Comparaison d'approches pour l'analyse de sentiments

Présentation

L'objectif est de classifier des textes selon le sentiment exprimé. Le jeu d'entraînement utilisé contient des données issues de Twitter, associées à un label binaire (sentiment positif ou négatif).

Pour être utilisées par les modèles, les données textuelles sont prétraitées, de manière à ne conserver que des mots en minuscule, dénués de ponctuation et d'accentuation. On supprime également des identifications (@<username>) et les liens.

Pour limiter la taille du vocabulaire, on ne conserve que les mots présents dans un dictionnaire de référence. Ici on utilisera celui de l'embedding GLOVE entraîné par des chercheurs de Stanford sur des données très similaires, également issues de Twitter. Cette approche présente deux avantages. D'abord, nous auront la garantie que tous les mots de notre corpus possèderont une représentation lorsque nous seront amenés à utiliser cet embedding (ce qui présente également une robustesse face aux orthographes multiples, car elles possèdent des représentations similaires). Ensuite, par rapport à une approche plus conventionnelle consistant à retirer les mots trop ou trop peu fréquents de manière arbitraire, on garde parmi eux ceux qui sont les plus susceptibles d'être porteurs de sens.

Le vocabulaire résultant est d'un peu moins de 200 000 mots, trois fois moins que le nombre de mots uniques du corpus original (incluant une proportion considérable de mots mal orthographiés et d'onomatopées diverses).

Pour réduire la dimension de notre vocabulaire, on utilisera deux prétraitements additionnels : le stemming (ou radicalisation) et la lemmatisation. On comparera les résultats obtenus avec et sans ces traitements (les données originales seront désignées comme « vanilla »).

Le stemming consiste à ne garder que le radical des mots (par exemple supprimer les terminaisons en « -ed », « -ly » ou « -ing » des mots Anglais). Les algorithmes les plus performants aujourd'hui apprennent des règles plus complexes à partir de jeux d'entraînement labelisés (entrainement supervisé).

La lemmatisation consiste à ne garder que la forme canonique des mots. Les modèles modernes apprennent en cherchant les mots au sens similaire. En anglais, « walking » deviendra « walk », comme avec le stemming. Mais en plus, « better » deviendra « good ».

Ces deux méthodes permettent de réduire la taille de notre dictionnaire entre 10% et 15%.

Pour évaluer l'ensemble de nos modèles, la métrique utilisée sera l'accuracy (ou exactitude). Cette métrique peut être biaisée si les données ne sont pas balancées (car les modèles tendront à prédire la classe majoritaire). Ce n'est pas un problème ici car la distribution de la variable cible est balancée.

Les hyperparamètres des modèles ont été optimisés par grid search (c'est-à-dire en testant différentes combinaisons de paramètres).

Modèles simples : Random Forest

Les random forest sont des modèles d'apprentissage automatique qui consistent à entraîner de multiples arbres de décision à partir de sous-ensembles des données (pour améliorer les performances et éviter les problèmes liés au surapprentissage).

Les données sont représentées en bag-of-words, c'est-à-dire transformées en vecteurs de dimension égale à la taille du vocabulaire. Dans notre cas, la dimension des données est impraticable, que soit en termes de mémoire ou de temps de calcul (ce qui aurait été le cas avec n'importe quel modèle de régression simple). Pour pouvoir s'en servir comme modèle de référence, on utilise un sous-ensemble des données d'origine, basé sur un vocabulaire contenant les quelques milliers de mots les plus fréquents.

Modèles avancés : Deep Learning

Le Deep Learning (ou réseaux de neurones profonds) est un ensemble de méthodes d'apprentissage automatique modernes. Il consiste à construire des couches de neurones pour traiter des données. Chaque neurone d'une couche est connecté à un certain nombre de neurones de la couche précédente et de la couche suivante. Un neurone peut réaliser un traitement paramétrable, auquel cas ce paramètre sera un poids appris lors de l'entraînement. La méthode d'entraînement la plus courante (que nous utiliseront ici) est la rétropropagation. Elle consiste à apprendre les poids en partant des sorties.

La fonction de perte (la fonction que l'on cherche à minimiser lors de l'apprentissage) utilisée pour l'ensemble des modèles est l'entropie croisée binaire. Elle exprime la divergence moyenne entre les prédictions du modèle et la variable cible.

L'optimiseur est l'algorithme qui sert à minimiser la fonction de perte. Le plus connu est la descente de gradient stochastique (qui étudie itérativement la répercussion sur la fonction de perte de petits changements sur les poids antérieurs). Pour nos modèles nous utilisons Adam, un algorithme qui estime les moments (c'est-à-dire l'amplitude de ces répercussions) d'ordre 1 et 2. Pour les modèles avec embeddings, nous utiliserons Adamax, une variante d'Adam efficace avec les embeddings.

Les premiers modèles sont des réseaux de neurones simples, composés de deux couches denses. Comme pour les random forest, les données en entrée sont les bag-of-words, avec la même problématique de dimensionnalité.

Ensuite deux modèles avec embedding (ou plongement de mots) ont été entraînés. Un embedding est une correspondance entre un mot et sa représentation dans un espace vectoriel de dimension finie. Deux embeddings pré-entraînés (par apprentissage non supervisé) sont utilisés ici.

L'entraînement de l'embedding GLOVE, présenté en début d'article, consiste à étudier les fréquences d'apparition des paires de mots dans les documents pour en construire une représentation sémantique.

L'autre embedding utilisé est word2vec, créé par Google. Il consiste en un modèle prédisant, au choix, un mot en fonction de ses voisins (CBOW) ou les voisins d'un mot (Skip-gram). Dans les deux cas, l'embedding est la dernière couche précédant la couche de prédiction. Pour une comparaison

équitable, le prétraitement des données a été réalisé de nouveau, en utilisant cette fois comme dictionnaire de référence celui du modèle entraîné par Google.

Le premier modèle avec embedding est un réseau de neurones convolutif (CNN). Une couche de convolution applique des filtres (dont les paramètres sont appris par le modèle) sur des fenêtres de mots de taille prédéfinie.

Le second modèle est un réseau avec une couche Long Short Term Memory (LSTM). Il s'agit d'une couche récurrente (qui traite les données de manière séquentielle), et qui possède une mémoire. A chaque élément d'une séquence, elle peut mettre à jour le contenu de sa mémoire, oublier de l'information, ou influer sur la sortie du neurone. Les paramètres de son comportement sont des poids appris par le modèle.

API AZURE Cognitive Services Text Analytics – Sentiment Analysis

Cette API proposée par Azure permet de classifier des documents selon trois classes (négatif, neutre ou positif). Le service se met en place simplement depuis l'interface graphique du portail Azure. En pratique ce service s'avère décevant car une grande portion du jeu d'entraînement est prédite comme probablement neutre par l'API, rendant les prédictions binaires associées de mauvaise qualité.

Résultats

On note que les modèles random forest et RN simples sont entraînés sur environ 10% du dataset, par conséquent leurs temps de calcul associés sont grandement sous évalués.

Modèle	Accuracy	Temps de calcul
Random forest (vanilla)	0.743	1h18 min
Random forest (stemming)	0.745	52 min
Random forest (lemmatisation)	0.744	1h10 min
RN simple (vanilla)	0.768	1h12 min
RN simple (stemming)	0.764	56 min
RN simple (lemmatisation)	0.768	1h07 min
GLOVE CNN	0.797	3.5 min
GLOVE LSTM	0.804	5 min
Word2vec LSTM	0.797	10 min
API AZURE	0.721	-