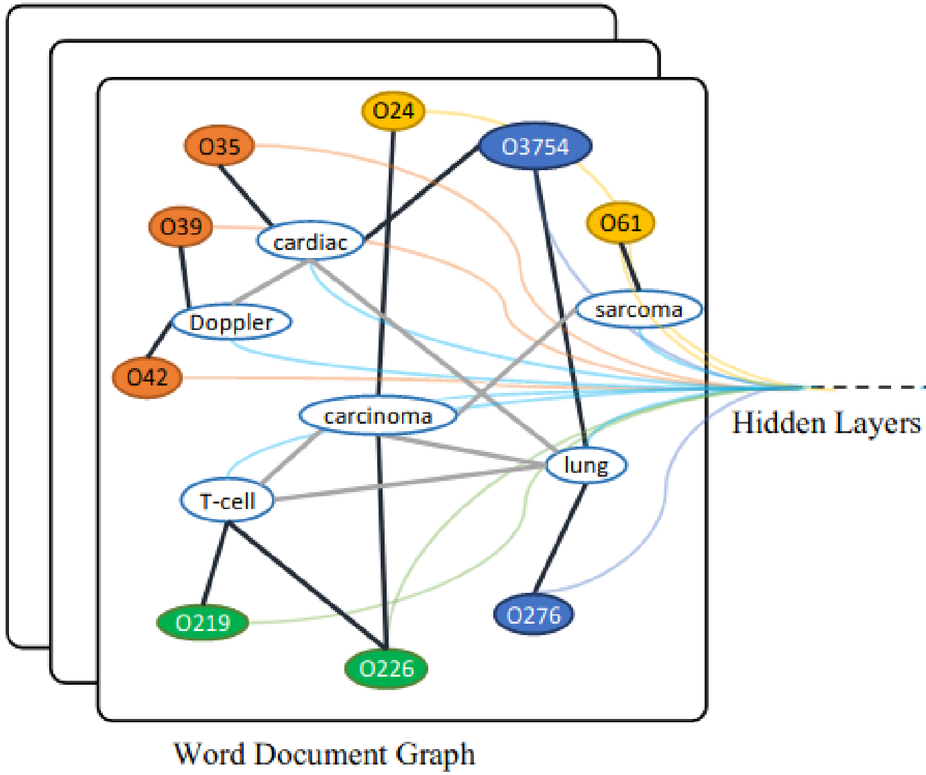
Projet 2 - Sciences des données **30 juin 2022**

Graph Convolutional Networks for Text Classification

Liang Yao, Chengsheng Mao, Yuan Luo (2018)



Word Document Graph

Clément Chauvet, Elise Chin, Mathilde Da Cruz

Introduction

Papier publié en 2018

Liang Yao, Chengseng Mao, Yuan Luo

IAAA

Présentation globale

Tâche de classification de documents

Représentation par graphe

r

GCN très simple (et pas de connaissances externes)
+ Apprentissage semi-supervisé

Apprentissage d'embeddings

Etat de l'art - Related works

Méthodes traditionnelles

Bag of words, n-grams

Deep Learning

Modèles basés sur les embeddings de mots

Graph Neural Networks

Graphe de noeuds mots

Contribution du papier

- Représentation du corpus sous forme d'un graphe hétérogène
- Apprend l'embedding de mots et de documents jointement
- Originalité du traitement du problème : transforme un problème de classification de documents en problème de classification de noeuds
- SATO pour classification sans modèle pré-entrainé ou connaissances extérieures
- Fonctionne bien avec faible % de labels

Méthode

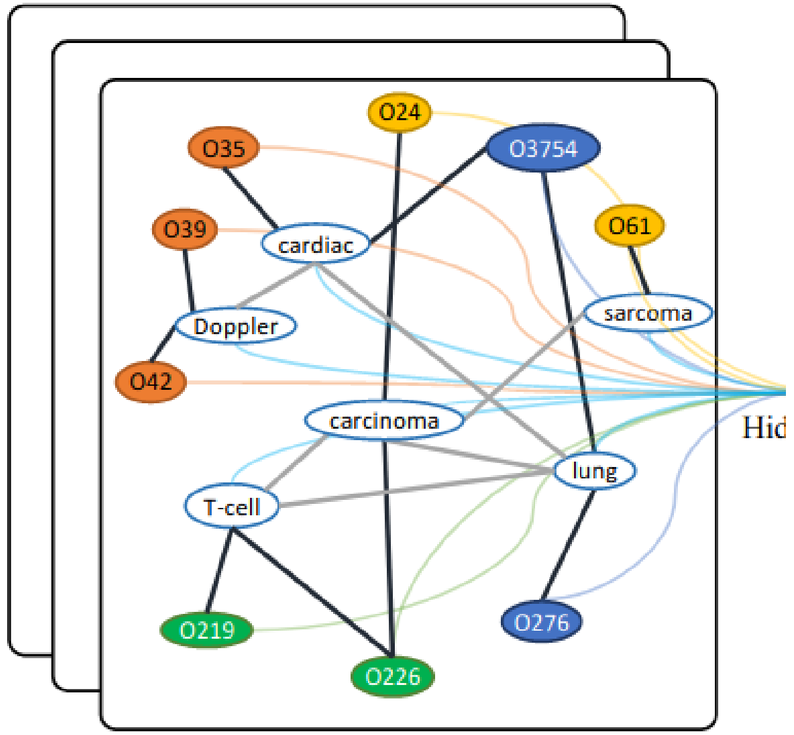
1. Construction d'un graphe hétérogène
2. Entraînement d'un GCN et inférence

Construction du graphe (1/2)

G = (V, E)

- V = {mots, documents}
- E = {arêtes entre deux mots, arêtes entre un mot et un document}
- Poids des arêtes :

$$A_{ij} = \begin{cases} \text{PMI}(i,j) & i,j \text{ are words, PMI}(i,j) > 0 \\ \text{TF-IDF}_{ij} & i \text{ is document, } j \text{ is word} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases}$$



Word Document Graph

Construction du graphe (2/2)

PMI(mot i, mot j)

$$p(i,j) = \frac{\#W(i,j)}{\#W}$$

$$p(i) = \frac{\#W(i)}{\#W}$$

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

Calculé sur l'ensemble du corpus :

- #W(i) est le nombre de fenêtres contenant le mot i
- #W(i, j) est le nombre de fenêtres contenant les mots i et j
- #W est le nombre total de fenêtres

TF-IDF(doc d, mot t)

$$\mathbf{tf}(t,d) = \frac{f_d(t)}{\max\limits_{w \in d} f_d(w)}$$

$$\mathbf{idf}(t,D) = \ln\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right)$$

$$\mathbf{tfidf}(t,d,D) = \mathbf{tf}(t,d) \cdot \mathbf{idf}(t,D)$$

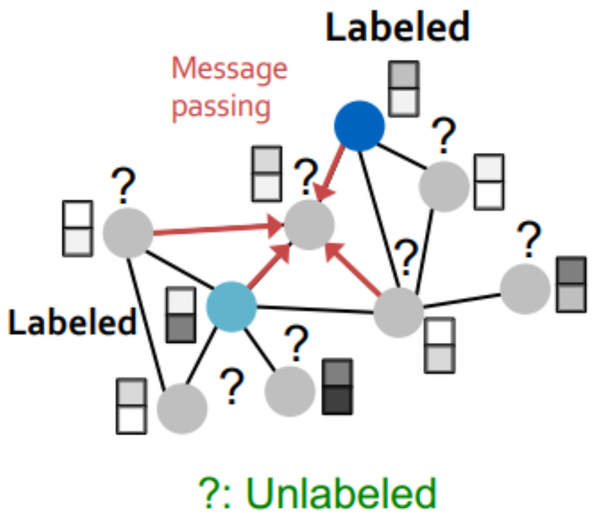
$f_d(t) :=$ frequency of term t in document d

$D :=$ corpus of documents

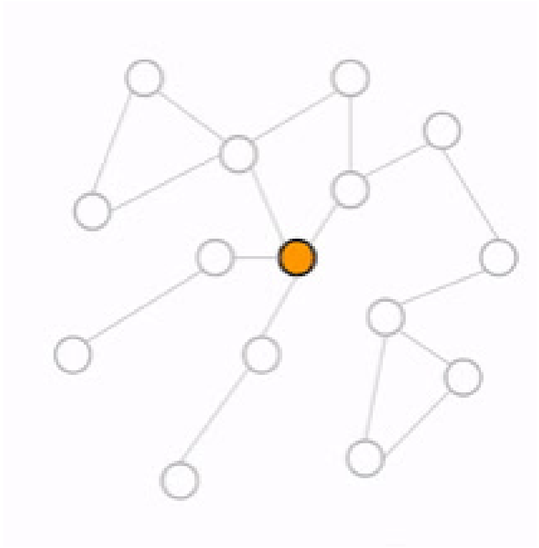
Entraînement GCN et classification (1/2)

Intuition

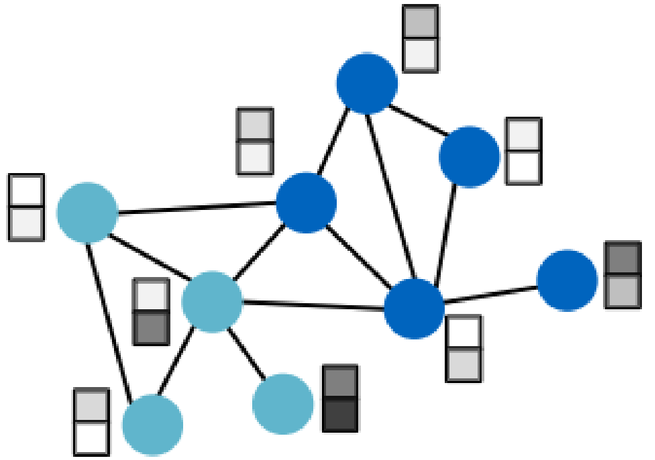
Entrée : graphe partiellement labellisé



Entraînement : propagation des features



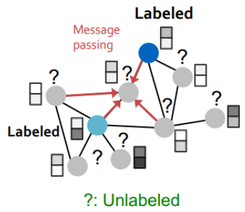
Inférence : classification des documents non labellisés



Entraînement GCN et classification (2/2)

Formulation

Entrée



G = (V, E), |V| = n

- Matrice de features X
  - X0 = I

Entraînement

Couche d'un GCN :

$$L^{(1)} = \rho(\tilde{A}XW_0)$$
$$L^{(j+1)} = \rho(\tilde{A}L^{(j)}W_j)$$

avec $\tilde{A} = D^{\frac{1}{2}}(A + I)D^{\frac{1}{2}}$

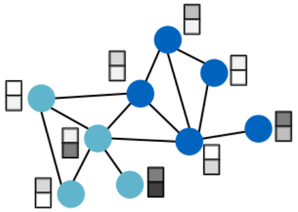
Dans le papier, GCN à deux couches :

$$Z = \text{softmax}(\tilde{A}\,\text{ReLU}(\tilde{A}XW_0)W_1)$$

Loss = cross-entropy entre les documents labellisés et prédictions

$$\mathcal{L} = -\sum_{d \in \mathcal{Y}_D} \sum_{f=1}^{F} Y_{df} \ln Z_{df}$$

Inférence



Sortie de la 2ème couche introduit dans un classifieur softmax

Résultats

Evaluation sur 2 taches et plusieurs datasets

- Le modèle peut-il atteindre des résultats satisfaisants dans la classification de textes, même avec des données limitées ?
- Le modèle peut-il apprendre des embeddings de mots et documents de bonne qualité ?

Table 1: Summary statistics of datasets.

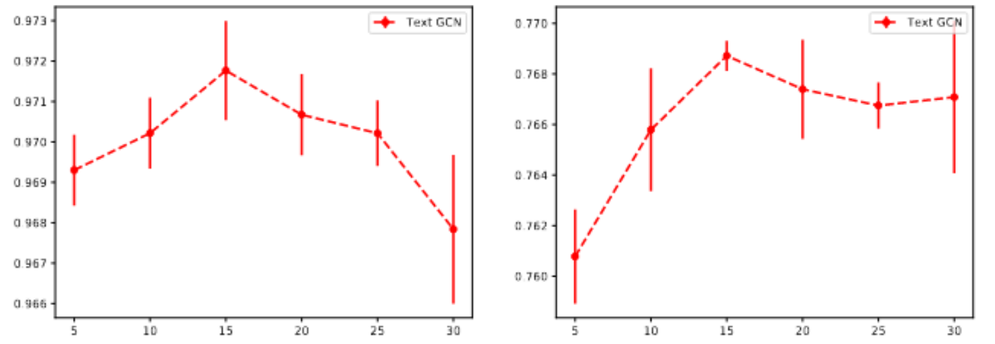
Dataset	# Docs	# Training	# Test	# Words	# Nodes	# Classes	Average Length
20NG	18,846	11,314	7,532	42,757	61,603	20	221.26
R8	7,674	5,485	2,189	7,688	15,362	8	65.72
R52	9,100	6,532	2,568	8,892	17,992	52	69.82
Ohsumed	7,400	3,357	4,043	14,157	21,557	23	135.82
MR	10,662	7,108	3,554	18,764	29,426	2	20.39

Résultats de la classification

Table 2: Test Accuracy on document classification task. We run all models 10 times and report mean ± standard deviation. Text GCN significantly outperforms baselines on 20NG, R8, R52 and Ohsumed based on student t-test ($p < 0.05$).

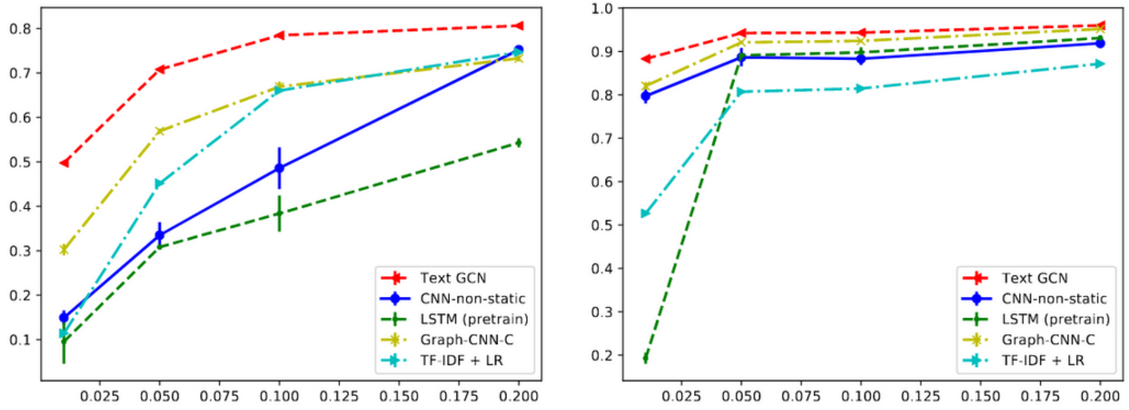
Model	20NG	R8	R52	Ohsumed	MR
TF-IDF + LR	0.8319 ± 0.0000	0.9374 ± 0.0000	0.8695 ± 0.0000	0.5466 ± 0.0000	0.7459 ± 0.0000
CNN-rand	0.7693 ± 0.0061	0.9402 ± 0.0057	0.8537 ± 0.0047	0.4387 ± 0.0100	0.7498 ± 0.0070
CNN-non-static	0.8215 ± 0.0052	0.9571 ± 0.0052	0.8759 ± 0.0048	0.5844 ± 0.0106	**0.7775 ± 0.0072**
LSTM	0.6571 ± 0.0152	0.9368 ± 0.0082	0.8554 ± 0.0113	0.4113 ± 0.0117	0.7506 ± 0.0044
LSTM (pretrain)	0.7543 ± 0.0172	0.9609 ± 0.0019	0.9048 ± 0.0086	0.5110 ± 0.0150	0.7733 ± 0.0089
Bi-LSTM	0.7318 ± 0.0185	0.9631 ± 0.0033	0.9054 ± 0.0091	0.4927 ± 0.0107	0.7768 ± 0.0086
PV-DBOW	0.7436 ± 0.0018	0.8587 ± 0.0010	0.7829 ± 0.0011	0.4665 ± 0.0019	0.6109 ± 0.0010
PV-DM	0.5114 ± 0.0022	0.5207 ± 0.0004	0.4492 ± 0.0005	0.2950 ± 0.0007	0.5947 ± 0.0038
PTE	0.7674 ± 0.0029	0.9669 ± 0.0013	0.9071 ± 0.0014	0.5358 ± 0.0029	0.7023 ± 0.0036
fastText	0.7938 ± 0.0030	0.9613 ± 0.0021	0.9281 ± 0.0009	0.5770 ± 0.0049	0.7514 ± 0.0020
fastText (bigrams)	0.7967 ± 0.0029	0.9474 ± 0.0011	0.9099 ± 0.0005	0.5569 ± 0.0039	0.7624 ± 0.0012
SWEM	0.8516 ± 0.0029	0.9532 ± 0.0026	0.9294 ± 0.0024	0.6312 ± 0.0055	0.7665 ± 0.0063
LEAM	0.8191 ± 0.0024	0.9331 ± 0.0024	0.9184 ± 0.0023	0.5858 ± 0.0079	0.7695 ± 0.0045
Graph-CNN-C	0.8142 ± 0.0032	0.9699 ± 0.0012	0.9275 ± 0.0022	0.6386 ± 0.0053	0.7722 ± 0.0027
Graph-CNN-S	–	0.9680 ± 0.0020	0.9274 ± 0.0024	0.6282 ± 0.0037	0.7699 ± 0.0014
Graph-CNN-F	–	0.9689 ± 0.0006	0.9320 ± 0.0004	0.6304 ± 0.0077	0.7674 ± 0.0021
Text GCN	**0.8634 ± 0.0009**	**0.9707 ± 0.0010**	**0.9356 ± 0.0018**	**0.6836 ± 0.0056**	0.7674 ± 0.0020

Performances de la classification avec différents paramètres



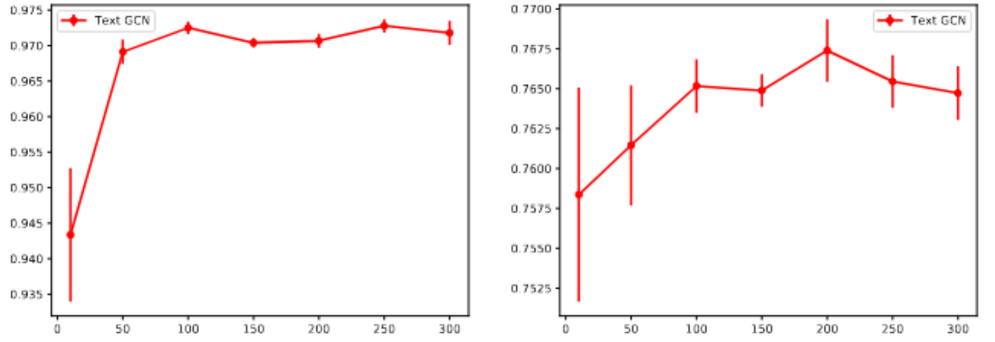
(a) R8 (b) MR

Figure 2: Test accuracy with different sliding window sizes.



(a) 20NG (b) R8

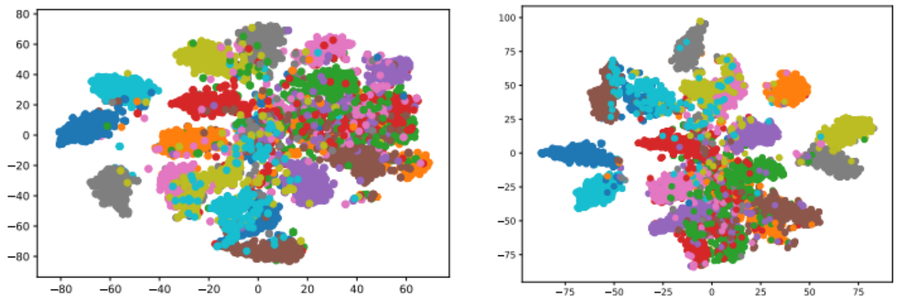
Figure 4: Test accuracy by varying training data proportions.



(a) R8 (b) MR

Figure 3: Test accuracy by varying embedding dimensions.

Embeddings



(a) Text GCN, 1st layer (b) Text GCN, 2nd layer

(c) PV-DBOW (d) PTE

Figure 5: The t-SNE visualization of test set document embeddings in 20NG.

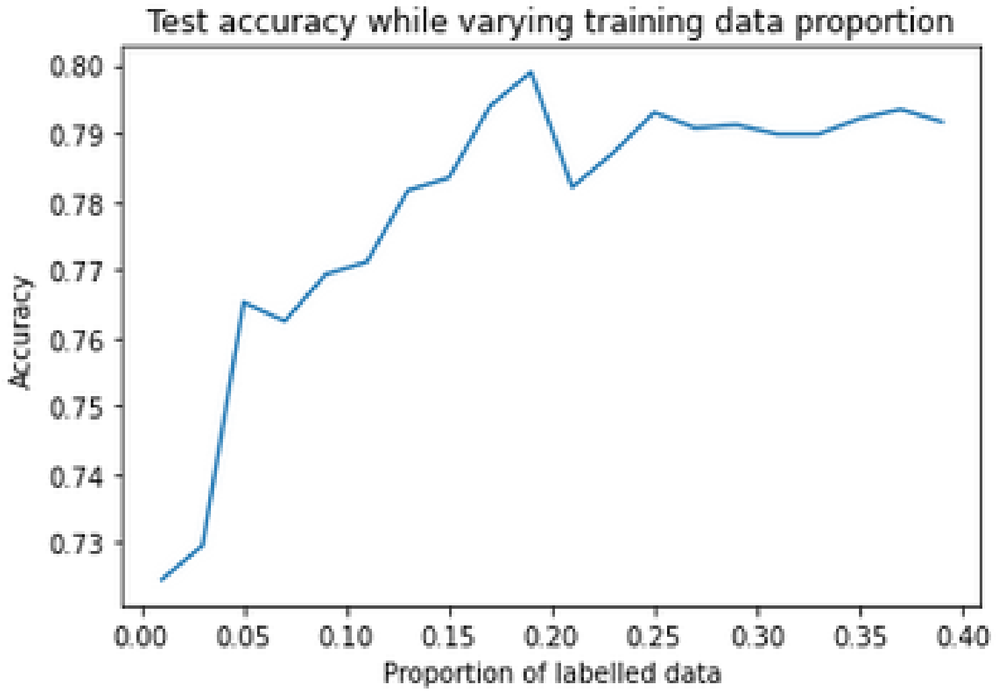
Table 3: Words with highest values for several classes in 20NG. Second layer word embeddings are used. We show top 10 words for each class.

comp.graphics	sci.space	sci.med	rec.autos
jpeg	space	candida	car
graphics	orbit	geb	cars
image	shuttle	disease	v12
gif	launch	patients	callison
3d	moon	yeast	engine
images	prb	msg	toyota
rayshade	spacecraft	vitamin	nissan
polygon	solar	syndrome	v8
pov	mission	infection	mustang
viewer	alaska	gordon	eliot

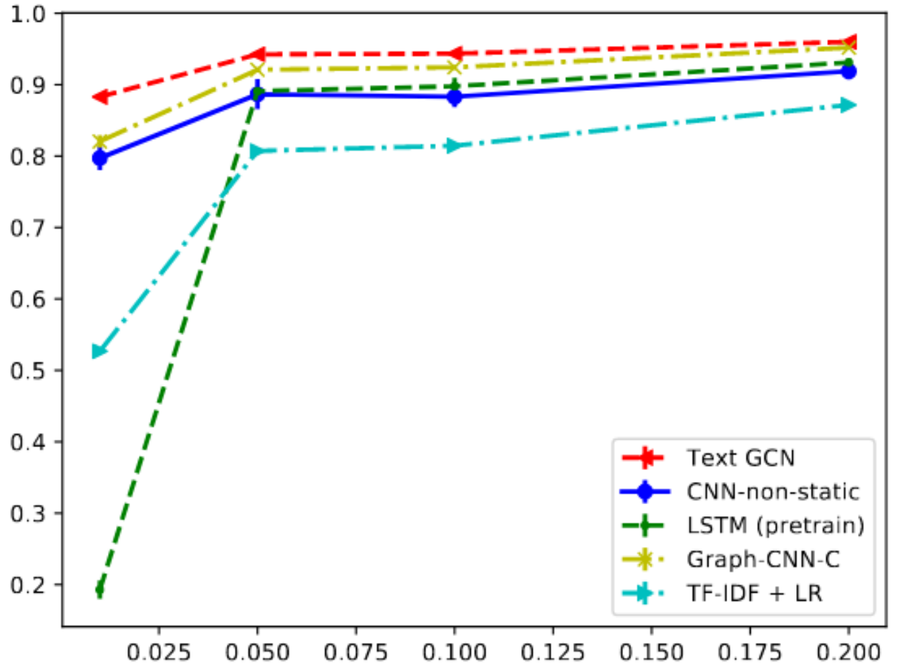
Expérimentations

Réimplémentation sur R8

Notre implémentation



Résultats annoncés dans le papier



Modification proposée

PMI(mot i, mot j)

$$p(i,j) = \frac{\#W(i,j)}{\#W}$$

$$p(i) = \frac{\#W(i)}{\#W}$$

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

Avantage : rend mieux compte de l'utilisation des mots et du contexte dans le cadre des documents étudiés

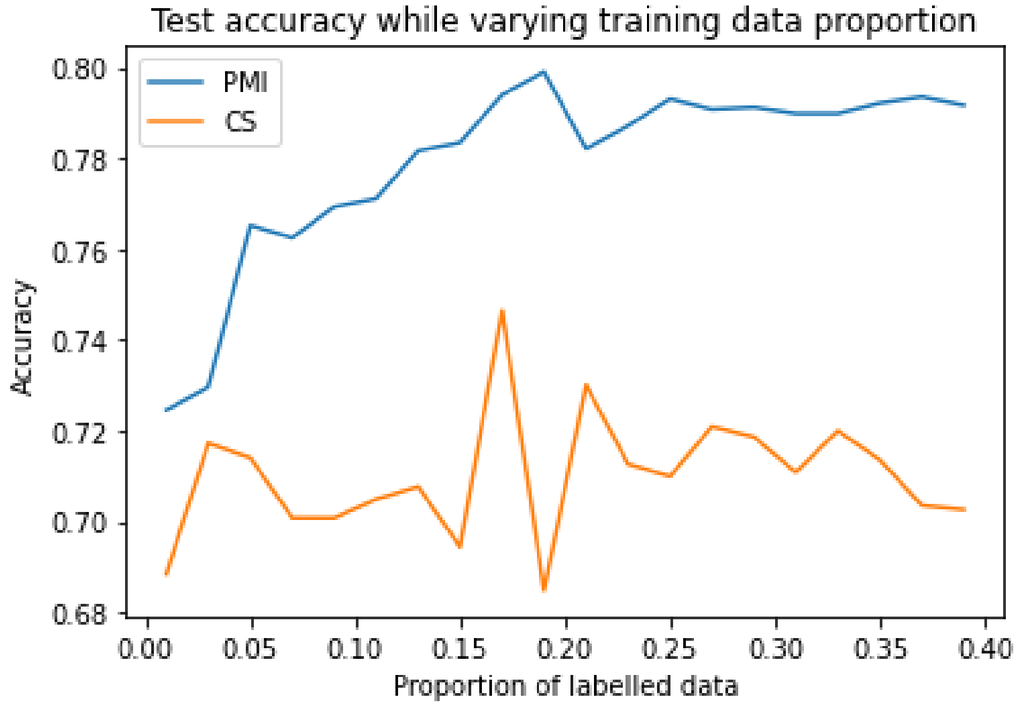
CS(mot i, mot j)

$$\mathrm{similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

Avantage : métrique indépendante de la taille du corpus

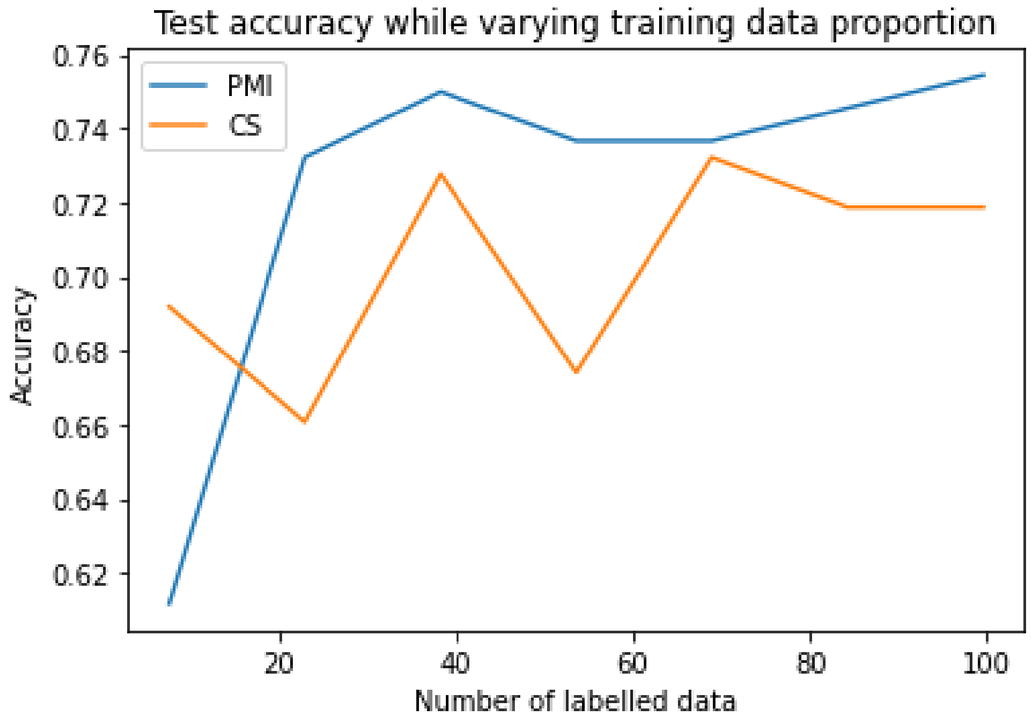
Résultats sur R8

PMI vs Cosine similarity

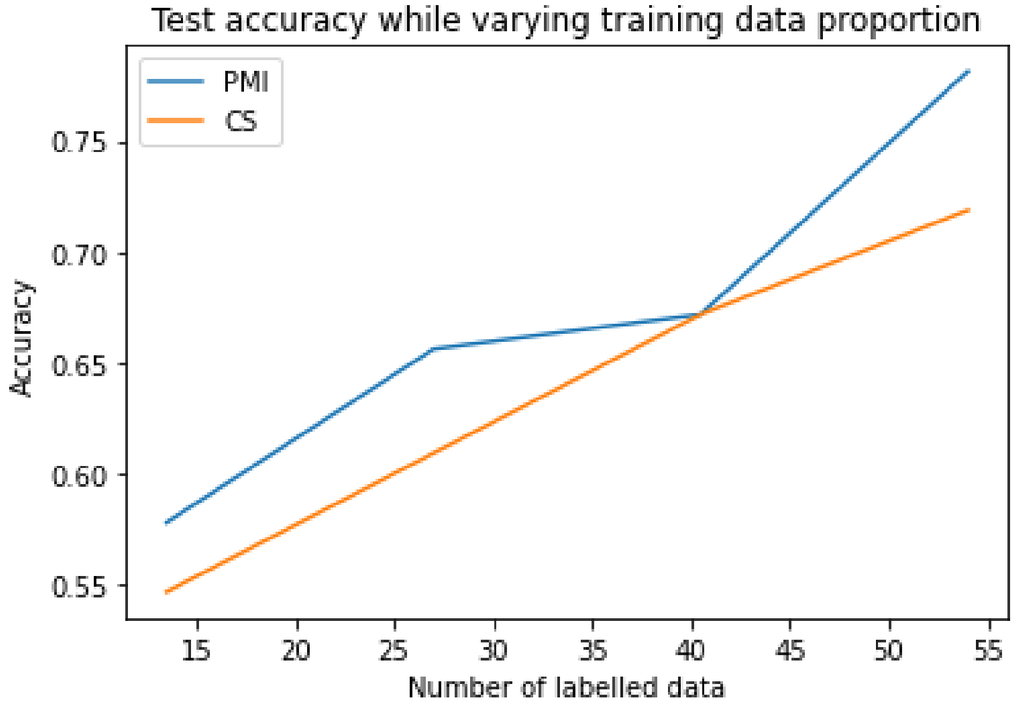


Résultats sur un echantillon de R8

Sur un dataset de 800 documents (10% de R8)



Sur un dataset de 200 documents (2,5% de R8)



Merci