle complétion donnerait peut-être « dans », etc.

R. Grin

LM

26

26

## Avertissement

- La fonctionnalité de base ne s'intéresse pas à la réalité ou à la vérité
- Les réponses générées avec un LM sont malgré tout souvent vraies si la phase d'apprentissage a utilisé des textes sélectionnés et de qualité, avec des critères de vérité
- Malgré tout, le LM peut répondre n'importe quoi, du moment que le texte généré ressemble à un texte écrit par un humain et semble plausible

R. Grin

LM

27

27

## Large Language Model - LLM

- LM, avec de très nombreux paramètres
- La distinction entre les LLMs et les LMs n'est pas clairement définie ; à partir de centaines de millions de paramètres on peut parler de LLM
- Pour la suite de ce support, les termes LLM ou LM seront utilisés indistinctement pour LM, LLM, SML (Small Language Model), sauf si la distinction doit être faite

R. Grin

LM

28

28

## Nombre de paramètres

- Llama 3 (Meta) a plusieurs modèles qui vont de 8 à 405 milliards de paramètres
- Pour GPT 4, on pense qu'il y a plus de 1000 milliards de paramètres
- Gemini 1.5 Flash a 8 milliards de paramètres, Gemini Pro en a 137 milliards

R. Grin

LM

29

29

## Exemples de LLM

- GPT de **OpenAI** (utilisé par ChatGPT) <https://chatgpt.com/>
- **Gemini** de Google <https://gemini.google.com/app>
- **Llama** de Meta (Facebook) <https://ai.meta.com/llama/>
- **Claude** de Anthropic <https://www.anthropic.com/index/clause-2>
- Le Chat de **Mistral** <https://chat.mistral.ai/chat>
- **Grok** de xAi <https://grok.com/>
- **DeepSeek** <https://www.deepseek.com/>
- **Qwen** de Alibaba <https://www.alibabacloud.com/en/solutions/generative-ai/qwen>

R. Grin

LM

30

30

## Prompt

- Question posée au LM par l'utilisateur du LM ; le LM va générer une réponse à la question ; aussi appelé « prompt utilisateur »
- Prompt système : instruction spéciale donnée au LM, avant toute autre interaction, pour définir son comportement, son rôle, le ton qu'il doit prendre, ou d'autres contraintes
- Le LM suit aussi les instructions du créateur du LM ; ces instructions sont cachées et non modifiables

R. Grin

LM

31

31

## Hyperparamètres

Différences avec paramètres ?

- Quelques hyperparamètres modifiables pendant l'inférence, qui auront un impact sur la réponse : max-tokens (limite le nombre de tokens de la réponse), température, top k, frequency penalty, presence penalty
- Remarque : ces hyperparamètres n'existent pas dans tous les LMs et les intervalles de variation et les valeurs par défaut de ces valeurs dépendent des LMs

R. Grin

LM

32

32

## Température (1/2)

- Hyperparamètre qui contrôle le degré de diversité des réponses du LM
- Plus la température est basse, plus le choix du mot va se rapprocher du choix du mot le plus probable
- Si la température est haute, le LM sera plus créatif (2 réponses à une même question pourront être bien différentes), mais avec un risque de moindre cohérence, ou de plus d'erreurs

R. Grin

LM

33

33

## Température (2/2)

- La température est souvent dans l'intervalle [0, 2] (Gemini et OpenAI) ou [0, 1]
- Valeur par défaut souvent proche de 0,7 (1 pour Gemini et OpenAI)
- A la valeur 0, le modèle choisit toujours le mot avec la plus haute probabilité
- Si on veut des informations fiables, une température faible est préférable (moins de 0,3) ; pour générer une histoire, une température élevée sera souvent un meilleur choix



R. Grin

LM

34

34

## Top k

- Nombre de tokens considérés à chaque génération de token
- Valeur par défaut environ 50, selon les LMs (toujours 64 pour Gemini)
- 0 pour aucune limitation

R. Grin

LM

35

35

## Top p

- Limite aussi les tokens qui sont considérés : la somme cumulative de leur probabilité ne doit pas dépasser cette valeur
- Valeur par défaut environ 0,9 ; dépend du LM (0,95 pour Gemini)
- 0 ou 1 : aucune limitation
- Valeur faible si on recherche des informations fiables ; plus élevée pour plus de créativité (comme température)

R. Grin

LM

36

36

## Exemple

- Voici les mots les plus probables avec leur probabilité :

1. mot<sub>1</sub> - 0,6
2. mot<sub>2</sub> - 0,2
3. mot<sub>3</sub> - 0,15
4. mot<sub>4</sub> - 0,1

Si top p est égal à 0,9, le choix se fera entre quels mots ?

R. Grin

LM

37

38

## Frequency/Presence penalty

- Pour éviter plus ou moins les répétitions dans la réponse
- Frequency penalty pénalise les tokens qui apparaissent déjà dans le prompt et le début de la réponse ; le nombre de fois qu'ils apparaissent est pris en compte
- Presence penalty est semblable ; la différence est que le nombre de fois n'est pas pris en compte

R. Grin

LM

38

## Ce que peut faire un LM

- Réponse à des questions
- Traduction de textes
- Résumé de contenus, extraction d'informations
- Corriger l'orthographe et le style d'un texte
- Aide à la programmation (code, documentation)
- Raconter une histoire
- Aide à l'apprentissage d'une langue
- Classification de textes
- Extraction d'informations
- Analyse de sentiments
- ...

R. Grin

LM

39

40

## Contexte

- C'est le contenu du prompt (prompt système et prompt utilisateur)
- Le contexte est limité en taille car l'algorithme de traitement (en particulier le mécanisme d'attention) dépend quadratiquement de sa longueur
- Le contexte contient souvent bien plus que la simple question posée par l'utilisateur, en particulier si le RAG est utilisé **Explications ?**

R. Grin

LM

40

## LM pour le développement

- Si on lui fournit du code informatique pendant sa phase d'apprentissage, un LM peut générer du code
- Le code peut être généré à partir d'une description écrite par le développeur ou/et en tenant compte de l'environnement de développement (code déjà écrit, commentaires,...)
- Aide pour des tâches répétitives et fastidieuses comme l'écriture des messages de commit Git
- Utilisation du langage naturel pour s'adresser aux APIs des LMs (voir TP 3)

R. Grin

LM

page 41

41

## Utilité et avantages des LMs

- Capacité à comprendre et générer un langage naturel complexe ; c'est la grande force des LMs : leur grande compétence linguistique
- Interaction avec les machines dans un langage naturel
- Accès rapide à de nombreuses connaissances dans des domaines très divers
- Peuvent être affinés, améliorés et adapté à des domaines spécifiques (fine-tuning, RAG, étudiés plus loin)

**Pourquoi accès rapide ? Accès rapide à des bases de données ?**

R. Grin

LM

42

## Problèmes des LMs (1/2)

- Attention aux hallucinations ! Problème principal des LMs : un LM peut inventer des faits ou des sources d'information plausibles mais faux, ou faire de mauvais raisonnements
- Non explicabilité des réponses : très difficile de comprendre pourquoi un LM a donné une certaine réponse, et donc de vérifier le « raisonnement » ou les données utilisées pour donner la réponse
- Protection des données : les utilisateurs peuvent divulgues des informations confidentielles dans leurs prompts

R. Grin

LM

page 43

43

## Problèmes des LMs (2/2)

- Attention à l'injection de prompt (semblable à l'injection SQL) : toujours filtrer les interactions entre l'utilisateur et les LMs pour éviter qu'un utilisateur n'obtienne des informations confidentielles ou ne lance des processus interdits
- Peut parfois fournir des réponses inappropriées ou offensantes : racisme, sexe, ..., ou au contraire trop autocensurées
- Ancienneté des données d'entraînement
- Coûts à surveiller
- Copyright, droit d'auteur ou copyleft

R. Grin

LM

page 44

44

## Hallucination

- Génération par un LM de contenu incorrect, inventé ou trompeur, qui n'est pas fondée sur les données d'entraînement ou les sources disponibles (par RAG, par exemple)
- L'information est factuellement erronée, ou bien le LM a inventé des noms, références ou faits qui n'existent pas ; il peut aussi déformer des informations (par exemple date modifiée)
- Le LM présente souvent ses hallucinations sur un ton convaincant et sûr de lui

R. Grin

LM

45

45

## Pourquoi ?

- Un LM est entraîné sur des textes choisis (mais pas infaillibles), et c'est pour cela que les réponses d'un LM sont souvent vraies
- Cependant, son but premier est d'écrire comme un humain, en construisant ses réponses token par token selon des probabilités, pas d'écrire toujours des informations vraies
- Un LM apprend en repérant des ressemblances entre les mots et les phrases. Parfois, il pense que deux informations se ressemblent beaucoup, même si elles ne sont pas exactement les mêmes. Cela peut l'amener à mélanger ou à inventer des faits

R. Grin

LM

46

46

## Eviter les hallucinations

- Essayer de vérifier les réponses (difficile)
- Utiliser le RAG pour vérifier les informations
- Eviter de mélanger dans un prompt ou dans les documents utilisés par le RAG des entités similaires mais en fait différentes, car le LM risque de les confondre

R. Grin

LM

47

47

## Sécurité et LMs

- Risque pour la sécurité quand on utilise un LM dans une application ; voir <https://genai.owasp.org/llm-top-10/> pour les risques principaux
- Aide apportée par un LM pour sécuriser les applications

R. Grin

LM

48

48

## Risques pour la sécurité

- Pendant le codage
  - Un LM peut générer du code erroné ou dangereux (exposition de clés, code non sécurisé, ...)
- Pendant l'exécution de l'application
  - Hallucinations
  - Injection de prompt
  - Réponses ou messages inappropriés à l'utilisateur, actions dangereuses lancées par le LM
  - Mauvaise gestion des autorisations
  - Extraction de données sensibles (données d'entraînement, informations personnelles, secrets d'entreprise, ...)

R. Grin

LM

49

49

## Bonnes pratiques pendant exécution

- Contrôler et journaliser (logging) les entrées-sorties des LMs, y compris les prompts intermédiaires (voir outils et agents)
- Conserver les métadonnées : date, utilisateur, contexte (messages système, ...), paramètres (température, ...)
- Détection et protection des données sensibles
- Demander une confirmation humaine pour les cas les plus à risques
- Gérer les autorisations d'exécution et d'accès aux données sensibles

R. Grin

LM

50

50

## Quelques outils pour la sécurité

- LLMSecGuard (<https://github.com/aryakvnust/LLMSecGuard>) pour générer du code plus sûr
- Pynt (test de sécurité des API exposant des menaces vérifiées pour les API à travers des attaques simulées ; <https://www.pynt.io/>)
- Rebuff (<https://blog.langchain.com/rebuff>), de LangChain, pour protéger contre l'injection de prompt
- Protecto (<https://www.protecto.ai>), Microsoft Presidio (<https://microsoft.github.io/presidio>) protègent les données sensibles
- Guardrails AI (<https://github.com/guardrails-ai/guardrails>) pour la détection des risques (en Python)
- NVIDIA Nemo Guardrails (<https://github.com/NVIDIA/NeMo-Guardrails>) impose des règles

R. Grin

LM

51

51

## Paradoxe des LMs généralistes

- Les LMs généralistes ont des bonnes connaissances linguistiques mais pas des connaissances d'expert dans tous les domaines
- Ils sont souvent utilisés par des personnes qui ne sont pas expertes dans le domaine sur lequel elles les interrogent
- Le problème est qu'il faut parfois très bien connaître un domaine pour repérer les possibles hallucinations

R. Grin

LM

page 52

52

## En résumé

- Les LMs sont bons en linguistique : ils sont capables de tenir une conversation comme le ferait un humain
- Ils ont des bonnes connaissances générales sur d'innombrables sujets
- Mais
  - Ils peuvent manquer de logique
  - Ils sont capables d'inventer (des faits, des liens,...)
  - Ils n'ont pas accès à des informations postérieures à leur date d'entraînement ni, évidemment, aux informations non publiques sur les entreprises

R. Grin

LM

page 53

53

## Précision

- Est-ce que ChatGPT (ou Claude, ou Gemini ou ...) est un LM ?

R. Grin

ML

54

54

## LM et Web

- La plupart des LMs des dernières générations sont secondés par des modules qui leur permettent de chercher des informations ou de vérifier ce qu'ils disent en consultant le Web
- En fait, c'est l'application IA qui utilise le LM qui ajoute cette fonctionnalité

R. Grin

LM

55

55

## Utilisation des LMs dans les entreprises

- Pour utiliser les LMs sur des tâches importantes dans les entreprises, il faudra
  - leur ajouter des connaissances liées à l'entreprise
  - les surveiller de près
  - vérifier les questions de l'utilisateur et les réponses du LM (pas facile...)

R. Grin

LM

page 56

56

## Conclusion

- Les LMs sont des outils qui peuvent être très efficaces aussi bien pour le développement que lorsqu'on les intègre à une application
- Cependant, il faut les utiliser avec prudence

R. Grin

LM

page 57

57

## Modèles de base

- Les LMs les plus connus deviennent tous multimodaux : on peut interagir avec eux non seulement avec du texte mais aussi avec des images, du son, et même des vidéos
- On parle alors de modèles de base (FM pour *Foundation Model*) puisque le modèle ne traite pas seulement de texte (Language) ; les FMs englobent les LMs

R. Grin

LM

58

58

## Agent de langage

- Un agent est un programme logiciel autonome qui peut percevoir son environnement, prendre des décisions basées sur ces perceptions et agir de manière autonome pour atteindre des objectifs spécifiques
- Un agent de langage est un agent avec lequel on peut interagir avec un langage naturel : on peut lui expliquer en langage naturel ce qu'il doit faire et quels types d'outils il peut utiliser pour cela
- Les agents de langage utilisent les LMs pour la compréhension du langage naturel et pour prendre des décisions

R. Grin

LM

59

59

## APIs des LMs

R. Grin

LM

60

60

10

## Standardisation ?

- Les APIs des LMs ne sont pas standardisées mais elles ont des points communs
  - API de type REST, avec échanges de données JSON (voir TP 1) ; le format JSON des requêtes et des réponses dépend du LM mais elles se ressemblent
  - sans état ; si le code que l'on écrit veut tenir une conversation suivie, il doit maintenir un état et le fournir à chaque nouvelle requête

R. Grin

LM

61

61

## Exemple

- Le code qui suit utilise les API de Gemini et d'OpenAI pour poser une question et obtenir une réponse
- Seul le code pour utiliser la clé secrète de l'API change (et aussi le code des méthodes associées au format JSON, qui ne sont pas données pour cet exemple)

R. Grin

LM

62

62

## Envoi de la requête

- Envoyer au endpoint de l'API du LLM une requête HTTP POST avec le JSON dans le corps de la requête
- Endpoint pour Gemini (la clé est passée en paramètre de l'URL) :
   
<https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent>
- Pour OpenAI (la clé est passée dans le header des requêtes) :
   
<https://api.openai.com/v1/chat/completions>

R. Grin

LM

63

63

## Connexion avec clé secrète - Gemini

```
// Récupération clé API
String key = System.getenv("GEMINI_KEY");
// URL endpoint avec la clé en paramètre
String url =
"https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent?key=" + key
// Création du client REST
Client clientRest = ClientBuilder.newBuilder();
// Prépare l'accès à la ressource REST avec l'URL
WebTarget target = clientRest.target(url);
// Le format JSON sera utilisé
Invocation.Builder request =
target.request(MediaType.APPLICATION_JSON_TYPE);
```

R. Grin

LM

64

64

## Connexion avec clé secrète - OpenAI

```
// Récupération clé API
String key = System.getenv("OPENAI_KEY");
// URL endpoint
String url = "https://api.openai.com/v1/chat/completions"
// Création du client REST
Client clientRest = ClientBuilder.newBuilder();
// Prépare l'accès à la ressource REST avec l'URL
WebTarget target = clientRest.target(url);
// Le format JSON sera utilisé
Invocation.Builder request =
target.request(MediaType.APPLICATION_JSON_TYPE);
// Clé dans header préfixée par « Bearer » (porteur)
request.header("Authorization", "Bearer " + chatgptKey);
```

R. Grin

LM

65

65

## Envoi requête

```
// Met la question au format JSON
String questionJSON = toJSONFormat(question);
// Met la question JSON dans entite (corps de la requête)
Entity<String> entite = Entity.entity(questionJSON,
MediaType.APPLICATION_JSON_TYPE);
// Envoie la question à l'API par requête POST
// et récupère la réponse
Response response = request.post(entite);
```

R. Grin

LM

66

66

## Traitement réponse

```
if (reponse.getStatus() == 200) { // Tout s'est bien passé
    // Récupère le corps de la réponse comme une String
    // au format JSON
    String reponseJson = reponse.readEntity(String.class);
    // Extrait le texte de la réponse à partir du JSON
    return extraitReponse(reponseJson);
} else { // Problème
    // On peut, par exemple, lever une exception
    throw new RequeteException(reponse.getStatus() + " : "
        + reponse.getStatusInfo());
}
```

R. Grin

LM

67

67

## Format JSON

- Les échanges de données entre le client et l'API utilisent le format JSON
- Ce format n'est pas standardisé

R. Grin

LM

68

68

## Format requête Gemini

```
{
    "contents": [
        {
            "role": "user",
            "parts": [ { "text": "Capitale du Maroc ?" } ]
        },
        {
            "role": "model",
            "parts": [ { "text": "Rabat" } ]
        },
        {
            "role": "user",
            "parts": [ { "text": "Et pour les US ?" } ]
        }
    ],
    ...
}
```

contents contient historique de la conversation (tableau JSON)

content pour la dernière question

R. Grin

LM

69

69

## Format requête OpenAI

```
{
    "model": "gpt-3.5-turbo",
    "messages": [
        {
            "role": "user",
            "content": "Capitale du Maroc ?"
        },
        {
            "role": "assistant",
            "content": "Rabat",
            "refusal": null
        },
        {
            "role": "user",
            "content": "Et pour les US ?"
        }
    ]
}
```

messages à la place de contents

R. Grin

LM

70

70

## Format réponse Gemini (1/2)

```
{
    "candidates": [
        {
            "content": {
                "parts": [ { "text": "La réponse ..." } ],
                "role": "model"
            },
            "finishReason": "STOP",
            "index": 0,
            "safetyRatings": [
                {
                    "category": "HARM_CATEGORY_HATE_SPEECH",
                    "probability": "NEGLIGIBLE"
                },
                ...
            ]
        }
    ]
}
```

R. Grin

LM

71

71

## Format réponse Gemini (2/2)

```
{
    "category": "HARM_CATEGORY_HATE_SPEECH",
    "probability": "NEGLIGIBLE"
},
...
],
"usageMetadata": {
    "promptTokenCount": 5,
    "candidatesTokenCount": 652,
    "totalTokenCount": 657
},
"modelVersion": "gemini-2.5-flash-001"
}
```

R. Grin

LM

72

72

## Format réponse OpenAI (1/2)

```
{
  "id": "chatcmpl-AOJuNe8Regmj92g3rK9UZKBHcd3Ln",
  "object": "chat.completion",
  "created": 1730359763,
  "model": "gpt-3.5-turbo-0125",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Washington DC",
        "refusal": null
      },
      "logprobs": null,
      "finish_reason": "stop"
    }
  ],
}
```

R. Grin

LM

73

73

## Format réponse OpenAI (2/2)

```
"usage": {
  "prompt_tokens": 78,
  "completion_tokens": 2,
  "total_tokens": 80,
  "prompt_tokens_details": {
    "cached_tokens": 0
  },
  "completion_tokens_details": {
    "reasoning_tokens": 0
  },
  "system_fingerprint": null
}
```

R. Grin

LM

74

74

## Rôles dans conversation

- user : l'utilisateur (pour une requête)
- model pour Gemini, assistant pour OpenAI : l'API (pour une réponse)
- Rôle système : ce rôle indique au LM comment il doit se comporter ; à placer au début d'une conversation ; la syntaxe dépend du LM :
  - Pour Gemini : system\_instructions
  - Pour OpenAI, plus simplement rôle « system »

R. Grin

LM

75

75

## Exemple rôle system - Gemini

```
{
  "system_instruction": {
    "parts": [
      { "text": "Tu parles comme une personne très timide" }
    ],
    {
      "contents": [
        {
          "role": "user",
          "parts": { "text": "Capitale du Maroc ?" }
        },
        {
          "role": "model",
          "parts": { "text": "..." }
        }
      ] } } }
```

R. Grin

LM

76

76

## Exemple rôle system - OpenAI

```
{
  "model": "gpt-3.5-turbo",
  "messages": [
    {
      "role": "system",
      "content": "You are a helpful assistant"
    },
  ]
}
```

R. Grin

LM

77

77

## LangChain

- Il est conseillé d'utiliser ce type de framework qui permet, entre autres, de faciliter l'utilisation des API des LMs et d'écrire du code indépendant du LM utilisé
- Framework Python open source pour faciliter l'utilisation des API des LMs
- LangChain4j est à l'origine une adaptation de LangChain à Java ; utilisé dans les TPs

R. Grin

LM

78

78

## Utilisation APIs des LMs

- 3 possibilités :
  - Directement avec l'API REST (TP 1)
  - SDK (Software Development Kit)
  - LangChain4j ou Spring AI (si application Spring)
- Le plus souvent les LMs offrent des SDKs standards en Python et JavaScript (et/ou TypeScript)
- Des produits offrent aussi un SDK standard en Java : OpenAI, Gemini, Anthropic par exemple
- Des librairies non standards ont aussi été développées

R. Grin LM 79

79

## Prompt engineering

R. Grin LM 80

80

## Prompt engineering

- Concevoir des requêtes efficaces pour interagir avec les LMs
- Plusieurs techniques
- **Attention**, ces techniques peuvent dépendre des LMs, et même des versions des LMs ; il faut tester

R. Grin LM 81

81

## Conseils généraux (1/5)

- Avant de passer aux techniques connues, voici quelques conseils généraux qu'il est souvent bon d'appliquer
- S'adapter à chaque cas particulier
- Prompt clair et précis ; éviter les ambiguïtés et les synonymes approximatifs
- Structurer les prompts (titres de paragraphes, listes à puces, numérotations, ...) ; éviter plusieurs demandes dans un seul prompt

R. Grin LM 82

82

## Conseils généraux (2/5)

- Donner le contexte :
  - Plutôt que de demander « Qu'est-ce que je pourrais manger ce soir à la maison ? »
  - Dire « Dans le réfrigérateur il y a ces légumes : .... et ces condiments : ...  
Je suis végétarien. Qu'est-ce que je pourrais manger ce soir à la maison ? »

R. Grin LM 83

83

## Conseils généraux (3/5)

- Définir le rôle du LM ; par exemple « Tu es un expert en cryptographie / Tu es un enseignant en .... »
- Définir son propre rôle ; « Je suis débutant en ... et je voudrais savoir ... »
- Définir le format de la réponse attendue ; « Donne-moi la réponse avec un format JSON / sous la forme d'une liste à puces »
- Préciser la réponse souhaitée ; « Donne-moi une réponse simple et courte », « Ne donne que le JSON, et rien d'autre »

R. Grin LM 84

84

## Conseils généraux (4/5)

- Si le LM semble perdu, découper la question en plusieurs sous-questions
- Si une question porte sur une réponse précédente du LM, bien préciser sur quelle partie de la réponse porte la nouvelle question ; « Dans la partie numérotée 3 de ta réponse, peux-tu me préciser ... »
- Demander des exemples concrets ; « La procédure que tu viens d'expliquer est complexe ; peux-tu me donner un exemple concret ? »

R. Grin

LM

85

85

## Conseils généraux (5/5)

- Un LLM ne dit jamais qu'il n'a pas compris ou qu'il ne peut pas répondre correctement à une question  
On peut lui dire
  - « Si tu ne comprends pas bien ma question ou si tu as besoin d'informations complémentaires, demande-moi. »
  - « Si tu ne sais pas répondre, dis-le »

R. Grin

LM

86

86

## Exemple avec question

- User : Je voudrais calculer le volume d'un cube que j'ai chez moi. Peux-tu m'aider ? Si tu as besoin d'informations complémentaires, demande-moi.
- LM : Pour calculer le volume de ton cube, j'ai besoin de connaître la **longueur de l'une de ses arêtes**.  
Peux-tu me donner cette mesure en centimètres, mètres, ou dans une autre unité de ton choix ?
- User : 30 cm
- LM : Le volume de ton cube avec une arête de 30 cm est de 27 000 cm<sup>3</sup>.

R. Grin

LM

87

87

## Techniques de prompts (1/4)

- **Zero shot prompting** : pour les questions simples, on peut juste poser la question, sans donner d'exemple
  - Plus simple et un peu moins coûteux
  - Mais souvent moins efficace que les techniques qui suivent pour les questions plus complexes

R. Grin

LM

88

88

## Techniques de prompts (2/4)

- Un LM a la capacité de faire de l'apprentissage contextuel (in-context learning) : il apprend à partir d'exemples fournis dans le prompt ; les paramètres du LM ne sont pas modifiés
- **Few-shot prompting** : donner quelques exemples de questions-réponses et poser ensuite la question en suivant le format des exemples :

Traduire en anglais :  
 loutre de mer => sea otter  
 girafe en peluche => plush giraffe  
 fromage =>

La réponse du LM :  
 fromage => cheese

R. Grin

LM

89

89

## Techniques de prompts (3/4)

- **Chain-of-Thought prompting (CoT)** :
  - Le but est d'inciter le LLM à donner les étapes à suivre pour arriver à la réponse finale (la chaîne de pensée) ; peut aider le LM à donner une meilleure réponse et l'utilisateur à mieux comprendre et vérifier la réponse
  - Pour cela on peut demander explicitement au LLM de donner les étapes à suivre : « Réfléchis étape par étape avant de donner ta réponse. »
  - On peut aussi donner dans le prompt un exemple de ce qu'on attend ou des réponses intermédiaires qu'il doit donner (few-shot + CoT)
  - Intéressant lorsque la réponse contient un raisonnement

R. Grin

LM

90

90

## Exemple de CoT

- User :** Un paysan doit traverser une rivière avec un tigre, un chat et une souris. Il ne peut les transporter qu'un seul à la fois. S'il laisse le tigre avec le chat, le tigre le mangera. S'il laisse le chat avec la souris, le chat la mangera.  
Comment peut-il tous les transporter en sécurité ? Donne les différentes étapes en les expliquant.
- LLM :** Le paysan doit planifier ses allers-retours pour éviter que les animaux ne se mangent. Il peut transporter une seule chose à la fois, le voyage se faisant en 4 étapes.
  - Étape 1 :** Le paysan prend le **chat** 🐱 et traverse la rivière. Il le dépose de l'autre côté de la rive et revient seul.
    - Sur la rive de départ, il ne reste que le tigre 🐆 et la souris 🐂, qui ne se présentent pas de danger l'un pour l'autre.
  - Étape 2 :** Le paysan prend le **tigre** 🐆 et traverse. Il le dépose de l'autre côté et prend le **chat** 🐱 pour le ramener sur la rive de départ.
    - Il laisse le tigre 🐆 sur la rive d'arrivée et ramène le chat 🐱 avec la souris 🐂 sur la rive de départ. Le tigre ne risque rien seul.
  - ...

R. Grin

LM

91

91

## Exemple de CoT avec few-shot

- User :** La somme des nombres impairs de ce groupe donne un nombre pair : 4, 8, 9, 15, 12, 2, 1.  
R : L'addition de tous les nombres impairs (9, 15, 1) donne 25. La réponse est Faux.  
La somme des nombres impairs de ce groupe donne un nombre pair : 15, 32, 5, 13, 82, 7.  
R :
- LLM :**
  - Somme :
    - $15 + 5 = 20$
    - $20 + 13 = 33$
    - $33 + 7 = 40$
  - La somme est 40, qui est **pair**.
  - R : Vrai**

R. Grin

LM

92

92

## Techniques de prompts (4/4)

- Prompt chaining :** l'utilisateur décompose sa question en plusieurs prompts ; après le 1<sup>er</sup> prompt, il pose la question suivante en utilisant le résultat à la 1<sup>ère</sup> question, et ainsi de suite
- Il existe de nombreuses autres techniques de prompt ; des articles de recherche sortent régulièrement pour en décrire de nouvelles

R. Grin

LM

93

93

## Injection de prompts

- Des instructions indésirables ou malveillantes sont insérées dans un prompt, dans le but de manipuler ou détourner son comportement
- Souvent ces injections essaient de faire révéler les informations privées sur l'entreprise ou le LM
- Les LLMs les plus évolués se protègent de mieux en mieux contre les injections de prompt mais il faut faire confiance à l'imagination des hackers pour détourner ces protections
- Il n'y a malheureusement pas de solution toute faite pour éviter l'injection de prompt et il faut être vigilant

R. Grin

LM

94

94

## Exemples d'injection de prompt

- En 2023, un étudiant découvre une méthode permettant de contourner les protections de Bing Chat (Microsoft), afin d'obtenir des directives internes de l'IA
- Demandez des informations privées sur lesquelles le LM a été entraîné (fine-tuning) ou qu'il a obtenues par RAG

R. Grin

LM

95

95

## Context engineering

- Avec les techniques les plus récentes de l'IA (exposés dans la suite du cours), on demande plus que du simple prompt engineering
- La notion de contexte est élargie à tout ce dont le LM dispose pour répondre aux questions :
  - ressources supplémentaires (RAG) et outils pour interagir avec l'environnement
  - mémoires à court et à long terme

R. Grin

LM

96

96

## LM local

R. Grin

LM

97

97

## LM local

- Installer un LM localement permet de ne pas avoir à payer pour accéder aux APIs des LMs
- Les réponses ne seront pas rapides si la machine n'est pas assez puissante (en particulier si elle n'a pas de GPUs ou de NPUs)
- Les LMs nécessitent un minimum de RAM pour fonctionner ; 16 Go, à la rigueur 8 Go pour les très petits modèles ; pour les grands modèles il faut au moins 32 à 64 Go

R. Grin

LM

98

98

## Installer LM local (llama2)

- Ollama peut s'installer localement et il est gratuit
- Fichier OllamaSetup.exe à télécharger depuis <https://ollama.com/download> et à lancer pour l'installation (rapide)
- Après l'installation, une fenêtre Windows s'affiche ; taper « ollama run llama2 » (pour démarrer le LM llama2) ; attendre 2 ou 3 minutes, et on peut commencer à poser des questions dans la fenêtre Windows
- On peut utiliser divers LMs : LLama3, Phi 3, Mistral, Gemma (voir <https://ollama.com/library>), ...

R. Grin

LM

99

99

## Utiliser API de llama2 avec LangChain4j (1/2)

- Dépendance Maven à ajouter :

```
<dependency>
    <groupId>dev.langchain4j</groupId>
    <artifactId>langchain4j-ollama</artifactId>
    <version>0.27.1</version>
</dependency>
```

LangChain4j étudié dans support suivant

R. Grin

LM

100

100

## Utiliser API de llama2 avec LangChain4j (2/2)

```
ChatLanguageModel model = OllamaChatModel.builder()
    .baseUrl("http://localhost:11434")
    .modelName("llama2")
    .build();

String answer = model.generate("List all the movies directed by Quentin Tarantino");
System.out.println(answer);
```

R. Grin

LM

101

101

## Avec container Docker (1/2)

- Testcontainers est une librairie Java qui peut fournir des instances légères et jetables de n'importe quel conteneur Docker. LangChain4j fournit plusieurs images Docker de Ollama avec différents LMs
- 

```
<dependency>
    <groupId>org.testcontainers</groupId>
    <artifactId>testcontainers</artifactId>
    <version>1.19.4</version>
    <scope>test</scope>
</dependency>
```

R. Grin

LM

102

102

## Avec container Docker (2/2)

```
GenericContainer<?> ollama =
    new GenericContainer<>("langchain4j/ollama-"
        + llama2 + ":latest")
    .withExposedPorts(11434);
ollama.start();

String baseUrl = String.format("http://%s:%d",
    ollama.getHost(), ollama.getFirstMappedPort());
ChatLanguageModel model = OllamaChatModel.builder()
    .baseUrl(baseUrl).modelName(MODEL_NAME).build();
String answer = model.generate("List all the movies directed
by Quentin Tarantino");
System.out.println(answer);

ollama.stop();
```

R. Grin

LM

103

103

## Outils pour tester et travailler avec LMs

R. Grin

LM

104

104

## Hugging Faces

- Plateforme communautaire axée sur le développement de modèles d'IA ; <https://huggingface.co/>
- Propose des outils et des bibliothèques pour la création, l'entraînement et le déploiement de modèles de ML :
  - Bibliothèque Transformers pour accès simplifié aux modèles d'IA et utilisation, ajustement de modèles déjà optimisés
  - Bibliothèque de datasets pour entraîner ou évaluer ses modèles
  - Partage de modèles et d'applications avec dépôt centralisé
  - Fine-tuning de modèles sur ses propres données
  - Fournit API indépendante des modèles

R. Grin

LM

105

105

## Code pour récupérer un modèle

```
HuggingFaceChatModel model =
    HuggingFaceChatModel.builder()
        .accessToken(System.getenv("HF_API_KEY"))
        .modelId("mistralai/Mistral-7B-Instruct-v0.2")
        .timeout(Duration.ofSeconds(15))
        .temperature(0.7)
        .maxNewTokens(20)
        .waitForModel(true)
        .build();
```

R. Grin

LM

106

106

## Ollama

- <https://ollama.com/>
- Gestion et utilisation locale de LMs (Hugging Face permet de travailler en local ou dans le cloud)
- Avantages
  - Installation et gestion simplifiée des modèles
  - Fournit une API commune aux LMs

R. Grin

LM

107

107

## Références

R. Grin

LM

108

108

## Prompt engineering

- Fiche métier pour prompt engineer : <https://www.studyrama.com/formations/fiches-metiers/informatique-electronique-numerique/prompt-engineer>
- Techniques pour les prompts : <https://reglo.ai/techniques-de-prompt-engineering/>, <https://www.promptingguide.ai/>
- Guide OpenAI : <https://platform.openai.com/docs/guides/prompt-engineering/six-strategies-for-getting-better-results>
- Guide Google : <https://ai.google.dev/gemini-api/docs/prompting-strategies?hl=fr>
- Diverses techniques : <https://learnprompting.org/docs/introduction>

R. Grin

LM

109

109

## Liens divers

- Tableaux de classement pour les modèles open source :
  - <https://huggingface.co/open-llm-leaderboard>
- Modèles d'embeddings :
  - <https://huggingface.co/spaces/mteb/leaderboard>
- Cours rapides sur LMs :
  - [www.deeplearning.ai/short-courses](https://www.deeplearning.ai/short-courses)
- Divers :
  - [www.pinecone.io/learn](https://www.pinecone.io/learn)

R. Grin

LM

110

110

## IA avec Java

- « AI for Java developers » par Microsoft :
 <https://www.youtube.com/watch?v=V45tKEYYAFs&list=PLPeZXICR7ew8sdUWqf2itkRG5BUE7GFsy>
- TensorFlow pour Java :
 <https://www.baeldung.com/tensorflow-java>  
<https://www.tensorflow.org/jvm/install?hl=fr>
- Tensorflow et Keras pour Java :
 <https://github.com/dhruvrajan/tensorflow-keras-java>  
 Deeplearning4j, <https://deeplearning4j.konduit.ai/>
- Playlist YouTube sur IA en Java :
 <https://www.youtube.com/playlist?list=PLZOgUaAUCiT7ooFAUWId7oeWatv6b3oCD>

R. Grin

LM

111

111

## API d'OpenAI

- OpenAI : <https://openai.com>
- API de OpenAI : <https://platform.openai.com/>
- Documentation sur l'API :
 <https://platform.openai.com/docs/api-reference/chat>
- Guide pour utiliser l'API de compléTION :
 <https://platform.openai.com/docs/guides/text-generation/chat-completions-api>

R. Grin

LM

112

112

## API de Gemini

- Formats des requêtes et réponses pour travailler directement avec l'API REST :
  - <https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/inference?hl=fr#parameters> (avec hyperparamètres et autre syntaxe)
  - <https://github.com/google/generative-ai-docs/blob/main/site/en/gemini-api/docs/get-started/rest.ipynb>
- Référence API : <https://ai.google.dev/api>
- Documentation sur l'API :
 <https://ai.google.dev/gemini-api/docs>
- Guide pour utiliser l'API de compléTION :
 <https://ai.google.dev/gemini-api/docs/text-generation>

R. Grin

LM

113

113