Université de Montréal

IFT6285 (TALN) — Devoir4
Expression multimots

Contact :
Philippe Langlais
RALI/DIRO
Université de Montréal
+1 514 343 61 11 ext: 47494
felipe@iro.umontreal.ca
http://www.iro.umontreal.ca/~felipe/

dernière compilation : 1er octobre 2020 (11:29)

Contexte

L'un des enjeux du traitement des langues est la profusion d'expressions multimots dans une langue (Sag et al. 2002). Il peut s'agir d'expression contigües de mots comme *(se faire) passer un sapin* ou des collotations[1] distantes comme celles reliant certains noms à certains adjectifs (on utilisera par exemple en anglais *strong* pour qualifier le nom *tea*, mais *heavy* pour qualifier le nom *traffic*, ce qui est pour le moins arbitraire). Pour plus à ce sujet, lire (Smadja 1993). En fait, le problème est si important que des workshops dédiés sont consacrés à l'étude des expressions multimots depuis presque 20 ans (reconnaissance, traduction, description linguistique).

Les méthodes de plongements de mots (embeddings) comme `word2vec` (Mikolov et al. 2013) qui ont boulversé le domaine du traitement des langues ne sont pas exempts de ces problèmes (surtout les plongements dits statiques comme `word2vec`) et des algorithmes sont disponibles qui permettent de regrouper des multimots dans un texte afin d'en créer (par la suite) une représentation adéquate.

Dans ce devoir, vous allez vous familiariser avec l'API Phraser de gensim, une boîte à outil populaire en python offrant entre autre une implémentation de `word2vec`. Cette API détecte des séquences de mots dans des textes à l'aide de variantes de l'information mutuelle vue en cours. Cette API est loin d'être parfaite et le but du devoir 4 est de l'explorer.

Données

Vous avez accès au 1 billion word (1Bwc) corpus (Chelba et al. 2013) dont une copie est disponible au DIRO :

`/u/demorali/corpora/1g-word-lm-benchmark-r13output/`.

Alternativement, vous pouvez travailler sur 1bshort.tar.gz qui contient les 9 premières tranches de ce répertoire (153Mo).

1. Une collocation est un ensemble de mots dont l'apparition n'est pas fortuite.

À faire

1. vous devez écrire du code permettant de lire tout ou partie des lignes du 1Bwc et qui calcule à partir de ces phrases les bigrammes et les trigrammes les plus saillants (les mieux notés selon l'information mutuelle ou une variante).
2. Vous devez produire un rapport d'au plus 3 pages (format pdf, en français ou en anglais) qui contient les informations suivantes :
   - une courbe montrant le nombre de bigrammes/trigrammes saillants trouvés dans les phrases "digérées" par votre programme en fonction du nombre de ces phrases.
     Par exemple, en lisant 900 phrases du 1Bwc à l'aide de Phraser, [2] j'obtiens : 809 bigrammes dont les scores varient de 6018 à 10, et dont les bigrammes les plus saillants (score de 6018), sont en ordre décroissant : *del Potro*, *de facto*, *polar cities*, *Notre Dame*, *Cross Country*, *Ban Ki-moon*, etc. En digérant le double de phrases, j'obtient 1646 bigrams (score allant de 10813 à 10.0) dont les trois plus saillants sont : *LOS ANGELES*, *Osama bin*. etc.
   - les dix meilleurs bigrammes selon Phraser lorsque vous utilisez les options par défaut et que vous lisez les 2.7M phrases des 9 tranches de 1bshort.tar.gz. Si vous ne parvenez pas à faire tourner Phraser sur ce corpus, indiquez le corpus le plus gros que vous avez réussi à traiter, et/ou jouer aves les options pour réduire les temps de calcul (indiquez votre configuration).
     Pour information, sur mon ordinateur (iMac, 16Go mémoire, 3,6 GHz Intel Core i7) cela prend 20 minutes à Phraser (valeurs par défaut) de traiter les 2.7M de phrases.
   - indiquez comment évolue le temps en fonction du nombre de phrases traitées. (Vous n'êtes pas obligé de faire cela par pas de 1 phrase !).
   - produisez de 3 à 5 contextes [3] d'apparition des deux bigrammes et des deux trigrammes les mieux notés par Phraser lorsqu'il traite toutes les phrases de 1bshort.tar.gz (ou moins si vous n'y parvenez pas). Vous préciserez la configuration de Phraser utilisée et

2. `min_count=1 threshold=10`, fonction de score par défaut.
3. On définit ici un contexte comme les 30 caractères qui précèdent ou qui suivent le ngram d'intérêt.

indiquerez le nombre total d'occurrences des ces bigrammes et trigrammes.
Par exemple, Phraser utilisé par défaut sur toutes les phrases de 1bshort.tar.gz produit 8 occurrences du bigramme *Daft Punk* (qui n'est pas le bigramme de plus haut score) dont voici trois contextes :

regalia , with the French duo	[[Daft Punk]]	*providing backup ; then he did*
He sampled a	[[Daft Punk]]	*track called " Harder Better S*
To quote	[[Daft Punk]]	*- work it harder , make it bet*

— discutez les limites de Phraser et suggérez des améliorations, par exemple à la lumière des techniques vues en cours.

What if ?

Si vous ne parlez pas python, ce devoir peut s'avérer difficile (sinon c'est juste une expérimentation sur du code très similaire à celui que je vous ai pointé). Deux solutions : apprendre python (courbe d'apprentissage non négligeable, mais pas insurmontable), ou programmer vous même une variante qui calcule l'information mutuelle (par exemple) de deux mots qui se suivent dans un document. Si vous êtes dans ce cas, indiquez le clairement dans votre rapport et réaliser ce que vous pouvez faire de ce qui est demandé : l'idée d'un devoir est d'explorer un outil ou une technique sans y passer (sauf intérêt particulier) la semaine.

Remise

La remise est à faire sur Studium sous le libellé `devoir4`. Vous devez remettre votre code et votre rapport (format pdf, texte en anglais ou en français) dans une archive (gzip, tar, tar.gz) dont le nom est préfixé de `devoir4-name1` ou `devoir4-name1-name2` selon que vous remettez seul ou a deux, où `name1` et `name2` sont à remplacer par l'identité des personnes faisant la remise (`prénom_nom`). Donc si j'avais à remettre seul mon solutionnaire au devoir4, je le ferais sous le nom `devoir4-philippe_langlais.tar.gz`. Assurez vous que le nom des personnes impliquées dans le devoir soit indiqué

sur tous les documents remis (code et rapport). Le devoir est à remettre en groupe d'au plus deux personnes au plus tard vendredi 9 octobre à 23h59. **Note : Aucune donnée ni modèle n'est demandé : juste le rapport et le code.**

Chelba, Ciprian et al. (2013). "One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling". In : *CoRR* abs/1312.3005. URL : http://arxiv.org/abs/1312.3005.

Mikolov, Tomas et al. (2013). "Efficient Estimation of Word Representations in Vector Space". In : *CoRR* abs/1301.3781. URL : http://arxiv.org/abs/1301.3781.

Sag, Ivan A. et al. (2002). "Multiword Expressions : A Pain in the Neck for NLP". In : *Computational Linguistics and Intelligent Text Processing*. Sous la direction d'Alexander Gelbukh. Berlin, Heidelberg : Springer Berlin Heidelberg, pages 1–15. ISBN : 978-3-540-45715-2.

Smadja, Frank (1993). "Retrieving Collocations from Text : Xtract". In : *Computational Linguistics* 19(1), pages 144–177.