Actes de CORIA-TALN 2026
Nantes, France, 29 juin au 3 juillet 2026

Affinés pour la réussite : Évaluation des représentations dans le domaine de la formation professionnelle

Alicia Breidenstein [1, 2] Marguerite Leang[3] Matthieu Labeau[1]
(1) LTCI, Télécom-Paris, Institut Polytechnique de Paris, France
(2) Caisse des Dépôts et Consignations, Paris, France
(3) Hi! PARIS Research Center, France
{alicia.breidenstein, matthieu.labeau}@telecom-paris.fr, marguerite.leang@ip-paris.fr

RÉSUMÉ

Dans le domaine de la formation professionnelle et du marché du travail, les données sont souvent structurées sous forme de graphes avec attributs textuels, reliant les offres de formation, les CV et les ontologies de certifications et de compétences. Cette configuration présente des défis uniques, car ces données sont organisées selon une hiérarchie de milliers d'étiquettes de classification, et le graphe comporte de nombreuses liaisons manquantes. Avec la difficulté supplémentaire que représente le vocabulaire spécifique au domaine, il est nécessaire d'utiliser une représentation du texte à la fois adaptée et efficace. Dans ce travail, nous évaluons un large éventail de représentations de texte, allant des représentations symboliques aux grands modèles de langue, sur des tâches réelles appliquées à notre jeu de données interne. Nous montrons que les représentations lexicales offrent les meilleures performances sans affinage, mais que les modèles basés sur BERT dominent une fois affinés sur des données spécifiques au domaine. En revanche, les LLM utilisés avec des instructions génératives directes sous-performent, limités par la complexité structurelle des données et le vocabulaire spécifique, et atteignent des performances comparables à celles des modèles basés sur BERT lorsqu'ils sont affinés sur le jeu de données.

ABSTRACT

Fine-tuned for Success : Evaluating Embeddings in the Vocational Training Domain

In the vocational training and labor market domains, data is often structured as a text-attributed graph, linking training offers, resumes, and ontologies of certifications and skills. This setup poses unique challenges, as this data is organized through a hierarchy of thousands of classification labels, and the graph comprises many missing links. With the added difficulty of the domain-specific vocabulary, it is necessary to use an appropriate and effective representation of text. In this work, we evaluate a broad spectrum of embeddings, ranging from symbolic representations to Large Language Models, across real-world tasks on our in-house dataset. We show that lexical representations perform best without fine-tuning, but BERT-based models dominate once fine-tuned on domain-specific data. However, LLMs used with direct prompting underperform, hindered by the data's structural complexity and specific vocabulary, and get similar results to BERT-based models when fine-tuned on the dataset.

MOTS-CLÉS : cas d'usage industriel, domaine de la formation professionnelle, affinage.

KEYWORDS: industrial use case, vocational training domain, fine-tuning.

Actes des 33ème Conférence sur le Traitement Automatique des Langues Naturelles.
Volume 1 : articles scientifiques originaux, pages 65–90.
Cette œuvre est mise à disposition sous licence Attribution 4.0 International.

1 Introduction

Les méthodes de TAL (Traitement Automatique des Langues) ont été largement appliquées au domaine du marché de l'emploi, notamment pour recommander des offres d'emploi (intitulés ou annonces) aux individus en fonction de leur CV ou de leurs compétences (Rosenberger *et al.*, 2025; Johary *et al.*, 2025; Gugnani & Misra, 2020). L'association des compétences aux offres d'emploi englobe plusieurs sous-tâches, telles que la détection, l'extraction, la standardisation et la classification des compétences (Senger *et al.*, 2024). Peu d'études intègrent cependant les données relatives aux offres de formation ou aux descriptions de cours. Lorsqu'elles sont utilisées, ces données sont généralement mises en correspondance avec des compétences (Ferreira *et al.*, 2025) ou intégrées à des systèmes de recommandation (Frej *et al.*, 2024; Weichselbraun *et al.*, 2022) pour aider les utilisateurs à identifier leurs besoins en formation, élargir leurs compétences et renforcer leurs capacités professionnelles. Dans ce domaine, les compétences occupent une place centrale : des ontologies comme ESCO (le Vrang *et al.*, 2014), ONET [1] et des référentiels nationaux sont utilisés pour standardiser les définitions des compétences et relier des sources de données variées (par exemple, CV, offres d'emploi, offres de formation), stimulant ainsi la recherche sur l'alignement d'ontologies (Neutel & de Boer, 2021; Hihn *et al.*, 2025). Notre jeu de données se démarque en se concentrant sur les descriptions d'offres de formation issues de la plateforme publique française de formation professionnelle Mon Compte Formation (MCF) [2]. Pour être autorisées sur la plateforme, ces offres doivent être associées à des certifications du RNCP ou du RS français [3], garantissant la qualité des programmes de formation. Cela implique un contrôle rigoureux de la conformité entre les offres et leurs certifications associées. Contrairement aux jeux de données existants, nos offres de formation et certifications se composent principalement de textes longs et ne sont pas directement liées à des compétences. Comme illustré par la Figure 1, chaque certification est associée à une structure hiérarchique de domaines thématiques, appelée l'arbre des *Formacodes*. Cette structure fonctionne comme une ontologie, similaire à l'ontologie des compétences ESCO. Des détails supplémentaires sur nos données sont disponibles en annexe A.1 et A.2.

Ce contexte soulève plusieurs défis : les *offres*, soumises par les organismes de formation, sont sujettes aux erreurs humaines et aux fraudes, et un contrôle manuel est irréalisable en raison de leur volume et de leurs mises à jour fréquentes. Une extension future pourrait inclure des offres *non certifiantes*, nécessitant un appariement avec la taxonomie complexe et déséquilibrée des *Formacodes*. Pour répondre à ces enjeux, des modèles robustes sont nécessaires afin d'aider les experts et les documentalistes dans la maintenance, le nettoyage et le contrôle de la base de données, sans accès à aucune étiquette ni à des exemples d'échantillons anormaux. Dans cet article, nous évaluons des méthodes de représentation textuelle à travers trois tâches : la *classification* des offres dans l'arbre des Formacode, l'évaluation de la *similarité* entre offres et certifications, ainsi que différentes formes de *détection d'anomalies*. Nos principales contributions sont la publication d'un jeu de données réel, nettoyé et prêt à l'emploi, ainsi que la définition de protocoles expérimentaux pour plusieurs tâches. Nous avons mené des expériences approfondies sur diverses méthodes de représentation, allant des sacs de mots (*Bag-of-Words*, BOW) aux Grands Modèles de Langue (*Large Language Models*, LLM), en incluant différentes adaptations et optimisations de modèles. Nos résultats indiquent que sur les données de formation professionnelle et en l'absence d'affinage, les représentations lexicales simples

1. https://www.onetonline.org/
2. https://www.moncompteformation.gouv.fr/espace-prive/html/
3. Répertoire National des Certifications Professionnelles et Répertoire Spécifique : https://www.francecompetences.fr/fiche/certifications-le-role-de-france-competences/

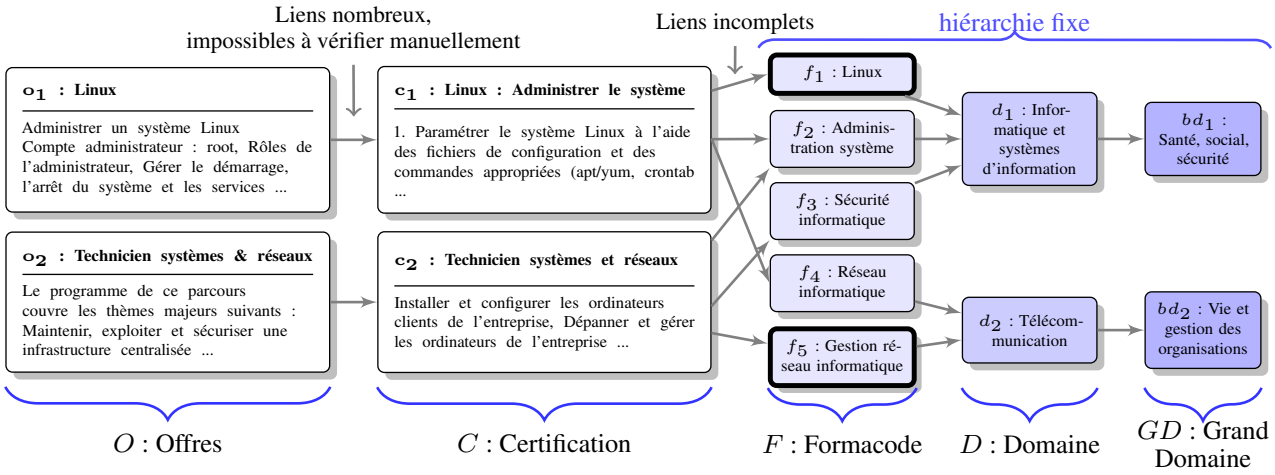

FIGURE 1 – Exemples d'*offres de formation*, de *certifications* et de l'*arbre des Formacode*. Chaque offre est liée à une certification, elle-même associée à un *Formacode principal* (indiqué par un contour épais) et jusqu'à quatre *Formacodes complémentaires*.

surpassent les autres. Cependant, une fois optimisés sur des données et tâches spécifiques au domaine, les modèles basés sur BERT les dépassent. Les LLM affinés atteignent des performances légèrement supérieures à celles des modèles basés sur BERT, mais à un coût de calcul d'entraînement très élevé[4].

2 Travaux connexes

Le contrôle des offres de formation publiées sur la plateforme peut être réalisé à l'aide de diverses méthodes, allant des systèmes à base de règles utilisant des mots-clés et des ontologies (Dörpinghaus *et al.*, 2023) aux approches basées sur la similarité exploitant différentes techniques de plongements. Le calcul de similarité est un sujet bien étudié dans la littérature dans le domaine de l'emploi, avec des méthodes allant des modèles traditionnels comme TF-IDF et les variantes de BERT (Rosenberger *et al.*, 2025) à l'utilisation généralisée de SentenceBERT (Gnehm *et al.*, 2022; Decorte *et al.*, 2024; Ferreira *et al.*, 2025), souvent avec de l'affinage, et plus récemment, des LLMs (Johary *et al.*, 2025). Cependant, dans notre contexte, les méthodes à base de règles ne sont pas applicables en raison du grand nombre de certifications, de types d'offres et de la diversité du vocabulaire. Nous expérimentons donc les méthodes mentionnées précédemment, bien que nous suspections que la complexité structurelle des données puisse également être difficile pour les LLMs.

Pour tirer parti de la structure des graphes enrichis de texte, Yan *et al.* (2023) distinguent deux grandes approches : d'une part, l'intégration de la modélisation du texte et du graphe dans une architecture en cascade ; d'autre part, l'utilisation d'un pré-entraînement topologique appliqué à un modèle de langue, afin de mieux saisir les liens entre les nœuds textuels. Dans le domaine des compétences, Zhang *et al.* (2023) appliquent ce type de pré-entraînement à la taxonomie ESCO, en adaptant un modèle XLM-Rlarge autour de deux objectifs : l'objectif classique d'entraînement du modèle de langue par masquage (*Masked Language Modeling*, MLM) (prédire les mots masqués dans un texte) et la prédiction des relations entre entités (ERP) (prédire les liens entre les nœuds). Nous reprenons leur modèle et appliquons leur méthodologie à nos propres données, comme expliqué en détail dans

4. Notre code est disponible à l'adresse https://github.com/abreidenstein/OCFproject.

la section 4.1. D'autres travaux, comme ceux de (Decorte *et al.*, 2024; Rosenberger *et al.*, 2025), optimisent des modèles de type SentenceTransformer (SentenceBERT ou GBERT pour l'allemand) en les entraînant sur des triplets (ancrage, exemple positif, exemple négatif). Ces triplets sont construits soit à partir des données elles-mêmes, soit à partir d'ontologies structurées comme ESCO. L'objectif est d'ajuster le modèle pour qu'il rapproche les représentations de l'ancrage et des exemples positifs, tout en les éloignant des exemples négatifs. Lorsque des éléments textuels sont sémantiquement liés, comme des compétences apparaissant ensemble dans une même offre d'emploi ou les différentes composantes d'une compétence (intitulé, synonymes, descriptions), le modèle apprend à produire des représentations qui reflètent ces relations, saisissant ainsi implicitement la structure sous-jacente des données. Nous suivons une démarche analogue pour l'affinage d'un modèle SentenceTransformer.

Pour finir, les tâches de classification dans le domaine des compétences restent peu courantes. La plupart se contentent de répartir les compétences en un nombre limité de classes, rarement plus de 15 (Butt *et al.*, 2025; Jiechieu & Tsopze, 2021). Notre approche, qui consiste à classer des documents dans une taxonomie hiérarchique étendue avec des étiquettes courtes, est souvent traitée comme un problème de similarité document-étiquette. Une étude utilise ainsi des représentations BERT associées à une tête de classification pour une classification multi-étiquettes extrême, comparable à notre classification en Formacode (Bhola *et al.*, 2020).

3 Description des tâches

Dans notre jeu de données, la majorité des erreurs proviennent des offres de formation et de leurs liens avec les certifications, établis par les organismes de formation et impossibles à vérifier manuellement dans leur intégralité. Les autres liens du graphe, entre certifications et Formacodes, sont annotés manuellement par des documentalistes et sont considérés comme corrects, bien qu'ils puissent être incomplets en raison des contraintes structurelles (par exemple, un maximum de cinq Formacodes peut être associé à une certification). Enfin, avec l'ajout éventuel d'offres non certifiantes au catalogue à l'avenir, leur rattachement à l'arbre des Formacodes devient un enjeu clé pour maintenir une recherche efficace. Ces trois défis justifient nos deux premières tâches : calculer un score de similarité entre une offre et sa certification, et classer les offres aux différents niveaux de la hiérarchie des Formacodes.

Calcul de la similarité entre offres et certifications : Pour ce faire, des modèles de plongements transforment le contenu textuel des offres et des certifications en représentations vectorielles. La similarité entre ces vecteurs est ensuite mesurée par similarité cosinus. L'évaluation des modèles repose sur plusieurs métriques liées au classement de la certification réelle parmi toutes les certifications possibles pour une offre donnée, en fonction du score de similarité. Nous supposons ici que les anomalies dans le jeu de données sont rares et que les offres sont correctement associées à leur certification dans l'ensemble d'apprentissage.

Classification des offres en Formacodes : Nous classons chaque offre selon les Formacodes, domaines ou grands domaines. Étant donné qu'une offre est liée à un Formacode principal et jusqu'à quatre Formacodes secondaires, nous abordons cette tâche à la fois comme un problème de classification simple étiquette (où le Formacode principal sert d'étiquette, voir Figure 1) et multi-étiquettes à chaque niveau hiérarchique. Pour la classification simple étiquette, nous utilisons une

	Rang Moyen ↓	Rang Médian ↓	Précision@1 ↑	Précision@5 ↑	Précision@10 ↑
BOW	31.32	1	0.46	0.69	0.78
TF-IDF	**21.75**	**0**	**0.52**	**0.76**	**0.84**
W2V	42.23	2	0.37	0.60	0.69
FastText	61.42	2	0.36	0.62	0.70
CamemBERT	657.01	639	0.08	0.14	0.17
CamemBERT-FT	9.76	0	0.57	0.85	0.92
ERP-CamemBERT	493.18	178	0.17	0.31	0.36
ERP-CamemBERT-FT	8.44	0	0.68	0.88	0.92
SentenceCamemBERT	48.91	2	0.33	0.59	0.69
SentenceCamemBERT-FT	**4.93**	**0**	**0.70**	**0.90**	**0.94**
Qwen-Transformer	165.7	5	0.29	0.50	0.57
E5	453.4	297	0.19	0.29	0.33
Qwen-SentenceTransformer	19.6	0	0.54	0.80	0.86
Qwen-SentenceTransformer-FT	**3.26**	**0**	**0.78**	**0.93**	**0.96**

TABLE 1 – Métriques de rang et de précision pour les méthodes de plongement sur la tâche de similarité. Les meilleurs résultats dans chaque catégorie de modèles (*lexicaux*, *basés sur CamemBERT* et *multilingues*) sont en **gras**.

régression logistique sur les plongements générés par nos modèles. Pour la classification multi-étiquettes, nous employons des classifieurs binaires parallèles, chacun dédié à un Formacode (ou autre niveau hiérarchique) spécifique. Les détails de notre protocole expérimental sont présentés en annexe B.

4 Modèles et protocole expérimental

Nous comparons plusieurs représentations lexicales et de plongements permettant de convertir le texte des offres et des certifications en représentations vectorielles. Les méthodes de représentation lexicale comme Bag-of-Words, TF-IDF ou les plongements statiques tels que Word2Vec (Mikolov *et al.*, 2013) et FastText (Bojanowski *et al.*, 2017) sont simples à mettre en œuvre et s'adaptent directement à notre vocabulaire spécifique. Nous évaluons également CamemBERT (Martin *et al.*, 2020) et SentenceCamemBERT, utilisés sous leur forme pré-entraînée puis affinés sur nos deux tâches. Leur entraînement sur des données francophones et leur capacité d'affinage les rendent particulièrement adaptés à notre contexte. Nous expérimentons aussi des modèles de plongements multilingues récents : E5 (Wang *et al.*, 2024) et Qwen (Zhang *et al.*, 2025), entraînés en tant que Transformer ou SentenceTransformer. Enfin, nous testons des LLMs, dont l'échelle pourrait améliorer les performances, mais qui pourraient être moins adaptés au vocabulaire spécifique et à la structure complexe de nos données. Une description plus détaillée de l'application de ces modèles est disponible en annexe B.1.

Affinage des modèles entraînés pour la similarité : SentenceCamemBERT, un modèle Sentence-Transformer, est spécifiquement entraîné pour rapprocher dans l'espace des plongements les phrases sémantiquement similaires tout en éloignant celles qui ne le sont pas. Cela le rend particulièrement adapté au calcul de similarité entre des offres et des certifications. Pour l'adapter à notre contexte, nous affinons le modèle en utilisant une stratégie d'entraînement basée sur des triplets. L'ensemble d'entraînement est réorganisé en triplets de la forme : $(\text{certification}, \text{offre positive}, \text{offre négative})$.

	Grand domaine		Domaine		Formacode	
	Simple étiquette	**Multi-étiquettes**	**Simple étiquette**	**Multi-étiquettes**	**Simple étiquette**	**Multi-étiquettes**
BOW	**85.3**	**83.9**	**76.5**	**76.1**	**57.3**	**50.9**
TF-IDF	85.0	76.4	71.4	54.1	34.7	15.5
W2V	73.9	66.1	65.7	54.1	48.9	24.8
FastText	70.1	58.3	62.2	50.4	49.1	23.6
CamemBERT	74.8	70.3	62.5	50.8	28.0	12.2
CamemBERT-FT	**97.4**	**89.7**	**95.2**	**85.6**	**78.3**	**63.0**
ERP-CamemBERT	84.3	79.0	74.4	66.6	44.0	19.8
ERP-CamemBERT-FT	92.2	81.8	84.5	68.4	68.3	24.5
SentenceCamemBERT	78.6	74.2	69.8	61.0	44.6	20.3
SentenceCamemBERT-FT	86.3	85.2	80.2	77.0	63.05	42.3
Qwen-Transformer	71.0	75.9	61.5	72.1	39.2	49.2
Qwen-Transformer-FT-LoRA	**92.1**	**91.6**	**87.2**	**87.0**	**70.1**	**64.0**
E5	75.6	72.5	69.9	59.5	41.1	16.2
Qwen-SentenceTransformer	81.5	73.8	71.0	54.9	31.6	11.1

TABLE 2 – Macro-F1 pour les méthodes de plongement sur la tâche de classification, en simple étiquette et multi-étiquettes, à chaque niveau de l'arborescence des Formacodes. Les meilleurs résultats dans chaque catégorie de modèles (*lexicaux*, *basés sur CamemBERT* et *multilingues*) sont en **gras**.

Chaque triplet contient la certification originale et une offre liée (positive), tandis qu'une deuxième offre non liée (négative) est échantillonnée aléatoirement. Le modèle est ensuite entraîné à rapprocher les plongements de la certification et de l'offre positive, tout en éloignant ceux de la certification et de l'offre négative. Cette approche correspond à l'objectif de pré-entraînement de SentenceCamemBERT.

Affinage pour la classification : Plusieurs modèles CamemBERT sont affinés pour classer les offres de l'ensemble d'entraînement dans les catégories Formacode appropriées, au niveau pertinent, et selon des configurations simple-étiquette ou multi-étiquettes, en utilisant la perte d'entropie croisée ou d'entropie croisée binaire. Des détails supplémentaires sur toutes les fonctions de perte utilisées sont fournis en annexe B.2.1.

4.1 Pré-entraîner CamemBERT à apprendre des liens hiérarchiques

Pour exploiter la structure spécifique de notre jeu de données, nous adaptons la méthodologie de Zhang *et al.* (2023) : initialement conçue pour les données multilingues ESCO avec XLM-Rlarge, nous choisissons de l'appliquer à notre jeu de données français via CamemBERT. Dans ce cadre, le modèle est entraîné pour évaluer la relation entre deux offres distinctes, en prédisant si elles partagent un Formacode, un domaine ou un grand domaine, ou n'ont aucun lien. Cet objectif prend la forme d'une perte de classification multi-étiquettes et vise à améliorer la compréhension par le modèle des relations hiérarchiques entre les textes des offres d'entraînement. Le modèle entraîné est ensuite utilisé directement pour générer des plongements ou est affiné sur nos tâches. Des détails supplémentaires sur ce processus sont fournis en annexe B.

4.2 Utilisation de grands modèles de langue

Par instructions : Nous explorons également l'utilisation de LLMs pour réaliser directement les tâches à partir d'instructions génératives (*prompt*) correspondants. Inspiré par (Hakimi Parizi *et al.*,

2023), notre instruction intégrait le texte de l'offre à classer ainsi que les étiquettes de catégories pertinentes. Nous avons évalué ces plongements à l'aide d'instructions génératives en anglais et en français, en utilisant des approches sans exemples (*zero-shot*) et avec peu d'exemples (*few-shot*). La classification a été limitée aux niveaux grand domaine et domaine, en n'utilisant que des étiquettes simples, car il s'agissait de nos tâches les moins complexes ; le nombre élevé de formacodes rendait la classification basée sur les LLMs peu pratique. Pour les expériences avec peu d'exemples, nous avons testé deux méthodes de sélection des exemples : d'abord, en en échantillonnant aléatoirement trois, de classes différentes, puis en sélectionnant un exemple par classe. Les exemples sélectionnés ont été utilisés de manière systématique dans chaque instruction afin d'améliorer la reproductibilité. Nous avons également attribué au LLM un rôle (Schulhoff *et al.*, 2024) d'assistant classifieur, ce qui a permis d'obtenir des résultats légèrement améliorés.

Nous avons également tenté d'appliquer directement les LLMs à la détection d'anomalies, en nous inspirant d'une instruction issue de (Yang *et al.*, 2025), en fournissant au LLM une offre et une certification et en lui demandant si les deux correspondaient, ainsi qu'un score d'anomalie.

Affinage des LLM via PEFT : Pour compléter notre analyse, nous avons étudié les performances des grands modèles de langage lorsqu'ils sont spécifiquement adaptés à notre tâche et à notre jeu de données. Étant donné que ces modèles sont très volumineux et difficiles à entraîner, nous avons appliqué la technique d'affinage économe en paramètres (*Parameter Efficient Fine-tuning* PEFT) Adaptation à rang faible (*Low-Rank Adaptation*, LoRA) afin de réduire le nombre de paramètres à entraîner et de diminuer le coût computationnel pour la tâche de classification. Nous avons choisi d'effectuer l'affinage du modèle Transformer Qwen3 0.6B avec LoRA pour la classification. Pour la tâche de similarité, l'affinage des SentenceTransformers avec LoRA s'est avéré impraticable en raison d'incompatibilités de bibliothèques, mais nous avons réussi à effectuer directement l'affinage complet du modèle Qwen3 0.6B-SentenceTransformer sur la tâche de similarité.

5 Résultats et analyse

Dans cette section, nous analysons les résultats obtenus par les modèles présentés à la section 4 sur nos deux tâches. Les performances en classification sont évaluées à l'aide des micro- et macro-F1. Pour la tâche de similarité, nous calculons le rang de la vraie certification selon la formule suivante :

$$\text{rank}(c_o^*, o) = \sum_{c \in C} \mathbb{1}\left[s(c, o) > s(c_o^*, o)\right].$$

où c_o^* désigne la vraie certification pour l'offre o et s la similarité cosinus entre c et o. Ce rang permet ensuite de calculer la médiane, la moyenne et la précision@k de la vraie certification. Les Tables 1 et 2 résument les principaux résultats pour les deux tâches.

Baselines : Les méthodes TF-IDF et BOW obtiennent souvent des performances solides, dépassant même des architectures plus complexes comme celles basées sur BERT ou les transformeurs Qwen lorsqu'elles ne sont pas affinées pour la tâche spécifique. Dans la tâche de classification, TF-IDF est moins performant que BOW, probablement parce qu'il surpondère certains tokens peu informatifs,

comme les durées de formation (qui devraient figurer dans des champs séparés), tandis que d'autres mots spécifiques aux classes peuvent être sous-pondérés. De plus, la régression logistique peut être induite en erreur par les fréquences de mots calculées au niveau du document plutôt qu'au niveau de la classe. En revanche, pour la tâche de similarité, TF-IDF se montre plus efficace. Bien que les représentations statiques directement entraînés sur nos données surpassent généralement les modèles non affinés, leurs performances restent inférieures à celles des meilleures représentations lexicales.

CamemBERT et affinage : Bien que les performances de CamemBERT soient moins bonnes que celles des représentations lexicales pour les deux tâches, son affinage sur la classification donne globalement les meilleurs résultats. Sans surprise, la version SentenceCamemBERT, entraînée pour la similarité, obtient de meilleurs résultats, et atteint les meilleures performances une fois affinée sur la tâche de similarité.

Pré-entraînement ERP : Notre modèle ERP CamemBERT, pré-entraîné pour apprendre les liens hiérarchiques à partir de nos données, améliore les performances de CamemBERT non affiné. Cependant, il reste moins performant qu'un affinage spécifique à chaque tâche. Nous émettons l'hypothèse que la longueur souvent importante des offres réduit l'impact des informations du graphe, et que le pré-entraînement ERP pourrait nuire à l'affinage.

Modèles multilingues : Les modèles de plongements plus récents que nous avons testés, comme E5 et Qwen, utilisés sans affinage, surpassent parfois les modèles basés sur CamemBERT non affinés. Ils restent toutefois moins performants que les modèles affinés, et dans la plupart des cas, même que les représentations lexicales. Les modèles français basés sur CamemBERT, plus légers et plus faciles à affiner, sont mieux adaptés à notre jeu de données, offrant ainsi de meilleures performances.

Utilisation d'instructions avec les LLMs : Les performances des LLMs se sont révélées systématiquement médiocres, bien en deçà des résultats obtenus par les autres modèles présentés dans cet article. Cela corrobore les résultats de (Vajjala & Shimangaud, 2025), qui montrent que les LLM sans affinage sous-performent sur les tâches de classification par rapport aux modèles traditionnels basés sur BERT, en particulier avec un grand nombre de classes. Les résultats détaillés sont disponibles en annexe 6.

Affinage des représentations de Qwen3 : nos résultats suggèrent que cette approche offre des performances légèrement supérieures à l'affinage de CamemBERT et SentenceCamemBERT pour notre tâche de similarité et en classification multi-étiquettes. Cependant, ce processus nécessite des temps d'entraînement significativement plus longs et des ressources computationnelles plus importantes, principalement en raison de l'échelle du modèle Qwen3.

6 Génération et détection d'anomalies

L'un de nos objectifs principaux étant de détecter les erreurs d'association entre les offres et les certifications, nous implémentons une tâche synthétique de détection d'anomalies, sur laquelle nous évaluons les représentations les plus performantes.

6.1 Génération d'anomalies

Nous simulons deux types d'anomalies : sur la *certification* et sur le *contenu*. Le premier type consiste à remplacer le lien entre une offre et sa certification associée par un lien incorrect. Étant donné que les organismes de formation sélectionnent eux-mêmes les certifications, ils peuvent commettre des erreurs ou associer intentionnellement des certifications incorrectes pour éviter des contrôles plus stricts. Pour un sous-ensemble aléatoire de notre jeu de données de test, nous remplaçons la certification liée par une certification tirée d'un sous-ensemble partageant au moins un grand domaine, un domaine ou un Formacode avec l'offre originale, ou sans aucun lien.

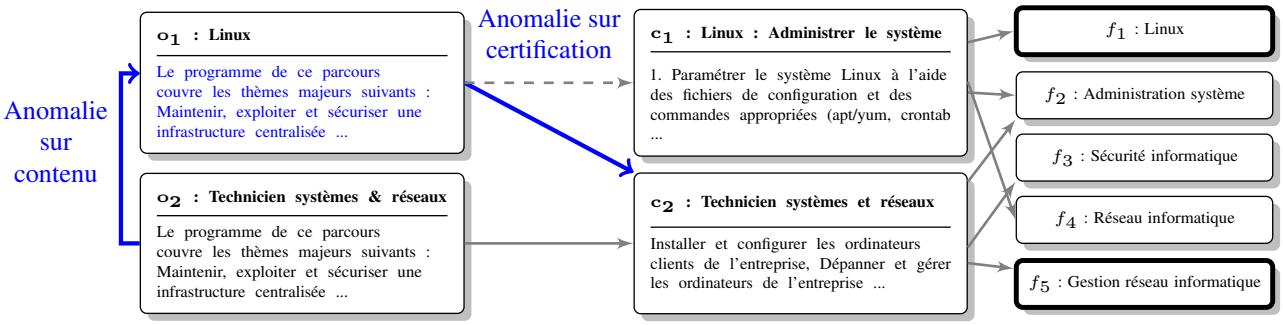

FIGURE 2 – Exemples d'anomalies sur le contenu et la certification au niveau des Formacode, indiquant que les anomalies sont générées à partir d'une autre offre partageant au moins un Formacode avec l'offre originale. Nous générons également des anomalies aux niveaux Domaine, Grand Domaine, ainsi que des anomalies aléatoires.

Le deuxième type d'anomalies consiste à remplacer une partie du texte d'une offre par un contenu incorrect. Cela se produit principalement lorsqu'un organisme de formation associe délibérément son offre de formation à une certification incompatible. Bien que le titre de l'offre corresponde à la certification, son contenu et ses objectifs diffèrent significativement, souvent en raison d'une mauvaise classification stratégique pour contourner des contraintes réglementaires ou profiter de certifications moins exigeantes. Pour simuler cette anomalie, nous sélectionnons aléatoirement un sous-ensemble de notre jeu de données de test et modifions le contenu textuel des offres, en conservant les titres inchangés. Le nouveau contenu est prélevé depuis une autre offre, échantillonnée soit à partir de l'ensemble du jeu de données, soit d'un sous-ensemble partageant au moins un grand domaine, un domaine ou un Formacode avec l'offre originale, soit sans aucun lien. La Figure 2 illustre ces deux types d'anomalies au niveau des Formacodes.

(a) Anomalies sur la certification

(b) Anomalies sur le contenu

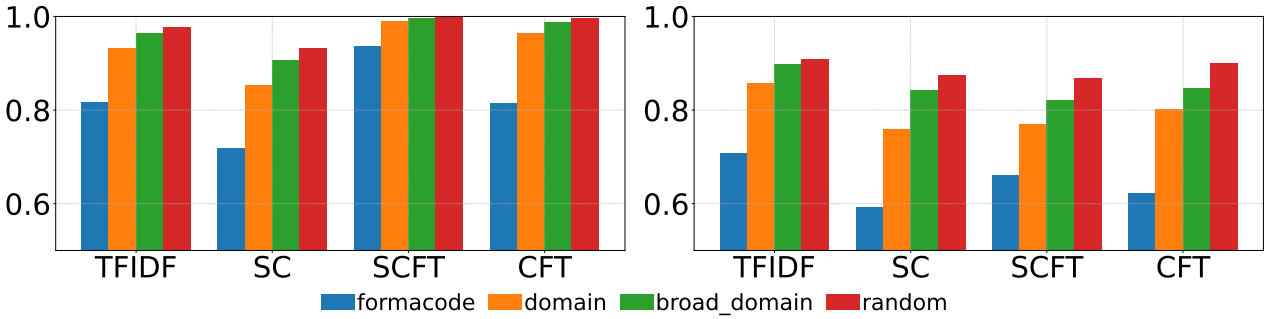

FIGURE 3 – Résultats de la Détection d'Anomalies en AUC sur les deux types d'anomalies.

6.2 Évaluation de la détection d'anomalies

Nous évaluons l'efficacité des différentes représentations par le calcul de similarité : les performances des modèles sont mesurées à l'aide de l'AUC, calculée sur les scores de similarité entre les offres et les certifications. Cette métrique permet de mesurer dans quelle mesure la représentation permet de distinguer les instances normales des anomalies, en intégrant le taux de vrais positifs par rapport au taux de faux positifs pour tous les seuils possibles de scores de similarité. La Figure 3 présente les résultats de la détection des deux types d'anomalies. Sans surprise, détecter les anomalies au niveau des Formacodes, où l'offre anormale est liée à une certification *proche*, s'avère plus difficile que d'identifier des anomalies aléatoires. La détection des anomalies basées sur le contenu est particulièrement ardue, car seule une partie du texte de l'offre est modifiée.

Pour les anomalies sur les certifications, le modèle SentenceCamemBERT affiné (**SCFT**) obtient les meilleurs résultats. Sur cette tâche, les modèles affinés sur le jeu de données se révèlent les plus prometteurs, tandis que SentenceCamemBERT non affiné (**SC**) donne les performances les plus faibles. À l'inverse, les représentations TF-IDF, directement dérivés des occurrences de mots, excellent dans la détection des anomalies de contenu des offres de formation. Ces résultats soulignent l'importance de combiner à la fois les représentations lexicales et contextuelles pour parvenir à une détection robuste des anomalies.

7 Conclusion

Dans cette étude, nous avons présenté deux tâches dans le domaine de la formation professionnelle : la classification des offres de formation selon l'arborescence des Formacodes et la mesure de la similarité entre les offres et leurs certifications correspondantes. Nos résultats ont démontré que la tâche de similarité permet de détecter efficacement des anomalies réalistes au sein du jeu de données. À travers une évaluation exhaustive des représentations, nous avons observé que les représentations lexicales simples surpassent souvent des approches plus complexes lorsqu'elles sont utilisées sans adaptation préalable. Cependant, l'affinage des modèles basés sur BERT s'avère significativement plus efficace dès lors que suffisamment de données textuelles sont disponibles, nous permettant ainsi d'exploiter pleinement leurs capacités dans un contexte spécifique au domaine. L'affinage des LLM donne des résultats légèrement meilleurs, mais à un coût computationnel très élevé, même avec l'utilisation de la technique PEFT LoRA. Cette étude met en lumière des enseignements clés pour

les applications industrielles : à l'ère des LLMs, des modèles de langue plus petits et spécifiques à une tâche, combinés à des représentations symboliques, peuvent offrir une solution plus robuste et efficace pour des structures de données et un vocabulaire complexes et propres à un domaine. Cela est particulièrement vrai pour des jeux de données réels où les représentations peuvent être utilisées pour détecter des anomalies, car le vocabulaire employé peut parfois être plus déterminant pour cette tâche que le sens capturé par les représentations contextuelles.

8 Limitations

Ce travail se concentre sur le développement de représentations flexibles, conçu pour supporter une large gamme d'applications, comme le démontrent les cas d'usage abordés, incluant l'association par similarité, la classification et la détection d'anomalies. Bien que nous ayons affiné le modèle sur nos tâches, nous avons délibérément évité des modèles de base trop spécialisés afin de conserver une polyvalence à la fois pour les scénarios de similarité et de classification. Par ailleurs, le catalogue des offres de formation professionnelle disponibles sur la plateforme française *MCF* est susceptible d'évoluer davantage, par exemple avec l'intégration de formations non certifiantes, donc non liées à des certifications. Étant donné la nature dynamique du secteur de la formation professionnelle, tant les cas d'usage que les données sous-jacentes continueront probablement d'évoluer.

Dans le contexte du secteur public, nos expérimentations se sont limitées à des modèles ouvertement disponibles, excluant les solutions propriétaires. Comme précisé dans le comparateur d'ouverture des IA génératives publié par les autorités françaises (PEReN - Pôle d'Expertise de la Régulation Numérique, 2024), il existe plusieurs niveaux d'ouverture, définis par divers critères. Nous avons sélectionné des modèles autorisant la redistribution et fournissant des poids accessibles publiquement. Ces modèles étaient également largement reconnus et utilisés dans la littérature existante.

Enfin, nous n'avons pas intégré de génération guidée par documents (*Retrieval Augmented Generation*, RAG) dans notre étude, car la nature de nos tâches ne correspondait pas à son usage prévu. La RAG est conçue pour récupérer des segments de texte spécifiques dans une base de données en réponse à des requêtes utilisateur concises, qui sont ensuite utilisés pour générer des réponses détaillées. Notre travail en revanche, s'est centré sur la comparaison de deux documents complets afin de déterminer leur degré de similarité ou de correspondance, plutôt que sur l'extraction ou la génération de texte à partir d'une entrée partielle. Les modèles RAG, optimisés pour la récupération et la génération plutôt que pour la similarité ou la classification au niveau du document, n'étaient donc pas adaptés aux objectifs de cette étude.

9 Remerciements

Ce travail a été soutenu par Hi ! PARIS et par le programme ANR/France 2030 (ANR-23-IACL-0005). Nous remercions l'IDRIS pour l'accès aux ressources de calcul haute performance sur les partitions A100 et H100 du supercalculateur Jean Zay, dans le cadre de l'allocation A0191016884, attribuée par le GENCI (Grand Équipement National de Calcul Intensif). Ce projet a été financé par l'État dans le cadre de France 2030, et financé par l'Union européenne - NextGenerationEU - dans le cadre du

plan France Relance. Nous remercions aussi Mike Zhang et Tim Luka Horstmann pour leur aide sur l'implémentation.

Références

AHO A. V. & CORASICK M. J. (1975). Efficient string matching : an aid to bibliographic search. *Communications of the ACM*, **18**(6), 333–340.

BHOLA A., HALDER K., PRASAD A. & KAN M.-Y. (2020). Retrieving skills from job descriptions : A language model based extreme multi-label classification framework. In D. SCOTT, N. BEL & C. ZONG, Éds., *Proceedings of the 28th International Conference on Computational Linguistics*, p. 5832–5842, Barcelona, Spain (Online) : International Committee on Computational Linguistics. DOI : 10.18653/v1/2020.coling-main.513.

BOJANOWSKI P., GRAVE E., JOULIN A. & MIKOLOV T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, **5**, 135–146. DOI : 10.1162/tacl_a_00051.

BRANCO R. & GOMEZ L. (2021). *Overlapy python package*. Rapport interne.

BROWN T., MANN B., RYDER N., SUBBIAH M., KAPLAN J. D., DHARIWAL P., NEELAKANTAN A., SHYAM P., SASTRY G., ASKELL A., AGARWAL S., HERBERT-VOSS A., KRUEGER G., HENIGHAN T., CHILD R., RAMESH A., ZIEGLER D., WU J., WINTER C., HESSE C., CHEN M., SIGLER E., LITWIN M., GRAY S., CHESS B., CLARK J., BERNER C., MCCANDLISH S., RADFORD A., SUTSKEVER I. & AMODEI D. (2020). Language models are few-shot learners. In H. LAROCHELLE, M. RANZATO, R. HADSELL, M. BALCAN & H. LIN, Éds., *Advances in Neural Information Processing Systems*, volume 33, p. 1877–1901 : Curran Associates, Inc.

BUTT S., CEBALLOS H. G. & MADERA-ESPÍNDOLA D. P. (2025). Tec-habilidad : Skill classification for bridging education and employment. In *Mexican International Conference on Artificial Intelligence*, p. 313–328 : Springer.

DECORTE J.-J., VAN HAUTTE J., DEMEESTER T. & DEVELDER C. (2024). Skillmatch : Evaluating self-supervised learning of skill relatedness. *arXiv preprint arXiv :2410.05006*.

DÖRPINGHAUS J., SAMRAY D. & HELMRICH R. (2023). Challenges of automated identification of access to education and training in germany. *Information*, **14**(10), 524.

ELANGOVAN A., HE J. & VERSPOOR K. (2021). Memorization vs. generalization : Quantifying data leakage in nlp performance evaluation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics : Main Volume*, p. 1325–1335.

FERREIRA A., RIBEIRO F. & NEVES A. (2025). Mapping esco skills taxonomy to educational offers using natural language processing and large language models. In *2025 6th International Conference of the Portuguese Society for Engineering Education (CISPEE)*, p. 1–7 : IEEE.

FREJ J., DAI A., MONTARIOL S., BOSSELUT A. & KÄSER T. (2024). Course recommender systems need to consider the job market. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 522–532.

GNEHM A.-S., BÜHLMANN E., BUCHS H. & CLEMATIDE S. (2022). Fine-grained extraction and classification of skill requirements in German-speaking job ads. In D. BAMMAN, D. HOVY, D. JURGENS, K. KEITH, B. O'CONNOR & S. VOLKOVA, Éds., *Proceedings of the Fifth Workshop on Natural Language Processing and Computational Social Science (NLP+CSS)*, p. 14–24, Abu Dhabi, UAE : Association for Computational Linguistics. DOI : 10.18653/v1/2022.nlpcss-1.2.

GUGNANI A. & MISRA H. (2020). Implicit skills extraction using document embedding and its use in job recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, p. 13286–13293.

HAKIMI PARIZI A., LIU Y., NOKKU P., GHOLAMIAN S. & EMERSON D. (2023). A comparative study of prompting strategies for legal text classification. In D. PREOȚIUC-PIETRO, C. GOANTA, I. CHALKIDIS, L. BARRETT, G. SPANAKIS & N. ALETRAS, Éds., *Proceedings of the Natural Legal Language Processing Workshop 2023*, p. 258–265, Singapore : Association for Computational Linguistics. DOI : 10.18653/v1/2023.nllp-1.25.

HIHN H., DITTRICH D. A., JESKE C., SOBRAL C. C., PAIS H. & LOCHMANN T. (2025). Ontology-aligned embeddings for data-driven labour market analytics. *arXiv preprint arXiv :2509.04942*.

JIECHIEU K. F. F. & TSOPZE N. (2021). Skills prediction based on multi-label resume classification using cnn with model predictions explanation. *Neural Computing and Applications*, **33**(10), 5069–5087.

JOHARY I., ROMERO R., MARA A. C. & DE BIE T. (2025). Jobhop : A large-scale dataset of career trajectories. *arXiv preprint arXiv :2505.07653*.

LE VRANG M., PAPANTONIOU A., PAUWELS E., FANNES P., VANDENSTEEN D. & DE SMEDT J. (2014). Esco : Boosting job matching in europe with semantic interoperability. *Computer*, **47**(10), 57–64.

LEE K., IPPOLITO D., NYSTROM A., ZHANG C., ECK D., CALLISON-BURCH C. & CARLINI N. (2022). Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 8424–8445.

MARTIN L., MULLER B., ORTIZ SUÁREZ P. J., DUPONT Y., ROMARY L., VILLEMONTE DE LA CLERGERIE É., SEDDAH D. & SAGOT B. (2020). CamemBERT : a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 7203–7219, Online. DOI : 10.18653/v1/2020.acl-main.645.

MENON A. K., JAYASUMANA S., RAWAT A. S., JAIN H., VEIT A. & KUMAR S. (2021). Long-tail learning via logit adjustment. In *International Conference on Learning Representations*.

MIKOLOV T., SUTSKEVER I., CHEN K., CORRADO G. S. & DEAN J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, p. 3111–3119.

NEUTEL S. & DE BOER M. H. (2021). Towards automatic ontology alignment using bert. In *AAAI Spring Symposium : Combining Machine Learning with Knowledge Engineering*, p. 1–12.

PEREN - PÔLE D'EXPERTISE DE LA RÉGULATION NUMÉRIQUE (2024). Comparateur d'ouverture de modèles d'iag. https://www.peren.gouv.fr/compare-os-iag/.

ROSENBERGER J., WOLFRUM L., WEINZIERL S., KRAUS M. & ZSCHECH P. (2025). Careerbert : Matching resumes to esco jobs in a shared embedding space for generic job recommendations. *Expert Systems with Applications*, **275**, 127043.

SCHULHOFF S., ILIE M., BALEPUR N., KAHADZE K., LIU A., SI C., LI Y., GUPTA A., HAN H., SCHULHOFF S. *et al.* (2024). The prompt report : a systematic survey of prompt engineering techniques. *arXiv preprint arXiv :2406.06608*.

SENGER E., ZHANG M., VAN DER GOOT R. & PLANK B. (2024). Deep learning-based computational job market analysis : A survey on skill extraction and classification from job postings. In E. HRUSCHKA, T. LAKE, N. OTANI & T. MITCHELL, Éds., *Proceedings of the First Workshop on*

Natural Language Processing for Human Resources (NLP4HR 2024), p. 1–15, St. Julian's, Malta : Association for Computational Linguistics.

VAJJALA S. & SHIMANGAUD S. (2025). Text classification in the llm era–where do we stand ? *arXiv preprint arXiv :2502.11830*.

WANG L., YANG N., HUANG X., YANG L., MAJUMDER R. & WEI F. (2024). Multilingual e5 text embeddings : A technical report. *arXiv preprint arXiv :2402.05672*.

WEICHSELBRAUN A., WALDVOGEL R., FRAEFEL A., VAN SCHIE A. & KUNTSCHIK P. (2022). Building knowledge graphs and recommender systems for suggesting reskilling and upskilling options from the web. *Information*, **13**(11), 510.

YAN H., LI C., LONG R., YAN C., ZHAO J., ZHUANG W., YIN J., ZHANG P., HAN W., SUN H. *et al.* (2023). A comprehensive study on text-attributed graphs : Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, **36**, 17238–17264.

YANG T., NIAN Y., LI L., XU R., LI Y., LI J., XIAO Z., HU X., ROSSI R. A., DING K., HU X. & ZHAO Y. (2025). AD-LLM : Benchmarking large language models for anomaly detection. In W. CHE, J. NABENDE, E. SHUTOVA & M. T. PILEHVAR, Éds., *Findings of the Association for Computational Linguistics : ACL 2025*, p. 1524–1547, Vienna, Austria : Association for Computational Linguistics. DOI : 10.18653/v1/2025.findings-acl.79.

ZHANG A. K., KLYMAN K., MAI Y., LEVINE Y., ZHANG Y., BOMMASANI R. & LIANG P. (2024). Language model developers should report train-test overlap. *arXiv preprint arXiv :2410.08385*.

ZHANG M., VAN DER GOOT R. & PLANK B. (2023). Escoxlm-r : Multilingual taxonomy-driven pre-training for the job market domain. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 11871–11890.

ZHANG Y., LI M., LONG D., ZHANG X., LIN H., YANG B., XIE P., YANG A., LIU D., LIN J., HUANG F. & ZHOU J. (2025). Qwen3 embedding : Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv :2506.05176*.

A Données

A.1 Description formelle des données

De manière plus formelle, les données peuvent être décrites comme un graphe tripartite $G(V, E)$, où les sommets V sont divisés en 3 groupes O, C et F. Les éléments de O et C sont composés de texte, tandis que les éléments de F contiennent une séquence ordonnée d'étiquettes, correspondant au chemin hiérarchique des Formacodes aux grands domaines. Étant donné que le graphe est tripartite, on a : $V = O \cup C \cup F$ et $O \cap C = C \cap F = O \cap F = \varnothing$. Les arêtes de E relient uniquement des éléments de O avec ceux de C et des éléments de C avec ceux de F. Dans notre cas, chaque élément de O est apparié à un seul élément de C, tandis que chaque élément de C peut être lié à entre 1 et 5 éléments de F. L'objectif est de classer les éléments de O dans les éléments de C ou F en exploitant au maximum les connaissances disponibles dans ce graphe.

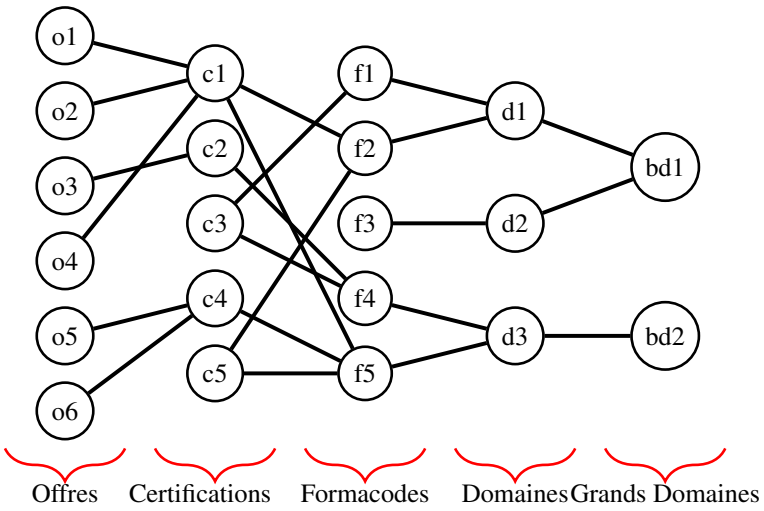

FIGURE 4 – Illustration de la structure des données et des liens entre les éléments.

A.2 Provenance et détails sur le jeu de données

Ce jeu de données a été construit par jointure à partir de plusieurs sources. Les offres proviennent de la plateforme Mon Compte Formation (MCF).[5]. Leur contenu textuel ainsi que les liens vers les certifications et les Formacodes sont consultables sur le site avec les données disponibles en open data de la Caisse des Dépôts et Consignations[6] . Les certifications et leur contenu textuel sont disponibles à la même adresse, mais sont gérées par France Compétences[7]. Les Formacodes et leurs liens avec les domaines ou grands domaines proviennent de la hiérarchie Formacode[8]. L'ensemble de nos données a été extrait en mai 2024 et ne reflète donc pas les informations actuellement disponibles sur ces plateformes.
Plusieurs difficultés découlent de ces données et de leur structure. Les données sont spécifiques au domaine de la formation professionnelle française et peuvent utiliser un vocabulaire spécialisé. Les données sont structurées sous la forme d'un graphe tripartite : chaque offre est reliée à une seule certification, tandis que chaque certification peut être associée à entre un et cinq Formacodes. Les liens entre les offres et les certifications peuvent être incomplets ou contenir des erreurs et certaines offres peuvent être presque identiques, ne différant que par de légères modifications de mots. Les Formacodes eux-mêmes forment une structure arborescente très déséquilibrée, comprenant plus de 1 000 Formacodes distincts — chacun défini uniquement par une étiquette — regroupés en 65 domaines, qui sont à leur tour organisés en 14 grands domaines. Certains Formacodes sont liés (via leur certification) à de nombreuses offres de formation, tandis que d'autres sont liés à très peu d'offres, allant de près de 20 000 à 1 (ou 0). Certains Formacodes peuvent être assez similaires les uns aux autres, et une seule offre peut être liée à plusieurs Formacodes. Étant donné que nous travaillons avec des données réelles, certains liens peuvent être incomplets ou erronés, et certains documents du jeu de données peuvent être très similaires les uns aux autres (avec seulement quelques mots changés, par exemple).
Un exemple des différents éléments du jeu de données est présenté dans la Figure 1, et la Figure 4

5. https://www.moncompteformation.gouv.fr/
6. https://opendata.caissedesdepots.fr/explore/dataset/moncompteformation_catalogueformation/
7. https://www.francecompetences.fr/
8. https://www.centre-inffo.fr/offre-formation-produits-services/le-formacode-v14

présente une vue d'ensemble de la structure des données.

A.3 Pré-traitement des données et filtrage

Lors de l'examen du jeu de données, nous avons observé que certaines offres étaient très similaires. Bien que les doublons exacts aient été supprimés lors de la phase initiale de nettoyage, certaines offres restaient très proches, ne différant que par quelques mots. Cette similarité survient souvent lorsque des organismes de formation proposent des offres comparables pour différentes certifications, par exemple pour divers types de permis de conduire. Afin d'éviter des résultats biaisés, il était essentiel d'éliminer les doublons entre les ensembles d'entraînement et de test. Cela garantit que les modèles ne soient pas exposés à des offres trop similaires pendant l'entraînement par rapport à celles utilisées pour l'évaluation en phase de test.

Dans la littérature, ce problème se pose principalement lors de l'entraînement de grands modèles de langue, car ces modèles utilisent souvent des jeux de données collectés à partir d'une multitude de sources, où des redondances peuvent survenir. Comme souligné dans l'article de prise de position de Zhang *et al.* (2024), le calcul du chevauchement entre les ensembles d'entraînement et de test est crucial pour s'assurer que les métriques dérivées des modèles ne soient pas biaisées et pour éviter une dégradation des performances entre les données vues et non vues. Plusieurs types de chevauchements ont été identifiés dans la littérature, basés sur des tokens ou des n-grammes (Brown *et al.*, 2020; Lee *et al.*, 2022; Elangovan *et al.*, 2021). Nous avons adopté la méthodologie décrite par (Brown *et al.*, 2020) pour rechercher des n-grammes communs entre les ensembles d'entraînement et de test. Cependant, nous avons choisi de supprimer uniquement les documents partageant au moins la moitié de leurs n-grammes avec d'autres documents. Étant donné que nos documents sont des offres de formation professionnelle, ils contiennent souvent des n-grammes communs car ils préparent à des certifications identiques ou similaires. Les organismes de formation peuvent intégrer des éléments de la certification dans leur texte ou reproduire des similarités à travers plusieurs offres. Toutefois, avoir plus de la moitié de leurs n-grammes en commun pourrait biaiser les résultats obtenus.

Pour ce faire, nous avons supprimé toutes les offres de l'ensemble de test qui partageaient plus de la moitié de leurs n-grammes avec une autre offre de l'ensemble d'entraînement. De plus, nous avons étudié l'impact de la suppression des redondances au sein même de l'ensemble d'entraînement. Un ensemble d'entraînement plus réduit pourrait potentiellement accélérer le processus d'entraînement et réduire les calculs inutiles. Par conséquent, nous avons également filtré l'ensemble d'entraînement pour éliminer les offres trop similaires à au moins une autre offre associée à la même certification.

Pour ces tâches de filtrage du jeu de données, nous avons utilisé la bibliothèque Python Overlapy (Branco & Gomez, 2021), qui repose sur l'algorithme d'Aho-Corasick (Aho & Corasick, 1975) et sur la méthodologie décrite dans les matériaux supplémentaires de (Brown *et al.*, 2020). L'algorithme d'Aho-Corasick utilise un automate fini pour améliorer l'efficacité de la recherche de chaînes de caractères. Pour calculer le chevauchement, l'algorithme est adapté pour utiliser les mots comme tokens. La taille des n-grammes est déterminée comme le cinquième percentile de la distribution des longueurs des exemples dans l'ensemble de test, avec un minimum de 8 et un maximum de 13, conformément à la méthodologie de (Brown *et al.*, 2020). Étant donné qu'une proportion substantielle des textes de notre corpus sont relativement longs, les n-grammes utilisés dans notre jeu de données sont de taille 13. L'ensemble de test est ensuite décomposé en n-grammes, et l'algorithme d'Aho-Corasick est utilisé pour rechercher ces n-grammes dans l'ensemble d'entraînement. Ce processus produit, pour chaque n-gramme commun, une liste composée de : [(numéro du document dans

l'ensemble de test, n-gramme, [liste des documents d'entraînement avec lesquels le n-gramme est partagé]]. La complexité de l'algorithme d'Aho-Corasick pour calculer ces n-grammes est :

$$\mathcal{O}(l + m + z)$$

où :

- l est la longueur en nombre de mots de l'ensemble d'entraînement.
- m est la somme des longueurs des éléments recherchés. Ici, $m = \sum$longueur des n-grammes $= 13 \times$ longueur de l'ensemble de test.
- z est le nombre total de correspondances rapportées par l'algorithme. Cet algorithme est dépendant de sa sortie : s'il existe de nombreux chevauchements entre les jeux de données, la complexité augmente.

Une fois les n-grammes communs entre les deux jeux de données calculés, les éléments à supprimer sont déterminés à l'aide de l'algorithme 1. Lors de l'étape 1, le nombre de n-grammes communs entre deux nœuds est calculé en traitant les résultats de l'algorithme d'Aho-Corasick. Ensuite, à l'étape 2, tous les éléments de l'ensemble de test qui présentent une proportion élevée d'éléments communs avec au moins un document de l'ensemble d'entraînement sont retirés de l'ensemble de test. Le seuil de proportion a été fixé à 0,5, ce qui signifie que tout document partageant plus de la moitié de ses n-grammes avec au moins un élément de l'ensemble d'entraînement est supprimé de l'ensemble de test. La complexité du processus de filtrage, après obtention des résultats de l'algorithme d'Aho-Corasick, est de $\mathcal{O}(z)$, car nous parcourons les correspondances retournées par l'algorithme.

Étant donné que les redondances n'apparaissent pas seulement entre les ensembles de test et d'entraînement, mais aussi au sein même de l'ensemble d'entraînement, nous avons décidé d'appliquer ce filtrage à l'ensemble d'entraînement pour éliminer les documents trop similaires entre eux. Cela réduit les redondances et la taille de l'ensemble d'entraînement. Ici, nous avons choisi de supprimer uniquement les offres similaires liées à la même certification, car les offres liées à des certifications différentes apportent des informations distinctes. Pour ce faire, nous avons conservé les étapes initiales de l'algorithme 1, mais nous les avons appliquées uniquement à l'ensemble d'entraînement, en exécutant toutes les étapes de manière itérative sur les offres liées à chaque certification. Un pseudo-code pour cet algorithme est fourni dans l'algorithme 2. Cependant, une difficulté apparaît : si deux documents de l'ensemble d'entraînement sont similaires, il faut en conserver un et ne pas supprimer les deux (ce qui se produirait avec l'algorithme 2, car ils seraient considérés comme proches d'au moins un autre document). Ainsi, la proximité des documents est d'abord calculée lors de l'étape 2, et lors de la suppression des éléments, seuls ceux qui n'ont pas tous leurs voisins supprimés sont eux-mêmes retirés. Il convient également de noter que la relation de voisinage n'est pas symétrique. Si un document est proche d'un autre, ce dernier peut ne pas être aussi proche du premier. Cette asymétrie est due aux différentes longueurs des documents. Un document plus long peut n'avoir qu'une petite proportion de son contenu en commun avec un document plus court, qui, lui, peut être presque entièrement contenu dans le document plus long.

De même, le processus de filtrage de l'ensemble d'entraînement a une complexité de $\mathcal{O}(z)$, car nous parcourons les correspondances retournées par l'algorithme ou une petite proportion de ces correspondances lors du traitement des nœuds qui sont détectés comme étant très similaires à au moins un autre nœud.

Voici les statistiques de filtrage arrondies que nous avons obtenues pour notre jeu de données, où la proportion d'éléments très similaires était assez élevée :

Algorithme 1 Filtrer les données de test par rapport aux données d'entraînement avec Aho-Corasick

Entrées :
— Données d'entraînement $Tr = [tr_1, tr_2, ..., tr_n]$
— Données de test $Te = [te_1, te_2, ..., te_m]$
— Seuil de similarité $seuil$ (par défaut : 0.5)
— Taille des n-grammes k (prédeterminée, par exemple 13)
— Correspondances Aho-Corasick $M = [(tr_i, ngram, [te_{j1}, te_{j2}, ...]), ...]$
Sortie : Données de test filtrées T'

Étape 1 : Initialiser les structures de données et traiter les résultats de l'algorithme d'Aho-Corasick
$compte_communs \leftarrow$ dictionnaire vide ▷ Clés : paires de documents, Valeurs : comptes de n-grammes communs
$total_ngrammes \leftarrow$ dictionnaire vide ▷ Clés : documents, Valeurs : comptes totaux de n-grammes
$voisins \leftarrow$ dictionnaire vide ▷ Clés : documents, Valeurs : ensembles de documents voisins
Pour chaque $doc \in T$ **faire**
    $total_ngrammes[doc] \leftarrow$ CompteTotalNGrammes(doc, k)
Fin Pour
Pour chaque $M = [(tr_i, ngram, [te_{j1}, te_{j2}, ...]), ...]$ **faire**
    Pour chaque $doc_j \in [te_{j1}, te_{j2}, ...]$ **faire**
        $compte_communs[(tr_i, te_{j1})] \leftarrow compte_communs[(te_{j1}, te_{j1})] + 1$
    Fin Pour
Fin Pour
Étape 2 : Déterminer et supprimer les documents similaires entre l'entraînement et le test
Pour chaque $(tr_i, te_{j1}) \in compte_communs.keys()$ **faire**
    $proportion \leftarrow \frac{compte_communs[(tr_i, te_{j1})]}{total_ngrammes[tr_i]}$
    Si $proportion \geqslant seuil$ **alors**
        $T'.supprime(doc)$
    Fin Si
Fin Pour
Retourne T'

— Jeu de données initial : environ 135 000 offres d'entraînement.
— Après suppression des doublons exacts : environ 106 000 offres.
— Répartition entraînement-test : Données d'entraînement : environ 80 000 offres ; Données de test : environ 26 000 offres.
— Après filtrage de l'ensemble de test avec l'algorithme 1 : Données d'entraînement : environ 80 000 offres ; Données de test, environ 10 000 offres.
— Après filtrage des similarités dans l'ensemble d'entraînement avec l'algorithme 2 : Ensemble d'entraînement : environ 46 000 offres ; Ensemble de test : environ 10 000 offres.

Les performances d'un ensemble d'entraînement filtré sur des tâches de similarité et de classification ont montré une légère amélioration pour les méthodes BOW et TF-IDF, et une légère baisse de performance pour les autres méthodes de plongements. Comme ces diminutions n'étaient pas significatives, nous avons décidé de conserver l'ensemble d'entraînement filtré pour les expériences afin de réduire le temps de calcul.

	Offres de formation			Certifications		
Grand domaine	min	max	médiane	min	max	médiane
Agriculture, environnement	4	408	70	20	669	87
Commerce, marketing, finance	3	775	112	14	2365	335
Développement des compétences	3	561	146	24	922	108
Énergie, électricité	3	496	101	24	1286	88
Génie civil, construction	3	480	110	45	1086	313
Mécanique, électronique	8	456	106	17	805	216
Production industrielle, transport, logistique	3	572	135	32	1750	163
Santé, social, sécurité	6	490	132	23	1349	278
Sciences	8	404	120	36	910	285
Sciences humaines, économie, droit, langues	7	489	123	36	1156	353
Sport, loisirs, tourisme	4	493	113	7	1682	238
Technologies de l'information et de la communication, arts	7	484	144	17	1321	189
Transformation matière produit	6	511	108	19	2179	386
Vie et gestion des organisations	6	470	116	35	1253	285
Tous les grands domaines	3	775	129	7	2365	198

TABLE 3 – Nombre de mots des offres de formation et des certifications par grand domaine

A.4 Statistiques générales du jeu de données

La table 3 présente des informations statistiques sur le nombre de mots des offres de formation et des certifications présentes sur la plateforme. Dans ce tableau, les offres certifiantes sont classées selon les grands domaines de la hiérarchie Formacode. Les comptages de mots proviennent de champs textuels agrégés : intitule, contenu et objectifs pour les offres de formation, et intitule, activités visées et capacités attestées pour les certifications. Avant l'analyse, les données textuelles subissent un nettoyage : suppression des accents, de la casse et des mots vides. La liste standard des mots vides a été étendue pour inclure des termes spécifiques au domaine, trop génériques pour apporter des informations pertinentes, tels que : formation, certification, diplôme ou examen. Une observation notable de la table 3 est la variation significative du nombre de mots entre les offres. Les offres de formation vont de quelques mots à plusieurs centaines, tandis que les certifications peuvent atteindre plusieurs milliers de mots. Cette disparité est attendue, car les certifications font généralement l'objet d'un contrôle plus strict et nécessitent des descriptions plus détaillées, ce qui entraîne des longueurs moyennes plus importantes que pour les offres de formation. Les offres avec un nombre minimal de mots manquent probablement de détails et de conformité, bien que la longueur médiane des offres de formation reste relativement élevée.

B Méthodologie expérimentale

B.1 Adaptation des modèles aux différentes tâches

Pour la tâche de similarité, la similarité est calculée entre les représentations de l'offre et de la certification. Pour chacune, cette représentation correspond à la moyenne des représentations de chaque token de la dernière couche du modèle.

Pour la tâche de classification, une régression logistique ou des classifieurs binaires (dans le cas multi-étiquettes) sont utilisés pour obtenir les résultats de classification. Lors de l'affinage sur la tâche de classification, nous avons retiré les têtes MLM et ERP de notre modèle et ajouté une tête de classification. Ces processus d'affinage ont suivi les mêmes méthodes d'optimisation et fonctions de perte que celles utilisées pour les modèles CamemBERT et SentenceCamemBERT standard.

La seule exception concerne le modèle CamemBERT, lorsqu'il a été affiné pour la tâche de classification. Dans ce cas, ses sorties peuvent être exploitées de deux manières : (i) en extrayant ses plongements et en entraînant un classifieur par régression logistique sur ceux-ci, comme pour les autres modèles, ou (ii) en utilisant directement les prédictions produites par le modèle affiné lui-même pour l'évaluation sur la tâche de classification. Nous avons testé les deux approches, mais les résultats étaient meilleurs en appliquant une régression logistique ou des classifieurs binaires sur les plongements de CamemBERT pour obtenir les résultats de classification.

B.2 Affinage des modèles

B.2.1 Fonctions de perte pour l'affinage

La fonction de perte utilisée pour l'affinage de SentenceCamemBERT sur la tâche de similarité est :

$$\mathcal{L} = \sum_{c,po,no} \max(0, ||X_c - X_{po}|| - ||X_c - X_{no}|| + \epsilon)$$

où X_c, X_{po} et X_{no} sont respectivement les plongements de la certification, de l'offre positive et de l'offre négative. Une fois affiné, le modèle SentenceCamemBERT est utilisé de la même manière que les autres modèles : les plongements sont calculés pour l'ensemble d'entraînement puis évalués sur les deux tâches.

Lors de l'affinage du modèle CamemBERT pour la classification, en simple étiquette, on utilise la perte d'entropie croisée, définie par :

$$\begin{aligned}\mathcal{L} &= nLLLoss(\log(softmax(x)), y) \\ &= -w_y \log \frac{\exp(x_y)}{\sum_{c=1}^{C} \exp(x_c)}\end{aligned}$$

où $nLLLoss$ est la perte de log-vraisemblance négative, x le vecteur de logits pour cet échantillon, y la classe réelle pour cet échantillon, C le nombre de classes et w les poids pour ce modèle. Des détails supplémentaires sur le calcul de ces poids sont disponibles en annexe B.2.2. Cette perte est ensuite moyennée sur les échantillons.
En multi-étiquettes, un classifieur séparé est entraîné pour chaque classe possible. Ici, on emploie la perte d'entropie croisée binaire (BCE) avec logits, qui combine une activation sigmoïde avec la BCE, et est définie par :

$$\mathcal{L} = -w \left[y \cdot \log \sigma(x) + (1 - y) \log \left(1 - \sigma(x)\right) \right]$$

où x est le logit pour un classifieur donné, y la vraie étiquette pour ce classifieur, et w le poids associé. Cette perte est moyennée sur les échantillons pour chaque classifieur.

Pour la perte ERP inspirée de (Zhang *et al.*, 2023), on échantillonne des paires d'offres d'entraînement $(o_1, o_2) \in O^2$, où o_1 et o_2 sont liées par l'une des relations suivantes $r \in$ (même formacode, même

domaine, même grand domaine, aléatoire). Le modèle CamemBERT reçoit alors les plongements $h_{[CLS]}$ du token $[CLS]$ correspondant à cette paire d'offres : $[CLS]X_{o_1}[SEP]X_{o_2}[SEP]$. La perte pour la tâche ERP est définie par :

$$\mathcal{L}_{ERP} = -\log p(r|h_{[CLS]})$$

Ainsi, comme défini dans (Zhang *et al.*, 2023), la perte générale pour le pré-entraînement de ce modèle est :

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{MLM} + \mathcal{L}_{ERP} \\ &= -\sum_i \log p(x_i|h_i) - \log p(r|h_{[CLS]})\end{aligned}$$

où x_i est la représentation du $i^{\text{ème}}$ token dans le texte de la paire d'offres et h_i sa représentation.

B.2.2 Calcul des poids pour l'affinage de CamemBERT en multi-étiquettes

Lors d'affinage de CamemBERT en classification multi-étiquettes, nous avons constaté que le modèle avait des difficultés à apprendre efficacement. Pour remédier à cela, nous avons introduit des poids spécifiques à chaque classe dans la fonction de perte, tout en conservant des poids uniformes $w_y = 1$ en simple étiquette.

Inspirés par l'approche de (Menon *et al.*, 2021), nous avons défini les poids comme l'inverse de la probabilité de la classe, c'est-à-dire :

$$w_y = \frac{1}{\mathbb{P}(y)} = \frac{\#\text{jeu de données}}{\#y},$$

où $\mathbb{P}(y)$ désigne la probabilité qu'une instance appartienne à la classe y. Pour garantir la robustesse, nous avons légèrement adapté cette formule :

$$w_y = \max\left(\frac{\#\text{jeu de données}}{\#y + 1}, 500\right).$$

Le terme $+1$ évite une division par zéro pour les classes absentes de l'ensemble d'entraînement, tandis que la borne supérieure de 500 atténue l'impact des classes extrêmement petites, qui entraîneraient sinon des poids disproportionnés. Nous avons expérimenté différents seuils, mais 500 a systématiquement donné les meilleures performances. Ces poids ont été appliqués lors de l'affinage de tous les modèles CamemBERT utilisés pour la classification multi-étiquettes.

B.3 Utilisation des grands modèles de langue (LLMs) avec apprentissage en contexte

Pour explorer davantage l'efficacité des différents types de représentations, nous avons également évalué l'utilisation des LLMs pour nos tâches. En nous inspirant de (Hakimi Parizi *et al.*, 2023), notre instruction intégrait le texte de l'offre à classer ainsi que les étiquettes de catégorie pertinentes. Nous avons évalué ces modèles en utilisant des instructions en anglais et en français, en employant à la fois des approches zero-shot et few-shot. Un exemple d'instruction utilisée pour la classification à

une seule étiquette est présenté dans la Figure 5. La classification a été limitée aux niveaux grand domaine et domaine, en utilisant uniquement des étiquettes simples, car celles-ci représentaient nos tâches les moins complexes ; le nombre élevé de formacodes rendait la classification basée sur les LLMs peu pratique. Pour les expériences few-shot, nous avons testé deux méthodes de sélection d'exemples : d'abord, en échantillonnant aléatoirement trois exemples étiquetés, puis en sélectionnant un exemple par classe. Ces exemples sélectionnés ont été utilisés de manière cohérente dans chaque prompt few-shot pour améliorer la reproductibilité. Nous avons également attribué au LLM un rôle (Schulhoff *et al.*, 2024) d'assistant de classification, ce qui a donné des résultats légèrement meilleurs.

```
 You are a classification assistant.

Classify the following French text into *exactly one* label from this
list of topics:

{json.dumps(topics, ensure_ascii=False, indent=2)}

Text:

"""{text}"""

Return your answer in **pure JSON** format like this (adapt the keys to
the attribute level, attribute_level can be 'grand_domaine', 'domaine' or
'formacode'):

{ "label_<attribute_level>": "<topic_code_attribute_level>",
"label_name_<attribute_level>": "<topic_name_attribute_level>",
"confidence_<attribute_level>": <float between 0 and 1>,
"explanation_<attribute_level>": "<short explanation why>"
}

Do not include any other text outside JSON.
```

FIGURE 5 – Exemple d'instruction en anglais pour la classification simple étiquette. Les Attribute_level, topics et le texte donné en entrée sont donnés en paramètres

Nous avons également tenté d'appliquer les LLMs directement à la détection d'anomalies, en utilisant une instruction inspirée de (Yang *et al.*, 2025), mais en donnant au LLM une offre et une certification et en lui demandant si elles correspondent et de fournir un score d'anomalie.

B.4 Détails d'implémentation des modèles

Pour les modèles de plongements simples, nous avons utilisé l'implémentation de scikit-learn pour BOW et TF-IDF. Word2Vec a été extrait de la bibliothèque gensim [9] et a été entraîné davantage sur nos données. FastText a été pris depuis la bibliothèque fasttext [10] sans entraînement supplémentaire, car cette version a donné les meilleurs résultats. Les modèles utilisés pour CamemBERT et Sentence-CamemBERT sont CamemBERT-base (Martin *et al.*, 2020) et SentenceCamemBERT-base [11]. Ces

9. https://radimrehurek.com/gensim/models/word2vec.html
10. https://pypi.org/project/fasttext/
11. https://huggingface.co/dangvantuan/sentence-camembert-large

modèles ont été choisis car ils sont spécifiquement entraînés sur des données en français.

L'affinage des modèles de classification a été réalisé avec un budget de 10 époques, et le meilleur modèle obtenu pendant l'entraînement a été sélectionné en fonction de ses résultats sur les données d'évaluation. Une exception a été faite pour l'affinage de CamemBERT au niveau des formacodes en contexte multi-étiquettes, car le modèle n'était pas encore complètement entraîné (ses pertes d'entraînement et d'évaluation diminuaient encore), il a donc bénéficié d'un budget de 20 époques. L'affinage du modèle SentenceCamemBERT a été effectué avec un budget de 5 époques, car 2 exemples négatifs étaient sélectionnés pour chaque exemple possible, amenant le modèle à voir chaque exemple 10 fois. Pour E5, nous avons utilisé E5-small [12], sans affinage. Pour Qwen, nous avons testé son utilisation en tant que Transformer et la version en tant que SentenceTransformer [13]. Nous avons choisi des versions réduites de E5 et Qwen pour garantir des calculs rapides, même sur un grand jeu de données et sans nécessiter une infrastructure excessivement puissante, afin de correspondre à notre contexte industriel.

Pour nos expériences utilisant directement des LLMs sans exploiter leurs plongements, nous avons sélectionné des modèles open-source à la fois largement adoptés dans la littérature et suffisamment compacts pour fonctionner localement sur nos GPUs : Llama 3.1 8B [14] et Mistral 7B [15]. Nous avons utilisés l'API d'Ollama avec une température de 0, un contexte de 128k pour Llama 3.1 8B et de 32k pour Mistral 7B. Les modèles étaient incités à produire des scores d'anomalie au format JSON, compris entre 0 et 1.

Pour l'affinage de Qwen3-SentenceTransformers sur la tâche de similarité, nous avons conservé les mêmes paramètres et le même nombre d'époques que pour l'affinage de SentenceCamemBERT. Pour l'affinage sur la tâche de classification, nous avons utilisé LoRA afin de réduire le nombre de paramètres entraînables et la complexité computationnelle. Nous avons utilisé un rang LoRA de 16, un paramètre alpha de LoRA de 32 et un *dropout* LoRA de 0,05. Pour la classification simple, nous avons entraîné sur le même nombre d'époques que CamemBERT (10 époques). Cependant, en mode multilabel, en raison des limitations en GPU haute capacité sur notre cluster de calcul, nous n'avons pu entraîner que sur 8 époques au niveau des grands domaines et des domaines, et 5 époques au niveau du formacode.

B.5 Description formelle de la génération d'anomalies

De manière plus formelle, les anomalies concernant les liens entre les offres et les certifications peuvent être décrites comme suit. Soit :

- $c_o \in C$ la certification correcte pour l'offre $o \in O$,
- $c_{oa} \in C$ la certification anormale pour l'offre o.

Si l'on cherche une certification ayant au moins un formacode en commun, où :

- $[f_p(c), \ldots, f_i(c)]$ sont les formacodes liés à la certification c,
- $f_p(c)$ est le formacode principal,
- $f_i(c)$ est le i-ème formacode,

alors une certification anormale pour l'offre o doit satisfaire :

12. https://huggingface.co/intfloat/multilingual-e5-small
13. https://huggingface.co/Qwen/Qwen3-Embedding-0.6B
14. https://huggingface.co/meta-llama/Llama-3.1-8B
15. https://mistral.ai/news/announcing-mistral-7b

$$\begin{cases} c_{oa} \neq c_o, \\ \exists i, \quad f_i(c_o) = f_i(c_{oa}). \end{cases}$$

Les anomalies concernant le contenu textuel de l'offre peuvent être décrites comme suit. Soit :

— $T(o)$ le titre de l'offre $o \in O$,

— $D(o)$ le contenu et les objectifs de l'offre o.

Une offre anormale o_a est construite comme suit :

$$o_a = T(o_a) \oplus D(o'),$$

où :

— $\oplus$ désigne la concaténation de texte,

— $o' \in O$ est une offre distincte, échantillonnée sous les contraintes ci-dessus,

— $T(o_a)$ reste inchangé (aligné avec la certification c_o).

C Résultats supplémentaires de classification

Des résultats supplémentaires de classification sont présentés dans le tableau 4.

Résultats Micro-F1 pour la classification						
	grand domaine		domaine		formacode	
	simple étiquette	multi-étiquettes	simple étiquette	multi-étiquettes	simple étiquette	multi-étiquettes
BOW	**90.8**	**89.7**	**89.2**	**86.0**	**84.3**	**79.5**
TF-IDF	90.9	87.4	88.0	79.6	77.6	63.0
CamemBERT	72.2	56.8	82.2	75.5	84.5	81.6
CamemBERT-FT	**98.5**	**93.6**	**97.7**	**91.9**	**94.7**	**87.0**
ERP-CamemBERT	90.3	87.2	87.7	81.8	78.1	64.6
ERP-CamemBERT-FT	94.7	87.9	92.5	81.9	89.5	60.7
SentenceCamemBERT	86.7	83.7	84.9	79.1	78.8	65.7
SentenceCamemBERT-FT	91.7	90.6	90.9	87.7	87.0	80.1
Qwen-Transformer	79.7	75.9	78.2	80.4	74.4	76.3
Qwen-Transformer-FT-LoRA	**94.6**	**94.1**	**92.6**	**91.7**	**88.4**	**83.5**
E5	83.8	80.8	84.1	77.7	79.7	63.1
Qwen-SentenceTransformer	87.4	83.3	85.0	76.9	74.6	57.3

TABLE 4 – Micro-F1 pour les méthodes de plongements sur la tâche de classification, en simple étiquette et multi-étiquettes, à chaque niveau de l'arborescence des Formacodes. Les meilleurs résultats dans chaque catégorie de modèles (*lexicaux*, *basés sur CamemBERT* et *multilingues*) sont en **gras**.

Les résultats concernant l'utilisation des LLMs avec apprentissage en contexte sont détaillés dans les Tableaux 5 et 6. Globalement, les performances des LLMs sur les tâches de classification ont été décevantes. Les prompts en anglais ont donné des résultats légèrement meilleurs que ceux en français, mais les deux sont restés largement inférieurs à ceux de nos autres modèles pour cette tâche. Nous attribuons cette sous-performance au caractère spécifique du domaine de nos données de formation professionnelle, qui dépasse probablement les capacités de généralisation de ces LLMs. Les expériences utilisant ces modèles en mode few-shot ont légèrement amélioré ces résultats, exposés dans la Table 7. Nos expériences sur la tâche de détection d'anomalies avec les LLMs avec apprentissage en contexte ont également été décevantes par rapport aux résultats présentés dans la Section 6.2, l'AUROC pour ces tâches n'ayant pas dépassé 0.61. Les LLMs affinés avec LoRA

sur la tâche de classification obtiennent des résultats similaires ou légèrement meilleurs que les modèles basés sur BERT (en particulier pour la classification multilabel avec la métrique F1-Macro). Cependant, ces modèles sont très volumineux à entraîner, nécessitent des GPU haute capacité (A100) qui n'étaient pas indispensables pour l'affinage des modèles basés sur BERT (qui pouvaient être entraînés sur des GPU A40), et demandent un temps d'entraînement très long. Pour obtenir de meilleurs résultats avec les modèles basés sur BERT, nous recommandons d'utiliser des versions de CamemBERT de plus grande taille (nous n'avons utilisé que CamemBERT-base dans nos expériences). Celles-ci permettraient probablement d'améliorer les performances à un coût inférieur à celui de l'affinage des LLMs.

Résultats Macro-F1 des LLMs pour la classification				
	grand domaine		**domaine**	
	simple étiquette	**multi-étiquettes**	**simple étiquette**	**multi-étiquettes**
Llama 3.1 8B sans exemple	19.0	19.9	18.3	15.5
Mistral 7B sans exemple	18.9	15.3	15.0	15.0

TABLE 5 – Résultats de l'utilisation des LLMs sur la tâche de classification avec la métrique F1-Macro pour des instructions en anglais.

Résultats Micro-F1 des LLMs pour la classification				
	grand domaine		**domaine**	
	simple étiquette	**multi-étiquettes**	**simple étiquette**	**multi-étiquettes**
Llama 3.1 8B sans exemple	17.7	22.9	19.8	14.6
Mistral 7B sans exemple	16.6	17.8	17.0	15.4

TABLE 6 – Résultats de l'utilisation des LLMs sur la tâche de classification avec la métrique F1-Micro pour des instructions en anglais.

Résultats des LLMs pour la classification au niveau grand domaine et simple étiquette		
	F1-Micro	**F1-Macro**
Llama 3.1 8B, instruction en français, un exemple par classe	26.6	20.3
Llama 3.1 8B, instruction en anglais, un exemple par classe	28.6	22.3

TABLE 7 – Résultats de l'utilisation des LLMs avec un exemple par classe sur la tâche de classification au niveau grand domaine et simple étiquette.

Algorithme 2 Filtrage interne de l'ensemble d'entraînement avec Aho-Corasick

Entrées :
— Données d'entraînement $Tr = [tr_1, tr_2, ..., tr_n]$
— Seuil de similarité $seuil$ (par défaut : 0.5)
— Taille des n-grammes k (prédéterminée, par exemple 13)
— Correspondances Aho-Corasick $M = [(tr_i, ngram, [tr_{j1}, tr_{j2}, ...]), ...]$
Sortie : Ensemble d'entraînement filtré T'

Étape 1 : Initialiser les structures de données et traiter les résultats d'Aho-Corasick
$compte_communs \leftarrow$ dictionnaire vide ▷ Clés : paires de documents, Valeurs : comptes de n-grammes communs
$total_ngrammes \leftarrow$ dictionnaire vide ▷ Clés : documents, Valeurs : comptes totaux de n-grammes
$voisins \leftarrow$ dictionnaire vide ▷ Clés : documents, Valeurs : ensembles de documents voisins
$supprims \leftarrow$ ensemble vide ▷ Ensemble des documents à supprimer potentiellement
Pour chaque $tr \in T$ **faire**
    $total_ngrammes[tr] \leftarrow$ CompteTotalNGrammes(tr, k)
Fin Pour
Pour chaque $(tr_i, ngram, [tr_{j1}, tr_{j2}, ...]) \in M$ **faire**
    Pour chaque $tr_j \in [tr_{j1}, tr_{j2}, ...]$ **faire**
        Si $tr_i \neq tr_j$ **alors**
            $compte_communs[(tr_i, tr_j)] \leftarrow compte_communs[(tr_i, tr_j)] + 1$
        Fin Si
    Fin Pour
Fin Pour
Étape 2 : Déterminer les documents similaires
Pour chaque $(tr_i, tr_j) \in compte_communs.cls()$ **faire**
    $proportion \leftarrow \frac{compte_communs[(tr_i, tr_j)]}{total_ngrammes[tr_i]}$
    Si $proportion \geqslant seuil$ **alors**
        $voisins[tr_i].ajoute(tr_j)$
        $supprims.ajoute(tr_j)$
    Fin Si
Fin Pour
Étape 3 : Appliquer la vérification des voisins pour le filtrage final
Pour chaque $tr \in supprims$ **faire**
    $garde \leftarrow$ Vrai
    Pour chaque $voisin \in voisins[tr]$ **faire**
        Si $voisin \notin supprims$ **alors**
            $garde \leftarrow$ Faux
        Fin Si
    Fin Pour
    Si non $garde$ **alors**
        $T'.supprime(tr)$ ▷ Supprimer effectivement le document des données d'entraînement
    Fin Si
Fin Pour
Retourne T'