Un algorithme d'apprentissage profond et semi-supervisé basé sur la représentation de graphes pour la classification des CV

Wissem Inoubli, Armelle Brun

Un algorithme d'apprentissage profond et semi-supervisé basé sur la représentation de graphes pour la classification des CV.

Wissem Inoubli * **, Armelle Brun **

*Keep In Touch, Strasbourg, France
** Université de Lorraine, LORIA, France

Résumé. Tout demandeur d'emploi cherche à repérer les offres d'emploi qui correspondent à son profil. Il en va de même pour les départements des ressources humaines, ou les recruteurs, qui visent à identifier les futurs collaborateurs, connus par leur CV, et qui correspondent à leurs attentes. Cependant, le nombre de demandeurs d'emploi et d'offres d'emploi est si important qu'aucun recruteur ni demandeur d'emploi n'est capable d'examiner manuellement tous les CV et toutes les offres. Pour pallier cette limite, les systèmes de recommandation sont utilisés dans le but de recommander aux demandeurs d'emploi et aux services des ressources humaines, des offres d'emploi et des CV respectivement. L'une des approches adoptées par la littérature repose sur l'identification automatique d'éléments de contenu dans les offres et dans les CV, qui contribuent à effectuer la correspondance automatiquement. Dans ce travail, nous proposons de représenter les données (les CV) sous forme de graphes et d'aborder ce problème de recommandation comme un problème de classification. Nous présentons DGL4C, un modèle d'apprentissage profond semi-supervisé à base de graphes. DGL4C est un modèle d'apprentissage profond qui apprend la représentation adéquate du graphe et entraîne un classifieur sur cette représentation latente. Les expériences menées sur un jeu de données publique de CV anonymisés montrent que DGL4C améliore significativement la précision d'un modèle traditionnel d'apprentissage profond, tel que sBERT, et confirme la pertinence de s'appuyer sur une structure de graphe pour la tâche de classification de CV.

1 Introduction

Le recrutement peut être vu comme le processus qui vise à faire correspondre des offres d'emploi et des CV. Cette correspondance est effectuée à la fois par les demandeurs d'emploi (manuellement) et par les services des ressources humaines (RH) (grâce à l'utilisation de systèmes de gestion des candidats (ATS), par exemple JobSCAN [1]). En raison de l'énorme volume de CV et d'offres, cette tâche ne peut plus être effectuée manuellement. Les algorithmes de recherche d'information (RI) sont traditionnellement utilisés pour effectuer cette

[1] https://www.jobscan.co/applicant-tracking-systems

tâche. Ils combinent des techniques d'extraction de caractéristiques avec un modèle de recherche (par exemple, le modèle standard booléen). Ainsi, dans Zaroor et al. (2017), l'objectif est de trouver des CV pertinents dans un corpus de CV en fonction des besoins d'un recruteur. Par ailleurs, les systèmes de recommandation sont désormais utilisés comme un outil commun conçu pour recommander aux RH les CV qui correspondent à une offre d'emploi donnée, ou à un demandeur d'emploi les offres pertinentes selon son profil Giabelli et al. (2021).

Bien que les offres d'emploi puissent être facilement collectées pour constituer un jeu de données, la collecte des CV est une tâche plus délicate, en raison de la confidentialité des données personnelles. Des ensembles de CV sont généralement collectés par des entreprises privées et ne sont pas disponibles gratuitement. Pire, très peu d'ensembles de données annotées, mettent en correspondance les offres et les CV. Ainsi, les approches par apprentissage automatique ne peuvent être directement utilisées. En outre, l'évaluation des modèles d'appariement CV/offre reste une étape difficile en raison du manque de base d'apprentissage contenant cet appariement. Par conséquent, l'approche basée sur le contenu, qui identifie les CV et les offres d'emploi similaires en termes de contenu, est l'approche la plus adéquate pour effectuer cette correspondance.

Pour faire face à cette limite, nous proposons d'effectuer, comme la littérature, cette correspondance en utilisant le profil professionnel qui décrit les CV et les offres. Concrètement, nous pensons que le profil d'un candidat, par exemple *informaticien*, et l'occupation d'une offre d'emploi, par exemple également *informaticien*, peuvent être utilisés pour effectuer cette mise en correspondance. A l'opposé de la littérature, nous proposons de considérer l'identification de ces informations de plus haut niveau comme un problème de classification, tel que proposé dans Yao et al. (2019). Le défi ici est donc de concevoir un classifieur de CV ou d'offres d'emploi, qui sont des documents en texte brut non structurés.

La littérature sur la classification de textes s'appuie traditionnellement sur le pré-traitement des textes. Par exemple, TF-IDF (*term- Frequency-inverse Document frequency*), LDA (*Latent Dirichlet Allocation*) Kim et al. (2019), et word2vec Mikolov et al. (2013) sont des modèles traditionnels d'extraction de caractéristiques et de représentation de textes. La classification est ensuite effectuée par des algorithmes d'apprentissage automatique supervisés, comme les forêts aléatoires, les arbres de décision et les machines à vecteurs de support, qui exploitent ces représentations Ding et al. (2021). La littérature a montré que les performances des classifieurs dépendent fortement de la qualité de la représentation des éléments à classifier, ici des CV ou des offres Hasan et al. (2020). L'apprentissage profond a récemment été étudié. Il effectue l'extraction de caractéristiques et la classification en une seule étape, contrairement aux travaux précédents. De nombreuses variantes de réseaux neuronaux ont été étudiées, comme la mémoire à long terme (LSTM) Huang et al. (2015), une architecture de réseau neuronal récurrent (RNN) Zhou et al. (2015), ou un réseau neuronal convolutif (CNN) Jiechieu et Tsopze (2021). L'apprentissage profond a montré une amélioration significative, par rapport aux approches d'apprentissage automatique. Cependant, dans le contexte de la classification des CV, l'apprentissage profond souffre toujours de taux d'erreur élevés et d'une faible précision de classification Zaroor et al. (2017). Cela peut probablement s'expliquer par la taille limitée des ensembles de données d'entraînement, puisque l'apprentissage profond nécessite généralement de grandes masses d'entraînement Li et al. (2017).

Les structures de graphes sont traditionnellement adoptées pour gérer des données riches et structurées. Récemment, des modèles d'apprentissage profond de graphes Yao et al. (2019),

qui permettent d'apprendre un espace non-euclidien de données, ont émergé. De manière surprenante, A notre connaissance ils n'ont pas été étudiés dans le contexte des RH, en particulier pour la classification de CV. Dans ce travail, nous proposons DGL4C, pour *Deep Graph Representation Learning for Classification*, un nouveau modèle de classification basé sur l'apprentissage profond des graphes. DGL4C est un modèle semi-supervisé, conçu pour la classification de CV, qui gère à la fois des données étiquetées (CV) et non étiquetées (éléments de CV).

Concrètement, nous proposons deux variantes de DGL4C. DGL4C-GCN, est un réseau neuronal convolutif de graphe de bout en bout, qui apprend toutes les étapes entre la phase initiale d'entrée et le résultat final de sortie (classification du résumé). DGL4C-GRL est composé de deux étapes : (i) la représentation du texte (CV) par une architecture GCN, et (ii) un classifieur basé sur l'apprentissage automatique.

La suite de ce document est organisée comme suit. La section 2 présente la littérature relative à la classification des CV. Dans la section 3, nous présentons DGL4C et ses deux variantes DGL4C-GCN et DGL4C-GRL. Ensuite, dans la section 4, les résultats expérimentaux sont décrits et analysés. Enfin, dans la section 5, nous concluons et proposons des perspectives.

2 État de l'art

Dans cette section, nous présentons des travaux relatifs à la classification de CV dans le domaine des ressources humaines.
La littérature a proposé plusieurs approches, que nous choisissons de diviser en trois catégories : (i) les modèles basés sur les ontologies, (ii) les modèles d'apprentissage automatique et (iii) les modèles d'apprentissage profond.

Considérons tout d'abord les modèles basés sur les ontologies. Une ontologie est un méta-modèle conceptuel qui représente une connaissance du domaine Fazel-Zarandi et Fox (2009). Après une étape d'extraction de caractéristiques, ces modèles utilisent des ontologies pour effectuer la classification. Quelques bases de connaissances internationales et nationales en matière de RH ont été publiées. Les plus connues sont le DISCO[2], la CITP[3] et l'ESCO[4]. de Groot et al. (2021). Ces bases de connaissances représentent des groupes de professions à différents niveaux de granularité.

En ce qui concerne les modèles d'apprentissage automatique, largement utilisés, ils reposent sur des données d'entraînement et nécessitent une étape de pré-traitement dédiée à l'extraction de caractéristiques des éléments à classer. Les modèles d'apprentissage automatique, tels que les forêts aléatoires, les arbres de décision et les machines à vecteurs de support, etc. ont montré une efficacité et des performances élevées pour la tâche de classification de CV Fareri et al. (2021).
Dans ces modèles, la qualité de l'étape d'extraction des caractéristiques a un impact important sur les performances de classification. À l'opposé des modèles basés sur les ontologies et de l'apprentissage automatique, les modèles d'apprentissage profond considèrent à la fois l'extraction de caractéristiques et la classification en une seule étape, ce qui réduit la potentielle perte d'informations dans l'étape d'extraction de caractéristiques. Les modèles d'apprentissage profond sont très populaires et ont montré une amélioration significative des perfor-

[2]Dictionnaire européen des aptitudes et des compétences
[3]Classification internationale type des professions
[4]European Skills, Competences, Qualifications and Occupations

mances. Plusieurs travaux ont été proposés pour la classification des offres d'emploi Jiechieu et Tsopze (2021); Giabelli et al. (2021); Sajid et al. (2022); Abdollahnejad et al. (2021) où les architectures de réseau neuronal convolutif 1-D (CNN) et de réseau neuronal récurrent (RNN) ont été adaptées dans le contexte des RH. Les techniques d'apprentissage de représentation de graphes sont apparues récemment et sont utilisées dans nombreuses applications. nombreuses applications diverses, notamment la recherche et la découverte de médicaments , la prédiction des accidents de la circulation, ainsi que la détection des fraudes Wu et al. (2022). La structure de graphe est un moyen traditionnel de représenter les données, mais les modèles qui s'appuient sur une telle représentation souffrent de la rareté des données et d'un manque de robustesse au bruit, ce qui diminue la performance des modèles prédictifs. Pour surmonter ces limites, l'apprentissage par représentation de graphes a été conçu pour transformer les données dans un nouvel espace de dimension réduite. Il a montré son efficacité pour les données non structurées telles que les images, les textes et les graphes Yao et al. (2023); Kumar et al.. L'apprentissage de la représentation de graphes peut être classé en trois familles : les modèles basés sur la factorisation matricielle (Zhang et al. (2021); **?**), les modèles basés sur la marche aléatoire (Grover et Leskovec (2016)) et les modèles basés sur les réseaux de neurones de graphes (GNNs - Graph Neural Networks) (Gori et al. (2005); Kipf et Welling (2016)). Ces derniers sont des réseaux neuronaux utilisés pour apprendre le plongement des nœuds en agrégeant les informations des nœuds voisins par le biais des arêtes. L'agrégation de voisinage consiste à transmettre et à recevoir des données entre les nœuds, à travers leurs voisins. Dans les GNNs, un nœud a un nombre illimité de voisins directs, alors que dans les autres architectures de réseaux neuronaux, le nombre de voisins directs est limité (par exemple, deux pour les architectures RNN et huit pour les architectures 2D-CNN). Ce nombre illimité de voisins directs a démontré sa capacité à encoder des informations à la fois structurelles et sémantiques (caractéristiques des nœuds), ce qui fait son succès Yao et al. (2023).

Les architectures de GNNs Gori et al. (2005) font l'objet d'une attention croissante dans le domaine de l'apprentissage de représentations pour les données de type graphe, et de nombreux modèles sont proposés dans la littérature Kumar et al. . En partant de la première architecture de réseaux convolutifs de graphes (GCN) Kipf et Welling (2016), GraphSage **?** a été proposé pour surmonter le problème de passage à l'échelle des GCN, en changeant la méthode de convolution. Dans le même contexte, un mécanisme d'attention a été proposé par une autre variante de GCN appelée GAT Veličković et al. (2017). À notre connaissance, l'architecture GNN n'a pas été étudiée pour la modélisation de CV ou d'offres d'emploi, et nous faisons l'hypothèse qu'ils pourraient améliorer les performances de classification, ce que nous proposons d'étudier ci-dessous.

3 DGL4C : un algorithme de classification basé sur les réseaux convolutifs de graphes

Partant du constat que la représentation des données est une étape essentielle pour les algorithmes de classification Hasan et al. (2020), nous proposons ici une nouvelle approche pour la représentation des CV, inspirée de Yao et al. (2019), et basée à la fois sur une structure de graphe et sur des informations contextuelles (les caractéristiques des noeuds). Concrètement, nous proposons DGL4C, un modèle d'apprentissage semi-supervisé de la représentation de

CV, basé sur une architecture GNN. Comme mentionné précédemment, la principale motivation pour le choix d'un apprentissage de représentation de graphes, et plus particulièrement d'une architecture GNN, est motivé par le fait que ces architectures exploitent des informations de voisinage (des voisins) lors de l'étape d'apprentissage, et qui permettent une représentation enrichie du graphe.

La quantité de données de CV étant généralement limitée, nous optons pour un algorithme d'apprentissage semi-supervisé, c'est-à-dire qui exploite à la fois des exemples étiquetés et non étiquetés ; le processus d'entraînement repose donc sur ces deux types de données Kipf et Welling (2016). DGL4C vise à apprendre une nouvelle représentation latente de qualité des CV, qui sera utilisée ensuite dans une étape de classification. Dans les sous-sections suivantes, nous présentons la façon dont nous proposons de construire un graphe de CV, puis l'idée centrale de l'apprentissage de représentation de graphes, et la manière dont nous développons DGL4C, conçu pour encoder un jeu de données de CV dans un espace vectoriel latent.

3.1 Construction du graphe

Avant l'étape d'apprentissage d'une représentation de données, la première phase consiste à construire le graphe des CV. Nous proposons de nous inspirer du travail mené dans Yao et al. (2019), notamment l'étape de construction de graphe.

Soit $D = (R, L)$ un jeu de données. R est l'ensemble des CV, $R_i \in R$ est un CV avec $R_i = \{wd_{i1}, wd_{i2}, ..., wd_{ij}\}$ est l'ensemble des j mots du CV R_i. $Wd = \bigcup_i R_i$ est le vocabulaire de D, c'est-à-dire l'ensemble des mots distincts dans R.

Y représente l'ensemble complet des étiquettes possibles(les classes), c'est-à-dire les classes auxquelles les CV peuvent appartenir. Chaque CV R_i est associé à une étiquette Y_i qui fait partie de cet ensemble Y. La cardinalité de Y est notée $|Y|$

Definition 3.1 (R-graphe) *Soit G un graphe hétérogène, avec attributs et non pondéré construit à partir de D.*

G = (V , E) avec V et E représentant respectivement les nœuds et les arêtes de G. L'ensemble de nœuds $V = R \cup Wd$ est l'union de l'ensemble des CV et de l'ensemble des mots uniques des CV. Par conséquent. Ainsi, V est composé de deux types de nœuds : les nœuds de type CV et les nœuds de type mots. L'ensemble des arêtes E est également divisé en deux types, les arêtes entre mot et CV et les arêtes mot à mot. En d'autres termes, une arête entre deux mots représente leur co-occurrence dans le même CV, tandis qu'une arête entre un mot et un CV indique la présence de ce mot dans ce CV.

De manière similaire à Yao et al. (2019), une arête entre deux nœuds existe si la similarité entre ces nœuds est positive. La façon dont cette similarité est évaluée dépend du type d'arête. Les arêtes mot à mot sont évaluées à l'aide du traditionnel TF-IDF, qui est calculé comme présenté en équation (1) :

$$\text{TF-IDF(wd,}r_i\text{,R)} = TF(wd, r_i)IDF(wd, R) \quad (1)$$

Avec $TF(wd, r_i)$ désigne le nombre de fois où le mot wd apparaît dans le CV r_i ; $IDF(wd, R)$ qui mesure l'importance de wd dans l'ensemble de CVs R. IDFIDF est calculé en prenant le logarithme du rapport entre le nombre total de CVs dans la collection (R) et le nombre de CVs contenant le mot wd. Les mots qui apparaissent fréquemment dans l'ensemble des CVs reçoivent un score IDF plus faible, car ils sont considérés comme moins discriminants ou des

mots vides. Les arêtes mot à mot sont évaluées par l'information mutuelle spécifique (PMI), voir équation (2).

$$PMI(i,j) = log\frac{p(i,j)}{p(i)p(j)} \tag{2}$$

$$p(i,j) = \frac{\#W(i,j)}{|R|} \tag{3}$$

$$p(i) = \frac{\#W(i)}{|R|} \tag{4}$$

Où $\#Wd(i)$ est le nombre de CV qui contiennent le mot wd_i, $\#W(i,j)$ est le nombre de CV qui contiennent à la fois les mots wd_i et wd_j et $|R|$ est le nombre de CV dans le corpus. L'équation (5) représente la matrice d'adjacence A du graphe G.

$$A_{i,j} = \begin{cases} 1 & i, j \text{ sont deux mots, et } PMI(i,j) > 0 \\ 1 & i \text{ est un CV et } j \text{ est un mot, et } TF\text{-}IDF_{i,j} > 0 \\ 1 & i = j \\ 0 & \text{sinon} \end{cases} \tag{5}$$

3.2 Apprentissage de représentation d'un graphe

Après avoir construit le graphe G, l'apprentissage de la représentation de graphe représente la seconde étape de DGL4C. Un GCN est un réseau de neurones convolutifs sur les graphes qui effectue des opérations similaires à celles du CNN, à l'exception qu'il applique une convolution sur un graphe au lieu d'une convolution sur un tableau 2-D Kipf et Welling (2016). Un GCN apprend une représentation latente en propageant l'information des voisins directs dans le graphe et applique une transformation linéaire. La procédure de propagation de l'information consiste à agréger les informations provenant des voisins directs et indirects. Ensuite, comme le perceptron, les GCN appliquent une transformation linéaire suivie d'une non-linéarité ponctuelle. La méthode de propagation GCN, définie dans Kipf et Welling (2016), est rappelée ici :

$$H^l = \sigma(\hat{A}H^{l-1}W^l),\ avec\ H^0 = X \tag{6}$$

Où $\hat{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ est la matrice d'adjacence symétrique normalisée et H^l et H^{l+1} sont respectivement la matrice d'état caché précédente et la nouvelle, W^l est une matrice de poids entraînable pour la couche l, et σ désigne toute fonction d'activation non linéaire (par exemple, ReLU). L'étape de convolution dans le GCN est basée sur le passage de messages qui est divisé en sous-étapes (i) collecte de messages et (ii) agrégation. La collecte des messages consiste à obtenir des messages de voisins, et l'agrégation consiste à normaliser tous les messages afin d'obtenir un plongement d'un nœud v.

Dans Yao et al. (2019), les auteurs ont utilisé les caractéristiques initiales des nœuds comme une matrice d'identité $X = I_{|V|}$. Dans DGL4C, la matrice des caractéristiques des nœuds X est le vecteur de caractéristiques de chaque nœud de G. Le modèle pré-entraîné bien connu sBERT (Reimers et Gurevych (2019)) est utilisé pour encoder à la fois les mots et les CV pour construire X. La dernière couche du GCN représente l'espace latent Z qui encode les CV sous forme de matrice numérique, et elle est normalisée par la fonction *softmax*. La fonction de perte est définie comme l'erreur d'entropie croisée uniquement sur les nœuds étiquetés :

$$L=-\sum_{r\in Y_r}\sum_{f}^{F} Y_{rf}\ \ln\ Z_{rf} \tag{7}$$

Où Y est l'ensemble des indices de CV qui ont des étiquettes et F est la dimension de la dernière couche du GCN, qui est le nombre de classes. Comme lesétiquettes des CV sont utilisées par la fonction de perte, seuls les nœuds de type CV sont utilisés.

Rappelons que le graphe contient des CV et des éléments de CV (mots). C'est cette particularité qui justifie l'apprentissage semi-supervisé de DGL4C. En outre, des expériences que nous avons menées ont montré que la méthode d'agrégation graphSage est plus performante que les méthodes d'agrégation GCN Kipf et Welling (2016) et GAT Veličković et al. (2017), c'est donc l'agrégation graphSage (avec l'agrégation moyenne) qui est utilisée par DGL4C.

Nous proposons deux variantes de DGL4C, qui diffèrent par le nombre d'étapes qui les composent.

- DGL4C-GCN est un modèle compact (de bout en bout) avec une étape unique qui apprend à la fois la représentation et la classification.
- DGL4C-GRL est un modèle composé de deux étapes : la représentation du texte (résumé) puis la classification.

4 Experimentations

Nous nous intéressons maintenant à l'évaluation de DGL4C, et en particulier des deux modèles DGL4C-GCN et DGL4C-GRL. Nous choisissons d'évaluer les modèles au travers de la précision, que nous comparons à celle d'algorithmes de l'état de l'art.

4.1 Protocole expérimental

Le jeu de données utilisé est un corpus de 2 484 CV librement disponibles et anonymisés [5]. Chaque CV est associé à une étiquette, qui représente le profil du CV. 24 profils (classes) sont disponibles. Chaque CV est écrit en langage naturel et contient des informations personnelles, la formation, des expériences, etc. Dans les expérimentations menées, nous nous intéressons en particulier à l'impact du nombre de classes (profils) sur la précision des modèles. Ainsi, nous formons plusieurs ensembles de données qui varient par le nombre de classes qu'il contient. Les statistiques relatives à chacun de ces ensembles de données ainsi que celles relatives aux graphes construits sont présentés dans le Tableau 1.

DGL4C-GCN et DGL4C-GRL ont été implémentés en utilisant le framework DGL [6] avec deux couches de convolution de l'architecture GraphSage **?** pour permettre le passage de messages entre les nœuds, et l'agrégation moyenne. D'un point de vue architectural, nous avons fixé la taille d'intégration de la première couche de convolution à 500. Nous avons réglé d'autres paramètres et fixé le taux d'apprentissage à 0,001, le *dropout* à 0,2. Pour chaque jeu de données, nous utilisons aléatoirement 80 % des CV pour l'entraînement et le reste pour le test.

[5] https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset
[6] https://www.dgl.ai/

Jeux de données	# CV (# Classes)	# Profiles # Noeuds	Représentation basée sur les graphes # Noeuds Annoté	# Arêtes	
D1	500	5	6,705	500	3,887,179
D2	1,000	11	10,304	1,000	6,724,780
D3	1,500	16	11,076	1,500	7,917,124
D4	2,000	20	12,838	2,000	9,566,638
D5	2,484	24	14,183	2,484	10,976,856

TAB. 1 – *Jeux de données*

4.2 Résultats expérimentaux

Pour évaluer l'efficacité de DGL4C-GCN et DGL4C-GRL, nous comparons leurs performances avec plusieurs modèles de la littérature, qui diffèrent soit par la représentation du texte, soit par l'étape du classification. Chaque modèle consiste en deux étapes, représentation de texte et un classifieur, à l'exception du modèle de bout en bout DGL4C-GCN. Les modèles de représentation de texte les plus populaires dans la littérature, et mentionnés dans la section 2, sont utilisés. La liste exhaustive de ces modèles est présentée dans la partie supérieure du Tableau 2, et les différentes versions de DGL4C sont presentées dans la partie inférieure.

Model Name	Text Representation	Classifier
TF-IDF+RF	TF-IDF Wu et al. (2008)	Random Forest
Word2Vec+RF	Word2Vec Mikolov et al. (2013)	Random Forest
sBERT+RF	sBERT Reimers et Gurevych (2019)	Random Forest
DGL4C-GCN	DGL4C-GCN	DGL4C-GCN
DGL4C-GRL+RF	DGL4C-GRL	Random Forest
DGL4C-GRL+LR	DGL4C-GRL	Logistic Regression
DGL4C-GRL+SVC	DGL4C-GRL	Support Vector Classification
DGL4C-GRL+MLP	DGL4C-GRL	Multi-Layer Perceptron

TAB. 2 – *Liste des modèles étudiés.*

4.2.1 Impact de la représentation du texte

Nous nous concentrons tout d'abord sur l'évaluation de l'impact de la représentation du texte, en fixant le classifieur. Nous choisissons ainsi d'utiliser l'algorithme populaire de la forêt aléatoire (RF) (voir Tableau 2). Le tableau 3 présente la précision de ces modèles, y compris DGL4C, en fonction du jeu de données utilisé.

Etudions tout d'abord la précision des modèles sur le jeu de données complet (D5). Comme attendu, TF-IDF+RF est le modèle le moins performant (précision=30,09), TF-IDF étant une représentation simple. Les représentations basées sur l'apprentissage profond : Word2Vec+RF et sBERT+RF sont plus performantes, avec une précision de 49,65% et 60,23% respectivement. sBERT+RF est plus performant que Word2Vec+RF, ce qui est conforme à la littérature. Précisons que sBERT est le modèle actuel le plus performant en traitement automatique des langues (TAL), spécifiquement sur la tâche de similarité sémantique textuelle Reimers et Gurevych (2019). En ce qui concerne les modèles de représentation basés sur les graphes, que

Modèle \ jeux de données	D1	D2	D3	D4	D5
TF-IDF+RF	70,50	42,43	35,65	34,65	30,09
Word2Vec+RF	78,43	63,16	51,76	52,24	50,64
sBERT+RF	92,23	73,15	68,14	66,97	61,65
DGL4C-GCN	93,54	**75,23**	**74,65**	73,66	62,43
DGL4C-GRL+RF	**94,38**	73,65	73,76	**75,76**	**67,87**

TAB. 3 – *Précision moyenne des modèles étudiés.*

nous proposons, DGL4C-GCN, modèle de bout en bout, est légèrement plus performant que sBERT+RF, mais cette augmentation n'est pas statistiquement significative. Quant à DGL4C-GRL+RF, il est plus performant que DGL4C-GCN et significativement plus performant que sBERT+RF. Nous pouvons conclure que la représentations de graphe est adéquate pour la tâche de classification de CV et par conséquent que l'utilisation de l'information de voisinage dans l'apprentissage de la représentation, qui combine à la fois l'information sémantique et structurelle permet d'améliorer la modélisation.

Intéressons-nous maintenant à l'impact du nombre de classes sur les performances des modèles, en étudiant les performances sur les jeux de données D1 à D5, c'est-à-dire de entre 5 et 24 classes. Comme attendu, les performances de chaque modèle sont négativement impactées par l'augmentation du nombre de classes. Par exemple, la précision de DGL4C-GRL+RF est de 94,38% avec 5 classes et diminue à 67,87% avec 24 classes. Cependant, cette performance ne diminue pas linéairement avec le nombre de classes. En particulier, les performances entre 11 et 20 classes restent stables. Une diminution significative se produit entre 20 et 24 classes, de 75,76% à 67,87%. Une diminution similaire se produit également pour l'autre modèle à base de graphe DGL4C-GCN. Cependant, ce n'est pas le cas pour les modèles basés sur l'apprentissage profond (Word2Vec), ni pour le TF-IDF. Il est par ailleurs important de noter que, quel que soit le nombre de classes, les modèles à base de graphe sont toujours ceux qui ont la précision la plus élevée.

Nous pouvons ainsi conclure que les modèles basés sur la représentation de graphe sont plus performants que les modèles récents basés sur l'apprentissage profond. Cependant, ils semblent être moins robustes lorsque le nombre de classes augmente. Des expériences supplémentaires mériteraient d'être menées pour identifier si la baisse de performance est due au nombre de classes ou aux caractéristiques des 4 classes supplémentaires de D5.

4.2.2 Impact du classifieur

Nous nous concentrons maintenant sur l'évaluation de l'impact du classifieur sur les performances de DGL4C-GRL. Nous évaluons plusieurs classifieurs populaires dans la littérature, à savoir la classification par vecteurs support, le perceptron multicouche, la régression logistique, que nous comparons à la forêt aléatoire précédemment étudiée. La liste des modèles étudiés est présentée dans le tableau 2. La précision moyenne de ces modèles est présentée dans le tableau 4, qui rappelle également les performances du meilleur modèle d'apprentissage profond sBERT+RF et du modèle DGL4C-GCN bout en bout basé sur les représentation de graphes. Tout d'abord, nous pouvons constater que quel que soit le classifieur utilisé, DGL4C-GRL est plus performant que sBERT+RF sur la plupart des versions des jeux de données. Si nous

Modèle \ jeux de données	D1	D2	D3	D4	D5
DGL4C-GRL+LR	**95,67**	73,99	74,23	74,85	67,10
DGL4C-GRL+SVC	94,20	72,03	74,60	75,65	**69.04**
DGL4C-GRL+MLP	95,08	72,25	72,96	73,76	65,77
DGL4C-GRL+RF	94,38	73,65	73,76	**75,76**	67,87
DGL4C-GCN	93,54	**75,23**	**74,65**	73,66	62,43
sBERT+RF	92,23	73,15	68,14	66,97	61,65

TAB. 4 – *Précision moyenne de DGL4C-GRL avec différents classifieurs*

nous concentrons sur l'impact du classifieur sur les performances de DGL4C-GRL, LR et SVC sont les deux classifieurs les plus performants, qui surpassent légèrement les performances de RF. Cependant, cette amélioration n'est pas statistiquement significative. Nous pouvons donc conclure que la nature du classifieur n'a pas d'impact significatif sur la performance du modèle. Au contraire, la représentation de graphe semble être l'étape la plus influente, ce qui confirme les résultats de la littérature. En ce qui concerne DGL4C-GCN, il s'agit du modèle le plus performant pour deux jeux de données (D2 et D3). Cependant, DGL4C-GCN a une performance significativement plus faible sur D5 (62,43% précision) par rapport au modèle le plus performant DGL4C-GRL+SVC (69,04% précision). Cela peut s'expliquer par le fait qu'un modèle de bout en bout a une fonction d'optimisation qui optimise en même temps l'apprentissage de la représentation et la classification, alors que DGL4C-GRL a deux fonctions d'optimisation utilisées séparément, ce qui rend le classifieur plus flexible.

4.2.3 Sensibilité des hyper-paramètres

Les hyper-paramètres jouent un rôle important dans l'apprentissage de la représentation des graphes, car ils déterminent la manière dont les plongements de nœuds seront générés. Dans cette sous-section, nous menons des expérimentations pour analyser l'impact de deux paramètres clés, à savoir la dimension de plongement et le nombre de couches de convolution. Le tableau 5 montre le comportement du modèle par rapport à la variation de la taille de plongement.

Jeux de données \ Taille de plogement	64	128	256	512	1024
D1	92,1	93,15	**94,23**	93,85	92,12
D2	70,20	72,03	72,43	**76,15**	72,05
D3	70,08	72,25	73,60	**76,26**	72,77
D4	71,23	73,65	73,56	**74,34**	71,87
D5	58,54	59,23	61,57	**63,69**	61,13

TAB. 5 – *Impact de taille de plongement sur la précision du modèle*

Dans cette expérimentation nous avons fait varier la dimension de plongement de 64 à 1024 pour toutes les données étudiées. Les résultats expérimentaux (voir Tableau 5) montrent clairement que le modèle obtient de meilleures performance avec une dimension de 1 024 pour les jeux de données D2, D3, D4 et D5 tandis que la meilleure précision de D1 est obtenue avec

une dimension de 512. Cela peut être expliqué par le fait que l'augmentation du nombre de dimensions d'un système de coordonnées permet de repérer d'une manière efficace un point dans l'espace, et qui permet d'avoir une meilleure représentation de données. Mais lorsque la taille de plongement dépasse un certain seuil, la performance diminue lentement. En outre, avec le jeu de données D1 la meilleure représentation était apprise avec seulement 256 dimensions. Cela peut être dû au fait que le jeu de données en question (D1) est modeste par rapport aux autres jeux de données.

Jeux de données \ Nombre de couche	1	2	3	4
D1.	92,41	**94,29**	89,65	90,75
D2	**77,32**	75,87	76,64	74,43
D3	**74,88**	73,76	71,43	66,32
D4	**73,93**	71,60	69,65	67,25
D5	65,54	**67,32**	63,28	62,32

TAB. 6 – *Impact du nombre de couches de convolution sur la précision du modèle.*

Nous étudions maintenant l'impact du nombre de couches de convolution de DGL4C sur la performance. Les résultats sont présentés dans le Tableau 6. Ils montrent que les meilleures performances ont été atteintes avec seulement une ou deux couches de convolution. En outre, à l'opposé de la taille de plongement, nous ne constatons aucun impact du jeu de données (ou de sa taille) sur la performance. En effet, la quasi totalité des jeux de données ont obtenu de bonnes performances avec une ou deux couches de convolution.

5 Conclusion

Dans cet article, nous avons proposé DGL4C, un modèle d'apprentissage profond semi-supervisé pour la classification des CV basé sur la représentation de graphes. Ce modèle peut être utilisé pour fournir des recommandations aux ATS, aux départements de ressources humaines et aux réseaux sociaux professionnels en ligne (par exemple Linkedin, Viadeo, Meetup, JobCase, etc). DGL4C s'appuie sur une approche d'apprentissage de représentation profonde et adapte l'architecture GCN à partir de données textuelles. Les expériences menées démontrent les très bonnes performances des deux variantes de DGL4C : DGL4C-GCN et DGL4C-GRL. En effet, les deux variantes sont plus performantes que les modèles de la littérature basés sur l'apprentissage automatique et l'apprentissage profond, en particulier que sBERT qui a montré de bonnes performances sur des cas d'usage proches. Dans nos travaux futurs, nous prévoyons d'adopter un apprentissage non supervisé de l'apprentissage de représentation de graphes au lieu d'un apprentissage semi-supervisé, et d'évaluer notre modèle avec des jeux de données plus larges.

?refname?

Abdollahnejad, E., M. Kalman, et B. H. Far (2021). A deep learning bert-based approach to person-job fit in talent recruitment. In *CSCI*. IEEE.

de Groot, M., J. Schutte, et D. Graus (2021). Job posting-enriched knowledge graph for skills-based matching. *arXiv preprint arXiv:2109.02554*.

Ding, Y., X. Zhao, Z. Zhang, W. Cai, et N. Yang (2021). Graph sample and aggregate-attention network for hyperspectral image classification. *Geoscience and Remote Sensing Letters 19*, 1–5.

Fareri, S., N. Melluso, F. Chiarello, et G. Fantoni (2021). Skillner: Mining and mapping soft skills from any text. *Expert Systems with Applications*.

Fazel-Zarandi, M. et M. S. Fox (2009). Semantic matchmaking for job recruitment: an ontology-based hybrid approach. In *ISWC*, Volume 525, pp. 2009.

Giabelli, A., L. Malandri, F. Mercorio, M. Mezzanzanica, et A. Seveso (2021). Skills2job: A recommender system that encodes job offer embeddings on graph databases. *Applied Soft Computing 101*, 107049.

Gori, M., G. Monfardini, et F. Scarselli (2005). A new model for learning in graph domains. In *IJCNN*, Volume 2, pp. 729–734.

Grover, A. et J. Leskovec (2016). node2vec: Scalable feature learning for networks. In *SIGKDD*.

Hasan, F., A. Roy, et S. Pan (2020). Integrating text embedding with traditional nlp features for clinical relation extraction. In *ICTAI*, pp. 418–425. IEEE.

Huang, Z., W. Xu, et K. Yu (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Jiechieu, K. et N. Tsopze (2021). Skills prediction based on multi-label resume classification using cnn with model predictions explanation. *Neural Computing and Applications*.

Kim, D., D. Seo, S. Cho, et P. Kang (2019). Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec. *Information Sciences*.

Kipf, T. N. et M. Welling (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kumar, V. S., A. Alemran, D. A. Karras, S. K. Gupta, C. K. Dixit, et B. Haralayya. Natural language processing using graph neural network for text classification. In *ICKES22*.

Li, P., J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, et K. Chen (2017). Multi-key privacy-preserving deep learning in cloud computing. *Future Generation Computer Systems*.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, et J. Dean (2013). Distributed representations of words and phrases and their compositionality. *NIPS 26*.

Reimers, N. et I. Gurevych (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Sajid, H., J. Kanwal, S. U. R. Bhatti, S. A. Qureshi, A. Basharat, S. Hussain, et K. U. Khan (2022). Resume parsing framework for e-recruitment. In *IMCOM*, pp. 1–8. IEEE.

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Lio, et Y. Bengio (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wu, H. C., R. W. P. Luk, K. F. Wong, et K. L. Kwok (2008). Interpreting tf-idf term weights as making relevance decisions. *ACM TOIS 26*(3), 1–37.

Wu, L., P. Cui, J. Pei, L. Zhao, et X. Guo (2022). Graph neural networks: foundation, frontiers and applications. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4840–4841.

Yao, D., Z. Zhi-li, Z. Xiao-feng, C. Wei, H. Fang, C. Yao-ming, et W.-W. Cai (2023). Deep hybrid: multi-graph neural network collaboration for hyperspectral image classification. *Defence Technology 23*, 164–176.

Yao, L., C. Mao, et Y. Luo (2019). Graph convolutional networks for text classification. In *AAAI*, Volume 33, pp. 7370–7377.

Zaroor, A., M. Maree, et M. Sabha (2017). Jrc: a job post and resume classification system for online recruitment. In *29th ICTAI*, pp. 780–787. IEEE.

Zhang, X., K. Xie, S. Wang, et Z. Huang (2021). Learning based proximity matrix factorization for node embedding. In *SIGKDD*, pp. 2243–2253.

Zhou, C., C. Sun, Z. Liu, et F. Lau (2015). A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Summary

The main goal of job seekers is to identify job offers that match their profile. The same stands for human resource departments that aim to identify candidates, through their resumes, that match the recruiter's expectations. However, the number of job seekers and job offers is so important that none of human resource employees nor job seekers is able to go through all the resumes and offers manually. Recommender systems have emerged these last years with the goal to recommend job seekers and human resource departments, job offers and resumes respectively. One of the approaches adopted by the literature relies on the identification of content elements in the offers and resumes that contribute to perform matching. We propose to represent data under the form of graphs and approach this problem as a classification problem. We present DGL4C, a semi-supervised graph deep learning model, that learns the adequate representation from a graph and trains a classifier on this latent representation. Experiments are carried out on an open dataset of anonymous resumes. Results show that DGL4C significantly improves precision and accuracy of a traditional deep learning models, such as sBERT and confirm the pertinence of relying on a graph structure for the classification task in HR domain.