Word2vec

https://code.google.com/archive/p/word2vec/

Introduction

Cet outil fournit une implémentation efficace des architectures de sac de mots continus et de skip-gram pour le calcul des représentations vectorielles des mots. Ces représentations peuvent ensuite être utilisées dans de nombreuses applications de traitement du langage naturel et pour des recherches ultérieures.

Comment ça marche

L'outil word2vec prend un corpus de texte en entrée et produit les vecteurs de mots en sortie. Il construit d'abord un vocabulaire à partir des données du texte d'apprentissage, puis apprend la représentation vectorielle des mots. Le fichier de vecteurs de mots résultant peut être utilisé comme fonctionnalités ou caractéristiques dans de nombreuses applications de traitement du langage naturel et d'apprentissage automatique.

Un moyen simple d'étudier les représentations apprises est de trouver les mots les plus proches d'un mot spécifié par l'utilisateur. L'outil de distance sert à cet effet. Par exemple, si vous saisissez 'france', distance affichera les mots les plus similaires et leurs distances par rapport à 'france', ce qui devrait ressembler à :

Word Cosine distance

spain	0.678515
belgium	0.665923
netherlands	0.652428
italy	0.633130
switzerland	0.622323
luxembourg	0.610033
portugal	0.577154
russia	0.571507
germany	0.563291
catalonia	0.534176

Il existe deux principaux algorithmes d'apprentissage dans **word2vec** : le bag-of-words continu et le skip-gram continu. L'option **-cbow** permet à l'utilisateur de choisir l'un de ces algorithmes d'apprentissage. Les deux algorithmes apprennent la représentation d'un mot qui est utile pour la prédiction d'autres mots dans la phrase. Ces algorithmes sont décrits en détail dans [1,2].

Propriétés intéressantes des vecteurs de mots

Il a été récemment démontré que les mots vecteurs capturent de nombreuses régularités linguistiques, par exemple les opérations vectorielles vecteur('Paris') - vecteur('France') + vecteur('Italie') aboutissent à un vecteur très proche de vecteur('Rome '), et vector('king') - vector('man') + vector('woman') est proche de vector('reine') [3, 1]. Vous pouvez essayer une démo simple en exécutant demo-analogy.sh.

Pour observer de fortes régularités dans l'espace vectoriel des mots, il est nécessaire d'entraîner les modèles sur un grand ensemble de données, avec une dimensionnalité vectorielle suffisante comme indiqué dans [1]. En utilisant l'outil word2vec, il est possible d'entraîner des modèles sur d'énormes ensembles de données (jusqu'à des centaines de milliards de mots).

Des mots aux phrases et au-delà

Dans certaines applications, il est utile d'avoir une représentation vectorielle de morceaux de texte plus volumineux. Par exemple, il est souhaitable de n'avoir qu'un seul vecteur pour représenter 'san francisco'. Ceci peut être réalisé en pré-traitant l'ensemble de données d'apprentissage pour former les phrases à l'aide de l'outil word2phrase, comme le montre l'exemple de script ./demo-phrases.sh. L'exemple de sortie avec les jetons les plus proches de 'san_francisco' ressemble à :

Word Cosine distance

los_angeles	0.666175
golden_gate	0.571522
oakland	0.557521
california	0.554623
san_diego	0.534939
pasadena	0.519115

seattle	0.512098
taiko	0.507570
houston	0.499762
chicago_illinois	0.491598

La linéarité des opérations vectorielles semble faiblement valable également pour l'addition de plusieurs vecteurs, il est donc possible d'ajouter plusieurs vecteurs de mots ou de phrases pour former une représentation de phrases courtes

Comment mesurer la qualité des vecteurs de mots

Plusieurs facteurs influencent la qualité des vecteurs de mots : * la quantité et la qualité des données d'apprentissage, la taille des vecteurs, l'algorithme d'apprentissage.

La qualité des vecteurs est cruciale pour toute application. Cependant, l'exploration de différents réglages d'hyper-paramètres pour des tâches complexes peut prendre trop de temps. Ainsi, nous avons conçu des ensembles de tests simples qui peuvent être utilisés pour évaluer rapidement la qualité du vecteur de mot.

Pour l'ensemble de test de relation de mots décrit dans [1], voir ./demo-word-accuracy.sh, pour l'ensemble de test de relation d'expression décrit dans [2], voir ./demo-phrase-accuracy.sh. Notez que la précision dépend fortement de la quantité de données d'entraînement ; nos meilleurs résultats pour les deux ensembles de tests sont supérieurs à 70 % de précision avec une couverture proche de 100 %.

Word clustering

Les vecteurs de mots peuvent également être utilisés pour dériver des classes de mots à partir d'énormes ensembles de données. Ceci est réalisé en effectuant un clusturing K-means sur les vecteurs de mots. Le script qui en fait la démonstration est ./demo-classes.sh. Le résultat est un fichier de vocabulaire avec des mots et leurs ID de classe correspondants, tels que :

carnivores 234 carnivorous 234 cetaceans 234 cormorant 234 coyotes 234 crocodile 234 crocodiles 234 crustaceans 234 cultivated 234 danios 234 . . . acceptance 412 argue 412 argues 412 arguing 412 argument 412 arguments 412 belief 412 believe 412 challenge 412 claim 412

Performances

La vitesse d'apprentissage peut être considérablement améliorée en utilisant l'apprentissage parallèle sur une machine à plusieurs processeurs (utilisez le commutateur '-threads N'). Le choix de l'hyperparamètre est crucial pour les performances (vitesse et précision), mais varie selon les applications. Les principaux choix à faire sont les suivants :

- l'architecture : skip-gram (plus lent, meilleur pour les mots peu fréquents) vs CBOW (rapide)
- l'algorithme d'apprentissage : softmax hiérarchique (meilleur pour les mots peu fréquents) vs échantillonnage négatif (meilleur pour les mots fréquents, meilleur avec des vecteurs de faible dimension)
- sous-échantillonnage des mots fréquents : peut améliorer à la fois la précision et la vitesse pour les grands ensembles de données (les valeurs utiles sont comprises entre 1e-3 et 1e-5)
- la dimensionnalité des vecteurs de mots : généralement plus est mieux, mais pas toujours
- taille du contexte (fenêtre) : pour le skip-gram, généralement autour de 10, pour le CBOW autour de 5.

Où obtenir les données d'entraînement

La qualité des vecteurs de mots augmente considérablement avec la quantité de données d'entraînement. À des fins de recherche, vous pouvez envisager d'utiliser des ensembles de données disponibles en ligne :

- Premier milliard de caractères de wikipedia (utilisez le script perl de prétraitement en bas de la page de Matt Mahoney)
- Dernier dump de Wikipédia. Utilisez le même script que ci-dessus pour obtenir un texte propre. Il devrait y avoir plus de 3 milliards de mots.
- Site WMT11 : données textuelles pour plusieurs langues (les phrases en double doivent être supprimées avant l'entraînement des modèles).
- Dataset from "One Billion Word Language Modeling Benchmark" Presque 1 milliard de mots, texte déjà prétraité.

- Corpus de la base de données Web de l'UMBC. Environ 3 milliards de mots, plus d'informations ici. Nécessite un traitement supplémentaire (principalement la tokenisation).
- Les données textuelles d'autres langues peuvent être obtenues sur statmt.org et dans le projet Polyglot.

Vecteurs de mots et de phrases pré-entraînés

Nous publions des vecteurs pré-entraînés formés sur une partie du jeu de données de Google News (environ 100 milliards de mots). Le modèle contient des vecteurs à 300 dimensions pour 3 millions de mots et de phrases. Les phrases ont été obtenues en utilisant une approche simple basée sur les données, décrite dans [2]. L'archive est disponible ici : GoogleNews-vectors-negative300.bin.gz.

An example output of ./distance GoogleNews-vectors-negative300.bin:

``` Enter word or sentence (EXIT to break): Chinese river

Word Cosine distance

| | |
|---|---|
| Yangtze_River | 0.667376 |
| Yangtze | 0.644091 |
| Qiantang_River | 0.632979 |

Yangtze_tributary 0.623527 Xiangjiang_River 0.615482 Huangpu_River 0.604726 Hanjiang_River 0.598110 Yangtze_river 0.597621 Hongze_Lake 0.594108 Yangtse 0.593442 ```

L'exemple ci-dessus calculera la moyenne des vecteurs des mots "chinois" et "rivière" et renverra les voisins les plus proches du vecteur résultant. D'autres exemples qui démontrent les résultats de l'addition de vecteurs sont présentés dans [2]. Notez que des vecteurs d'entités plus précis et désambiguïsés peuvent être trouvés dans l'ensemble de données suivant qui utilise le nommage Freebase.
```

Pre-trained entity vectors with Freebase naming

Nous proposons également plus de 1,4 million de vecteurs d'entités pré-entraînés avec le nommage de Freebase. Ceci est particulièrement utile pour les projets liés à l'extraction de connaissances.

- Entity vectors trained on 100B words from various news articles: freebase-vectors-skipgram1000.bin.gz
- Entity vectors trained on 100B words from various news articles, using the deprecated /en/ naming (more easily readable); the vectors are sorted by frequency: freebase-vectors-skipgram1000-en.bin.gz

Here is an example output of ./distance freebase-vectors-skipgram1000-en.bin:

``` Enter word or sentence (EXIT to break): /en/geoffrey_hinton

**Word Cosine distance**

/en/marvin_minsky 0.457204
/en/paul_corkum 0.443342

/en/william_richard_peltier 0.432396 /en/brenda_milner 0.430886 /en/john_charles_polanyi 0.419538 /en/leslie_valiant 0.416399 /en/hava_siegelmann 0.411895 /en/hans_moravec 0.406726 /en/david_rumelhart 0.405275 /en/godel_prize 0.405176 ```

**Final words**

Merci d'essayer cette boîte à outils, et n'oubliez pas de nous faire savoir quand vous obtiendrez des résultats étonnants ! Nous espérons que les représentations distribuées amélioreront de manière significative l'état de l'art en PNL.
```