

Nouveau

Formation Fine-tuning et agents IA : spécialiser un LLM et l'intégrer dans vos agents

Réf. SF34746

Apprenez à spécialiser un modèle open source pour vos usages métier et à en faire le moteur fiable et économique d'un agent IA.

2 j
Durée

1 600 € HT
Inter / participant

0
Sessions à venir

Objectifs pédagogiques

- ✓ Choisir, pour un cas d'usage agentique, entre prompt engineering, modèle généraliste et modèle spécialisé par fine-tuning
- ✓ Développer un agent LLM capable de planifier et d'appeler des outils externes (function calling, MCP)
- ✓ Préparer un jeu de données d'instructions et d'appels d'outils et fine-tuner un modèle open source avec LoRA et QLoRA
- ✓ Intégrer un modèle spécialisé dans un agent et le comparer à un modèle généraliste
- ✓ Évaluer un système agentique : fiabilité des appels d'outils, qualité des réponses, coût et latence

Public visé & prérequis

Public visé Data scientists, Développeurs, Ingénieurs Big Data

Prérequis- Pratique de Python (niveau intermédiaire) - Bases du machine learning : jeux d'entraînement et de validation, sur-apprentissage, métriques - Avoir déjà utilisé un LLM par API ; notions de JSON et d'API REST

Programme détaillé

Agents LLM et spécialisation de modèles : poser le cadre

- Agent ou chatbot : boucle agentique, degrés d'autonomie, du prompt unique à l'agent outillé
- Design patterns agentiques : utilisation d'outils, planification, réflexion, orchestration multi-agents
- Fine-tuning d'un LLM : principe, bénéfices, limites, fine-tuning complet ou adaptation efficace des paramètres
- Ce qu'un modèle spécialisé apporte à un agent : appels d'outils fiables, sorties structurées, routage, vocabulaire et règles métier
- Critères de décision : prompt engineering et exemples d'abord, fine-tuning quand le gain en qualité, coût ou latence est mesurable
- Choisir un modèle open source : taille, licence, fenêtre de contexte, prise en charge de l'appel d'outils

Activités pratiques :

- Comparaison d'un prompt direct et d'un workflow agentique sur une même tâche métier
- Construction d'une grille de décision « prompt, modèle généraliste ou modèle spécialisé » pour plusieurs cas d'usage

Concevoir un agent outillé : function calling, MCP et planification

- Function calling : déclaration d'outils par schéma (nom, paramètres, description) et sorties JSON structurées
- MCP (Model Context Protocol) : connecter l'agent à des services et à des sources de données de façon standardisée
- Pattern ReAct : alterner raisonnement, action et observation jusqu'à la réponse
- Pattern Plan & Execute : décomposer une demande complexe en étapes et les exécuter
- Réflexion et auto-correction : relire et corriger une sortie avant de la rendre
- État et mémoire de l'agent ; recherche documentaire exposée comme un outil parmi d'autres
- Ouverture vers les systèmes multi-agents : quand un superviseur et des agents spécialisés se justifient

Activités pratiques :

- Transformation de fonctions Python en outils utilisables par un agent

- Développement d'un agent de support interne (consultation d'un dossier, création de ticket, escalade) avec un petit modèle open source
- Relevé des erreurs de l'agent : outil mal choisi, arguments invalides, format non respecté

Constituer un jeu de données d'instructions et d'appels d'outils

- Format conversationnel et chat templates : structurer messages, appels d'outils et résultats d'outils pour l'entraînement
- Sources de données : historiques anonymisés, annotation manuelle, exemples rédigés par les experts métier
- Données synthétiques : générer des exemples avec un modèle plus puissant, puis les filtrer et les relire
- Qualité avant volume : nettoyage, déduplication, équilibrage des cas, exemples négatifs où aucun outil ne doit être appelé
- Découpage entraînement, validation et test ; jeu d'évaluation figé pour comparer les modèles
- Données personnelles et confidentialité : anonymisation, droits d'usage des données d'entraînement

Activités pratiques :

- Constitution d'un jeu d'exemples d'appels d'outils à partir des erreurs relevées sur l'agent de référence
- Génération et filtrage d'exemples synthétiques avec un modèle généraliste

Fine-tuner un modèle open source avec LoRA et QLoRA

- Supervised fine-tuning (SFT) avec l'écosystème Hugging Face : Transformers, Datasets, PEFT et TRL
- LoRA : adapter un modèle en entraînant un petit nombre de paramètres (rang, modules ciblés)
- QLoRA : quantification 4 bits pour fine-tuner sur un seul GPU ; estimer les besoins mémoire
- Hyperparamètres essentiels : taux d'apprentissage, nombre d'époques, taille de lot, longueur de séquence
- Suivi de l'entraînement : courbes de perte, détection du sur-apprentissage, points de contrôle
- Adaptateurs : sauvegarde, fusion avec le modèle de base, versionnement des données et des modèles pour des expériences reproductibles

Activités pratiques :

- Fine-tuning QLoRA du modèle de l'agent sur le jeu de données d'appels d'outils
- Comparaison de deux configurations d'entraînement et choix argumenté de la meilleure

Intégrer le modèle spécialisé dans l'agent

- Exposer le modèle fine-tuné derrière une API compatible OpenAI avec un serveur d'inférence
- Brancher le modèle spécialisé dans l'agent à la place du modèle de base, sans réécrire la logique de l'agent
- Robustesse : valider les appels d'outils, gérer les sorties invalides et les erreurs d'outils, prévoir des relances
- Architecture mixte : un modèle généraliste pour planifier, un petit modèle spécialisé pour une tâche étroite et fréquente

Activités pratiques :

- Bascule de l'agent de support sur le modèle fine-tuné
- Mise en place d'une architecture mixte modèle généraliste / modèle spécialisé

Évaluer et fiabiliser le système agentique

- Évaluer le modèle fine-tuné : validité des appels d'outils, exactitude des arguments, non-régression sur les capacités générales
- Évaluer l'agent : evals composant par composant et de bout en bout, métriques de réussite des tâches
- LLM-as-a-judge : noter automatiquement la qualité des réponses, et ses limites
- Analyse d'erreurs : identifier ce qui relève du modèle, des outils ou de la conception de l'agent
- Coût et latence : comparer modèle spécialisé et modèle généraliste sur un même volume de requêtes
- Garde-fous : permissions des outils, injection de prompt via les données d'outils, validation humaine des actions sensibles
- Bonnes pratiques et enjeux éthiques de l'adaptation des modèles

Activités pratiques :

- Banc d'évaluation comparant l'agent sur modèle de base, sur modèle fine-tuné et sur modèle généraliste par API
- Analyse d'erreurs et recommandation argumentée sur le modèle à retenir pour la mise en production

Pourquoi choisir Sparks Formation ?

20 ans d'expertise IT, des formateurs experts et des sessions garanties contre l'annulation.

20 ans
d'expertise

1 068+
formations

11 178
apprenants formés/an

+67
Net Promoter Score

Avis des participants

4,9/5 — 190 avis

Anne-Lise C. — Très bonne adaptation du formateur au public et au niveau d'attente.

Marie G. — La formatrice était très pédagogue et bienveillante ! Elle a su nous partager sa passion et donner envie de pratiquer à la suite Voilà bien longtemps que je...

Ils nous font confiance

Cette année notre filière Intelligence artificielle & Data Science a accompagné plus de 544 entreprises dont :

SOPRA STERIA THALES GLOBAL SERVICES ARTELIA AXA ASSISTANCE FRANCE GRANT THORNTON FRANCE AKKODIS DIGITAL SAS
MEDIABRANDS CEGEDIM COVAGE INFRA SNCF - E.SNCF SOLUTIONS +534 autres

Financement & certifications

Cette formation est éligible aux financements OPCO et FNE-Formation. Nos équipes vous accompagnent dans le montage de votre dossier.



Réservez votre place dès maintenant

20 ans d'expertise · 1 000+ formations · NPS +67 · réponse sous 24h ouvrées

0 805 950 800

demande@sparks-formation.com