Fine-tuning des Modèles de Langage Large (LLMs) pour l'alignement d'entités au sein des graphes de connaissances (GCs)

Bill Gates Happi Happi*, Géraud Fokou Pelap**
Danai Symeonidou*** Pierre Larmande*

*Université de Montpellier, 75 Av. Augustin Fliche, 34090 Montpellier, France,
{bill.happi+pierre.larmande}@ird.fr
**Université de Dschang, Ouest Cameroun, geraud.fokou@univ-dschang.org
***INRAE, SupAgro, UMR MISTEA, 2 place pierre viala 34060 Montpellier, France,
danai.symeonidou@inrae.fr

Résumé. La recherche d'entités similaires dans les graphes de connaissances a toujours été un défi complexe. L'arrivée des LLMs a ouvert de nouvelles perspectives, notamment grâce au fine-tuning qui permet à ces modèles de se spécialiser pour des tâches spécifiques. Cet article propose d'utiliser les modèles GPT-2 et BERT pour développer un modèle généralisé permettant de résoudre les problèmes d'alignement d'entités (AE) sur divers jeux de données. Un protocole basé sur le réseau de Kolmogorov-Arnold (KAN) est également présenté pour pallier les limites des LLMs en termes d'interprétabilité et de coût computationnel. Les résultats montrent que GPT-2 surpasse BERT et KAN, offrant de meilleures performances de score F1 pour les défis d'alignement d'entités. Cette approche offre une meilleure capture des similarités linguistiques, syntaxiques et sémantiques entre les entités.

1 Introduction

L'alignement d'entités (AE) est essentiel pour relier des entités équivalentes dans différents graphes de connaissances (GCs). Traditionnellement basé sur des règles (Zou et Özsu (2017)), l'AE s'est amélioré avec les techniques modernes d'embedding et de deep learning (Lu et al. (2023)). Les LLMs, grâce aux transformeurs (Vaswani et al. (2017)), offrent une interprétation automatique des descriptions d'entités, mais leur coût et leur interopérabilité posent des défis (Tan et al. (2024)). En parallèle, les réseaux KAN (Liu et al. (2024)), fournissent une alternative plus transparente pour capturer des relations complexes. Cet article explore la généralisation de l'AE via le fine-tuning des modèles de language GPT-2 et BERT et l'entraînement from-scratch de réseaux KAN. Nous avons évalué ces approches sur 8 ensembles de données divers. GPT-2 et BERT ont d'abords été fine-tunés sur des jeux de données individuels, puis sur des données combinées, et finalement testés sur des ensembles de données inconnus pour évaluer leur capacité à généraliser . KAN a également été entraîné from-scratch pour comparer ses performances. Nos contributions concernent le Fine-tuning de GPT-2 et BERT pour l'AE ;

Le développement et entraînement de réseaux KAN ; l'évaluation des capacités de généralisation. La suite de l'article présente l'état de l'art en section 2, la méthodologie en section 3, les résultats en section 4 et la discussion en section 5.

2 Travaux associés

Les approches traditionnelles d'AE utilisent des ontologies (DLinker (Happi et al. (2022)), Limes (Ngonga Ngomo et al. (2021)), Logmap (Jiménez-Ruiz et Cuenca Grau (2011))) et des mesures de similarité (Jaro-Winkler (Keil (2019)), N-gram (Kim et Shawe-Taylor (1994))) pour réduire les paires d'entités candidates pour un alignement. En apprentissage automatique, des embeddings exploitant les connexions locales des entités (Zhu et al. (2021)) permettent d'apprendre des alignements positifs et négatifs (Trisedya et al. (2019)). Les perceptrons multicouches (MLP) (Shoeybi et al. (2020)) et le réseau Kolmogorov-Arnold (KAN) (Liu et al. (2024)), utilisant des splines, capturent efficacement des relations complexes. Les LLMs, introduits par les transformeurs (Vaswani et al. (2017)), ont révolutionné le traitement du langage naturel. GPT-2 (Radford et al. (2019)), avec 1,5 milliard de paramètres, prédit les tokens suivants, tandis que BERT (Devlin et al. (2019)) analyse les relations bidirectionnelles entre mots et s'adapte facilement à des tâches spécifiques via fine-tuning. Récemment, LLM4OM (Giglou et al. (2024)) a utilisé des LLMs pour appariements d'ontologies. Notre approche explore le fine-tuning de GPT-2, BERT et l'entraînement de zéro d'un KAN pour l'alignement d'entités.

3 Méthodologie

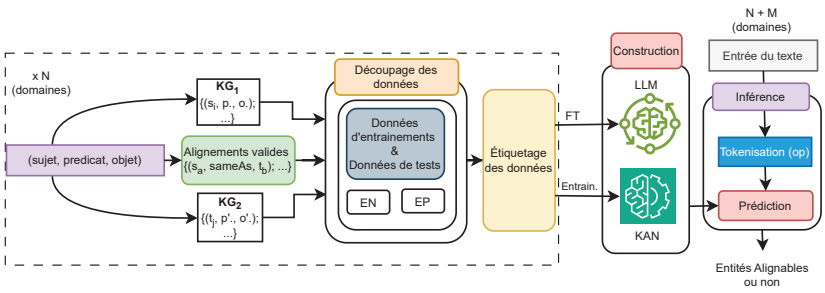

FIG. 1 – *Vue d'ensemble du pipeline pour l'AE. EN désigne l'échantillonnage négatif, EP désigne l'échantillonnage positif, op signifie optionnel, FT représente le fine-tuning. La "tokenisation" est nécessaire pour l'architecture KAN lors de l'étape d'étiquetage des données.*

Cette étude explore la généralisation des modèles d'EA à des données non vues en évaluant l'impact et la diversité des données d'entraînement. Nous visons à concevoir des stratégies pour que les modèles fonctionnent sur divers graphes de connaissances, même bruités ou limités. La Figure 1 présente le processus d'alignement d'entités. Nous formalisons le problème dans la Section 3.1, basée sur la structure des graphes RDF. La première étape du pipeline concerne

l'échantillonnage des jeux de données nécessaires pour l'entraînement des modèles. Pour cela 3 entrées sont nécessaires (KG1, KG2 et alignements valides) pour créer des ensembles de test et d'entraînement via des échantillonnages positifs (PS) et négatifs (NS). Par la suite, les traitements se distinguent entre le fine-tuning des LLMs et l'entraînement des réseaux KAN. Dans le premier, les entrées textuelles sont étiquetées (1 ou 0) pour ajuster GPT-2 et BERT-BASE, avant l'inférence sur N+M domaines (M pour non vus). Dans le second, un protocole transforme les textes en vecteurs réduits avec une analyse en composante principale (PCA) pour évaluer l'architecture KAN. La méthode est détaillée pour un seul ensemble de données dans la section suivante.

3.1 Préliminaires

Considérons une source de données RDF (un graphe) comme un ensemble de triplets défini par : $G = \{(s, p, o) \in (R \cup B) \times (R) \times (R \cup B \cup L)\}$, où R est l'ensemble de toutes les ressources sous forme d'IRI (Identifiant de Ressource Internationalisé), B est l'ensemble des nœuds blancs, et L est l'ensemble des littéraux. Nous désignons l'ensemble des graphes de données RDF par D. Soient $S(G)$, $P(G)$, et $O(G)$ les ensembles respectifs de sujets, prédicats et objets d'un ensemble de données G. Soit $\mathbb{N}$ l'ensemble des nombres naturels, incluant 0. Considérons les variables numériques suivantes (appartenant à $\mathbb{N}$), qui peuvent ou non dépendre de l'expression de la fonction qu'elles définissent et du contexte d'application. Soit $KG_1 = \{(s_i, p_k, o_k^i); 0 \leq i < m, 0 \leq k < k_{max}^i\} \in D$, l'ensemble des entités sources contenant m triplets, où k_{max}^i représente l'indice maximal possible de l'objet du $i^{\text{ème}}$ sujet. Et $KG_2 = \{(t_j, p'_l, o_l^j); 0 \leq j < n, 0 \leq l < l_{max}^j\} \in D$, l'ensemble des entités cibles contenant n triplets, où l_{max}^j représente l'indice maximal possible de l'objet du $j^{\text{ème}}$ sujet. Les objets ne sont pas nécessairement ordonnés. Nous déclarons $A_p =$ `owl:sameAs` comme étant le prédicat d'alignement. Nous considérons également l'ensemble suivant : $V_a = \{(s_{i'}, A_p, t_{j'})_c; \forall i', j' \in \mathbb{N}, s_{i'} \in S(KG_1), t_{j'} \in S(KG_2)\}$, avec KG_1 et $KG_2 \in D; 0 \leq c < c_{max}$, où $s_{i'}$ et $t_{j'}$ sont des entités alignables de leurs graphes respectifs. V_a est l'ensemble des alignements valides qui contient des alignements fondamentaux valides ($V_a \subset D$) et où A_p est le prédicat de liaison d'entités.

3.2 Formatage des données d'entrée et de sortie pour le modèle

Le processus d'échantillonnage vise à générer des alignements positifs (EP) et négatifs (EN) pour l'entraînement (70%) et le test (30%). Les EP sont extraits des alignements valides, puis des EN équivalents sont créés en associant aléatoirement sans duplication, des entités source et cibles. Les EP et EN sont fusionnés pour constituer les ensembles d'entraînement et de test. $\overline{A_p}$ représente les alignements incorrects, et $Rd(E)$ sélectionne un élément aléatoire dans E.

Pour les LLMs Pour optimiser les performances des LLMs, nous structurons le texte d'entrée pour minimiser le bruit et éviter les séquences excessivement longues ou tronquées, facilitant ainsi leur fine-tuning. Considérons les entités s_i et t_j décrites par des triplets :
$KG_1 = \{(s_i, p_k, o_k^i); 0 \leq k < k_{max}^i\}$ et $KG_2 = \{(t_j, p'_k, o_k^j); 0 \leq k < k_{max}^j\}$.
Pour s_i, $Concat(s_i)$ correspond à "$s_i : s_i \quad p_0 \quad o_0^i; \quad p_2 \quad o_2^i; \quad ... \quad p_{k_{max}^i - 1} \quad o_{k_{max}^i - 1}^i$."

Pour l'étiquetage, $Concat(s_i)+Concat(t_j)$ est associé à 0 (alignement négatif) ou 1 (positif). Le tokeniseur du LLM transforme ces séquences en matrices pour ajuster les poids lors du fine-tuning.

Pour l'architecture KAN Considérons s_i et t_j formatés en $F = Concat(s_i)+Concat(t_j)$. F est transformé en un vecteur numérique v de taille fixe d, complété si nécessaire pour correspondre à la dimension requise par KAN. Une normalisation (eq.2) ajuste v à une plage adaptée aux réseaux de neurones. Le tokeniseur génère des vecteurs numériques à partir des chaînes d'entrée (I) en sommant les vecteurs unitaires des symboles présents. Cependant, ces vecteurs de haute dimension ($h \in \mathbb{N}$) compliquent la classification pour KAN. Pour réduire cette complexité, nous utilisons l'Analyse en Composantes Principales (PCA) (Greenacre et al. (2022)) pour transformer les données en une représentation réduite (d_f), capturant les informations clés et facilitant la classification dans KAN.

$$\begin{array}{c|c|c} (1)\quad T_k : \mathrm{V}^* \longmapsto \mathbb{N}^h & (2)\quad N : \mathrm{R}^d \longmapsto \mathrm{R}^d & (3)\quad Vs : \mathrm{R}^d \longmapsto \mathrm{R}^{d_f} \\ I \longmapsto tokeniseur(I) = & & \\ [t_{k_0}, ..., t_{k_{h-1}}], (t_{k_*} \in \mathbb{N}) & x \longmapsto \frac{x-\min(x)}{\max(x)-\min(x)} & x \longmapsto PCA(N(x), d_f) \end{array}$$

Mathématiquement, nous pouvons exprimer le flux de la transformation de F, en appliquant les étapes suivantes : $v = T_k(F)$ et Le vecteur d'entrée $(V_s(v))$ est associé à la sortie 0 (pour NS) ou 1 (pour PS).

4 Expérimentations

Nous présentons les résultats d'expériences démontrant l'efficacité de notre approche de généralisation pour l'alignement d'entités. Nous avons utilisé deux LLMs : GPT-2 (124M paramètres) et BERT-BASE (340M), disponibles sur Hugging Face [1], ainsi qu'un modèle KAN implémenté en Python. Les expériences ont été menées sur un ordinateur équipé d'un processeur Intel Core i9-11950H (16 cœurs, 32 Go RAM, 7,5 Go GPU), sous Ubuntu 20.04.4 LTS. Nous avons utilisé des ensembles de données de IM@OAEI (2010, 2011, 2016, 2019), chacun avec une complexité et des caractéristiques uniques, fournis par la communauté OAEI [2].

4.1 Hyperparamètres de fine-tuning

Nous utilisons d'abord "gpt2", fine-tuné sur 3 epoch avec 70% des alignements pour l'entraînement et 30% pour le test (*lr=5e-5*, batch=1, AdamW avec $\beta 1 = 0.9$, $\beta 2 = 0.999$, $\epsilon = 10^{-8}$, régularisation=0.0). Ensuite, "google/bert-base" est fine-tuné sur 2 epoch (*lr=2e-5*, batch=1, gradient accumulation=4, AdamW avec $\epsilon = 10^{-6}$, régularisation=0.01, entropie croisée binaire). Enfin, le modèle KAN a trois couches : entrée ($d = 20$), cachée ($d + 1$), et sortie (1 neurone), avec une grille de 10 intervalles et un polynôme de degré $k = 3$. L'entraînement utilise LBFGS (Ge et al. (2018)) sur 20 itérations. Les évaluations prennent plus de 24 heures.

1. https ://huggingface.co/
2. https ://oaei.ontologymatching.org/

4.2 Résultats

Des expériences ont été menées pour fine-tuner des modèles sur des ensembles de données individuels ou fusionnés (Tableaux 1). GPT-2 atteint une précision élevée mais un F1-score faible sur les ensembles individuels. Sa précision s'améliore sur des ensembles fusionnés et dépasse celle de BERT et KAN.

Modèl.	M	NLL-DB	Yago-Wiki	Person	Restaurant	Doremus	Anatomy	SS19	SL19
GPT-2	P	1.0 (1.0)	0.97 (0.98)	1.0 (1.0)	1.0 (1.0)	1.0 (0.74)	1.0 (0.71)	1.0 (1.0)	0.0 (0.96)
	R	0.03 (0.86)	0.38 (0.93)	1.0 (0.99)	0.05 (0.97)	0.01 (0.70)	0.30 (0.60)	0.95 (0.96)	0.0 (0.93)
	F	0.05 (**0.93**)	0.55 (**0.95**)	**1.0 (0.99)**	0.08 (**0.98**)	0.03 (**0.72**)	0.46 (**0.65**)	**0.98 (0.98)**	0.0 (**0.95**)
BERT	P	0.5 (0.50)	0.0 (0.50)	1.0 (0.50)	0.0 (0.50)	1.0 (0.50)	1.0 (0.0)	1.0 (0.48)	0.0 (0.50)
	R	1.0 (1.0)	0.0 (1.0)	0.98 (1.0)	0.0 (1.0)	0.49 (1.0)	0.96 (0.0)	0.63 (1.0)	0.0 (1.0)
	F	0.66 (0.66)	0.0 (0.66)	0.99 (0.66)	0.0 (0.66)	**0.66 (0.66)**	**0.98 (0.0)**	0.77 (0.65)	0.0 (0.67)
KAN	P	0.74 (0.53)	0.80 (0.60)	0.85 (0.54)	0.82 (0.53)	0.57 (0.57)	0.62 (0.53)	0.80 (0.60)	0.79 (0.58)
	R	0.74 (0.53)	0.80 (0.60)	0.85 (0.54)	0.82 (0.53)	0.57 (0.57)	0.62 (0.53)	0.80 (0.60)	0.79 (0.58)
	F	**0.74 (0.53)**	**0.80 (0.60)**	0.85 (0.53)	**0.82 (0.53)**	0.57 (0.57)	0.62 (0.53)	0.80 (0.60)	**0.79 (0.58)**

TAB. 1 – *Comparaison des modèles sur les données individuels (métriques hors parenthèses) et fusionnées (métriques entre parenthèses). Les valeurs soulignées (resp. en gras) représentent les meilleures précisions (resp. F pour le F1-score) des modèles de meilleures performances.*

4.2.1 Fine-tuning sur un ensemble de données individuel

Dans le Tableau 1, GPT-2 affiche une précision élevée mais un faible rappel, entraînant un F1-score faible, sauf pour "Person", où il identifie parfaitement les alignements. BERT n'est pas bon sur Restaurant et SL19 (F1=0.0) et reste inférieur à GPT-2 sur Yago-Wikidata, Restaurant, SS19 et SL19, mais surpasse GPT-2 sur Doremus et Anatomy. KAN est stable mais sans exceller, obtient de meilleures performances sur NLL-DB, Yago-WIKI, Restaurant et SL19.

4.2.2 Fine-tuning sur des ensembles de données fusionnés

Dans le Tableau 1, GPT-2 obtient les meilleurs F1-scores, dépassant 0.98 sur Person, Restaurant et SS19. Nell-DBPedia et Yago-Wikidata affichent des F1-scores de 0.93 et 0.95, respectivement, mais les performances diminuent sur Doremus et Anatomy. BERT montre des F1-scores faibles (0.6) et échoue sur Doremus (F1=0), avec une précision de 0.50 et un rappel de 1.0, révélant des difficultés à distinguer les alignements. KAN est stable (0.5) et manque de performance mais reste sans biais marqué. Les ensembles de données Nell-DBPedia et Yago-Wikidata, tous deux faisant partie du track commonKG, avaient été précédemment évalués par LLM4OM (Giglou et al. (2024)), obtenant des F1-scores respectivement de 0.9426 et 0.9219.

5 Discussions

Dans le Tableau 1, GPT-2 affiche des F1-scores faibles, en partie à cause du manque de diversité des tokens et contextes dans les données d'entraînement (Wolfe et Caliskan (2021); Mosbach et al. (2021)). BERT montre un surapprentissage, avec une précision faible et un

rappel élevé, indiquant une difficulté à distinguer alignements positifs et négatifs (Zhang et al. (2020)). KAN, bien que précis sur de petits jeux données, souffre également sur des jeux plus volumineux, en raison de la réduction de la dimensionnalité via ACP (Liu et al. (2024)). Les jeux Doremus et Anatomy posent des défis : Doremus, avec des textes en français, donne un faible score individuel (F1=0.027 pour GPT-2), mais améliore ses résultats (F1=0.724) après fusion. Anatomy, avec des données littérales courtes et peu de contexte, obtient de faibles performances, aggravées par la dispersion des tokens. Ces observations confirment l'avantage de GPT-2 sur BERT et KAN dans les métriques globales.

Évaluation de notre généralisation sur des ensembles de données non vus. Pour évaluer la généralisation de nos modèles, nous les avons fine-tunés sur quatre ensembles de données (Anatomy, Doremus, SS19 et SL19) et testés sur quatre autres (Person, Restaurant, Yago-Wikidata et Nell-DBPedia). Après évaluation des quatre ensembles de données non vus, les résultats rapportés dans le Tableau 2 montrent que GPT-2 généralise et fait mieux que les approches existantes (RIMOM (Achichi et al. (2016)), Logmap [3] [4], AML (Achichi et al. (2016)), Lily [5]). Nous avons rapporté le temps d'exécution pour chaque modèle dans le Tableau 3. GPT-2 prend plus de temps que les autres lors des évaluations de généralisation.

Dataset-Outil	RIMOM	Logmap	AML	Lily	GPT-2	BERT	KAN
Doremus	0.813	-	**0.862**	-	0.751	0.0	0.535
Anat.	-	0.881	0.943	0.83	**0.962**	0.0	0.560
SS19	**0.992**	0.841	0.864	0.991	0.952	0.0	0.55
SL19	**0.995**	0.785	0.86	**0.995**	0.940	0.0	0.571
Y-W (A)	-	**0.86**	0.0	-	0.66	0.0	0.615
N-D (A)	-	**0.88**	0.0	-	0.68	0.0	0.697
Person (A)	0.97	-	-	-	**0.990**	0.0	0.510
Rest. (A)	0.81	-	-	-	**0.967**	0.0	0.5

TAB. 2 – *Comparaison entre les modèles SOTA et nos modèles fine-tunés sur quatre ensembles de données et prédits sur quatre ensembles de données non vus (Person et Restaurant). Les valeurs du Tableau correspondent au F1-score et les meilleurs outils sont mis en gras. "A" : absent dans les données d'entraînement.*

6 Conclusion

Cet article a exploré la généralisation de l'alignement d'entités (EA) entre graphes RDF via deux approches : le fine-tuning de modèles pré-entraînés (GPT-2, BERT) et l'évaluation d'une nouvelle architecture (KAN). Les résultats montrent que GPT-2, avec un fine-tuning sur des

3. https ://hobbit-project.github.io/OAEI_2022.html
4. https ://oaei.ontologymatching.org/2020/results/anatomy/index.html
5. https ://hobbit-project.github.io/OAEI_2021.html

Dataset-Tool	NLL-DB	Yago-Wiki	Person	Restaurant	Doremus	Anatomy	SS19	SL19
GPT-2	8.31	14.16	22.09	6.93	8.45	58.87	13.25	57.64
BERT	5.98	11.93	10.52	4.83	2.05	19.42	3.32	18.85
KAN	**0.108**	**0.132**	**0.208**	**0.113**	**0.129**	**0.211**	**0.129**	**0.169**

TAB. 3 – *Temps d'exécution (en secondes) des modèles lors des tests sur les ensembles de données. Le modèle le moins gourmand en temps est indiqué en gras.*

données substantielles, surpasse BERT et KAN, même sur des données hors domaine. Nos travaux futurs viseront à exploiter GPT-2 pour développer des méthodes robustes d'EA et créer un pipeline d'apprentissage par renforcement pour améliorer la gestion des scénarios d'alignement non vus. Notre code est disponible à cette adresse : **url** [6].

Références

Achichi, M., M. Cheatham, Z. Dragisic, J. Euzenat, Faria, et al. (2016). Results of the ontology alignment evaluation initiative 2016. In *OM : Ontology matching*.

Devlin, J., M.-W. Chang, K. Lee, et K. Toutanova (2019). Bert : Pre-training of deep bidirectional transformers for language understanding.

Ge, F., Y. Ju, Z. Qi, et Y. Lin (2018). Parameter estimation of a gaussian mixture model for wind power forecast error by riemann l-bfgs optimization. *Ieee Access 6*, 38892–38899.

Giglou, H. B., J. D'Souza, F. Engel, et S. Auer (2024). Llms4om : Matching ontologies with large language models.

Greenacre, M., P. J. Groenen, T. Hastie, A. I. d'Enza, A. Markos, et E. Tuzhilina (2022). Principal component analysis. *Nature Reviews Methods Primers 2*(1), 100.

Happi, B., G. F. Pelap, D. Symeonidou, et P. Larmande (2022). Dlinker results for oaei 2022. In *OM@ ISWC*, pp. 166–173.

Jiménez-Ruiz, E. et B. Cuenca Grau (2011). Logmap : Logic-based and scalable ontology matching. In *ISWC 2011*.

Keil, J. M. (2019). Efficient bounded jaro-winkler similarity based search. In *Datenbanksysteme für Business, Technologie und Web*.

Kim, J. Y. et J. Shawe-Taylor (1994). Fast string matching using an n-gram algorithm. *Software : Practice and Experience 24*(1), 79–88.

Liu, Z., Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, et M. Tegmark (2024). Kan : Kolmogorov-arnold networks.

Lu, D., G. Han, Y. Zhao, et Q. Han (2023). Review of deep learning-based entity alignment methods. In *International Conference on Green, Pervasive, and Cloud Computing*.

Mosbach, M., M. Andriushchenko, et D. Klakow (2021). On the stability of fine-tuning bert : Misconceptions, explanations, and strong baselines.

6. https ://github.com/BillGates98/Fine-Tuning-LLMs-4-EA-in-KG

Ngonga Ngomo, A.-C., M. A. Sherif, K. Georgala, M. M. Hassan, K. Dreßler, K. Lyko, D. Obraczka, et T. Soru (2021). Limes : A framework for link discovery on the semantic web. *KI - Künstliche Intelligenz 35*(3), 413–423, doi: 10.1007/s13218-021-00713-x.

Radford, A., J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog 1*(8), 9.

Shoeybi, M., M. Patwary, R. Puri, P. LeGresley, J. Casper, et B. Catanzaro (2020). Megatron-lm : Training multi-billion parameter language models using model parallelism.

Tan, Z., T. Chen, Zhang, et al. (2024). Sparsity-guided holistic explanation for llms with interpretable inference-time intervention. In *AAAI Conference on Artificial Intelligence*.

Trisedya, B. D., J. Qi, et R. Zhang (2019). Entity alignment between knowledge graphs using attribute embeddings. In *Proceedings of the AAAI conference on artificial intelligence*.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, Jones, et al. (2017). Attention is all you need.

Wolfe, R. et A. Caliskan (2021). Low frequency names exhibit bias and overfitting in contextualizing language models.

Zhang, T., F. Wu, A. Katiyar, K. Q. Weinberger, et Y. Artzi (2020). Revisiting few-sample bert fine-tuning.

Zhu, Y., H. Liu, Z. Wu, et Y. Du (2021). Relation-aware neighborhood matching model for entity alignment.

Zou, L. et M. T. Özsu (2017). Graph-based rdf data management. *Data Science and Engineering 2*, 56–70.

Summary

Finding similar entities across diverse and heterogeneous data sources in knowledge graphs (KGs) remains a major challenge. The emergence of LLMs has introduced new research opportunities. Fine-tuning LLMs has been rapidly adopted due to their ability to specialize in specific tasks. This challenge focuses on capturing subtle linguistic, syntactic, and semantic similarities between entities. In this paper, we propose a fine-tuning approach for GPT-2 and BERT to address the generalization of entity alignment (EA) across multiple datasets using a single model. Additionally, we introduce a protocol based on the Kolmogorov Arnold Network (KAN) to overcome the limitations of LLMs regarding interpretability, redundancy, and computational cost. Our evaluations demonstrate that the fine-tuned GPT-2 model significantly outperforms BERT and KAN in entity alignment tasks, offering better performance and reliability.