TreeCloud & Unitex: une synergie accrue

Claude Martineau

UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

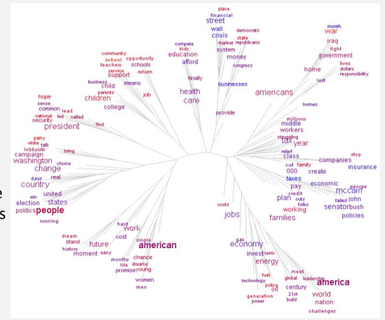
LABORATOIRE D'INFORMATIQUE GASPARD-MONGE

Sous la co-tutelle de :
CNRS
ÉCOLE DES PONTS PARISTECH
ESIEE PARIS
UPEM • UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

Projet PEPS CNRS/UPE Eclavit

TreeCloud: visualisation du texte en forme de Nuage Arboré

TreeCloud construit une visualisation du texte sous la forme d'un nuage arboré, dans lequel les étiquettes sont affichées autour d'un arbre pour refléter la distance de cooccurrence entre les mots dans le texte.



Le nuage arboré ci-contre donne une vue d'ensemble des discours de la campagne présidentielle 2008 de Barack Obama.

Construction du nuage arboré

Texte d'entrée
↓
Suppression des mots vides — *antidico français, anglais, espagnol, etc.*
↓
Sélection des mots — *Les n mots les plus fréquents*
↓
Recherche des cooccurrences — *Fenêtre de cooccurrence (fenêtre glissante, version online) ou caractère séparateur*
↓
Calcul des distances entre mots — *12 formules de distance de cooccurrence*
↓
Construction de l'arbre — *Appel à SplitsTree* (*version téléchargeable*), *intégré* (*version online*)
↓
Taille des mots — *Fréquences ou valeurs personnalisées*
↓
Couleur des mots — *Fréquences, chronologie, dispersion ou valeurs personnalisées*
↓
Dessin du nuage arboré — *Appel à SplitsTree* (*version téléchargeable*), *intégré* (*version online*)

- L'antidictionnaire contient les mots grammaticaux, les auxiliaires et les verbes de modalité.

  Par exemple, en anglais: *the*, *of*, *is*, *was*, *have*, *may*, *can*, etc.

  Les mots interdits sont habituellement les plus communs d'une langue. Pour construire un arbre significatif, tous ces mots doivent être tout d'abord **supprimés** du texte d'entrée.
- Lorsque une fenêtre glissante est utilisée, sa taille (c.a.d lenombre de mots) doit être donné en paramètre

Calcul des cooccurrences et distances

Soient deux mots *A* et *B* et:

- O_{11}, le nombre observé de fenêtres glissantes contenant **à la fois** *A* et*B*
- O_{12}, le nombre observé de fenêtres glissantes contenant *A* **mais pas** *B*
- O_{21}, le nombre observé de fenêtres glissantes **ne** contenant **pas** *A* **mais** *B*
- O_{22}, le nombre observé de fenêtres glissantes **ne** contenant **ni** *A* **ni** *B*

Les variable suivantes sont définies:

- $R_1 = O_{11} + O_{12}$, le nombre de fenêtres glissantes contenant *A*
- $R_2 = O_{21} + O_{22}$, le nombre de fenêtres glissantes **ne** contenant **pas** *A*
- $C_1 = O_{11} + O_{21}$ le nombre de fenêtres glissantes contenant *B*
- $C_2 = O_{12} + O_{22}$, le nombre de fenêtres glissantes **ne** contenant **pas** *B*
- $N = R_1 + R_2 = C_1 + C_2$, number of sliding windows
- $E_{11} = (R_1C_1/N$ le nombre attendu de fenêtres glissantes contenant **à la fois** *A* et *B*
- $E_{12} = (R_1C_2/N)$, le nombre attendu de fenêtres glissantes contenant *A* **mais pas** *B*
- $E_{21} = (R_2C_1/N)$, le nombre attendu de fenêtres glissantes **ne** contenant **pas** *A* **mais** *B*
- $E_{22} = (R_2C_2/N)$, le nombre attendu de fenêtres glissantes **ne** contenant **ni** *A* **ni** *B*

Les définitions des formules de **cooccurrence** sont les suivantes:

- **jaccard**: $1 - O_{11} / (O_{11} + O_{12} + O_{21})$
- **liddell**: $1 - (O_{11}O_{22} - O_{12}O_{21}) / (C_1C_2)$
- **dice**: $1 - 2O_{11} / (R_1 + C_1)$
- **hyperlex**: $1 - \max(O_{11} / R_1, O_{11} / C_1)$
- **poissonstirling**: $O_{11}(\log O_{11} - \log E_{11} - 1)$
- **chisquared**: $1000 - N(O_{11} - E_{11})^2 / (E_{11}E_{22})$
- **zscore**: $1 - (O_{11} - E_{11}) / \mathrm{sqr}(E_{11})$
- **ms**: $1 - \min(O_{11} / R_1, O_{11} / C_1)$
- **oddsratio**: $1 - \log((O_{11}O_{22}) / (O_{12}O_{21}))$
- **loglikelihood**: $1 - 2(O_{11}\log(O_{11}/E_{11}) + O_{12}\log(O_{12}/E_{12}) + O_{21}\log(O_{21}/E_{21}) + O_{22}\log(O_{22}/E_{22}))$
- **gmean**: $1 - O_{11}/\mathrm{sqr}(R_1C_1) = 1 - O_{11}/\mathrm{sqr}(NE_{11})$
- **mi** (mutual information): $1 - \log(O11/E_{11})$
- **ngd** (normalized Google distance): $(\max(\log R_1, \log C_1) - \log O_{11}) / (N - \min(\log R_1, \log C_1))$

Plusieurs versions de TreeCloud

Version téléchargeable en Python

- 2009: TreeCloud 1.3 pour Windows, Linux, Mac, dev. par Philippe Gambette Utilisation de SplitsTree 4.10 pour dessiner l'arbre

Version online en C

- 2009: 1ère version C,developpée par Jean-Charles Bontemps
- 2012: Transition à Unicode developpée par Claude Martineau
- 2014: 1ère implémentation d'Unitex developpée par Claude Martineau

Unitex/GramLab est un outil d'analyse et d'annotation de corpus

- **Fondé sur les notions d'automate et de RTN avec sorties**
- **Multilingue**: Jusqu'à 22 langues (français, anglais,..., grec, ... , coréen, thaï)
- **Unicode 3.0** (UTF8, UTF16LE, UTF16BE)
- **Multiplateforme**: Linux, macOS, Windows
- **Open source**: https://github.com/UnitexGramLab
- **Site web et installeurs binaires**: http://unitexgramlab.org
- **Développé depuis 2001 par une communauté de développeurs et d'utilisateurs**

Unitex/GramLab utilise *des ressources linguistiques*:

- **DELA (LADL dictionnaires électroniques)**

  Une entrée de type DELA est constituée d'une forme fléchie de mot simple ou composé, suivi d'un lemme et d'une information grammaticale. Chaque entrée peut comporter des attributs syntaxiques et sémantiques et des informations flexionnelles:

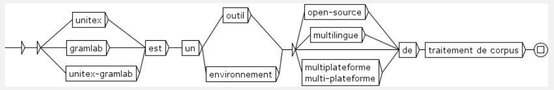
  f**orme_fléchie,lemme.infos_grammaticales+attributs:infos_flexionnelles**

  Exemple: considérons le mot simple *avocat et le mot composé avocat d'affaires, les représentations DELA sont les suivantes:*

  avocat,avocat.N+Hum+Prof:ms — avocat d'affaires,avocat d'affaires.N+Hum+Prof:ms
  avocate,avocat.N+Hum+Prof:fs — avocate d'affaires,avocat d'affaires.N+Hum+Prof:fs
  avocats,avocat. N+Hum+Prof:mp — avocats d'affaires,avocat d'affaires. N+Hum+Prof:mp
  avocates,avocat.N+Hum+Prof:fp — avocates d'affaires,avocat d'affaires.N+Hum+Prof:fp

  1 forme fléchie
  2 ,forme canonique
  3 .catégorie grammatical e
  4 +attributs sémanticques
  5 :information flexionnelle (m: masculin, f: féminin, s: singulier, p: pluriel)

- **Règles syntaxiques ou sémantiques appelées «grammaires locales» représentées par des graphes**
  - Les grammaires locales sont représentées sous la forme de graphes
  - Un motif est reconnu s'il existe un chemin allant de l'état initial à l'état final



Quelques exemples de séquences reconnues et non reconnues par la grammaire ci-dessus:

Unitex-GramLab est un outil multilingue de traitement de corpus [**RECONNUE**]
Unitex-GramLab est un outil de traitement de corpus [**RECONNUE**]
Unitex-GramLab est difficile à apprendre [**ECHEC**]
Unitex-GramLab est [**ECHEC**]

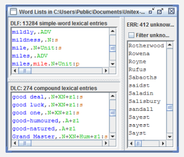
Un exemple d'analyse

Application d'un dictionnaire; obtention du **dictionnaire du texte; application d'une grammaire locale**

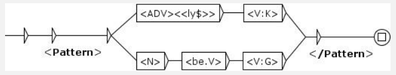
La grammaire ci-dessous, pour l'anglais contient deux chemins:

- Un **adverbe** (<***ADV***>) finissant en ***-ly*** suivi d'un **participe *passé*** (<***V:K***>)
- Un **nom**(<***N***>) suivi d'un **verbe** à la forme **progressive** (<***be.V***> <***V:G***>)

Un masque lexicale comme ***<V:K>*** fait appel au dictionnaire du texte.
Les séquences reconnues sont entourées par l'étiquette <**pattern**>. Les résultats sont représentés sous la forme de concordances.



Dictionnaire du texte



Une grammaire locale très simple

by one who has <Pattern>unwittingly fallen</Pattern> under your disple
.nce John, "his <Pattern>heart is sinking</Pattern>; I am jealous lest
rchers, having <Pattern>previously determined</Pattern> by lot their o

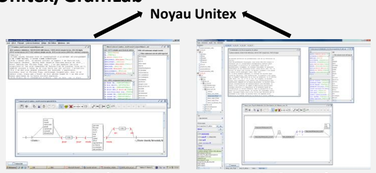
Sample of concordances

Remarquons que Unitex/GramLab peut également produire un texte annoté avec tous les motifs reconnus étiquetés

Plusieurs manières d'utiliser Unitex/GramLab

Deux interfaces écrits en **JAVA**:
- **IDE Unitex** (classique)
- **IDE GramLab (**orienté projet)

Ils appelent le:
Noyau Unitex écrit en **C/C++**



IDE Unitex **IDE GramLab**

Utiliser l'API C et JAVA (JNI) qui donne accès à
- **Un système de fichier virtuel**
- **La persistance des ressources en mémoire (alphabets, dictionaries and corpora)**

Lignes de commandes ou appels système avec Perl, Python, etc.

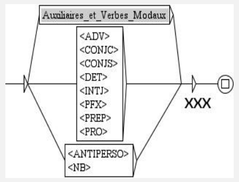
Comment et Pourquoi intégrer Unitex dans TreeCloud ?

Construction de l'arbre avec Unitex

1) **Unitex transforme** le ***texte d'entrée*** en un ***nouveau texte***
avec tous les **mots interdits remplacés** par le mot ***XXX***

2) Le ***nouveau texte*** est envoyé à TreeCloud avec ***XXX*** comme unique mot interdit

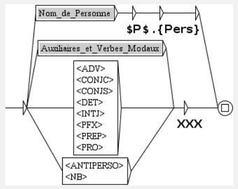
Obtenir une couverture des mots interdits plus précise et étendue



La grammaire locale ci-contre contient trois chemins :

- La boîte grisée représente un appel à un **sous-graphe** qui reconnaît des ***auxiliaires*** et des ***verbes modaux.***
- La seconde boîte utilise des **masques lexicaux** (<ADV>, <CONJ>, ...) pour reconnaître les **mots *grammaticaux.***
- Le masque lexicale <ANTIPERSO> dans la dernière boîte reconnaît la liste des mots interdits fournie par l'utilisateur. Chaque mot de cette liste est étiqueté avec ANTIPERSO. <***NB***> reconnaît les nombres.

Faire apparaître des multimots dans l'arbre

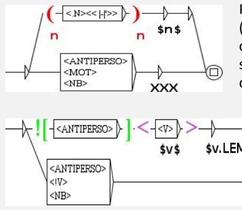


Si un dictionnaire contient des mots composés ceux-ci peuvent être gardés dans l'arbre mais toutes les sortes de multimots ne peuvent être listées dans un dictionnaire.

La grammaire ci-contre contient un chemin qui reconnaît les noms de personnes . La variable P contient le nom d'une personne capturé par le sous-graphe Name_of_Person.

L'étiquette .{Pers} est ajoutée à la sortie (version online 2017).

Effectuer une forte sélection des mots dans l'arbre



Puisque le nombre de mots gardés dans l'arbre est très faible (de 50 à 150) et dépend de leur fréquence, pour qu'une catégorie particulière soit affichée, sa sélection doit être soigneusement effectuée. La grammaire ci-contre ne garde que les noms composés.

Si l'on souhaite afficher des verbes dans l'arbre, il peut être utile de connaître leur **LEMME** (puisque ils ont de nombreuses formes fléchies). La grammaire ci-contre est établie dans ce but. Un processus similaire est utilisé pour les langues à cas (par ex: serbe ou grec).

Version online 2017 de TreeCloud: une implémentation améliorée

Notion de fichier de traitement

Dans la version online de 2014, il n'y avait qu'un seul traitement Unitex pour chaque langue. La liste de ressources nécessaires était codée en dur dans le code du programme.

Dans la nouvelle version 2017, il peut y avoir plusieurs versions pour chaque langue. En outre, la langue serbe (Latin et Cyrillique) a été ajoutée. Afin de gérer ces paires (langue, traitement) la notion de fichier de traitement a été mise en place.

Par exemple, dans le fichier de traitement ci-dessous, la première ligne indique le chemin des ressources pour le français, ensuite les trois dictionnaires (dictionnaire général du français, dictionnaire de prénoms, dictionnaire de toponymes) doivent être appliqués au texte. A la fin la grammaire locale est appliquée en mode REPLACE.

```
REP:TreeCloud_WS/French/src
NB_DICOS=3
        FICHIER=Dela/dela-fr-public.bin
        FICHIER=Dela/prenom-s.bin
        FICHIER=Dela/Prolex-Unitex_1_2_TOPONYMES.bin
GRAMMAIRE:
        FICHIER=Graphs/Treecloud_N_Pers_Top_v1_FR.fst2
        MODE=REPLACE
```

Tirer parti du travail déjà effectué par Unitex

Etapes d'analyse Unitex/GramLab

programme appelé	fichier créé
Normalize	*text.snt*
Tokenize	*tokens.txt, text.cod*
Dico	*dlf, dlc, err*
Locate	*concord.ind*
Concord	

Concordances — **Texte annoté**

A la fin de l'analyse, *text.snt* contient un texte normalisé (normalisation des caractères séparateurs), ***text.cod*** contient la liste de positions dans le fichier liste des tokens ***tokens.txt***.

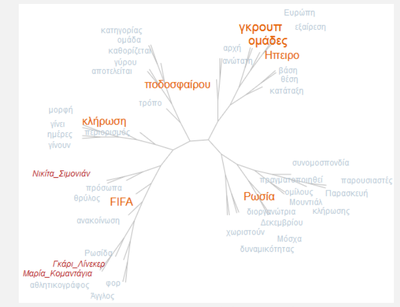
dlf, *dlc*, *err*, contiennent respectivement les mots simples, composés, et inconnus

concord.ind contient les séquences reconnues avec leur position dans le texte (XXX, et multimots)

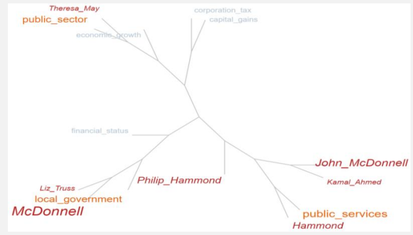
Pour obtenir le *«nouveau texte»*, nous «**retokenizons»** le texte avec les **séquences reconnues** du fichier ***concord.ind*** comme les nouveaux tokens du texte. De nouveaux fichiers ***token.txt*** et ***text.cod*** sont créés. Ce processus évite une double lecture du texte et une double division en mots.

Grâce à l'**API Unitex** et au **système de fichiers virtuels**, tout ce travail est effectué en **mémoire**.

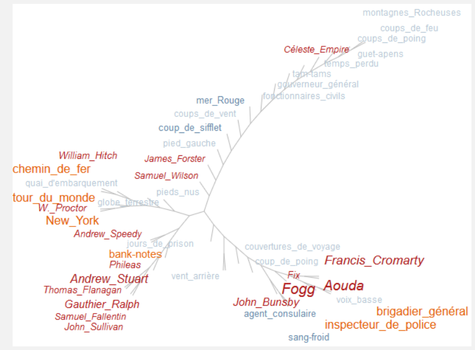
Quelques exemples d'arbres en différentes langues



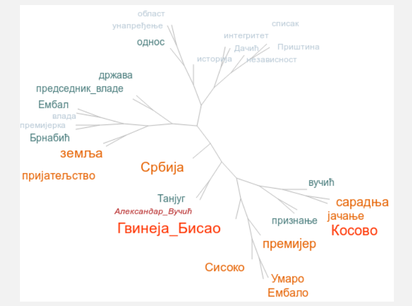
Arbre des 50 mots les plus fréquents d'un article grec



Arbre des 20 noms composés et noms de personne les plus fréquents d'un article anglais



Arbre des 45 noms composés et noms de personne les plus fréquents d'un roman de Jules Verne



Arbre des 30 mots les plus fréquents d'un article serbe

Conclusion

Intégrer **Unitex** dans **TreeCloud** permet:

- **Une représentation plus fine des mots interdits**
- **A toute sorte de multimots d'être reconnus dans le texte et affichés dans l'arbre**
- **Une visualisation de la catégorie grammaticale ou sémantique de certains mots.**
- **Une construction plus rapide de l'arbre (grâce à une utilisation fine de l'API Unitex)**

http://treecloud.univ-mlv.fr