WSdsm.R
Mode d'emploi simplifié

Les fonctions contenues dans ce script sont destinées à faciliter et à accélérer l'utilisation de la bibliothèque R wordspace (*wordspace* package, http://download.r-forge.r-project.org/src/contrib/wordspace_0.1-16.tar.gz) dans le cadre de recherches de sémantique historique, appliquées à des corpus enregistrés sous CWB. Plus particulièrement des corpus latins, prétraités avec les outils OMNIA (http://www.glossaria.eu/treetagger/).
Les principes qui sont au fondement de cette bibliothèque sont bien exposés par Peter D. Turney et Patrick Pantel, « From Frequency to Meaning : Vector Space Models of Semantics, *Journal of Artificial Intelligence Research*, 37-2010, pp. 141-188 (https://www.jair.org/media/2934/live-2934-4846-jair.pdf).
On utilise aussi deux autres bibliothèques, nécessaires pour les affichages graphiques : *ade4* et *circular* (se trouvent sur le CRAN).

On peut envisager l'utilisation de la bibliothèque 'wordspace' à des fins très diverses ; ici, il s'agit essentiellement de sémantique historique : on cherche à mieux cerner le sens de mots contenus dans des textes anciens, mots que l'on croit comprendre, mais dont la 'traduction' génère constamment illusions et contresens.
Cette perspective (statistique lexicale appliquée à la recherche du sens) se fonde avant tout sur les proximités entre vocables : le 'profil' d'un lemme est représenté par l'ensemble de ses cooccurrents 'significatifs'. La première méthode, la plus simple, que l'on peut appeler celle des 'cooccurrents simples' (ou 'cooccurrents de premier ordre'), consiste à rechercher et à calculer ces cooccurrents en utilisant toutes les occurrences du lemme dans le corpus considéré. La seconde méthode, dont il est question ici, est celle que l'on peut appeler des 'cooccurrents du second ordre', ou 'cooccurrents généralisés', elle est également désignée comme 'analyse de l'espace de mots' (wordspace WS) ou 'modèle de sémantique distributionnelle' (distributional semantic model, DSM). Elle consiste à calculer tous les cooccurrents de tous les mots d'un corpus, et à rechercher, pour un mot (lemme) donné, les mots présentant les 'profils' les plus semblables (i.e. ayant l'ensemble de cooccurrents le plus voisin).
En théorie, la première méthode permet d'observer des relations syntagmatiques, la seconde des relations paradigmatiques. Par exemple, pour le lemme 'chien',
syntagmatiques : laisse, niche, berger, bruyant, aboyer, mordre
paradigmatiques : caniche, épagneul, chat, carnivore.
En pratique, la distinction n'est pas toujours facile à établir, et les deux méthodes, appliquées au même lemme, font souvent apparaître, en partie, les mêmes résultats. Il est rationnel de considérer les deux méthodes comme complémentaires.

La recherche des cooccurrents de second ordre se heurte à deux types de difficultés. Les premières sont matérielles et techniques : dès que le corpus atteint une certaine taille (nécessaire en fait pour obtenir des effectifs utilisables), la quantité de calculs et la taille des objets en RAM atteignent des proportions qui peuvent rendre les opérations impossibles (que l'on songe à l'idée d'une matrice de 40000 lignes sur 40000 colonnes : 1600000000, un milliard et demi d'emplacements...). Des outils récents (le programme de calcul 'cwb-scan-corpus' et la bibliothèque 'wordspace') permettent de résoudre ces difficultés, à condition d'être employés à bon escient. Mais d'autres difficultés sont plus retorses : la construction d'un 'objet dsm' implique le choix de toute une série de paramètres, car il existe une grande variété de possibilités pour établir le tableau des cooccurrents généralisés. Et, comme le dit parfaitement Stefan Evert, « Despite recent efforts to carry out systematic evaluation studies, the precise effect of these parameters and their relevance for different application settings are still poorly understood ». S'agissant de recherche de sémantique historique appliquée au latin, on en est au stade des expériences préliminaires : quelques tendances semblent se dégager, mais elles restent à explorer, à préciser, à valider.

MODUS OPERANDI

L'objectif est d'aboutir à une identification des éléments du 'champ sémantique' d'un lemme, avec une possible représentation sous forme graphique. Trois étapes : 1. balayer le corpus pour récupérer tous les cooccurrents souhaités, et tabuler, 2. introduire ces data dans le workspace de R et les transformer en un objet formaté utilisable (de class 'dsm'), 3. exploiter cet objet dsm pour rechercher les plus proches voisins du lemme considéré, et représenter graphiquement la matrice de distances ainsi obtenue.
L'opération sur le corpus entier peut-être appliquée à des tranches du corpus, ce qui permet d'obtenir de précieuses indications sur l'évolution du champ d'un lemme.

corpus2scan(corp, reg="", dis=3, posA="QLF|SUB|VBE", posB="QLF|SUB|VBE", objetA="lemma", objetB="lemma", D=0, F=Inf, attr="", val="", destination, flag=T)

1ère étape : balayage à la recherche des cooccurrents.
Fonction destinée à créer un fichier totalisant tous les cooccurrents (éventuellement définis par leurs pos) de tous les lemmes (éventuellement définis par leur pos) : 1. appel à une fonction extérieure très rapide (cwb-scan-corpus) [merci Krzysztof], 2. utilisation de stockages provisoires sur disque dur ; le résultat (par simple concaténation) est enregistré sur le disque dur. Pas d'autre sortie.
NB. le résultat est généralement volumineux, voire très volumineux, prévoir suffisamment de place dans le répertoire de destination.

En entrée :
corp : nom d'un corpus enregistré sous CWB
reg : adresse du registry du corpus (permet d'utiliser des corpus dans différents répertoires)
dis : largeur de la fenêtre (d'un côté) pour le calcul des cooccurrents (fenêtre totale = dis*2)
posA : restriction pour les pivots (les noms des pos doivent être séparés par une barre verticale, c'est une regexp)
posB : restriction pour les cooccurrents (id.)
objetA : choix : lemma ou word (ou autre définition, selon le corpus)
objetB : id.
D /F : sélection d'un sous-corpus par bornes de début et de fin
attr : nom d'un attribut pour sélection d'un sous-corpus
val : valeur de l'attribut
destination : nom complet du fichier d'enregistrement du résultat
flag : afficher ou non le temps d'exécution

En sortie :
↠ un fichier enregistré sur le disque dur
5 colonnes séparées par des tabulations
* eff (effectif cooccurrent)
* lemma1 (pivot)
* pos1 (pos du pivot)
* lemma2 (cooccurrent)
* pos2 (pos du cooccurrent)

ce fichier est facile à récupérer, si nécessaire, avec la fonction read.table()

scan2dsm(scan, efftt="", seuil=9, coeff="simple-ll", nproj="", flag=T)

2ème étape : création d'un objet dsm dans le workspace de R.
Lecture d'un fichier issu de la fonction précédente, construction d'un objet dsm avec choix de paramètres.

En entrée :
scan : nom exact du fichier (sur disque) produit par `corpus2scan()`
efftt : effectif total du corpus (pas indispensable)
seuil : seuil minimal de cases (en ligne comme en colonne) différentes de zéro (= correspondant à une coocurr.)
coeff : type de filtre à appliquer aux effectifs de cooccurr. bruts pour obtenir un indice de significativité
nproj : nombre de dimensions à conserver en cas de réduction du nombre de dimensions
flag : afficher ou non le temps d'exécution

En sortie :
↠ un objet de class 'DSM' (essentiellement : une matrice de cooccurrences, en format S4)

corpus2dsm(corp, reg="", dis=3, posA="QLF|SUB|VBE", posB="QLF|SUB|VBE", objetA="lemma", objetB="lemma", D=0, F=Inf, attr="", val="", destination, scan, efftt="", seuil=9, coeff="simple-ll", nproj="", flag=T)

Fonction qui regroupe les deux précédentes : les paramètres sont la somme des paramètres des deux fonctions précédentes. Permet de passer directement d'un corpus à un objet 'dsm' (avec enregistrement intermédiaire d'un fichier de scan sur le disque), on gagne ainsi du temps et on limite le risque d'erreur.

En entrée :
corp : nom d'un corpus enregistré sous CWB
reg : adresse du registry du corpus (permet d'utiliser des corpus dans différents répertoires)
dis : largeur de la fenêtre (d'un côté) pour le calcul des cooccurrents (fenêtre totale = dis*2)
posA : restriction pour les pivots (les noms des pos doivent être séparés par une barre verticale, c'est une regexp)
posB : restriction pour les cooccurrents (id.)
objetA : choix : lemma ou word (ou autre définition, selon le corpus)
objetB : id.
D /F : sélection d'un sous-corpus par bornes de début et de fin
attr : nom d'un attribut pour sélection d'un sous-corpus
val : valeur de l'attribut
destination : nom complet du fichier d'enregistrement du résultat
efftt : effectif total du corpus (pas indispensable)
seuil : seuil minimal de cases (en ligne comme en colonne) différentes de zéro (= correspondant à une coocurr.)
coeff : type de filtre à appliquer aux effectifs de cooccurr. bruts pour obtenir un indice de significativité
nproj : nombre de dimensions à conserver en cas de réduction du nombre de dimensions
flag : afficher ou non le temps d'exécution

En sortie :
↬ un fichier enregistré sur le disque dur
5 colonnes séparées par des tabulations
* eff (effectif cooccurrent)
* lemma1 (pivot)
* pos1 (pos du pivot)
* lemma2 (cooccurrent)
* pos2 (pos du cooccurrent)

↬ un objet de class 'DSM' (essentiellement : une matrice de cooccurrences, en format S4)

dsm2af(dsm, lm, nppv, cex=.9, decal=T)

3ème étape : étant donné un lemme, extraction des `nppv` plus proches voisins ; on utilise par défaut la distance du cosinus ; les coefficients sont rangés dans une matrice de similarités ensuite traitée par AFC (analyse factorielle des correspondances) transcrite en graphique (axes 1 et 2).

En entrée :
dsm : nom de l'objet 'dsm' à utiliser
lm : lemme pivot
nppv : nombre de plus proches voisins à rechercher
cex : taille des caractères sur le graphique
decal : supprimer les recouvrements d'étiquettes, ou non

En sortie :
↬ un objet de type 'list' à 3 éléments :
* matrice de similarités
* vecteur indiquant les ppv (avec coefficient de similarité)
* coordonnées des points (une seule liste, la matrice étant symétrique)

↬ un graphique factoriel fournissant une image du champ sémantique

Le deuxième groupe de fonctions est tout à fait parallèle au premier ; la différence tient au fait que l'on applique ici les procédures à des tranches de corpus, de manière à examiner des évolutions. Dans les noms des fonctions, on a seulement ajouté un ***m*** *(multiple) pour indiquer que l'on travaille ici sur des tranches.*

corpus2scanm(corp, reg="", dis=3, posA="QLF|SUB|VBE", posB="QLF|SUB|VBE", objetA="lemma", objetB="lemma", attr="", val="", destination, efftt, trnch, flag=T)

1ère étape : balayage à la recherche des cooccurrents.
Fonction destinée à créer un fichier totalisant tous les cooccurrents (éventuellement définis par leurs pos) de tous les lemmes (éventuellement définis par leur pos) : 1. appel à une fonction extérieure très rapide (cwb-scan-corpus) [merci Krzysztof], 2. utilisation de stockages provisoires sur disque dur ; les résultats (par simple concaténation), un pour chaque tranche, sont enregistrés sur le disque dur. Pas d'autre sortie.
NB. le résultat est généralement volumineux, voire très volumineux, prévoir suffisamment de place dans le répertoire de destination.

En entrée :
corp : nom d'un corpus enregistré sous CWB
reg : adresse du registry du corpus (permet d'utiliser des corpus dans différents répertoires)
dis : largeur de la fenêtre (d'un côté) pour le calcul des cooccurrents (fenêtre totale = dis*2)
posA : restriction pour les pivots (les noms des pos doivent être séparés par une barre verticale, c'est une regexp)
posB : restriction pour les cooccurrents (id.)
objetA : choix : lemma ou word (ou autre définition, selon le corpus)
objetB : id.
attr : nom d'un attribut pour sélection d'un sous-corpus
val : valeur de l'attribut
destination : nom complet du radical des noms de fichiers d'enregistrement du résultat
efftt : effectif total du corpus (obligatoire)
trnch : nombre de tranches
flag : afficher ou non le temps d'exécution

En sortie :
`trnch` fichiers enregistrés sur le disque dur
5 colonnes séparées par des tabulations
* eff (effectif cooccurrent)
* lemma1 (pivot)
* pos1 (pos du pivot)
* lemma2 (cooccurrent)
* pos2 (pos du cooccurrent)

ces fichiers sont faciles à récupérer, si nécessaire, avec la fonction read.table()

scan2dsmm(scan, seuil=9, coeff="simple-ll", nproj="", trnch, flag=T)

2ème étape : création d'un objet multi-dsm dans le workspace de R.
Lecture des fichiers issus de la fonction précédente, construction d'un objet de type 'list', regroupant `trnch` dsm, avec choix de paramètres.

En entrée :
scan : nom exact du fichier (sur disque) produit par `corpus2scanm()`
seuil : seuil minimal de cases (en ligne comme en colonne) différentes de zéro (= correspondant à une coocurr.)
coeff : type de filtre à appliquer aux effectifs de cooccurr. bruts pour obtenir un indice de significativité
nproj : nombre de dimensions à conserver en cas de réduction du nombre de dimensions
efftt : effectif total du corpus (pas indispensable)
flag : afficher ou non le temps d'exécution

En sortie :
↬ un objet 'list' regroupant `trnch` objets de class 'DSM' (essentiellement : matrices de cooccurrences, en format S4)

corpus2dsmm(corp, reg="", dis=3, posA="QLF|SUB|VBE", posB="QLF|SUB|VBE", objetA="lemma", objetB="lemma", trnch, attr="", val="", destination, scan, efftt, seuil=9, coeff="simple-ll", nproj="")

Fonction qui regroupe les deux précédentes : les paramètres sont la somme des paramètres des deux fonctions précédentes. Permet de passer directement d'un corpus à un objet multi-'dsm' (avec enregistrement intermédiaire des fichiers de scan sur le disque), on gagne ainsi du temps et on limite le risque d'erreur.

En entrée :
corp : nom d'un corpus enregistré sous CWB
reg : adresse du registry du corpus (permet d'utiliser des corpus dans différents répertoires)
dis : largeur de la fenêtre (d'un côté) pour le calcul des cooccurrents (fenêtre totale = dis*2)
posA : restriction pour les pivots (les noms des pos doivent être séparés par une barre verticale, c'est une regexp)
posB : restriction pour les cooccurrents (id.)
objetA : choix : lemma ou word (ou autre définition, selon le corpus)
objetB : id.
trnch : nombre de tranches
attr : nom d'un attribut pour sélection d'un sous-corpus
val : valeur de l'attribut
destination : nom complet du radical des fichiers d'enregistrement des résultats
efftt : effectif total du corpus (indispensable)
seuil : seuil minimal de cases (en ligne comme en colonne) différentes de zéro (= correspondant à une coocurr.)
coeff : type de filtre à appliquer aux effectifs de cooccurr. bruts pour obtenir un indice de significativité
nproj : nombre de dimensions à conserver en cas de réduction du nombre de dimensions

En sortie :
↬ `trnch` fichiers enregistrés sur le disque dur
5 colonnes séparées par des tabulations
* eff (effectif cooccurrent)
* lemma1 (pivot)
* pos1 (pos du pivot)
* lemma2 (cooccurrent)
* pos2 (pos du cooccurrent)

↬ un objet 'list' regroupant `trnch` objets de class 'DSM' (essentiellement : matrices de cooccurrences, en format S4)

dsmm2af(dsm, lm, nppv, xax=1, yax=1, cex=.9, decal=T)

3ème étape : étant donné un lemme, extraction des `nppv` plus proches voisins dans chaque tranche ; on utilise par défaut la distance du cosinus. Ici, on procède autrement que pour un corpus entier : à partir de l'ensemble des listes de lemmes extraites, on forme une liste totale, et on établit les indices de similarité entre le pivot et les lemmes extraits, tranche par tranche ; on obtient ainsi une matrice de trnch colonnes × n lemmes semblables (n indéterminable à l'avance, puisque cela varie selon les propriétés des distributions lexicales de chaque tranche et leur nombre) ; cette matrice doit être nettoyée pour éliminer les outliers qui peuvent bloquer l'analyse factorielle ; on procède enfin à l'AFC et à l'affichage des points sur les 2 premiers axes. On a ici des points-colonnes (les périodes) et des points-lignes (les lemmes 'voisins'). On joint les points-colonnes dans l'ordre des tranches-périodes.

En entrée :
dsm : nom de l'objet list (multi-'dsm') à utiliser
lm : lemme pivot
nppv : nombre de plus proches voisins à rechercher
xax : orientation du premier axe (mettre -1 pour inverser)
yax : orientation du deuxième axe (mettre -1 pour inverser)
cex : taille des caractères sur le graphique

`decal` : supprimer les recouvrements d'étiquettes, ou non

En sortie :
↬ un objet de type 'list' à 6 éléments :

* matrice de similarités brute
* matrice de similarité traitée
* variance des lignes
* variance des colonnes
* matrice de similarité en forme de tableau disjonctif
* coordonnées de l'ensemble des points sur les 2 premiers axes

↬ un graphique factoriel fournissant une image de l'évolution du champ sémantique