TP07 Bases de l'IA : Traitement automatique du langage naturel

Partie 4. Classification des textes : analyse de sentiments (suite)

Dans ce TP, nous allons continuer à explorer quelques modèles pour l'analyse de sentiments (classement de textes). On utilisera toujours le même dataset d'analyse de sentiments dans le contexte de la finance (`https://www.kaggle.com/datasets/sbhatti/financial-sentiment-analysis`), et les mêmes librairies de la semaine dernière (**pandas**, **scikit-learn** et **nltk**).[1].

1. Entrainement

- Créer un modèle de type **MLPClassifier** avec une couche cachée et un maximum d'itérations de 50.
- Entrainer le modèle sur les représentations vectorielles obtenues comme résultat de l'étape 1 de la semaine dernière.
- Prédire les classes en appliquant le modèle sur les représentations d'entrainement.
- Afficher le rapport de classification.

2. Test

- Prédire les classes en appliquant le modèle sur les représentations vectorielles de test.
- En utilisant **timeit**, calculer le temps de prédiction.
- Afficher le rapport de classification.
- Comparer les résultats obtenus avec ce modèle aux resultats obtenus avec SVM : qu'observez-vous ?

Partie 5. Améliorations possibles

Nous voulons maintenant améliorer ce modèle, qui a quelques inconvénients (rien n'est parfait). Dans cette partie du TP, vous devez étudier quelques limites de ce modèle et les améliorer. Il faut au moins améliorer une chose sans détériorer le reste (une petite détérioration est acceptable).

- taille d'encodage : un modèle avec moins de paramètres est préférable
- Précision (micro-avg)
- Rappel (micro-avg)
- Temps de test : utile lorsqu'on veut utiliser ce modèle avec une application qui doit être rapide comme moteur de recherche.

Ressources utilisées

- Vous pouvez changer d'algorithme d'apprentissage : utiliser les autres algorithmes de scikit-learn
- Vous pouvez paramétrer le MLP existant : ajouter des couches, changer la fonction d'activation, etc.
- Vous pouvez utiliser d'autres méthodes de vectorisation (quelques modèles entrainés sont fournis avec ce TP)

Voici une description des modèles fournis où :

- **Modèle** : le nom du modèle
- **Algorithme** : l'algorithme utilisé pour entrainer le modèle
- **vecteur** : la taille du vecteur de représentation
- **fenêtre** : la fenêtre utilisée pour calculer le contexte d'un mot
- **phrases** : est-ce que le modèle peut représenter les phrases

[1] Remerciements : ARIES Abdelkrime

- **mots** : est-ce que le modèle peut représenter les mots

Modèle	Algorithme	vecteur	fenêtre	phrases	mots
gensim_lsa_100	LSA	100	/	oui	oui
gensim_lsa_50	LSA	100	/	oui	oui
gensim_lsa_10	LSA	100	/	oui	oui
gensim_word2vec_100_w2	Word2Vec-CBOW	100	2	non	oui
gensim_word2vec_100_w4	Word2Vec-CBOW	100	4	non	oui
gensim_word2vec_50_w2	Word2Vec-CBOW	50	2	non	oui
gensim_word2vec_50_w4	Word2Vec-CBOW	50	4	non	oui
gensim_word2vec_10_w2	Word2Vec-CBOW	10	2	non	oui
gensim_word2vec_10_w4	Word2Vec-CBOW	10	4	non	oui
gensim_word2vec_100sg_w2	Word2Vec-Skip-gram	100	2	non	oui
gensim_word2vec_100sg_w4	Word2Vec-Skip-gram	100	4	non	oui
gensim_word2vec_50sg_w2	Word2Vec-Skip-gram	50	2	non	oui
gensim_word2vec_50sg_w4	Word2Vec-Skip-gram	50	4	non	oui
gensim_word2vec_10sg_w2	Word2Vec-Skip-gram	10	2	non	oui
gensim_word2vec_10sg_w4	Word2Vec-Skip-gram	10	4	non	oui

Comparer les résultats obtenus avec ce modèle aux resultats obtenus avec SVM : qu'observez-vous ?