La théorie des fonctions lexicales appliquée pour tester des grands modèles de langage: une approche par verbalisation

Viviane Binet (20244728)[1], Alessandra Mancas (20249098)[1], sup. Philippe Langlais[1]

[1]Département d'informatique et de recherche opérationnelle (DIRO)

20 décembre 2024

Résumé

Notre recherche s'inscrit dans le domaine du traitement du langage naturel. Nous observons, notamment, les fonctions lexicales (FLs) telles que décrites par Igor Mel'čuk et Alain Polguère dans la Théorie Sens-Texte pour se demander : pouvons-nous verbaliser les FL, de sorte que les relations entre lexies soient compréhensibles par un modèle de langue large ? Pour y répondre, nous avons tenté de verbaliser treize fonctions lexicales. Notre jeu de données provient du Réseau Lexical du Français (RL-fr), une ressource (non-exhaustive) qui fournit une représentation de la langue française sous forme d'un graphe. Les noeuds (les lexies) sont reliés, entre autres, par les fonctions lexicales. Nous avons généré plusieurs échantillons aléatoires correspondant aux fonctions lexicales de notre choix. Sur ces derniers, nous avons évalué les performances du modèle de langage Llama3.2. On a créé une série de questions correspondant à chaque FL, ainsi qu'un ensemble d'exemples pour poser ces questions en *k-shots*. Nous avons ensuite comparé les réponses du modèle à nos questions avec celles "attendues" du RL-fr. À la fin de ce projet, nous concluons que même si le modèle n'a pas de très bonnes performances objectives, llama3.2 semble comprendre, plus ou moins, les relations entre lexies telles qu'énoncées par les FLs.

1 INTRODUCTION

1.1 Qu'est-ce qu'une fonction lexicale ?

La théorie Sens-Texte (TST ou ST) est une approche linguistique proposée d'abord par Igor Mel'čuk en 1965, puis approfondie par Alain Polguère. Son objectif principal est de formaliser les représentations d'énoncés linguistiques entre eux par rapport à leurs représentations textuelles, puis par rapport à leurs sens. Elle vise à faire cette formalisation de la manière la plus "universelle" possible, afin de construire des modèles linguistiques généralisables à toute langue humaine [4, p.2] en reposant sur des concepts élémentaires. Elle utilise un langage formel pour représenter les énoncés linguistiques et encoder leurs règles de manipulation et leurs interactions [4, p.3]. Ce formalisme rend les composantes d'un modèle ST calculables par un système logique, tel un ordinateur ou un programme. (idem)

Les fonctions lexicales (FLs) sont un outil développé dans le cadre de l'approche ST qui permet de modéliser formellement des phénomènes linguistiques fondamentaux de toute langue. En effet, selon Mel'čuk et Polguère, les FLs sont "une partie intégrante de toute langue" [2, p.76]. Pour les comprendre et, notamment, les développer, il faut une bonne maîtrise de notions linguistiques telles la sémantique, la syntaxe, la morphologie, la syntaxe de dépendance, etc... (*idem*)

> "Une fonction lexicale f appliquée à l'unité lexicale L – ce qui est noté f(L) – fournit pour le sens 'σf' [c'est à dire le contenu sémantique] associé à f un ensemble d'expressions alternatives de ce sens dont la sélection est contrainte par L." [2, p.75]

Les FLs peuvent être vues comme des fonctions mathématiques ou des méthodes au sein d'un programme. Elles retournent une liste unique de lexies selon leur comportement et la lexie fournie en argument. Notons également que les valeurs d'un ensemble de retour pour une lexie donnée sont plus ou moins synonymes [2, p.80]. De plus, la TST définit un langage formel et des raccourcis pour représenter chaque FL ainsi que les différences subtiles entre elles.

Par exemple :

$\mathtt{A_1(}$*douter*$\mathtt{)\ =}$ *dubitatif, dans le doute*

Ici, $\mathtt{A_1}$ est la FL désignant le qualificatif adjectival typique ($\mathtt{A_i}$) du premier actant syntaxique profond (le sujet) de son argument ($\mathtt{i} = 1$). On peut, en effet, dire que "X (le sujet) est *dans le doute*". [2, p.79]

Pour ce projet, nous nous en sommes tenues aux FLs dites *standard*. Ces dernières ont les propriétés d'admettre un grand nombre d'arguments (pour être compatible avec beaucoup de lexies), puis de retourner un grand nombre de valeurs différentes (pour ne pas être une relation triviale) [2, p.83]. Ainsi, elles ont une nature universelle, c'est-à-dire qu'on peut les retrouver dans la plupart des langues du monde (*idem*).

Nous avons travaillé avec les données du *Réseau Lexical du Français* (RL-fr) (v.3.1). Le RL-fr est un "modèle formel du lexique du français contemporain". Selon Polguère, on peut voir la langue comme étant un graphe, appelé Système Lexical, dont les noeuds sont des lexies (ou des collocations) et les arcs sont les liens entre ces dernières au moyen des FLs [3, 5]. Via la plateforme Ortolang [1] , nous pouvons gratuitement télécharger une base de données sous la forme de fichiers `.xml` et `.csv`, qui décrivent respectivement divers systèmes et hiérarchies (comme les FLs), puis les données du modèle (les noeuds et leurs propriétés, leurs relations).

1.2 Motivation

L'objectif de notre projet est de vérifier si un modèle de langue permet de retrouver en partie l'information du RL-fr en lui posant des questions. En effet, nous avons utilisé les fonctions lexicales présentées dans la théorie des fonctions lexicales ainsi que leurs exemples pour voir si il était possible de créer des questions qui généralisaient bien chaque fonction, peu importe la classe du mot en entrée (verbe, nom, clausation, etc). De plus, nous voulions pouvoir mesurer la performance d'un modèle

1. https ://www.ortolang.fr/market/lexicons/lexical-system-fr

de langage avec certaines questions définies à l'avance. Nous voulions aussi voir si il était possible d'imposer une forme de sortie au modèle (réponse en un seul mot en français).

2 MÉTHODOLOGIE

2.1 Exploration

Nous avons commencé en lisant *Les fonctions lexicales dernier cri* de Mel'čuk et Polguère pour se familiariser avec la théorie des fonctions lexicales. Nous avons aussi pris connaissance de la documentation de la base de données du RL-fr afin de comprendre comment aller chercher les exemples de mots d'entrée et de sortie des fonctions lexicales.

2.1.1 L'API

Alexander Petrov, un étudiant à la maîtrise sous la direction du professeur Langlais, nous a partagé une version d'un API qu'il a créé pour lire la ressource dans le cadre de sa recherche. Il convertit les fichiers du RL-fr en Dataframes puis emploie des dictionnaires Python pour rapidement chercher et manipuler les informations en temps constant.

On s'en est servi pour :

— Obtenir une lexie et le nom d'une relation grâce à leurs identifiants.

— Obtenir un dictionnaire qui indique toutes les paires source-cible et leur relation à partir du fichier `15-lslf-rel.xml`.

— Distinguer les FLs paradigmatiques des FLs syntagmatiques.

Quant aux changements du code source, nous avons d'abord ajusté les méthodes existantes pour accélérer leur exécution puis on a ajouté des méthodes qui nettoient les données et qui génèrent des échantillons sur lesquels nous allons tester notre modèle.

Voici un aperçu de nos ajouts :

— Générer des ensembles aléatoires d'échantillons selon des critères donnés : le nombre minimal d'exemples disponibles, la taille de l'ensemble, le nombre d'ensembles générés. Mettre ces derniers dans un (ou plusieurs) fichiers .csv qu'on utilisera plus tard pour générer les requêtes.

Avoir plusieurs ensembles aléatoires nous permettra d'évaluer le comportement du modèle sur des données plus variées. Cela nous permettra d'avoir une vue plus globale sur le comportement du modèle en combinant les résultats des ensembles.

— Supprimer des tables originales les rangées dont les termes contiennent des caractères numériques.

Durant notre exploration, nous avons constaté que, la plupart du temps, les réponses du modèle ne contiennent pas des chiffres. De plus, notre but est de verbaliser les mécanismes linguistiques des FLs. Parmi les milliers de noeuds du RL-fr, nous avons décidé que la fraction qui contient des chiffres n'est pas pertinente pour notre analyse.

Par exemple, le RL-fr propose `Syn(`*trois*`) =` *3*.

— Supprimer des tables originales les instances d'exemples que nous avons utilisé pour effectuer le roulement en k-shots. Comme les exemples sont donnés dans la requête, on veut éviter de

fournir la réponse au modèle par accident.

Voici un extrait du fichier .csv des exemples qui est généré par l'API, pour la relation `Anti`. Chaque ligne représente les résultats possibles d'une lexie fournie en entrée selon le RL-fr. Le premier mot est le mot "source", puis tous les suivants sont les "cibles" de la FL précisée plus haut. Notons que les mots cibles ont, entre eux, une certaine distance sémantique, tel que décrit par le séparateur "," (faible distance), " ;" (distance assez notable) et "<" (distance correspondant à une variation d'intensité) [6, p.22]. Bref, les lexies cibles sont toujours plus ou moins des synonymes.

```
>>> Anti
oublier ,se souvenir ,se rappeler
tendreté ,dureté
ordinaire ,extraordinaire
```

2.1.2 Choix des fonctions lexicales

Une fois que nous avions un fichier traitable, nous devions décider quelles fonctions nous voulions verbaliser et tester. Un critère fut de prendre les fonctions ayant au moins 70 exemples, afin d'avoir un échantillon suffisamment grand. De plus, en pensant à des questions à poser aux modèles de langage qui verbalisaient les fonctions, nous nous sommes rendues compte que certaines des FL étaient difficiles à généraliser à tout type de lexie, voire même difficile à comprendre pour nous. Par exemple, $\texttt{Conv}_{\texttt{ij}}$ est décrite comme

> $\texttt{Conv}_{\texttt{ij}}$(L) est un substitut sémantique possible de L modulo une permutation actancielle $L_{(ij)} \rightarrow f(L)_{(ji)}$ au niveau syntaxique profond."

Voici quelques exemples :

> $\texttt{Conv}_{21}$(inclure) = appartenir, faire partie
> $\texttt{Conv}_{321}$(prêter) = emprunter

Nous n'avons pas verbalisé cette FL car nous avions de la difficulté à bien la comprendre. En effet, notre compréhension personnelle des FLs fut un facteur important dans nos choix de questions. Pour certaines autres fonctions, le problème n'était pas de comprendre mais plutôt de créer une question qui était assez généralisée pour fonctionner avec tous les mots possibles en entrée, et qui était suffisamment précise pour refléter le sens de la fonction lexicale.

En regardant les FLs dans la ressource [2, p. 95], nous avons d'abord choisi celles qu'on pourrait mettre en nos mots le plus facilement possible, afin de générer des questions représentatives.

Notre choix fut d'abord motivé par notre compréhension personnelle des FLs, c'est-à-dire des mécanismes sémantiques sous-jacents de chacune et notre habileté à les mettre en nos propres mots. Nous en avons également choisi qui sont plus complexes (`Contr`, $\texttt{A}_2\texttt{Perf}$), pour voir si, même avec une explication possiblement plus maladroite, le modèle pourrait comprendre notre intention.

On aimerait également souligner que certaines FLs sont d'usage plus marginal que d'autres, puis cela pourrait affecter notre compréhension de leur comportement [2, p.96].

Les FLs standards se divisent d'abord en deux catégories : simples et complexes. Une FL *simple* accorde un certain sens à une lexie, de manière directe et singulière. Une FL *complexe* s'agit d'une

"union" de plusieurs sens d'une FL. Par exemple, dans `MagnSon(`*moteur*`) =` *hurler, rugir* on dit qu'un *moteur* qui produit un son plus fort (`Magn`) hurle ou rugit [2, p. 90]. Pour faciliter leur explicativité, nous avons, pour la plupart, regardé les FLs simples (sauf $\texttt{A}_2\texttt{Perf}$)

Ensuite, les FLs standards peuvent également être classées comme étant *paradigmatiques* (para) ou *syntagmatiques* (synta).

Une FL para modélise la dérivation sémantique entre deux lexies, c'est-à-dire que dans une FL telle que f(L) → L', L' est sémantiquement construite à partir de L. Simplement dit, on choisit un nouveau mot qui est dérivé du premier [2, p.85-86]

— $\texttt{S}_1$ (une FL para) : nom d'agent dans $\texttt{S}_1$`(`*soigner*`) =` *médecin* et $\texttt{S}_1$`(`*lire*`) =` *lecteur*

Les FLs synta, quant à elles, modélisent les collocations (juxtapositions), de lexies. Toutes les lexies dans un syntagme L' retourné par une FL L → L' synta sont en co-occurence avec L. Ainsi, chaque FL synta modélise un type de collocation [2, p.86-87]. Par exemple, `Magn(`*boire*`) =` « *comme un trou* » représente l'intensification dans l'expression "boire comme un trou". On ajoute ici L' à la suite de L.

On a choisi de travailler uniquement avec les FLs para, car la variabilité des réponses possibles est très élevée lorsqu'on travaille avec des collocations. (et, selon notre méthode d'évaluation, les scores du modèle seraient probablement moins bons par rapport aux réponses du RL-fr, car la ressource n'est pas exhaustive).

2.1.3 Construction du modèle

Ensuite, une fois que nous avions des échantillons de mots pour chaque FL que nous voulions tester, nous avons pu commencer à nous familiariser avec les outils de requêtes aux modèles de langages. Nous avons tout d'abord utilisé la fonctionnalité *chat* d'Ollama pour tester les réponses aux questions que nous avions composées pour quelques FL. Ollama est un *framework* pour rouler des modèles de langages sur une machine locale. Toutefois, c'était plutôt lent, en prenant plusieurs minutes pour traiter une cinquantaine de requêtes (et donc une cinquantaine de mots). Nous avons considéré utiliser ChainForge, un environnement de programmation visuel et *open source* pour évaluer des modèles de langage. Toutefois, ce n'était pas adapté pour nous, puisqu'il aurait fallu mettre nos données dans un nouveau format .xml et donc recommencer le processus de formattage des entrées et des sorties. De plus, ça ne nous aurait pas nécéssairement permis un gain d'efficacité. Afin de pouvoir rouler une plus grande quantité d'exemples de façon locale et sans GPU, Philippe Langlais nous a recommandé le traitement en batch d'Ollama, qui permet de créer un modèle à partir d'un autre modèle (ici, llama3.2). Nous avons spécifié une question générale au modèle ainsi que certains paramètres, et nous posions les questions directement à notre modèle, ce qui a accéléré le processus.

Voici le modèle que nous avons construit pour l'interroger.

```
FROM llama3.2
PARAMETER num_ctx 500
SYSTEM """ Tu es un modèle en français, ne me donne que des réponses d'un mot.
Réponds à la question passée en input en français en un mot. """
```

Notre modèle est construit à partir de llama3.2, et nous avons modifié le paramètre num_ctx. Ce paramètre représente la grandeur de la fenêtre de contexte utilisée pour générer la prochaine réponse. La valeur par défaut est 4096, donc en la réduisant, notre modèle se base moins sur les questions et réponses précédentes pour générer les prochaines réponses. Ensuite, dans le contexte général donné au modèle, nous spécifions qu'il doit répondre en français en un mot car c'est le seul format que nous pouvons traiter. Nous ne voulions pas que les réponses contiennent par exemple : "Voici la réponse : [*réponse*]" ou des retours à la ligne.

2.2 Choix des exemples ajoutés

Nous avons décidé d'ajouter des exemples avec les questions sous les suggestions de Philippe Langlais. Avant sa suggestion, nous avions intégré un exemple à une question pour voir le résultat, sans connaître le concept de k-shot. En effet, ceci correspond à ajouter k exemples à la question posée au modèle afin que la question soit plus claire. Nous avons testé chaque question avec aucun exemple, 1 exemple, 3 exemples et 5 exemples. Pour choisir les exemples, nous avons pris 5 mots mis en exemple pour chaque relation dans le papier de Mel'čuk et Polguère [2], car nous pensions que ces auteurs sont mieux placés que nous pour choisir des exemples représentatifs. S'il n'y avait pas d'exemples fournis dans le document pour une certaine FL, on a parcouru les noeuds du RL-fr et nous avons alors choisi les lexies qui semblaient bien représenter les cas possibles d'une FL. Pour chaque question pour une même relation, les mêmes exemples sont utilisés parmi les 5 exemples pris dans la ressource. L'ordre des exemples dans notre liste (et l'ordre dans lequel ils seront fournis à chaque k-shot) est déterminé par la précision de l'exemple. Le premier est une lexie qui représente plus généralement le fonctionnement de la FL, puis le cinquième est un cas plus particulier. On voulait ajouter de la complexité aux exemples plus on en donne au modèle.

2.3 Rédaction des questions

Toutes les questions qu'on envoie au modèle suivent la même structure. La première partie s'agit d'une "traduction" de la FL, suivie d'une instruction pour la sortie attendue. Dans la première partie, on varie entre l'usage du terme "mot" et des termes associés aux parties du discours de l'entrée d'une FL (nom, adjectif, verbe...). Cette deuxième partie dépend des parties du discours qui sont ciblées par la FL. Par exemple, la FL d'adjectivisation retourne toujours un adjectif, que son entrée soit un nom, un verbe, un adverbe ou une clausation. Ainsi, on demanderait au modèle quelque chose comme :
`Quel est l'adjectif correspondant au mot [x]? Donne un seul adjectif sans ponctuation.`
Pour les requêtes de chaque k-shot, nous avons ajouté k exemples correspondant à la FL, tels que :

```
Quel est l'adjectif correspondant au mot [x]? Donne un seul adjectif sans ponctuation.
Voici 3 exemples:
Pour "temps", la réponse est "temporel".
Pour "se nourrir", la réponse est "nutritionnel".
Pour "rotation", la réponse est "giratoire".
```

Nous précisons ici au modèle qu'il devrait nous répondre avec un adjectif.

Pour lire toutes les questions que nous avons posé au modèle, voir l'annexe.

2.4 Construction des tests

Pour tester les questions, nous avons créé trois échantillons de 50 entrées choisies de manière aléatoire pour chaque fonction lexicale. Chaque entrée provient de la ressource du RL-fr et contient un mot source ainsi que les mots en sortie possibles, pour la fonction lexicale correspondante. Nous avons choisi de générer des échantillons de taille 50 car nous voulions des échantillons de la taille la plus grande possible pour avoir des résultats plus représentatifs, tout en gardant une vitesse d'exécution plutôt rapide puisque nous avions beaucoup de questions à tester.

Chaque question a donc été testée sur 12 lots de 50 mots, c'est-à-dire une fois pour chaque échantillon (3 échantillons), et cela pour 0-shot, 1-shot, 3-shot, 5-shot.

Voici un extrait d'un fichier de sortie pour la relation `Anti` (1-shot) qui suit la structure :
`mot source : mot(s) cible(s) : réponse du modèle`

```
inintelligence : ,intelligence : Intelligence
trouvable : ,introuvable : Indisponible
démarrer : ,arrêter ,s'arrêter : Arrêter
déraison : ,raison : Raisonnement
```

2.5 Calcul des scores

Pour chaque échantillon de 50 mots, le score de l'échantillon représente la proportion de réponses données par le modèle qui font partie des réponses contenues dans la ressource. Le score représente donc l'exactitude des résultats. Par exemple, si pour le mot "voler", la réponse de la ressource est "aile, avion" et que la réponse du modèle est "Avion", alors c'est une bonne réponse. La casse n'est pas prise en compte dans le calcul du score, c'est-à-dire que toutes les lettres sont transformées en minuscules avant de comparer les mots. Si la réponse du modèle avait été n'importe quel mot différent de ceux de la ressource, ce serait compté comme une mauvaise réponse. Les scores pour chaque question ont été calculés en faisant une moyenne sur les scores des 3 échantillons. La variance est celle entre les 3 échantillons aussi. La "meilleure question" pour une relation donnée avec un certain nombre d'exemples est celle dont le score moyen sur 3 échantillons est le meilleur.

3 RÉSULTATS ET DISCUSSION

3.1 Présentation des résultats

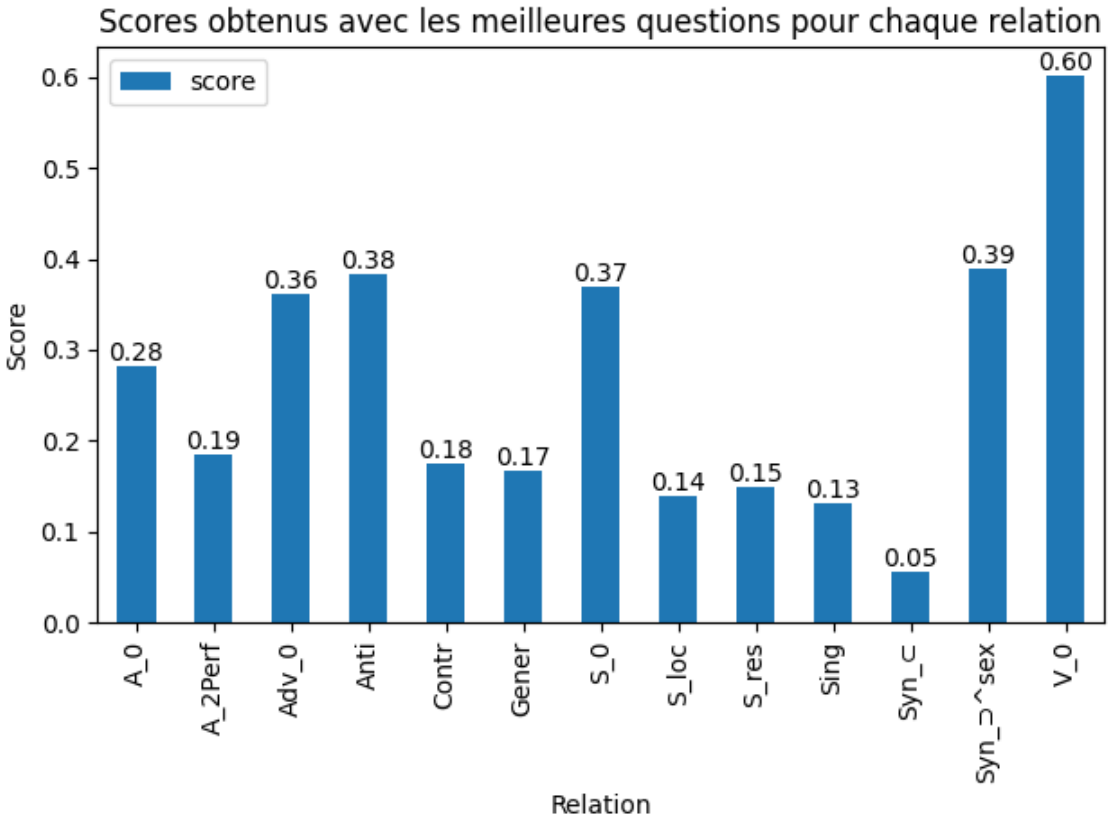

FIGURE 1 – Meilleur score pour chaque relation

3.2 Interprétation des scores

Comme vu dans la figure 1, certaines fonctions lexicales ont donné de meilleurs résultats que d'autres. La fonction avec le meilleur résultat est $\texttt{V}_0$, avec un score moyen de 0.60 pour tous les échantillons. Plus de détails sur $\texttt{V}_0$ se trouvent dans la figure 3. D'autres fonctions qui ont eu des scores plus élevés sont $\texttt{Syn}_{\supset sex}$ (score 0.39), $\texttt{Anti}$ (score 0.38), $\texttt{S}_0$ (score 0.37) et $\texttt{Adv}_0$ (score 0.36). Certaines des fonctions lexicales ont donné des scores beaucoup plus bas, comme $\texttt{Syn}_{\supset}$, avec un score moyen de seulement 0.05.

Nous avions fait l'hypothèse que les tests avec des exemples (1-shot, 3-shot, 5-shot) auraient des résultats significativement meilleurs que ceux sans exemples (0-shot). Pourtant, on observe que l'ajout des exemples n'a pas un grand effet mélioratif. Il est possible que la façon dont nous avons ajouté les exemples soit mélangeante pour le modèle de langage et qu'il n'intègre pas correctement les exemples dans son traitement des questions.

Dans la Figure 2, on s'intéresse à la variation des scores obtenus avec chacune des meilleures questions (décrites ci-dessus) par k-shot. La variation entre les scores des k-shots pour une meilleure questions n'est pas élevée, tel que décrit dans la Table 1, où la variance la plus élevée fut de 0.0044.

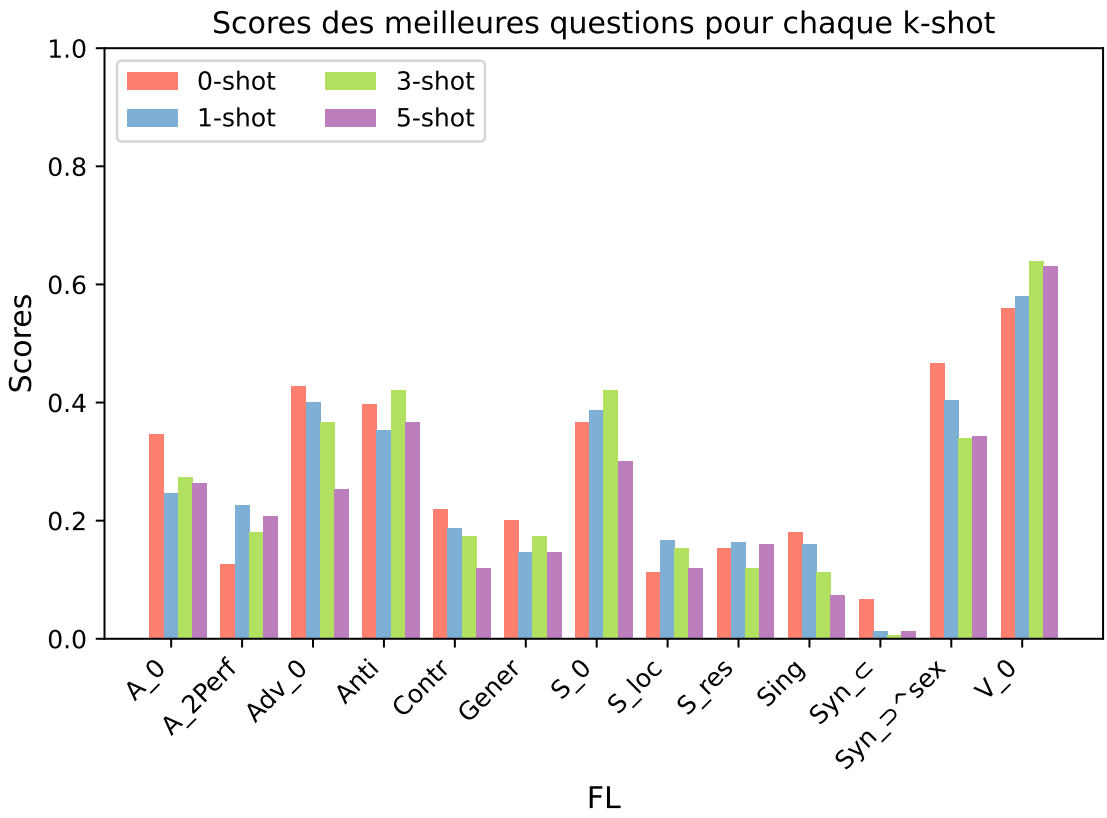

FIGURE 2 – Scores des meilleures questions pour chaque k-shot

En combinant ce tableau avec la Figure 2 on constate également que la plupart des "meilleures questions" ont le mieux performé lorsque k=0 (7 sur les 13 FLs).

Donc, contrairement à nos attentes, les k-shots n'ont pas toujours aidé les performances du modèle.

3.3 Limitations

Une des limitations à nos expériences était notre compréhension des fonctions lexicales. Nous n'avons pas testé une grande partie des fonctions parce que nous avions de la difficulté à les comprendre et à les généraliser. Pour celles que nous avons testé, il aurait probablement été possible de former de meilleures questions si notre compréhension des fonctions lexicales avait été meilleure.

De plus, nous avons fait nos tests uniquement avec llama3.2, nous n'avons pas utilisé d'autres modèles. Certains modèles de langage comme GPT auraient probablement donné des scores plus élevés [1, p. 7]. De plus, cela nous aurait permis de comparer les performances des modèles sur les mêmes tâches. Il est également possible que le modèle que nous avons étayé puisse être raffiné en ajustant davantage ses paramètres, comme nous avons fait pour `num_ctx`.

TABLE 1 – Meilleur score et son k-shot correspondant pour la meilleure question de chaque FL

FL	Question	k-shot	Score	σ^2(score)
A_0	2	0	0.3467	0.0015
A_2Perf	6	1	0.2267	0.0014
Adv_0	2	0	0.4267	0.0044
Anti	2	2	0.42	0.0007
Contr	2	0	0.22	0.0013
Gener	4	0	0.2	0.0005
S_0	4	2	0.42	0.0019
S_loc	2	1	0.1667	0.0005
S_res	2	1	0.1633	0.0003
Sing	3	0	0.18	0.0017
Syn_$\subset$	2	0	0.0667	0.0006
Syn_$\supset$ˆsex	4	0	0.4667	0.0027
V_0	2	2	0.64	0.0011

Sur chaque ligne, on a la fonction lexicale traitée, et le numéro de la question pour cette fonction qui a produit le meilleur score parmi les questions de cette fonction (Voir la table 2 pour les questions correspondantes). Ensuite, il y a le nombre d'exemples fournis pour avoir ce score (le meilleur score pour cette fonction). Il y a le score lui-même qui est la moyenne sur 3 échantillons. Il y a la variance du score qui est comptée sur les 3 échantillons qui ont servi pour la moyenne.

En observant les mots en sortie donnés par llama3.2, nous avons fait le constat que certains mots pourraient être une sortie valide d'une fonction lexicale donnée, mais ne figurent pas dans la ressource. Par exemple, pour la fonction `Gener`, on cherche un mot qui généralise le mot en entrée. Par exemple, dans la ressource, on a :

`Gener`(capitaine) = officier

Toutefois, la sortie du modèle était "commandant". Nous nous sommes demandées si le mot "commandant" ne serait pas une sortie valide elle aussi et si la ressource pourrait être complétée avec davantage de mots. Il y avait plusieurs cas similaires.

Nous avons nettoyé les mots en entrée en enlevant ceux qui contenaient des nombres, mais nous aurions pu également enlever ceux qui contenaient des collocations. En effet, les expressions contenant plusieurs mots ont souvent donné des résultats erronés de la part du modèle, et il y avait plusieurs collocation que nous n'avions jamais entendues (par exemple "chiffon rouge", "pisseur de copie"), ce qui pourrait, encore une fois, affecter notre compréhension des FLs.

Enfin, comme nous n'avions pas accès à un ordinateur avec GPU et nous ne voulions pas abîmer nos ordinateurs personnels, nous avons dû contraindre notre projet à une plus petite échelle (3 échantillons aléatoires seulement, un modèle simple, une seule exécution par question pour chaque échantillon).

4 CONCLUSION

Notre démarche nous a permis de réaliser que verbaliser des fonctions lexicales est une tâche difficile pour nous et que le modèle que nous avons testé, llama3.2, performait en général plutôt mal pour les questions que nous lui demandions. Afin de nous aider dans notre compréhension des FL on aurait pu collaborer avec des linguistes de l'OLST ou qui ont travaillé sur le RL-fr. Il serait aussi intéressant de voir si la ressource pourrait être augmentée en utilsant à des modèles de langage.

Une autre approche aurait pu être de ne pas tenter de créer une question qui se généralise bien pour tout type d'entrée (partie du discours, par exemple) mais plutôt des questions plus spécifiques et de trouver une façon de séparer les mots des échantillons pour faire correspondre chaque question à ces derniers.

Il serait aussi intéressant d'étudier plus en profondeur les raisons pour lesquelles certaines questions fonctionnent mieux que d'autres pour une fonction donnée. En effet, il y a parfois de grandes différences de scores entre les questions qui ont mieux fonctionné et celles qui ont moins bien fonctionné, parfois en changeant quelques mots dans la question. De plus, on pourrait analyser davantage l'effet de l'aléatoire dans les scores, en roulant plus de fois le modèle sur le même échantillon de mots par exemple. Ainsi, on pourrait voir si le modèle est consistant dans ses réponses ou si le facteur aléatoire prend beaucoup d'importance.

5 TABLES ET FIGURES

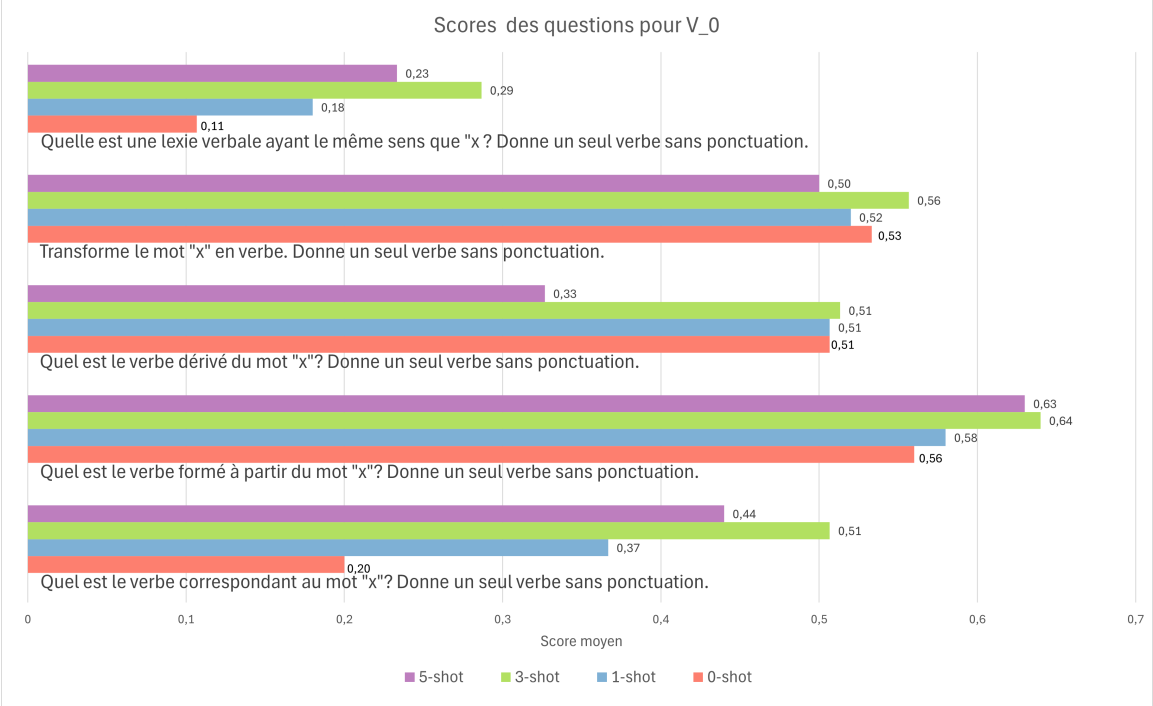

FIGURE 3 – Scores des questions pour V_0

Dans cette figure, les cinq questions ayant servi à tester la fonction lexicale V_0 sont présentées avec leurs scores respectifs. On voir que certaines questions fonctionnent mieux que d'autres, par exemple la question "Quelle est une lexie verbale ayant le même sens que "x ? Donne un seul verbe sans ponctuation." a donné un score maximal de 0.29.

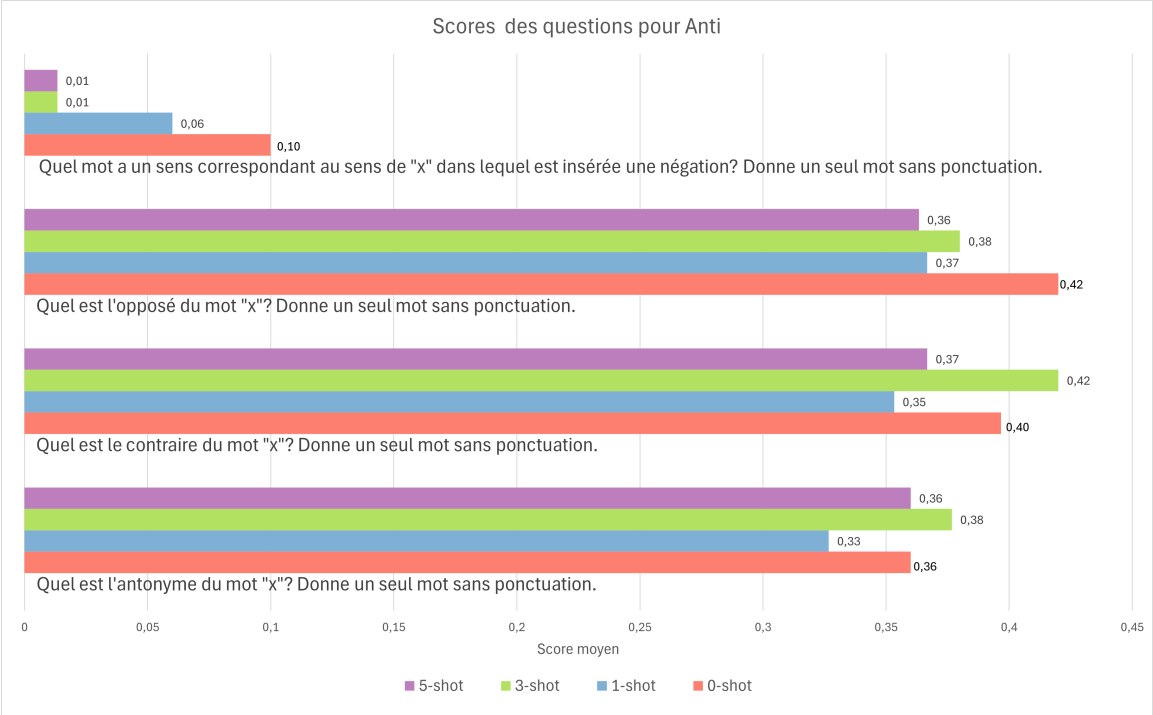

FIGURE 4 – Scores des questions pour `Anti`

Dans cette figure, les cinq questions ayant servi à tester la fonction lexicale `Anti` sont présentées avec leurs scores respectifs.

6 ANNEXE

6.1 Exemple de sortie du modèle

Voici la sortie du modèle pour la relation V_0 (la verbalisation), avec la question "Quel est le verbe formé à partir du mot "x" ? Donne un seul verbe sans ponctuation." et en ne fournissant aucun exemple. La structure d'une ligne est la suivante :
mot source : mots cibles contenus dans la ressource : réponse du modèle.

```
dévoration : ,dévorer ,dévorer : devorer
   envol : ,s'envoler : Envoler
   vomissement : ,vomir : Vomitait
   appauvrissement : ,s'appauvrir : Pauvre
   conception : ,concevoir : Concevoir
   condition : ,conditionner : Conditionner
   ski : ,skier : Skier
   marquage : ,marquer : Marque
   maquillage : ,maquiller : Maquillage
   fourniture : ,fournir : fournit
```

trébuchement : ,trébucher : Tomber
ordre : ,ordonner : ordonner
persuasion : ,persuader : Persuader
assassinat : ,assassiner : Tuer
espoir : ,espérer ,espérer : Espérer
description : ,décrire : Décrit
indice : ,indiquer : Décliner
provocation : ,provoquer : Provocer
admiration : ,admirer : Admire
séjour : ,séjourner : Demarrer
contradictoire : ,contredire : Contrarier
agacement : ,s'agacer : Agace
détournement : ,détourner ,détourner : Détourner
union : ,unir : Unirse
grognement : ,grogner : Gronder
privation : ,priver : Privaison
gonflage : ,gonfler : Gonfle
chuchotement : ,chuchoter : Chuchoter
éloignement : ,s'éloigner : Éloigner
atterrissage : ,atterrir : Atterrir
vol : ,voler ,voler ,voler : Voler
moquerie : ,se moquer : Mojer
épuisement : ,s'épuiser : Épuiser
insistance : ,insister : Insiste
numérotation : ,numéroter : Numéroter
hâte : ,se hâter : Hater
aboiements : ,aboyer : Boyer
pique-nique : ,pique-niquer : ique
échappée : ,échapper : Échapper
léchage : ,lécher : Lècher
croissance : ,croître ;grandir : Gronde
approche : ,approcher : Approche
tremblement : ,trembler : Trembler
énumération : ,énumérer : Liste
affection : ,affectionner : Affectionne
jet : ,jeter : Jeter
jardinage : ,jardiner : Arroser
bénédiction : ,bénir : Bénir
descente : ,descendre : descend
exécution : ,exécuter : Exécute

6.2 Questions créées

Table 2 – Questions et leur numéro correspondant

FL	No	Question
A_0	0	Quel est l'adjectif correspondant au mot "x" ? Donne un seul adjectif sans ponctuation.
A_0	1	Transforme le mot "x" en adjectif. Donne un seul adjectif conjugué au masculin, et sans ponctuation.
A_0	2	Quel est l'adjectif formé à partir du mot "x" ? Donne un seul adjectif conjugué au masculin, et sans ponctuation.
A_0	3	Quel est l'adjectif dérivé du mot "" ? Donne un seul adjectif conjugué au masculin, et sans ponctuation.
A_0	4	Quelle est une lexie adjectivale ayant le même sens (ou presque) que "x" ? Donne un seul adjectif conjugué au masculin, et sans ponctuation.
A_2Perf	0	Quel est l'adjectif correspondant à l'aboutissement de "x" ? Donne un seul adjectif sans ponctuation.
A_2Perf	1	Quel est l'adjectif correspondant à la fin de "x" ? Donne un seul adjectif sans ponctuation.
A_2Perf	2	Quand on aboutit la chose suivante, quel est l'adjectif approprié pour le définir : : "x" ? Donne un seul adjectif sans ponctuation.
A_2Perf	3	Comment peut-on qualifier la chose suivante lorsqu'elle est aboutie : "x" ? Donne un seul adjectif sans ponctuation.
A_2Perf	4	Comment qualifie-t-on l'actant du mot "x" lorsqu'il finit l'acte associé au mot donné ?
A_2Perf	5	Quel est le qualificatif adjectival pour désigner l'aboutissement du fait "x" ? Donne un adjectif sans ponctuation.
Adv_0	0	Quel est l'adverbe correspondant au mot "x" ? Donne un seul adverbe sans ponctuation.
Adv_0	1	Quel est l'adverbe formé à partir du mot "x" ? Donne un seul adverbe sans ponctuation.
Adv_0	2	Quel est l'adverbe dérivé du mot "x" ? Donne un seul adverbe sans ponctuation.
Adv_0	3	Transforme le mot "x" en adverbe. Donne un seul adverbe sans ponctuation.
Adv_0	4	Quelle est une lexie adverbiale ayant le même sens que "x" ? Donne un seul adverbe sans ponctuation.
Anti	0	Quel est l'antonyme du mot "x" ? Donne un seul mot sans ponctuation.
Anti	1	Quel est le contraire du mot "x" ? Donne un seul mot sans ponctuation.
Anti	2	Quel est l'opposé du mot "x" ? Donne un seul mot sans ponctuation.
Anti	3	Quel mot a un sens correspondant au sens de "x" dans lequel est insérée une négation ? Donne un seul mot sans ponctuation.
Contr	0	Quel mot crée un contraste avec, mais n'est pas l'antonyme de, "x" ? Donne un seul mot sans ponctuation.
Contr	1	Quel est l'opposé, mais pas l'antonyme de, "x" ? Donne un seul mot sans ponctuation.
Contr	2	Quel mot est l'opposé de "x", sans être son contraire ? Donne un seul mot sans ponctuation.
Contr	3	Quel mot est en opposition contrastive sur le plan sémantique avec "x" ? Donne un seul mot sans ponctuation.
Gener	0	Quel mot crée un contraste avec, mais n'est pas l'antonyme de, "x" ? Donne un seul mot sans ponctuation.
Gener	1	Quel est l'opposé, mais pas l'antonyme de, "x" ? Donne un seul mot sans ponctuation.
Gener	2	Quel mot est l'opposé de "x", sans être son contraire ? Donne un seul mot sans ponctuation.
Gener	3	Quel mot est en opposition contrastive sur le plan sémantique avec "x" ? Donne un seul mot sans ponctuation.
S_0	0	Quel est le nom commun correspondant au verbe ou à l'adjectif "x" ? Donne un seul nom commun sans ponctuation.
S_0	1	Quel est le nom commun du verbe ou l'adjectif "x" ? Donne un seul nom commun sans ponctuation.
S_0	2	Quel est le nom commun formé à partir du mot "x" ? Donne un seul nom commun sans ponctuation.
S_0	3	Quel est le nom commun formé à partir du verbe ou de l'adjectif "x" ? Donne un seul nom commun sans ponctuation.
S_0	4	Quel est le nom commun dérivé du mot "x" ? Donne un seul nom commun sans ponctuation.
S_0	5	Quel est le nom commun dont "x" est la racine ? Donne un seul nom commun sans ponctuation.
S_0	6	Transforme le mot "x" en nom commun. Donne un seul nom commun sans ponctuation.
S_0	7	Quelle est une lexie nominale ayant le même sens que "x" ? Donne un seul nom commun sans ponctuation.

Table 3 – Questions et leur numéro correspondant (suite)

FL	No	Question
S_loc	0	Quel est un nom qui décrit la localisation de ¨, "¨ ? Donne un seul nom sans ponctuation.
S_loc	1	Donne le lieu typique de "x" ? Donne un seul nom sans ponctuation.
S_loc	2	Donne le lieu ou le moment typique de "x" ? Donne un seul nom sans ponctuation.
S_loc	3	À quel endroit se trouve "x" ? Donne un seul nom sans ponctuation.
S_loc	4	Quelle est une lexie nominale qui désigne un circonstant de lieu typique de, ou d'un fait lié à "x" ? Donne un seul nom sans ponctuation.
S_res	0	Quel est un résultat typique de l'acte associé au mot "x" ? Donne un seul mot sans ponctuation.
S_res	1	Quel est un résultat de l'acte associé au mot "x" ? Donne un seul mot sans ponctuation.
S_res	2	En quoi résulte l'acte de "x" ? Donne un seul mot sans ponctuation.
S_res	3	Quelle est une lexie nominale qui désigne un circonstant de résultat typique de "x" ? Donne un seul mot sans ponctuation.
Sing	0	Quel est un mot désignant une unité de "x" ? Donne un mot sans ponctuation.
Sing	1	Comment appelle-t-on une unité de "x" ? Donne un mot sans ponctuation.
Sing	2	Comment appelle-t-on une seule partie de "x" ? Donne un mot sans ponctuation.
Sing	3	Donne une lexie qui désigne une unité de "x" ? Donne un mot sans ponctuation.
Syn_⊂	0	Quel est le synonyme avec un sens plus large du mot "x" ? Donne un seul mot sans ponctuation.
Syn_⊂	1	Quel mot a un sens plus large "x" ? Donne un seul mot sans ponctuation.
Syn_⊂	2	Donne un seul mot qui englobe le sens du mot "x".
Syn_⊂	3	Donne un mot plus général pour signifier "x".
Syn_⊃ˆsex	0	Quel est le mot féminin correspondant au mot "x" ? Donne un seul mot sans ponctuation.
Syn_⊃ˆsex	1	Quel est le correspondant féminin du mot "x" ? Donne un seul mot sans ponctuation.
Syn_⊃ˆsex	2	Quel est l'équivalent féminin du mot "x" ? Donne un seul mot sans ponctuation.
Syn_⊃ˆsex	3	Conjugue le mot "x" au féminin. Donne un seul mot sans ponctuation.
V_0	0	Quel est le verbe correspondant au mot "x" ? Donne un seul verbe sans ponctuation.
V_0	1	Quel est le verbe formé à partir du mot "x" ? Donne un seul verbe sans ponctuation.
V_0	2	Quel est le verbe dérivé du mot "x" ? Donne un seul verbe sans ponctuation.
V_0	3	Transforme le mot "x" en verbe. Donne un seul verbe sans ponctuation.
V_0	4	Quelle est une lexie verbale ayant le même sens que "x" ? Donne un seul verbe sans ponctuation.

Références

[1] OpenAI et al. Gpt-4 technical report, 2024.

[2] Alain Mel'čuk, Igor et Polguère. Les fonctions lexicales dernier cri. *La Théorie Sens-Texte. Concepts-clés et applications*, pages 75–155, 2021.

[3] (n.d). Lexical systems.

[4] Alain Polguère. La théorie sens-texte. *Dialangue*, pages 9–30, 1998.

[5] Alain Potlinger, Sandrine et Polguère.

[6] Alain Potlinger, Sandrine et Polguère. Distribution des systèmes lexicaux version 3.1, 2024.