

Article

Faire passer l'agrégation de Grammaire à un système Agentique : Le cas de l'exercice d'histoire phonétique

Wainstain Tom
UFR Linguistique
Université Paris Cité
Paris, France
tom.wainstain@etu.u-paris.fr

Triantafyllou Filippos
UFR Linguistique
Université Paris Cité
Paris, France
filippos.triantafyllou@etu.u-paris.fr

Marly-Léger Rose
UFR Linguistique
Université Paris Cité
Paris, France
rose.marly-leger@etu.u-paris.fr

Abstract—L'épreuve de phonétique de l'agrégation consiste à décrire les changements d'un mot latin jusqu'à son reflet français. Peu d'études ayant automatisé ce processus par des moyens computationnels, nous mettons en place une architecture de système agentique basé sur un LLM (Llama 3.1 8B) afin de répondre au mieux à cette tâche, le comparons à un modèle identique (sans amélioration agentique) ainsi qu'à un modèle plus gros (GPT-5.2). Notre agent produit des résultats légèrement meilleurs que sa version sans outils, mais reste loin d'une réponse idéale, et peine à battre un modèle plus récent. Nous concluons par la possibilité d'un Reinforcement Learning ou l'intégration d'un outil basé sur des règles phonologiques, réduisant l'importance du LLM au sein du système.

Mot-Clés—Agentic, AI, RAG, LLM, Pipeline, Computer Science, Linguistics, Phonetics, Phonology, Diachrony, Latine, French

I. INTRODUCTION ET ÉTAT DE L'ART

A. Introduction

De tout temps l'homme a voulu passer l'agreg' de grammaire sans apprendre par cœur des règles phonologiques. L'exercice de phonétique diachronique consiste à identifier et décrire l'évolution d'un mot latin jusqu'au français

moderne. La tâche est d'autant plus compliquée en français, par rapport aux autres langues romanes, par la diversité des voyelles aujourd'hui présentes dans la langue, malgré la disparition (exceptée en Suisse¹), de la distinction de longueur. Cette tâche est pourtant largement résolue et documentée par l'état actuel de la linguistique historique. Son automatisation reste cependant peu étudiée (Section II), et c'est cette épreuve que notre étude tentera d'accomplir, par le moyen d'un LLM (*Large Language Model*) amélioré d'outils et d'une architecture appropriée (Section III).

B. Définition du problème

Nous voulons qu'à partir d'un mot latin et son reflet en français moderne, notre agent puisse générer, selon les consignes de l'agrégation, un tableau donnant chaque forme intermédiaire, l'époque correspondante et le ou les phénomènes linguistiques en jeu.

Le but de notre projet est de concevoir une implémentation d'un LLM relativement petit, *Llama 3.1* ("seulement" 8 milliards de paramètres), accompagné d'une suite d'outils pour constituer une IA agentique, qui soit capable de produire des résultats équivalents ou supérieurs

¹Le dialecte français de suisse distingue *faites* [fɛt] de *fête* [fe:t].

à ceux d'un modèle récent et immense (*GPT-5.2*), mais sans outil.

C. État de l'art

Les modèles d'IA agentiques sont issus des développements récents au sein de l'étude des LLMs, et commencent donc à peine à se développer. Il n'existe actuellement que très peu de recherches sur l'usage d'une IA, agentique ou non, pour étudier et décrire l'évolution d'un mot au fil du temps (par analyse *diachronique*). On trouve, à l'inverse de notre sujet d'étude, l'usage de LLMs pour dater des pièces de théâtres françaises (Romanello et al., 2025) ou une étude sur l'influence des LLM dans l'évolution diachronique de l'anglais (Anderson et al., 2025). Des modèles agentiques ont pu être utilisés pour modéliser de la variation linguistique selon des critères sociaux (Cai et al., 2024), enseigner une langue (Pan et al., 2025) ou, plus proche de notre objectif, pour identifier les facteurs moteurs du changement linguistique (Dekker, 2024). Mais notre travail en tant que tel reste inédit, en partie dû à l'originalité de notre tâche.

La littérature linguistique sur l'évolution du latin au français moderne, est elle bien plus florissante. De nombreux modèles phonologiques existent: de la *Sound Pattern of English* de Chomsky et Halle (Chomsky, 1968), à la théorie autosegmentale de Goldsmith (Goldsmith, 1976) ou la théorie de l'optimalité (Violin-Wigent, 2006). Que cela soit en traits distinctifs ou en phonèmes, on pourrait, à l'aide des deux précis de référence du sujet (Joly, 2003) (Laborderie, 2015), concevoir une computation successive des formes du latin classique au français moderne, par exemple sous forme de règles et candidats, triées selon la théorie de l'optimalité (Roberts, 2012) et (Marr, 2017), ou par une grammaire formelle non-contextuelle.

L'enjeu du modèle génératif que nous tenterons de construire est de dépasser les limitations des modèles basés sur des règles, qui nécessitent une description exhaustive et détaillée de chaque phénomène, en comptant sur l'agentivité et les connaissances du modèle pour choisir la voie à suivre.

II. ARCHITECTURE DE L'AGENT

A. Architecture conceptuelle globale

D'abord, présentons l'architecture globale que nous avons fini par mettre au point. Comme on le voit dans la Fig. 2, l'agent suit une logique que nous avons arbitrairement déterminée en essayant de reproduire un cheminement de pensée humain face à une copie d'examen.

À savoir, une compréhension des termes clés de la question, leur lien avec des concepts clés du domaine, puis la mobilisation de connaissances. L'objectif étant de produire une réponse formellement semblable à celle de la Fig. 1, choisie parmi les corrections proposées par le jury.

II ^e -III ^e	[kénèrə]	Bouleversement vocalique (mutation vocalique d'aperture). Les oppositions de quantité du latin classique sont remplacées par des oppositions d'aperture en latin vulgaire. Les voyelles brèves s'ouvrent et les voyelles longues se ferment. Ici, [i] s'ouvre en [ɛ] et [ɛ] en [e]. Le bouleversement accentuel, qui entraîne l'allongement des voyelles toniques par le passage d'un accent de hauteur à un accent d'intensité, a lieu parallèlement.
III ^e	[t̪_séndrə]	Palatalisation de [k] + [e] en position forte : - renforcement de l'articulation ; mouvement vertical d'élévation vers la voûte palatine : la langue s'élève et vient s'étaler largement contre le palais : [k] > [k̪]. - dentalisation ; déplacement de point d'articulation (mouvement horizontal vers l'avant) : le phonème avance jusqu'à passer dans la zone dentale : [k] > [t̪]. - assibilation : la siffante [s] apparaît dans une phase de détente après la tenue ; elle se combine avec la dentale pour former une consonne affriguée palatale : [t̪] > [t̪_s] Amuïssement de la voyelle pénultième atone. Alors, la mise en contact de la nasale [n] et de la liquide [r] provoque l'apparition d'une consonne épenthétique sonore. La voyelle tonique, désormais entravée, ne pourra pas se diptonguer.
VII ^e	[tséndrə]	Dépalatalisation de l'affriguée initiale.
VII ^e	[tséndrə]	Dépalatalisation de l'affriguée initiale. La voyelle finale autre que [a] ne s'amüit pas mais s'affaiblit en [e] de soutien au groupe consonantique conjoint [dr].
XI ^e	[tséndrə] [tsándrə]	Nasalisation de la voyelle orale [é] devant la consonne nasale [n] par abaissement anticipé du voile du palais : en prévision de la prononciation de la consonne nasale, le voile du palais s'abaisse trop tôt (avant la fin de la tenue vocalique) et la voyelle orale prend une résonance nasale du fait de l'air qui circule dans les fosses nasales. La consonne nasale reste articulée.
		Vers la fin du même siècle, ouverture de la voyelle nasalisée : [é̄n] > [án]. Prononcer une voyelle nasalisée suppose de relâcher les muscles du voile du palais, ce qui entraîne une décontraction des muscles élévateurs de la langue et une action des muscles abaisseurs ; cela provoque donc un abaissement de la langue, d'où une ouverture de la voyelle.
XIII ^e	[sándrə]	Réduction de l'affriguée par amuïssement de son élément occlusif.
XV ^e	[sándrə]	Labialisation de [e] central en [œ].
vers 1600	[sádroə]	Dénasalisation partielle : la nasalisation a produit deux sons nasalisés (une voyelle et une consonne) ; toutefois, on s'accorde mal de devoir articuler deux nasales successives et, pendant l'époque classique, se produit une dissimilation au détriment du phonème le plus faible ; en syllabe fermée, la consonne (implosive) s'amüit et la voyelle reste nasalisée. L'ancienne consonne nasale se conserve dans la graphie comme signe diacritique pour signaler le timbre nasal de la voyelle.
XVIII ^e	[sádR]	Le [r] apico-alvéolaire se transforme en un [R] dorso-uvulaire. La voyelle finale devient caduque : elle n'est plus prononcée que comme voyelle de soutien dans un groupe de mots (« la cendre du foyer », « la cendre devient poussière »).

Fig 1. Proposition de corrigé de la session 2021, pour la question : "Retracer l'évolution, du latin au français moderne, de cendre (v. 691) < CINEREM."

B. Architecture détaillée

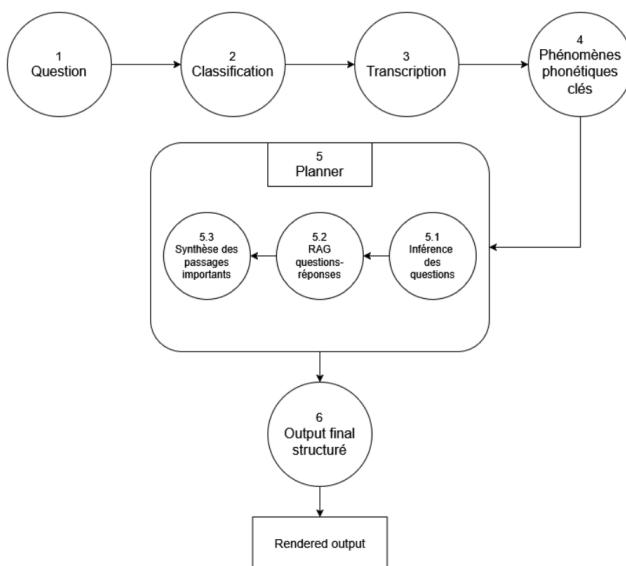


Fig 2. Architecture de l'Agent

Voici, pour chacun des 8 nœuds du graphe de la Fig. 2, leur fonction détaillée. Notre code est disponible sur GitHub suivant le lien suivant: <https://github.com/phtryll/agreg-agent>.

1 - Le simple input, une question de la forme : “Retracer, du latin jusqu’au français moderne, l’histoire phonétique de **X < Y**”, ou “**X** (éymon : **Y**)”, avec **X** un mot en français moderne et **Y** son étymon latin.

2 - À ne pas confondre avec un modèle dit ‘classifieur’ en machine learning, notre étape de classification correspond simplement à identifier les éléments clés du sujet. Dans notre cas, à identifier lequel des deux mots est en français moderne et lequel est son étymon latin. C’est bien sûr une étape primordiale, mais très largement réussie car la différence entre les mots est en quelque sorte pré-formatée par le sujet.

3 - L’agent transcrit **X** et **Y** dans le but de les étudier sur le plan phonétique. Le modèle étant bien sûr entraîné sur un langage textuel, il est connu qu’ il a beaucoup de mal à effectuer cette ‘traduction’ de l’écrit vers l’oral. Il s’agit donc de l’une des étapes les moins bien réussies, ce qui cause la plupart du temps un effet boule de neige sur la suite des étapes, particulièrement pour la n°4. De meilleures implémentations pour la transcription ont été imaginées mais pas

implémentées, faute de temps (voire Section IV - Discussion)

4 - L’autre étape la plus complexe pour le modèle : il doit chercher, ayant dans le prompt une liste de phénomènes donnés, lesquels d’entre eux semblent avoir eu lieu lors de l’évolution phonétique de **Y** à **X**. Par sa dépendance à la phonétique, elle est quasiment certaine de rater si la transcription n’est pas bonne. Cette étape correspondrait pour un candidat à la mobilisation de concepts phonétiques clés.

5 - Ici, le mot “planner” est utilisé pour désigner le processus qui va reproduire la mobilisation des connaissances, et qui va donc créer un plan de réponse nécessaire à la résolution de chaque phénomène phonétique. Pour cela, il procède en trois temps :

5.1 - Un modèle génératif va transformer chaque concept phonétique en posant le problème du concept sous forme de question.

5.2 - Un modèle CrossEncoder est utilisé pour chercher dans la base de données les morceaux de textes (chunks) les plus pertinents pour répondre à la question. Ces modèles de langage sont entraînés sur des paires [question, réponse] et attribuent un score censé représenter la qualité de la réponse vis-à-vis de la question.

5.3 - Le modèle génératif écrit un résumé des données mobilisées pour répondre à la question. Les étapes 5.(1,2,3) sont répétées jusqu’à ce qu’il n’y ait plus de questions.

6 - Enfin, un output structuré est produit générativement pour répondre à la question. L’intégrité de sa structure est vérifiée : si elle n’est pas conforme, l’agent renvoie simplement le log de toutes les étapes passées, faute de réponse convenablement transmise.

L’output structuré est ensuite déterministiquement (sans l’usage de modèle ni de prompt) transformé en une suite d’étapes d’évolution phonétique telle qu’on en trouve dans les corrections proposées par le jury du concours d’agrégation de Grammaire.

III. RÉPONSES DE RÉFÉRENCE ET ÉVALUATIONS

Nous avons généré 4 réponses de référence (*baselines*), seuils plus accessibles que la correction idéale du jury de l'agrégation. Nous avons généré 2 réponses avec le LLM *GPT-5*, modèle massif très puissant, l'une avec un simple prompt de la consigne, le second avec une prompt détaillant le contenu et la forme désirée.

En parallèle et avec les mêmes prompts, nous avons généré 2 réponses avec *LLama 3.1 8B*, le modèle relativement petit que nous avons ensuite intégré au sein de notre agent. Nous souhaitions donc vivement un progrès entre les résultats de notre agent et ceux du modèle “à nu”, et visions de dépasser la force “brute” de *GPT-5* par notre outillage et notre pipeline.

Cette analyse qualitative est performée par des auteur·ice·s formés à la linguistique, mais loin d'être des experts de la phonétique historique. Nous nous baserons donc principalement sur une comparaison avec la réponse attendue du jury.

A. GPT-5.2

1) Simple prompt

Le modèle propose, en résumé, l'évolution suivante:

mansionem
 → *mansione*
 → **masjone*
 → *maison / maisun*
 → *maison*

Soit un total de 5 étapes, contre les 16 attendues, et ni *mansione* ni **masjone* ne figurent dans les étapes de l'évolution du mot.

L'astérisque indiquant une forme reconstituée est une pratique courante en linguistique diachronique, mais ici, dans le cadre d'une évolution phonétique pas à pas au fil des siècles, elle est inappropriée et témoigne de l'incohérence du propos.

Sans consigne supplémentaire, le modèle donne très peu de dates, plaçant uniquement l'ancien français entre les IX^e et XII^e siècle (ce qui est trop réducteur).

Est également notable la disparition, sans aucune justification, du /n/ en latin tardif, erreur

visiblement commise par de nombreux candidat·e·s cette année-là. Le terme “simplification” pour désigner le passage de /ʃ/ à /s/, est lui plus original, mais tout aussi incorrect. Les termes de *palatalisation* et *sonorisation* sont cependant correctement utilisés.

2) Prompt contextualisé

GPT propose cette fois 8 étapes et sépare clairement orthographe et phonologie:

- /man.si 'o:.nem/
- /man.sjo 'ne/
- /man 'sjo.ne/
- /man 'sjo.ne/
- /maj 'zõn/
- /mɛ 'zõn/
- /mɛ 'zð/
- /mɛ.zð/

Malheureusement, en mélangeant la chronologie des différentes mutations, les seules formes entièrement correctes que le modèle nous propose sont les étapes initiales et finales, soit celles déjà renseignées par la consigne.

Le modèle propose de plus nombreuses étapes, même si l'on est encore loin du corrigé, mais elles semblent surtout arbitraires, voire purement hallucinées (“Roman commun”). Il qualifie également la chute du /n/ (encore une fois bien trop tardive par ailleurs), d'une nasalisation progressive.

L'ellipse injustifiée du VI^e au IX^e siècle a permis la chute sans aucune justification du /e/ final. *GPT* justifie, par la chute du premier /n/, la nasalisation de la voyelle “précédente”... quand elle se trouve en réalité plus loin.

B. *Llama 3.1*

1) Simple prompt

La réponse intégrale du modèle est la suivante:

Mansionem est la forme latine de « demeure », ou plus simplement « une maison » et pas seulement « un château ». Par ailleurs, l'empereur n'était pas nécessairement le seigneur du château, mais il était le chef d'un domaine administratif.

La réponse est donc un hors-sujet total, mais pour sa défense, le *zero-shot* sans aucune

contextualisation est loin d'être l'usage le plus honnête et efficace d'un grand modèle de langage.

2) *Prompt contextualisé*

Llama propose maintenant 5 étapes:

- *mansionem*
- *maisonon* (*forme intermédiaire*)
- *maison*
- *maysoun*
- *maison*

Là encore, l'évolution proposée est loin des 15 étapes attendues par le jury. Les justifications proposées par le modèle sont des hallucinations lunaires de bout en bout:

Le son [t] devient [d], dû à l'influence du français.

Le modèle n'a jamais proposé ni [t], ni [d] dans ses formes.

La diphongue "ou" se simplifie en [i].

Une simplification audacieuse, d'autant plus lorsque Llama ne propose aucune diphongue "ou" au préalable.

Le son [s] initial peut être conservé ou disparu, mais ici il n'est pas clair de dire si l'évolution est due à un changement phonétique spécifique.

Le son [s] n'est ici pas initial, et il n'est pas clair de comprendre de quelle évolution le modèle parle, quand il a pourtant postulé que le [s] reste stable de bout en bout.

C. *Notre modèle*

Llama 3 produit donc un résultat déplorable. Notre implémentation lui permet-elle un progrès significatif, afin d'atteindre une performance comparable à GPT-5 ?

Après avoir généré (de manière erronée) comme points de départ et d'arrivée /mансо. 'н̄/ et /maj.sɔ/, notre agent reconstitue 10 formes:

- manso.'н̄
- manso.'ne
- mansom
- mansøm
- mansöm
- mansøm
- mansøm
- maisonem

- maisonem
- maison
- maison

Notre agent propose des évolutions presque aussi hallucinantes qu'auparavant:

/mn/ > /nm/

Une métathèse très classique, mais introuvable dans cet exercice, puisque /mn/ n'y apparaît jamais. Ou encore:

/i/ et /u/ > ii

Quand ni /u/ ni /ü/ n'ont été proposés. Il n'est d'ailleurs pas explicitement dit quel phonème ü représenterait.

On constate l'usage du terme *paroxyton* (mot dont l'accent tonique est placé sur la pénultième syllabe), mais sans pour autant indiquer l'accent à aucune étape. On trouve des hallucinations dans les étapes mêmes, ainsi que leurs datations: On alterne entre XV^e et XVI^e siècle, certaines étapes sont dupliquées (maison > maison).

Bien que l'on puisse qualifier ce résultat de progrès, par rapport au Llama d'origine, les avancées à effectuer sont encore très grandes avant que l'on puisse espérer voir Llama 3, ou même GPT 5, devenir professeur...

IV. DISCUSSION ET TRAVAUX FUTURS

A. *Discussion des résultats*

Plus précisément, nous avons réfléchi à plusieurs possibilités d'améliorations de l'agent que nous n'avons pas pu implémenté, principalement par manque de temps, nous profitons donc de cette section pour discuter de ces optimisations, dont voici les plus plausibles et les plus pertinentes (classées par ordre du noeud qu'elles prévoient d'optimiser) :

- Pour la transcription (étape 3), on pourrait considérer utiliser l'API d'un site qui référence les transcription phonétiques des mots latins tel que www.howtopronounce.com, ou utiliser une base de données de l'alphabet phonétique international au français et au latin, qu'il aurait fallu chercher en détails.

- Pour l'étape 4, nous pensons, en nous penchant encore plus sur les phénomènes d'évolution phonétique, que donner une liste plus précise au modèle pourrait améliorer ces performances.
- En les découplant proprement en chunks, il nous paraît également intuitif que rajouter les corrections d'annales aux données accessibles par le retriever (étape 5.2) améliorerait considérablement ses performances. Cela équivaudrait à la manière dont un candidat se sert des annales pour répondre à la question, c'est d'ailleurs ce que le jury conseille, en affirmant qu'il s'agit d'une des manières les plus efficaces de répondre à ce genre de questions considérées 'classiques'.
- Toujours en utilisant les corrections d'annales, nous pourrions nous en servir pour fournir un In Context Learning à l'étape 6, afin que le modèle y génère un output structuré en s'appuyant sur des exemples validés par les correcteurs. Créer un environnement de Few-shot prompting est en effet connu pour augmenter significativement les performances des agents.

Concernant les limites de notre agent, il convient de noter les défauts qui émergent de nos tentatives. D'abord, nous avons appliqué notre structure agentique à un seul modèle pouvant tourner en local, et ne contenant que 8 milliards de paramètres, ce qui majore bien sûr les performances atteignables. D'autre part, notre architecture crée une grosse tension aux étapes 3 et 4, qui déterminent entièrement la direction globale prise par l'agent. Pour nous, cette pression constitue le plus gros défaut conceptuel de l'architecture : aucune étape ultérieure ne peut rattraper une erreur commise aux étapes 3 ou 4. Il faudrait donc soit subdiviser le travail accompli lors de ces étapes, soit les rendre infaillibles, ce qui ne serait possible que par une expertise et une heuristique très poussées de notre part.

B. Travaux futurs

Outre les améliorations possibles de l'agent, nous considérons aussi les domaines de recherches parallèles qui pourraient contribuer à améliorer l'agent, sans être directement intégrables avec seul le temps comme facteur limitant, mais bien d'autres approches possibles au même problème.

D'abord, le Reinforcement Learning semble être une option efficace et prometteuse pour la résolution d'une tâche spécifique, d'autant plus que ceux-ci ont déjà fait leurs preuves pour des tâches par LLM génératif (Hou & Niu, 2024). On peut en plus l'aligner en l'entraînant directement sur la base des annales disponibles. Il faudrait alors comparer les différentes possibilités agentiques pour déterminer la plus performante.

Ensuite, nous avons déjà mentionné les systèmes formels de grammaire guidant les règles d'évolution phonétique. La conception d'un tel modèle formel pourraient tout à faire être une piste d'automatisation de notre problème, ou, à minima, servir de guide à un LLM génératif pour donner un modèle agent hybride (règles explicites et IA), ce qui est souvent une option prometteuse (Panchendarajan & Zubia, 2024).

D'une manière plus générale, nous pensons qu'un agent fonctionnellement capable de retracer l'évolution phonétique d'un mot du latin au français moderne pourrait avoir des applications pédagogiques, et pourrait être utilisé pour donner des exemples interactifs aux candidats, de la même manière que cela existe déjà en médecine sur la base des données médicales.

V. BIBLIOGRAPHIE

Anderson, B., Galpin, R., & Juzek, T. S. (2025). Model Misalignment and Language Change: Traces of AI-Associated Language in Unscripted Spoken English. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 8(1), 179–191. <https://doi.org/10.1609/aies.v8i1.36540>

Armaselu, F., Liebeskind, C., & Valunaite Oleskeviciene, G. (2024).

- Self-Evaluation of Generative AI Prompts for Linguistic Linked Open Data Modelling in Diachronic Analysis. In G. Sérasset, H. G. Oliveira, & G. V. Oleskeviciene (Eds.), *Proceedings of the Workshop on Deep Learning and Linked Data (DLnLD) @ LREC-COLING 2024* (pp. 86–91). ELRA and ICCL. <https://aclanthology.org/2024.dlnld-1.8/>
- Bac philo 2025: ChatGPT et Le Chat testés par un prof agrégé.* (n.d.). Retrieved December 28, 2025, from <https://www.ekole.fr/blog/on-plafonne-fait-passer-bac-philo-ia-chatgpt-chat-voici-quen-pense-prof>
- Bastings, J., Titov, I., Aziz, W., Marcheggiani, D., & Simaan, K. (2017). Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 1957–1967. <https://doi.org/10.18653/v1/D17-1209>
- Cai, J., Li, J., Zhang, M., Li, M., Wang, C.-S., & Tei, K. (2024). Language Evolution for Evading Social Media Regulation via LLM-Based Multi-Agent Simulation. *2024 IEEE Congress on Evolutionary Computation (CEC)*, 1–10. <https://doi.org/10.1109/CEC60901.2024.10612015>
- Chomsky, N. (with Internet Archive). (1968). *The sound pattern of English*. New York, Harper & Row. <http://archive.org/details/soundpatternofn0000chom>
- Dekker, P. (2024). *Identifying drivers of language change using agent-based models*. Brussel.
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). [No title found]. *Proceedings of the 2019 Conference of the North*, 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- Du, X., & Cardie, C. (2018). Harvesting Paragraph-level Question-Answer Pairs from Wikipedia. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1907–1917. <https://doi.org/10.18653/v1/P18-1177>
- Elsahar, H., Gravier, C., & Laforest, F. (2018). Zero-Shot Question Generation from Knowledge Graphs for Unseen Predicates and Entity Types. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 218–228. <https://doi.org/10.18653/v1/N18-1020>
- Feng, Y., Chen, X., Lin, B. Y., Wang, P., Yan, J., & Ren, X. (2020). Scalable Multi-Hop Relational Reasoning for Knowledge-Aware Question Answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1295–1309. <https://doi.org/10.18653/v1/2020.emnlp-main.99>
- Ganea, O.-E., & Hofmann, T. (2017). Deep Joint Entity Disambiguation with Local Neural Attention. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2619–2629. <https://doi.org/10.18653/v1/D17-1277>
- Goldsmith, J. A. (1976). *Autosegmental phonology* [Thesis, Massachusetts Institute of Technology]. <https://dspace.mit.edu/handle/1721.1/16388>
- GPT-5 is here | OpenAI. (2025). <https://openai.com/gpt-5/>
- Habscheid, S., Hector, T., Hoffmann, D., & Waldecker, D. (Eds.). (2024). *Voice Assistants in Private Homes: Media, Data and Language in Interaction and*

- Discourse (1st ed., Vol. 7). transcript Verlag.
<https://doi.org/10.14361/9783839472002>
- Han, X., Cao, S., Lv, X., Lin, Y., Liu, Z., Sun, M., & Li, J. (2018). OpenKE: An Open Toolkit for Knowledge Embedding. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 139–144.
<https://doi.org/10.18653/v1/D18-2024>
- Hou, Z., Niu, Y., Du, Z., Zhang, X., Liu, X., Zeng, A., Zheng, Q., Huang, M., Wang, H., Tang, J., & Dong, Y. (2024). ChatGLM-RLHF: Practices of Aligning Large Language Models with Human Feedback (No. arXiv:2404.00934). arXiv.
<https://doi.org/10.48550/arXiv.2404.00934>
- Huo, C., Ma, S., & Liu, X. (2022). Hotness prediction of scientific topics based on a bibliographic knowledge graph. *Information Processing & Management*, 59(4), 102980.
<https://doi.org/10.1016/j.ipm.2022.102980>
- Ji, H., Ke, P., Huang, S., Wei, F., Zhu, X., & Huang, M. (2020). Language Generation with Multi-Hop Reasoning on Commonsense Knowledge Graph. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 725–736.
<https://doi.org/10.18653/v1/2020.emnlp-main.54>
- Joly, G. (2003). *Précis de phonétique historique du français*.
<https://www.dunod.com/lettres-et-arts/précis-phonétique-historique-du-français>
- Karpukhin, V., Ong, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020). Dense Passage Retrieval for Open-Domain Question Answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6769–6781.
<https://doi.org/10.18653/v1/2020.emnlp-main.550>
- Koncel-Kedziorski, R., Bekal, D., Luan, Y., Lapata, M., & Hajishirzi, H. (2019). [No title found]. *Proceedings of the 2019 Conference of the North*, 2284–2293.
<https://doi.org/10.18653/v1/N19-1238>
- Laborderie, N. (2015). *Précis de phonétique historique*.
<https://www.dunod.com/lettres-et-arts/précis-phonétique-historique>
- Lawrence, J., & Reed, C. (2020). Argument Mining: A Survey. *Computational Linguistics*, 45(4), 765–818.
https://doi.org/10.1162/coli_a_00364
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880.
<https://doi.org/10.18653/v1/2020.acl-main.703>
- Liu, Z., Xiong, C., Sun, M., & Liu, Z. (2020). Fine-grained Fact Verification with Kernel Graph Attention Network. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7342–7351.
<https://doi.org/10.18653/v1/2020.acl-main.655>
- Marr, C. (2017). *Towards a computational model of long-term diachronic change: Simulating the development of Classical Latin to Modern French*.
- Marr, C., & Mortensen, D. (2023). Large-scale computerized forward reconstruction yields new perspectives in French diachronic phonology.

- Diachronica*, 40(2), 238–285.
<https://doi.org/10.1075/dia.20027.mar>
- Meta-llama/Llama-3.1-8B* · Hugging Face. (2024, December 6).
<https://huggingface.co/meta-llama/Llama-3.1-8B>
- Narayan, S., Cohen, S. B., & Lapata, M. (2018). Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1797–1807.
<https://doi.org/10.18653/v1/D18-1206>
- Noraset, T., Lowphansirikul, L., & Tuarob, S. (2021). WabiQA: A Wikipedia-Based Thai Question-Answering System. *Information Processing & Management*, 58(1), 102431.
<https://doi.org/10.1016/j.ipm.2020.102431>
- Pan, M., Kitson, A., Wan, H., & Prpa, M. (2025). ELLMA-T: An Embodied LLM-agent for Supporting English Language Learning in Social VR. *Proceedings of the 2025 ACM Designing Interactive Systems Conference*, 576–594.
<https://doi.org/10.1145/3715336.3735786>
- Peters, M. E., Neumann, M., Logan, R., Schwartz, R., Joshi, V., Singh, S., & Smith, N. A. (2019). Knowledge Enhanced Contextual Word Representations. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 43–54.
<https://doi.org/10.18653/v1/D19-1005>
- Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P., Bakhtin, A., Wu, Y., & Miller, A. (2019). Language Models as Knowledge Bases? *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2463–2473.
<https://doi.org/10.18653/v1/D19-1250>
- Peyrard, M., Josifoski, M., & West, R. (2025). *Agentic AI: The Era of Semantic Decoding* (No. arXiv:2403.14562). arXiv.
<https://doi.org/10.48550/arXiv.2403.14562>
- Roberts, P. (2012). *Towards a computer model of the historical phonology and morphology of Latin* [Http://purl.org/dc/dcmitype/Text, Oxford University, UK].
<https://ora.ox.ac.uk/objects/uuid:d3ef315c-3d5c-486b-8fbe-0fa6fdbb8219>
- Romanello, M., Gabay, S., & Carboni, N. (2025, September). Predicting the Fictional Time and Space of French Theatre Plays by Using Large Language Models. *IEEE International Conference on Cyber Humanities*.
<https://hal.science/hal-05157140>
- Saedi, C., Branco, A., António Rodrigues, J., & Silva, J. (2018). WordNet Embeddings. *Proceedings of The Third Workshop on Representation Learning for NLP*, 122–131.
<https://doi.org/10.18653/v1/W18-3016>
- Sennrich, R., & Haddow, B. (2016). Linguistic Input Features Improve Neural Machine Translation. *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, 83–91.
<https://doi.org/10.18653/v1/W16-2209>
- Si, J., Zhou, D., Li, T., Shi, X., & He, Y. (2021). Topic-Aware Evidence Reasoning and Stance-Aware Aggregation for Fact Verification. *Proceedings of the 59th Annual Meeting*

- of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 1612–1622.*
<https://doi.org/10.18653/v1/2021.acl-long.128>
- Stasaski, K., & Hearst, M. A. (2017). Multiple Choice Question Generation Utilizing An Ontology. *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, 303–312.
<https://doi.org/10.18653/v1/W17-5034>
- Violin-Wigent, A. (2006). OPTIMALITY THEORY: CONSTRAINT INTERACTION IN GENERATIVE GRAMMAR. *Studies in Second Language Acquisition*, 28(1), 160–160.
<https://doi.org/10.1017/S0272263106220060>
- Wang, D., Liu, P., Zheng, Y., Qiu, X., & Huang, X. (2020). Heterogeneous Graph Neural Networks for Extractive Document Summarization. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6209–6219.
<https://doi.org/10.18653/v1/2020.acl-main.n.553>
- Wang, X., Shen, Y., Cai, J., Wang, T., Wang, X., Xie, P., Huang, F., Lu, W., Zhuang, Y., Tu, K., Lu, W., & Jiang, Y. (2022). DAMO-NLP at SemEval-2022 Task 11: A Knowledge-based System for Multilingual Named Entity Recognition. *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, 1457–1468.
[https://doi.org/10.18653/v1/2022.semeva1-1.200](https://doi.org/10.18653/v1/2022.semева1-1.200)
- Xu, J., Gan, Z., Cheng, Y., & Liu, J. (2020). Discourse-Aware Neural Extractive Text Summarization. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 5021–5031.
- <https://doi.org/10.18653/v1/2020.acl-main.451>
- Yang, Z., Xu, Y., Hu, J., & Dong, S. (2023). Generating knowledge aware explanation for natural language inference. *Information Processing & Management*, 60(2), 103245.
<https://doi.org/10.1016/j.ipm.2022.103245>
- Yasunaga, M., Ren, H., Bosselut, A., Liang, P., & Leskovec, J. (2021). QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 535–546.
<https://doi.org/10.18653/v1/2021.naacl-main.45>
- Yu, D., Yang, B., Liu, D., Wang, H., & Pan, S. (2023). A survey on neural-symbolic learning systems. *Neural Networks*, 166, 105–126.
<https://doi.org/10.1016/j.neunet.2023.06.028>
- Zhang, H., Liu, Z., Xiong, C., & Liu, Z. (2020). Grounded Conversation Generation as Guided Traverses in Commonsense Knowledge Graphs. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2031–2043.
<https://doi.org/10.18653/v1/2020.acl-main.n.184>
- Zhang, Q., Chen, S., Fang, M., & Chen, X. (2023). Joint reasoning with knowledge subgraphs for Multiple Choice Question Answering. *Information Processing & Management*, 60(3), 103297.
<https://doi.org/10.1016/j.ipm.2023.103297>
- Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., & Liu, Q. (2019). ERNIE: Enhanced Language Representation with

Informative Entities. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1441–1451.

<https://doi.org/10.18653/v1/P19-1139>

Zhong, W., Xu, J., Tang, D., Xu, Z., Duan, N., Zhou, M., Wang, J., & Yin, J. (2020). Reasoning Over Semantic-Level Graph for Fact Checking. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6170–6180.

<https://doi.org/10.18653/v1/2020.acl-main.549>

Zhou, J., Han, X., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2019). GEAR: Graph-based Evidence Aggregating and Reasoning for Fact Verification. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 892–901.

<https://doi.org/10.18653/v1/P19-1085>

Rapport

Group report

Link to the repository: <https://github.com/phtryll/agreg-agent>

1) The task

The final goal of the project is to build an agent that is capable of accurately answering agrégation-style questions on the phonetic evolution of French, from Latin to today. The starting point was simply to build an agent that is capable of “passing” the agrégation exam in lettres modernes, but as this task is arguably unfeasible for multiple reasons:

- The LLMs that we can reliably use are relatively small models (8B) and ideally should be run locally, making their “reasoning” capabilities and internal “knowledge” relating to the very specific and difficult tasks of the agrégation limited.
- Extending these capabilities would require extensive documentation, and although the agrégation program is available, many of the fundamental books and textbooks required are simply not available.
- Evaluation of such a large task would require at least a qualitative assessment of the output of the agent (and developing a quantitative evaluation would be almost impossible), and we are underqualified to write a similar dissertation, let alone evaluate it, even if we somehow compared it to past correction templates.

There is one task in the agrégation de lettres modernes that is feasible given the constraints above: the phonetic evolution of French. Indeed:

- The knowledge is precise enough that we can at least attempt to enrich an LLM to learn phonetic evolution patterns.
- The required documentation is available: there are two fundamental textbooks (“Précis de Phonétique Historique” by Nöelle Laborderie and “Précis de Phonétique Historique du Français” by Geneviève Joly). In addition we can use the rapport de jury from the previous years (2020-2025).
- The rapports du jury provide detailed corrections in the form of a table, each line represents a phonetic change over time. For each line there are three entries: a chronological date/period, the phonetic form of the word at that time, and a description of the changes.

The task is therefore simple, a question of standard format: “Retracer l’évolution, du latin au français moderne, de [Latin word] > [Modern French word]”. Since the expected output is also straightforward (i.e. the agent should output a table with three entries as mentioned above, the hard part is actually making sure that the provided answers are correct and not gibberish), this makes it possible, or at least attemptable, to build a small AI agent to tackle this task.

2) Starting point

We can think of our agent as a graph where the nodes are “states” and the edges are “transformations” or functions that transform the states. The starting structure is simple: input question → something happens here → structured “table” answering the question, with chronological phonetic evolutions of the given words and short descriptions of the changes. So the goal is to figure out the middle part.

We can start by determining a baseline: we take two LLMs, one state-of-the-art and another “small” model and ask the evaluation question (“Retracer l’évolution, du latin au français moderne, de [Latin word] > [Modern French word]”) with two prompts:

- Simple zero-shot prompt that consists just of the question.
- Contextualized prompt where we tell the LLM that it is passing the aggregation and that it must answer this question with a precise format in mind.

The prompts that we will use for the baselines are the following:

- “Retracer l’évolution, du latin au français moderne, de [Latin word] > [Modern French word].”
- “Tu es un candidat à l’agrégation de lettres modernes. Tu passes l’épreuve d’étude grammaticale d’un texte antérieur à 1500, et tu dois répondre à la question de phonétique. La question est [Prompt 1]. Tu dois répondre de manière précise en suivant de près les instructions suivantes: Tu dois faire un tableau où chaque ligne représente une évolution phonétique d’une période à une autre. Pour chaque changement, indique la période/siècle où ce changement est observé et la forme phonétique du mot à cette époque ainsi qu’une courte description du ou des changements en question.”

The models that we will use will be GPT-5.2 for the state-of-the-art model and Llama-3.1 for the “small” model, it is this last model that we will integrate in our agentic workflow hoping to improve its response accuracy.

Some remarks on the baseline responses:

- We noticed that Llama3.1 was completely unable to answer the question with the simple prompt.
- ChatGPT-5.2 was much more performant, but still was very imprecise, and often wrong in terms of the actual content of its response.
- The “contextualized” prompt was enough to at least constrain the models to produce outputs that resemble the final format.
- From these simple tests it is clear that the main goal will be to break down the task in such a way that the model will improve its performance. Furthermore, including some sort of retrieval system seems inevitable, if we want to attempt to improve the accuracy of its responses.

3) Starting off with a simple LLM call

We start by simply starting a chat with our LLM (from now on, when we say LLM we mean Llama3.1-8B). We do not care about having a conversation with our agent, since the goal is to provide an answer to a very specific question.

Question: Why use this specific model, and not another one available from the Ollama library? A question we asked ourselves is that of the quality of the pre-training data relative to the task at hand. For example, shouldn't we expect better results from a model like mistral-7B, that has (supposedly) been trained on more french data than Llama3.1? We experimented with these two models, and early tests led us to the conclusion that Llama3.1 was overall a better performing model, but this was done before the full implementation of the agentic pipeline. Heuristically, we chose Llama3.1 because (1) it is slightly larger and (2) because it is more recent. If we had a solid pipeline, testing with different LLMs to see how performance varies would have been an interesting experiment, but since our agent architecture was going to change over time, in order to be able to minimally control the changes, we picked one model and stuck with it.

We build the simplest possible pipeline as a starting point. This is equivalent to simply having a script that opens up a chat with the LLM. We start with a simple script called “main dot py”.

- Loading Llama3.1 works fine, we are able to open up a conversation.
- We have no structured output, the model answers our questions freely.
- The model can obviously not answer the question.

Our agent has to produce a structured answer, reassembling as much as possible to the output expected by the correctors at the agrégation. There are two problems: (1) having the structure of the answer fit the required format, (2) having the contents of the answer be accurate and rigorous to the standards of the exam. The first part is much simpler, as we have seen with the baselines, the models are completely capable of following the structure, just by specifying it in the prompt. The hard part will be making sure that the LLM answers accurately to the questions.

- We want to tackle the second problem immediately.
- The “best” way to guide the model to generate an answer with such precise knowledge is to point it directly at the books that are recommended by the agrégation.
- There will be a problem with context so we cannot simply pass the books as context.
- Fine-tuning is out of the question as for this we need input label pairs? What labels will there be for such input? And this breaks the purpose of building an agent in the first place...
- Solution: build a small retrieval system and see if the model’s response works.
- “pymupdf” works well and converts the pdf to text. Images are not rendered, tables are broken...

- Tried to use “marker-pdf” that has image and table support, generally is a much better solution, it can also convert pdf to structured markdown and we could hope to build a sort of graph-based retrieval to improve our queries.
- Unfortunately, we encountered dependency problems that appeared to have to do with our hardware (this was tested on mac silicon). Had to fall back to the first solution, all be it not perfect, it could at least convert the pdf to text.
- Chunking was successful, defined initial chunk size to 500 words (we do splitting to space) with an overlap of 100. We will play with these parameters later.
- We use simple similarity embeddings from “sentence_transformers”, we then simply encode the query to embeddings, and compute cosine similarity between the query and the chunks, taking the top 5 results.

After a lengthy search online, we were able to “find” only one of the two historical phonetics manuals that are recommended in the agrégation syllabus: the “Précis de phonétique historique” from Noëlle Laborderie. We used this as our source document to include in our retrieval system.

This test implementation of a retriever has to be passed to the LLM in some way. An agentic architecture would have the retrieval built as an available tool. But we are not yet testing for this, we simply want to know if using the retriever, the model can improve the accuracy of its answers. Our first pipeline is as follows:

- We pass the question to the LLM and ask it to produce a query based on the question (it knows that this will be passed to a retrieval system).
- We pass the query to the retrieval system and extract the retrieved chunks.
- We pass back the retrieved chunks with the original question and ask it to generate a response.

We tried this with the question from 2025: Retracer l'évolution, du latin au français moderne, de INGENIUM > engien. The answer was very much better: the model was able to find the relevant information because it included the latin and french forms in its query. Turns out that the manual had information on that precise derivation, and it was able to accurately describe “some” of the phonetic phenomena. To verify the consistency of this result, we also tried with the previous year's question: “Retracer l'évolution, du latin au français moderne, de *MA(N)SIONEM > maison”. The results were completely underwhelming. It appears that this derivation is not in the manual... and although it retrieved chunks, the final answer was in no way more accurate. It basically used the retrieved information to hallucinate a more “plausible” response.

Despite all that, the retrieval system works and is a useful safeguard from free generation. It seems that the problem is shifted to the architecture of our agent. Our version right now is basically like instantiating two chats independently of each other and simply having a retrieval system in the middle. We need to think about how to structure the response of our agent in a consistent way.

4) Designing a structured chain-of-thought pipeline

Let's reinstate our goal: build an agent that can reliably (and successfully) answer exam questions asked in the agrégation de lettres modernes, precisely the historical phonetics question. We assume that performance wise, our agent won't be very successful, but that is ok; if we can at least break down the problem to sub-parts that can be “optimized”, that is, attempt at translating the task to a series of optimization problems, it would be a great start. So the question becomes, how can we break down the problem into relatively simple steps that can be executed by an LLM.

Our intuition is to build some sort of directed workflow, where a general “state” is being modified by different functions or steps, with different LLM calls in the process. Once the theoretical steps of solving the task are settled upon, adding the relevant tools becomes much easier. Furthermore, what is interesting about the agent is not the specific tools we are going to use, but the “logic” we are trying to implement, and how it helps us tackle the very complicated task at hand.

Our starting intuition is that we need somehow to simulate the working process of an actual student that passes this exam. None of us have actually attempted the agrégation, but we at least have some ideas about how one can try to answer such a question:

- When we are faced with the question, a trivial step for us is to identify the terms that we are going to be working with;
- We then need to transcribe them in their phonetic forms or IPA, since these will be the starting and ending forms, and will indicate or hint at the phonetic evolution of the word;
- What does a student do then? (if they have not memorized the precise derivations by heart, that would be equivalent to an LLM having seen the specific phonetic evolution during training): they might start asking questions or making hypotheses about the phonetic phenomena that occurred, perhaps being informed by some sort of chronological order of changes, and trying to formulate a draft or plan of how to tackle the question;
- We assume (as is the case?) that the agrégation students have access to a library with books? Either way we assume that we have access to some sort of documentation, otherwise, as we have seen with the baseline (and unfortunately even with the previous experiment with the retrieval system), without some external source to at least attempt some sort of grounding, the LLM will most definitely underperform and hallucinate;
- Ideally, the hypotheses formulated earlier will guide the research made with the documentation, and based on this research, we (the agent) can formulate a final plan of phonetic evolution.
- To summarize, if we were to tackle the question on our own: we would transcribe the two terms in their IPA forms, formulate hypotheses about phonetic changes, verify these

claims and research information around these intuitions, and with this formulate a plan of a chronological phonetic evolution.

Arguably the task is a lot harder, there are numerous additional questions that arse:

- During the phonetic transcription of the Latin and French words in the question, how can we be sure that we have correctly transcribed them in IPA?
- Same goes for the hypotheses we make, how can we be sure of the relevance of these claims? Say believe there is at some point a nasalisation of a consonant, but once the textbook is consulted, there is no relevant information, do we conclude that our hypothesis is false, or that the textbook just did not contain the required information?
- We assume one single way of trying to figure out what are the phonetic changes: that is we look at the first and final phonetic form, and pretty much “guess” possible relevant changes for that word, but another approach would be to simply start from the latin word, and based on the sounds start deriving a “tree” of different changes over the centuries (like some sort of rule system), and slowly, “bottom-up” try to find the correct path that leads to the final form. We do not choose this approach because we assume that our agent does not have this type of extensive knowledge of all the phonetic changes specific to French over 20 centuries. But nothing apart from this arbitrary decision tells us that this process is any less worse than our original intuition.

We try to implement this approach: we need to build an agent that does the following:

- Looks at the question: identifies the words we are working with;
- Takes the two terms, and transcribes them in IPA;
- Looks at the IPA forms, starts making hypotheses about the possible phonetic changes over time;
- These hypotheses are translated into “research questions”, more like simple queries, that we can use with our retrieval system;
- Based on the retrieved information, the LLM plans its response, making sure that it provides data for the required output: a table with a chronological marker, a phonetic form of the word in use at that time, and a short description of the phonetic changes that have taken place;
- The LLM can then generate an “informed” response based on that plan, this is the final output.

This pipeline is very deterministic. Our reasoning is the following: if we start giving the LLM autonomy about “how” it can go around answering the question, if it fails, we have no way of controlling what is going on in its reasoning steps. Furthermore... Llama3.1 is not a reasoning model... therefore, we really have no way of knowing how it generates its output. Finally, we have to address the problem in “exam conditions”. It’s no fun if in the input (i.e. the first prompt with the question) we provide extensive information other than what is given in the exam: the student has a text in french anterior to 1500, and a set of questions, the first being the historical

phonetics question (after the translation) without any other context. Again, the second approach that we proposed previously would be more accurate in simulating a student that has prepared the agrégation for a year, if not more, but we need to start off simple... we have to start somewhere...

5) Implementation

Going back to our codebase, we have a main script that simply calls the LLM once to read the question and generate a query, the query is passed to a retrieval system, and then the retrieved chunks plus the question get passed again to the LLM to generate the final answer. Not very elegant.

A. The agent state

Our approach is similar to a “structured chain-of-thought”, it can be thought of as a graph, where there is an “agent state” that gets modified through different steps. We start by designing the agent state:

- It will simply be a python class containing different fields that will get filled step by step:
 - The question;
 - The words that we are working with, as present in the question;
 - Something to store their IPA forms, probably a dictionary;
 - The hypotheses;
 - The queries;
 - The “plan”, again some sort of dictionary?
 - Then finally the answer,
 - Perhaps some log, especially to record at each step what went wrong.

Question: Should we start using standard libraries such as Langchain, Langraph and whatnot? Well at this stage, we argue that it is a bit of an overkill. For instance, we have no real “tools” that our agent can call yet. Either way, what tool can we use for this type of task? A dictionary of phonetic forms? This will be tackled at the phonetic transcription stage, but for example Wiktionary that was proposed by the professor does not have at all the phonetic transcriptions of the Latin terms in the agrégation. Most of the terms are in the accusative singular and they have the IPA transcription only for the nominative singular. Ok arguably we could retrieve the nominative and then ask for the accusative, the LLM would have to do a little less guessing, but still, there is no tool calling, it's all deterministic, the model has to use Wiktionary.

The pipeline we have defined is basically some glorified chain-of-thought “simulation” by having multiple LLM calls enrich a “state”. But for the task at hand, it seems appropriate, if we want to build an agent that reliably solves a problem, we need to reliably define “how” it can

solve it, and if it does not, (even more important) we need to have a clear idea of where and how it went wrong (as much as possible). Importantly we need to break down the task to simple “optimisable” sub-tasks. Then we could think about tool calling, multi-agent eval and fancy stuff.

Importantly, there needs to be structure, the agent state is what really matters, and the LLM output will be structured in a way that directly fills the slots of the agent state, for this we will use the old faithful json schemas and structured output in Ollama that is as simple as passing a Pydantic BaseModel class to the “format” argument of the “call” method of the Ollama library. This can guarantee that the output of the LLM fits nicely with our needs and feeds our agent state as we move down the pipeline.

B. Ollama

As we said we are calling multiple times Llama3.1 with different prompts, at different stages of the process. Therefore we have one template call. We define a global “system” prompt that will be shared by all the nodes of our agentic pipeline:

“Tu es un candidat à l’agrégation de lettres modernes ; tu passes l’épreuve d’étude grammaticale d’un texte antérieur à 1500 et tu dois répondre à une question de phonétique historique. Tu dois produire UNIQUEMENT du JSON valide conformément au schéma fourni, sans aucune explication supplémentaire ni clé additionnelle. Question (contexte) : [question].”

For each node, we specify a user prompt that will represent the “sub-task” to be executed by the model at that node, and we also tell the model to expect a BaseModel class json schema, different one each time.

C. Classify and transcribe

The first nodes of our agent. The first is fairly straightforward, simply identify the terms we are going to be working with in the question. The second is a bit more tricky and arguably one of the most fundamental nodes in our pipeline: if the LLM makes a mistake here, it is most certain that it will only amplify the mistake in the later stages.

Results: the LLM always identifies the words correctly, this is a trivial task. Concerning the phonetic transcription, it is sometimes accurate, sometimes completely wrong. We tried changing the prompt, but the results are the same. One improvement would be to include Wiktionary data, as mentioned before. Correction, we could not correctly parse for the IPA. For example the “wiktionsparser” library does not have IPA information, and making HTTP requests to wiktionary failed. Here we can mark this node in the areas of improvement: if we ever get to tool

calling, perhaps a simple web search tool and a prompt pointing to Wiktionary can render better results.

D. Triggers and queries

The third and fourth nodes of our agent. With the transcription step, they are the most important, as the retrieved passages, and therefore the quality of the final “plan”, depend on the type of hypotheses the agent makes based on the phonetic form of the words. Why not have a single step that looks at the sounds and derives queries? This is exactly what we tried in the beginning.

The planner was not decomposed into triggers → queries → retrieval → plan, but it was simply a two step process with queries → retriever and then the “generate” node was tasked to generate a response based on the retrieved passages. The problem is that one task, generating queries for the retriever, should happen after the much more complex task of inferring what possible phonetic evolutions could have happened between the first and last form. Translating some span of text into a question is typically the simple task that an LLM can do almost effortlessly. On the other hand, inferring useful information is difficult. Separating these two tasks in two different nodes allows for improvement in the future. We argue that the “trigger” step, or the hypothesis building step is the most crucial step with transcribing the words into IPA.

If we had more time, we would have liked to build two simple tools, one that consistently checks phonetic transcriptions of lexemes, and another to specifically tackle this problem of inferring possible evolutions, perhaps with some rule-based system?

E. Retriever

The retriever has been slightly reworked compared to the previous implementation in the early versions of our agent. The main difference is that before, we simply had an embeddings model encode the chunks and the queries, and then we performed a simple cosine similarity search. What we have done now is switched this model for a cross-encoder model re-ranker. The key difference is that the cross-encoder processes the query and the chunks at the same time, whereas the simple cosine similarity search first computes the embeddings and then proceeds to compute similarities. Since our “task” is to find the most relevant document based on the query, we are basically performing classification between all the chunks in our database, cross-encoders are much more reliable in classification than simple bi-encoders.

The switch between the two is seamless: we still use the “sentence_transformers” library and use “cross-encoder/ms-marco-MiniLM-L12-v2” as a model, and indel 3 lines of code. Although our implementation is not optimal (indexing happens at every run), it is still extremely fast (a couple of seconds) which for a normal use case poses no problems.

F. Planner and generation

The planner is the final stage of the “thinking” process of the agent. This is basically the scratchpad where all the information collected is stored in a structured way in the form of a “phonological fact” class (the name is very bland...). In theory, it would store a list of phonetic phenomena relevant to the question, with a chronological marker, information about the graphemes, the source excerpt where the information was taken from, as well as a short description, equivalent to what we can find in the third column of the agrégation corrections, in the form of a dictionary. Based on this, the generation node can simply take this information and write an answer. Again, the point is to separate as much as possible the trivial tasks (rewriting, generation, locating key words), to the non-trivial tasks that require some thinking or external knowledge, and can be augmented later on.

The main problem at this stage is the context window. The LLM generates a discrete number of triggers that get translated into queries, playing around with the size of the chunks and the top k documents we retrieve, these results are multiplied by the number of queries. The simple solution is increasing the size of the context window within Ollama, but we are faced with another limitation of our project: the local LLM is unable to run fast enough with such a large context window. One solution would be to build the list of phonological facts incrementally, passing one query or even one document at a time, but this would create other artefactual problems such as having to determine at what rate the model generates and validates one “phonological fact”: is it one query one entry or one document one entry? And if not, how can the model decide “ok there is no extra information for this fact, let’s move on?”. The temporary solution is to reduce the chunk sizes in order to fit in one call, but this of course reduces the quality of the retrieval.

G. Final output, verification, issues logger

The final stages do not involve an LLM, they are therefore completely deterministic. Since the generator node also outputs a structured output, we have a simple parser that reads the contents and prints them neatly. The verifier checks if the LLMs output conforms to a small set of presentation rules and also prints all the log messages recorded during the pipeline; mostly empty labels, wrong IPA formatting and whatnot.

6) Final notes

Our agent performs very poorly. In fact, despite all our efforts, it never managed to even get a single thing right. Despite this there are a couple of future improvements that can be done fairly easily, that would pave the way of a boost in accuracy:

- The pipeline is “hand crafted” in the sense that it does not use the standard agentic libraries. The reason being, when breaking down the task, implementing an agent that follows a specific thought process rather than doing anything as long as it works seemed much more interesting, and simple structured LLM calls modifying an external environment seemed the appropriate first step.
- This can only be considered a first step. We gave examples of minor improvements that we were not successful at implementing here (for example integrating Wiktionary phonetic data), but that would be absolutely essential to move beyond our scaffold.
- Importantly, if the same architecture is tested with better resources, namely a proper LLM called through an API, we can really test the strengths and weaknesses of our pipeline, and move beyond a strictly deterministic workflow, perhaps integrating tool autonomy to our agent. Additionally, it was a nice challenge to see if a small local model would even compete in such a difficult task. Our current answer is clearly, without a doubt, and perhaps for the better, no.

Baseline

ChatGPT-5.2 responses:

Prompt 1:

Voici l'évolution **phonétique, morphologique et sémantique** du mot latin *mansionem* jusqu'au français moderne **maison**, étape par étape.

1. Le point de départ : le latin classique

Latin classique : *mānsiō, mānsiōnem*

- Sens : « action de rester », puis « lieu où l'on s'arrête », « demeure », « gîte »
- Dérivé du verbe *manēre* (« rester »)

👉 On part de l'accusatif **mansionem**, forme qui sert de base à l'évolution romane.

2. Passage au latin tardif / latin vulgaire

a) Évolution phonétique générale

- Chute du *-m* final :
mansionem → *mansione*
- Accent tonique sur **-si-** :
mansione

b) Palatalisation

- Le groupe **-si-** suivi d'une voyelle (*-o-*) se palatalise :
/si/ → */sj/* → */ʃ/* (ch)

👉 On obtient une forme proche de :

***majone**

3. Ancien français (IX^e–XII^e siècles)

Formes attestées : *maison, maisun, meson*

a) Évolutions phonétiques clés

- /ʃ/ > /s/ (simplification en français)
- Fermeture de la voyelle finale :
-one → -un → disparition finale

b) Évolution morphologique

- Perte de la déclinaison latine
- Fixation d'une forme unique

👉 **maison** est déjà bien établie en ancien français.

4. Moyen français → français moderne

a) Stabilisation phonétique

- /z/ apparaît par sonorisation du /s/ intervocalique
- Orthographe fixée au XVI^e siècle

b) Évolution sémantique

- Du sens général de « lieu où l'on s'arrête »
- Au sens spécifique de « bâtiment destiné à l'habitation »

👉 **maison** (sens moderne)

5. Schéma récapitulatif

mansionem
→ mansione
→ *maſone
→ maison / maisun
→ maison

6. Remarque comparative (pour mieux comprendre)

Le même étymon a donné :

- **italien** : *mansione* (plutôt « étape, séjour »)
- **espagnol** : *mansión* (demeure luxueuse)
- **anglais** (emprunt) : *mansion*

👉 Le français a **spécialisé et généralisé** le sens.

Conclusion

Le mot **maison** illustre parfaitement :

- la **palatalisation romane**
- la **chute des finales latines**
- la **spécialisation sémantique progressive**

Si tu veux, je peux aussi te fournir une **ligne du temps phonétique détaillée** avec transcription API, ou comparer *maison* à *demeure* et *habitation*.

Prompt 2:

Voici une réponse **conforme aux attentes de l'agrégation**, centrée sur la **phonétique historique**, avec une présentation **chronologique en tableau**, chaque ligne correspondant à **un changement phonétique identifié**, depuis le latin classique jusqu'au français moderne.

Évolution phonétique de **mansionem** → **maison**

Période / siècle	Forme phonétique (API)	Forme graphique	Description du changement phonétique
Latin classique (I ^{er} s. av.–I ^{er} s. ap. J.-C.)	/man.si'o:nem/	<i>mansionem</i>	Mot quadrisyllabique. Le groupe <i>-ns-</i> est pleinement articulé. Voyelle longue /o:/ . Accent tonique sur <i>-o-</i> (pénultième longue).
Latin tardif (III ^e –IV ^e s.)	/man.sjo'ne/	<i>mansion(e)</i>	Réduction des finales : chute du <i>-m</i> final. Palatalisation naissante de /si/ > /sjo/. Brève affaiblissement de la quantité vocalique.
Latin vulgaire (IV ^e –V ^e s.)	/man'sjo.ne/	<i>mansione</i>	Accent déplacé sur l'avant-dernière syllabe après la perte de quantité vocalique distinctive. Stabilisation du groupe /sj/.
Roman commun (VI ^e s.)	/man'sjo.ne/	<i>mansione</i>	Maintien du groupe <i>-nsj-</i> . Début de l'instabilité du <i>n</i> devant <i>s</i> (nasalisation progressive).
Ancien français (IX ^e –X ^e s.)	/maj'zõn/	<i>maison</i>	Chute de la consonne nasale <i>n</i> devant <i>s</i> avec nasalisation de la voyelle précédente . Sonorisation de /s/ intervocalique > /z/. Diphthongaison de /a/ en /ai/.

Ancien français tardif (XI ^e –XII ^e s.)	/mɛ'zõn/	<i>maison</i>	Monophthongaison de /ai/ > /ɛ/. Conservation de la voyelle nasale /õ/.
Moyen français (XIV ^e –XVe s.)	/mɛ'zõ/	<i>maison</i>	Chute du <i>-n</i> final graphique et phonétique. La nasalité est portée uniquement par la voyelle.
Français moderne (XVII ^e s. → aujourd’hui)	/mɛ.zõ/	<i>maison</i>	Stabilisation du système vocalique nasal. Conservation de /z/ issu de la sonorisation ancienne de /s/. Forme moderne fixée.

Remarques de méthode (niveau agrégation)

- Le **changement clé** est la **nasalisation compensatoire** liée à la chute de *n* devant *s*.
- La **sonorisation de /s/ en /z/** est régulière en position intervocalique en gallo-romain.
- La **diphthongaison puis monophthongaison** de /a/ > /ai/ > /ɛ/ est caractéristique de l'évolution vocalique de l'ancien français.
- La continuité graphique (*maison*) masque une évolution phonétique complexe et étagée.

Si tu le souhaites, je peux aussi te proposer **une version encore plus concise**, exactement calibrée pour une **réponse orale de 5 minutes**, ou bien une **comparaison avec raison, poison, saison** pour renforcer l'argumentation comparative.

Llama3.1 responses:

Prompt 1:

Mansionem est la forme latine de « demeure », ou plus simplement « une maison » et pas seulement « un château ». Par ailleurs, l'empereur n'était pas nécessairement le seigneur du château, mais il était le chef d'un domaine administratif.

Prompt 2:

Voici le tableau détaillant l'évolution de la forme "mansionem" au latin, jusqu'à sa version actuelle en français moderne :

Étape	Forme	Périod e / Siècle	Changement
Latin	mansionem	-	Pont de départ avec une terminaison latine -em.
Latin évolué (fin Moyen Âge)	maisonon (forme intermédiaire)	13e-14e siècle	Le son [t] devient [d], dû à l'influence du français.
Français ancien	maision	14e-15e siècle	La diphtongue "ou" se simplifie en [i].
Français moyen (Renaissance)	maysoun	16e siècle	Le son [s] initial peut être conservé ou disparu, mais ici il n'est pas clair de dire si l'évolution est due à un changement phonétique spécifique.
Français moderne	maison	17e siècle et suivant s	La forme actuelle du mot s'installe dans la langue française standard.

Remarques :

- Les changements linguistiques, surtout en phonétique, peuvent être subjectifs et influencés par les différents dialectes ou régions de langues.

- Ce tableau ne prend pas en compte les variations qui se sont produites au cours des siècles, comme des formes archaïques d'expression.
- L'évolution peut ne pas être linéaire ; les changements peuvent s'effectuer par étapes et être influencés par divers facteurs culturels, sociaux et linguistiques.

Notre agent:

mansionem manso. 'nɛ > maison maj.sɔ̃

IIe et le IVe siècle

manso. 'nɛ

- Reduction de deux diphthongues, oe et ae

IIe et le IVe siècle

mansom. 'nɛ → manso. 'ne

- /mn/ > /nm/

avant le IIIe s.

mansom (en supposant l'effacement de la voyelle finale)

- Effacements précoce de voyelle finale

IIe et le IVe siècle

mansom → mansøm

- /oe/ > œ (ouvert ou fermé)

VIIe-VIIIe s.

mansöm → mansöm

- Effacement de la voyelle finale des paroxytons et atones

XVe siècle (hypothétique)

mansöm → maisonem

- mutation de la voyelle finale en -em-

XVIe siècle

maisonem → maisonem → maison

- /e/ > œu /ieu > œ/yœ

- mutation de la voyelle finale en -ion-

XVe siècle (hypothétique)

maisonem → maison

- mutation de la voyelle finale en -em- ou en -ion-

XVI^e siècle

maison (en supposant l'effacement de la voyelle finale)

- mutation de la voyelle finale en -em- ou en -ion-

XVI^e siècle

maison (en supposant l'effacement de la voyelle finale)

- /a/ > /e/ dans proparoxytons, sauf après les diphthongaisons de o ouvert

XVI^e siècle

maison (en supposant l'effacement de la voyelle finale)

- /i/ et /u/ > ü

- Effacement de la voyelle finale des paroxytons et atones

Correction du jury dans le rapport de 2024:

Le jury ne peut que rappeler, cette année encore, la nécessité de s'entraîner régulièrement à faire des évolutions complètes de mots, telles qu'on peut les trouver dans bon nombre de manuels de phonétique historique. La pratique régulière de l'exercice permet d'acquérir des automatismes très utiles le jour de l'épreuve ; par ailleurs, la question de phonétique historique au concours ne cherche pas à piéger les candidats : elle porte sur des évolutions classiques, sur lesquelles on peut travailler, à l'aide desdits manuels, pendant l'année de préparation. Si le jury a été indulgent sur les détails de l'évolution de *s + y* dans *ma(n)sionem* > *maison*, il ne peut excuser ni les incohérences conduisant à déplacer l'accent en cours d'évolution, ni l'oubli de phénomènes courants dans l'histoire phonétique du français comme la diphtongaison spontanée de la voyelle [o], tonique et libre, ou la nasalisation du produit de cette diphtongaison. Il était également regrettable de ne pas tenir compte des indications du sujet : dès lors que [n] était placé entre parenthèses dans l'étymon *ma(n)sionem*, on pouvait difficilement le maintenir jusqu'à l'ancien français comme si de rien n'était : ce [n] implosif disparaît à date latine, dès le début de l'évolution, et c'est bien cela qu'indiquent les parenthèses. Ont donc été valorisées les copies qui proposaient des explications claires et raisonnées de l'évolution phonétique de *ma(n)sionem* jusqu'à *maison*. On a valorisé également la distinction nette entre transcription phonétique et graphies : les bonnes copies sont celles qui savent expliquer les différentes étapes de l'évolution et en donner une transcription phonétique correcte. Si de surcroît elles commentent avec pertinence les graphies qui en résultent, c'est parfait !

Proposition de corrigé

Mansionem [mansíoñem] > *maison* [męz̪ɔ̄].

	[mansíoñem]	- Accent : l'accentuation de l'aboutissement français sur la dernière voyelle prononcée [ɔ̄] permet de placer l'accent sur la pénultième latine [o]. Comme cette dernière est en syllabe ouverte, elle est forcément longue par nature ⁸ . - Quantités étymologiques : la voyelle [i], en hiatus, est nécessairement brève. - <i>m-</i> initial, en position forte, se maintient sans changement du latin au français moderne.
II-I ^{er} s. avt	[ma(n)síoñem]	- Amuïssement de [n] implosif devant [s], en latin vulgaire et dès l'époque archaïque. La langue savante tente de rétablir la prononciation de ce [n], mais n'y réussit que pour les préfixes <i>in-</i> et <i>con-</i> : il ne sera pas restitué dans <i>ma(n)sionem</i> . - Amuïssement de la consonne finale [m].
I ² avt	[masyóne(m)]	- Consonification de la voyelle en hiatus : [i] en hiatus passe à la semi-consonne [y].
II ^e s.	[masyóne]	- Bouleversement vocalique : l'opposition des voyelles selon leur quantité (voyelle longue vs voyelle brève) n'est plus pertinente. Elle est remplacée par une opposition d'aperture (voyelle ouverte vs voyelle fermée) : [ö] > [ɔ̄]. - Bouleversement quantitatif : [ö], voyelle tonique et libre, devient longue. À l'inverse, les voyelles atones [a] et [e] deviennent brèves. - Palatalisation de [s] suivi de yod : . la constrictive alvéolaire sourde recule son point d'articulation jusque dans la zone médiopalatale. La palatalisation de ce phonème se fait difficilement : on aboutit à une semi-palatale notée [s'] ⁹ .
	[ma's'óne]	

		. afin de faciliter l'articulation de ce phonème, un [i] de transition se développe à l'avant de la consonne palatalisée ¹⁰ .
IV ² s.	[maiz'óne]	- ce [i] forme avec la voyelle qui précède une diphongue par coalescence [ai].
VI ^e s.	[maiz'óøne]	- Sonorisation de la semi-palatale sifflante, sourde, qui se trouve en position intervocalique : [s'] > [z'].
	[maiz'óune]	- Début de la diphongaison spontanée française :
		. La voyelle [ó], longue depuis le bouleversement quantitatif, s'allonge jusqu'à se segmenter.
		. Différenciation : le second segment se ferme d'un degré : [ø] > [u].
VII ^e s.	[maizóun]	- Dépalatalisation de la constrictive alvéolaire sonore.
X ^e s.	[maizóün]	- Chute de la voyelle finale atone.
XI ² s.	[mēizóün]	- Nasalisation du second segment de la diphongue, sous l'influence de la consonne nasale [n].
XII ^e s.	[mēzóün]	- Diphongue par coalescence : sous l'influence du second segment, très fermé, le premier segment se ferme d'un degré.
		- Monophtongaison, par assimilation du second segment au premier, de la diphongue par coalescence.
		- Nasalisation du premier segment de la diphongue issue de la diphongaison française de [o] fermé.
XII ² s.	[mēzóøn]	- Monophtongaison de la diphongue nasalisée :
		. sous l'influence du premier segment de la diphongue, le second segment s'ouvre d'un degré ;
		. puis le second segment est assimilé par le premier.
XIII ^e s.	[mēzõn]	- Effet ouvrant de la nasalisation : [ó] > [ø].
XVI ² - XVII ¹	[mēzõ]	- Dénasalisation partielle : en syllabe fermée, la voyelle garde son timbre nasal ; la consonne, en finale absolue (position fragilisante), s'amuït.
		Remarques sur la graphie : le digramme -on transcrit la voyelle nasalisée [ø], tandis que le digramme -ai-, graphie conservatrice, transcrit e ouvert [e]. Enfin, en position intervocalique, s correspond à la sifflante sonore [z].