Segmenteur Étiqueteur Markovien (SEM)

Table des matières

1 Préface

1.1 Présentation de SEM

Segmenteur Étiqueteur Markovien (SEM) [Tellier et al.2012b] est un logiciel d'annotation syntaxique du français.

Il permet la segmentation de texte brut en phrases, elles-même découpées en unités lexicales, mais il est tout-à-fait en mesure de traiter un texte pré-segmenté. Les unités multi-mots peuvent être gérées de deux manières différentes : soit comme une seule unité lexicale où chaque mot est relié par le caractère '_', soit comme une suite de mots ayant une annotation particulière précisant que les mots sont reliés entre eux et possèdent globalement la même fonction syntaxique.

SEM propose trois niveaux d'annotation : le premier est une annotation morpho-syntaxique de chaque unité lexicale du texte selon le jeu d'étiquettes défini par Crabbé et Candito (TALN 2008). Le deuxième est une annotation en chunks selon le modèle BIO (Begin In Out), le programme permettant d'obtenir un étiquetage selon un chunking complet ou bien partiel, auquel cas il ne reconnaîtra que les groupes nominaux (le chunking partiel étant soumis à des règles différentes que le chunking complet, l'un n'est donc pas un sous-ensemble de l'autre). Le troisième est une annotation en entités nommées sur l'ensemble des types défini par [**?**].

Toutes les commandes du manuel sont mises entre guillemets pour les distinguer clairement du reste du texte, mais elle doit être écrite sans eux.

2 Installation

Sur la page suivante se trouvent toutes les informations nécessaires :

https ://github.com/YoannDupont/SEM

SEMdoit être téléchargé pour être installé, le script d'installation se lance avec la commande :
python setup.py install
qui se chargera d'installer SEM avec tous les prérequis.
Il existe deux possibilités pour télécharger la dernière version.

2.1 Si GIT est installé

Il faut alors aller dans un terminal et taper la commande suivante :

git clone https ://github.com/YoannDupont/SEM.git

Cela va créer un dossier de SEM dans le répertoire où est tapée la commande.

Il s'agit de la branche GIT (dépôt), qui sert à gérer les différentes versions du logiciel. Il ne faut en AUCUN cas modifier le contenu de ce dossier (c'est surtout vrai si on prévoit de mettre-à-jour la branche, mais c'est une habitude à prendre immédiatement). Pour utiliser SEM, il faut copier les différents fichiers et dossiers dans un autre répertoire. Un dossier .git est présent : étant caché il ne sera pas copié si on n'active pas l'affichage des fichiers cachés, sinon il faut le déselectionner. C'est ce dossier qui contient les informations de versionnement.

L'intérêt ici est de pouvoir mettre-à-jour simplement le logiciel en tapant la commande "git pull" dans la branche. Cela mettra à jour uniquement les fichiers qui doivent l'être, ce qui est pratique quand (comme ici) le contenu est assez lourd.

2.2 Si GIT n'est pas installé

Sur https ://github.com/YoannDupont/SEM cliquer sur "clone or download" puis "Download ZIP".

L'avantage de cette méthode est qu'il s'agit des fichiers non-versionnés, il n'est donc pas nécessaire d'être aussi précautionneux avec le contenu du dossier. L'inconvénient est que pour mettre à jour, il faut tout retélécharger.

2.3 Wapiti

Wapiti [Lavergne et al.2010] est un logiciel implémentant les CRF, il permet d'apprendre des annotateurs à partir de corpus annotés et d'effectuer l'annotation.

La dernière version de Wapiti compatible avec SEM est disponible dans le dossier ext. Les consignes d'intallation sont disponibles avec. SEM est prévu pour fonctionner avec le Wapititel qu'il est fourni dans le dossier ext, il faut le compiler pour pouvoir appeler Wapiti avec SEM. Depuis la version 3.0.0 de SEM, Wapiti est automatiquement compilé à l'installation.

2.3.1 Erreurs de compilation

Wapiti est un logiciel ayant recours à certaines spécificités du matériel et du système d'exploitation pour améliorer ses performances. En conséquence, il est possible d'avoir des erreurs à l'étape I/3- étant dues à l'abscence de ces spécificités sur votre machine.

La plus fréquente semble être dûe à la fonction "__sync_bool_compare_and_swap" présente dans le fichier "gradient.c". Si la commande make provoque une erreur et vous af-

fiche des messages relatifs à cette fonction, la procédure est très simple.

Dans le fichier "wapiti.h", cherchez la ligne :
//#define ATM_ANSI
Et supprimez la chaîne "//" en début de ligne pour obtenir :
#define ATM_ANSI
Sauvegardez et reprenez la procédure d'installation.

3 Corpus, annotations et ressources linguistiques

3.1 French Treebank (FTB)

SEM a été appris automatiquement sur le French Treebank (FTB) [Abeillé et al.2003].

3.2 Jeu d'annotation PoS

L'annotation PoS se base sur le jeu d'étiquettes défini par [Crabbé and Candito2008] :

ADJ : adjectif
ADJWH : adjectif
ADV : adverbe
ADVWH : adverbe
CC : conjonction de coordination
CLO : clitique objet
CLR : clitique réfléchi
CLS : clitique sujet
CS : conjonction de subordination
DET : déterminant
DETWH : déterminant
ET : mot étranger
I : interjection
NC : nom commun
NPP : nom propre
P : préposition
P+D : préposition
P+PRO : préposition
PONCT : ponctuation
PREF : préfixe
PRO : pronom
PROREL : pronom
PROWH : pronom
VINF : verbe à l'infinitif
VPR : verbe au participe présent
VPP : verbe au participe passé
V : verbe à l'indicatif
VS : verbe au subjonctif
VIMP : verbe à l'impératif

3.3 Annotation en chunks

Le chunking utilise les annotations définies dans [Tellier et al.2012a] :

AP : groupe adjectival
AdP : groupe adverbial
NP : groupe nominal
VN : noyau verbal
CONJ : conjonction
UNKNOWN : chunk inconnu
PP : chunk prépositionnel

3.4 Annotation en entités nommées

Pour effectuer la reconnaissance des entités nommées, SEM se base sur les annotations définies par [Sagot et al.2012] :

Person : les personnes
Location : les lieux
Organization : toute organisation ou association à but non lucratif
Company : les entreprises
POI : Point Of Interet (exemple : l'Opéra)
FictionCharacter : les personnages fictifs
Product : les produits

3.5 Lexique des Formes Fléchies du Français (LeFFF)

Le Lexique des Formes Fléchies du Français (LeFFF) [Clément et al.2004] est un lexique du Français riche fournissant des information morphologiques et syntaxiques. SEM utilise le LeFFF en tant que dictionnaire afin d'améliorer la qualité de son annotation PoS.

4 Formats des fichiers

SEM permet de traiter deux types de fichiers en entrée : les fichiers dits linéaires et les fichiers dits vectorisés.

4.1 fichiers linéaires

Un fichier linéaire est un fichier dans lequel les mots sont (souvent) séparés par un espace. Ils représentent la majorité des textes (texte brut). SEM considère qu'un retour à la ligne termine une phrase, lorsqu'il fournit en sortie un fichier linéaire, chaque phrase sera séparée par un retour à la ligne. Si un fichier en entrée est un fichier linéaire, SEM pourra le segmenter en tokens et phrases.

SEM ne peut traiter en entrée que les fichiers de texte brut.

4.1.1 Exemples

exemple 1 : texte brut
Le chat dort.
exemple 2 : texte annoté en PoS
Le/DET chat/NC dort/V ./PONCT
exemple 3 : texte annoté en PoS et en chunks
(NP Le/DET chat/NC) (VN dort/V) ./PONCT

4.2 fichiers vectorisés

Un fichier vectorisé en un fichier où chaque mot est sur une ligne, les phrases étant séparées par une ligne vide. Dans un fichier vectorisé, chaque token peut contenir plusieurs informations, ces informations sont séparées par des tabulations. Chaque information est donc sur une « colonne » qui lui est spécifique.

4.2.1 Exemples

exemple 1 : texte brut vectorisé

```
Le
chat
dort
.
```

exemple 2 : texte vectorisé enrichi avec l'information « le mot commence-t-il par une majuscule ? »

```
Le      oui
chat    non
dort    non
.       non
```

exemple 3 : texte vectorisé annoté en PoS

```
Le      DET
chat    NC
dort    V
.       PONCT
```

exemple 4 : texte vectorisé annoté en PoS et en chunks

```
Le      DET     B-NP
chat    NC      I-NP
dort    V       B-VN
.       PONCT   0
```

5 Utilisation

SEM dispose de module indépendants les uns des autres, le programme principal faisant alors office d'aiguilleur vers le module à lancer.

Pour obtenir la liste des modules disponibles et la syntaxe générale pour les lancer :
python -m sem (–help ou -h)

Pour connaître la version de SEM :
python -m sem (–version ou -v)

Pour connaître les informations relatives à la dernière version de SEM :

python -m sem (–informations ou -i)

Pour lancer un module, la syntaxe générale est :
python -m sem <nom_du_module> <arguments_et_options_du_module>

Les différents modules seront détaillés un par un.

5.1 annotate

description

arguments

options

–help ou -h : switch
affiche l'aide

–input-encoding : string
définit l'encodage du fichier d'entrée. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–output-encoding : string
définit l'encodage du fichier de sortie. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–encoding : string
définit l'encodage du fichier d'entrée et de sortie. Si un encodage est fourni pour un fichier, cette valeur est surchargée (défaut : UTF-8).

–log ou -l : string
définit le niveau de log : info, warn ou critical (défaut : critical).

–log-file : fichier
le fichier où écrire le log (défaut : sortie terminal).

5.2 chunking_fscore

description

arguments

options

–help ou -h : switch
affiche l'aide

–input-encoding : string

définit l'encodage du fichier d'entrée. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–output-encoding : string
définit l'encodage du fichier de sortie. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–encoding : string
définit l'encodage du fichier d'entrée et de sortie. Si un encodage est fourni pour un fichier, cette valeur est surchargée (défaut : UTF-8).

–log ou -l : string
définit le niveau de log : info, warn ou critical (défaut : critical).

–log-file : fichier
le fichier où écrire le log (défaut : sortie terminal).

5.3 clean

description

clean_info permet de supprimer des colonnes d'informations dans un fichier vectorisé.

arguments

infile : fichier
le fichier d'entrée. Format vectorisé.

outfile : fichier
le fichier de sortie. S'il existe déjà, son contenu sera écrasé.

ranges : string
les colonnes à garder. Il est possible de donner soit un numéro de colonne soit une portée. Une portée se constitue de deux nombres séparés par le symbole « : ». Il est possible de fournir plusieurs valeurs en les séparant par le symbole « , ».

options

–help ou -h : switch
affiche l'aide

–input-encoding : string
définit l'encodage du fichier d'entrée. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–output-encoding : string
définit l'encodage du fichier de sortie. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–encoding : string
définit l'encodage du fichier d'entrée et de sortie. Si un encodage est fourni pour un fichier, cette valeur est surchargée (défaut : UTF-8).

–log ou -l : string
définit le niveau de log : info, warn ou critical (défaut : critical).

–log-file : fichier
le fichier où écrire le log (défaut : sortie terminal).

5.4 enrich

description

Permet de rajouter des informations à un fichier vectorisé. Les informations rajoutées sont déclarées dans un fichier de configuration xml. Il est possible d'ajouter deux types d'informations. Premièrement des informations endogènes, que l'on peut déduire à partir des informations déjà présentes. Deuxièmement des informations exogènes, c'est-à-dire des informations issues de dictionnaires.

arguments

infile : fichier
le fichier d'entrée, format vectorisé.
infofile : fichier
fichier pour ajouter des informations, format xml.
outfile : fichier, format vectorisé.
le fichier de sortie

options

–help ou -h : switch
affiche l'aide
–input-encoding : string
définit l'encodage du fichier d'entrée. Prioritaire sur la valeur de –encoding (défaut : –encoding).
–output-encoding : string
définit l'encodage du fichier de sortie. Prioritaire sur la valeur de –encoding (défaut : –encoding).
–encoding : string
définit l'encodage du fichier d'entrée et de sortie. Si un encodage est fourni pour un fichier, cette valeur est surchargée (défaut : UTF-8).
–log ou -l : string
définit le niveau de log : info, warn ou critical (défaut : critical).
–log-file : fichier
le fichier où écrire le log (défaut : sortie terminal).

5.5 export

description

arguments

options

–help ou -h : switch
affiche l'aide

–input-encoding : string
définit l'encodage du fichier d'entrée. Prioritaire sur la valeur de –encoding (défaut : –encoding).
–output-encoding : string
définit l'encodage du fichier de sortie. Prioritaire sur la valeur de –encoding (défaut : –encoding).
–encoding : string
définit l'encodage du fichier d'entrée et de sortie. Si un encodage est fourni pour un fichier, cette valeur est surchargée (défaut : UTF-8).
–log ou -l : string
définit le niveau de log : info, warn ou critical (défaut : critical).
–log-file : fichier
le fichier où écrire le log (défaut : sortie terminal).

5.6 label_consistency

description

arguments

options

–help ou -h : switch
affiche l'aide

–input-encoding : string
définit l'encodage du fichier d'entrée. Prioritaire sur la valeur de –encoding (défaut : –encoding).
–output-encoding : string
définit l'encodage du fichier de sortie. Prioritaire sur la valeur de –encoding (défaut : –encoding).
–encoding : string
définit l'encodage du fichier d'entrée et de sortie. Si un encodage est fourni pour un fichier, cette valeur est surchargée (défaut : UTF-8).
–log ou -l : string
définit le niveau de log : info, warn ou critical (défaut : critical).
–log-file : fichier
le fichier où écrire le log (défaut : sortie terminal).

5.7 tagging

description

arguments

options

–help ou -h : switch
affiche l'aide

–input-encoding : string
définit l'encodage du fichier d'entrée. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–output-encoding : string
définit l'encodage du fichier de sortie. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–encoding : string
définit l'encodage du fichier d'entrée et de sortie. Si un encodage est fourni pour un fichier, cette valeur est surchargée (défaut : UTF-8).

–log ou -l : string
définit le niveau de log : info, warn ou critical (défaut : critical).

–log-file : fichier
le fichier où écrire le log (défaut : sortie terminal).

5.8 segmentation

description
Prend un fichier texte linéaire et segmente le texte en phrase et tokens et donne un fichier vectorisé.

arguments

infile : fichier
le fichier d'entrée. Format texte brut.

outfile : fichier
le fichier de sortie. Format vectorisé.

options

–help ou -h : switch
affiche l'aide

–input-encoding : string
définit l'encodage du fichier d'entrée. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–output-encoding : string
définit l'encodage du fichier de sortie. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–encoding : string
définit l'encodage du fichier d'entrée et de sortie. Si un encodage est fourni pour un fichier, cette valeur est surchargée (défaut : UTF-8).
–log ou -l : string
définit le niveau de log : info, warn ou critical (défaut : critical).
–log-file : fichier
le fichier où écrire le log (défaut : sortie terminal).

5.9 compile

description
Compile (sérialise) un fichier dictionnaire qui pourra alors être utilisé en ressource dans SEM.
arguments
input : fichier dictionnaire
Le dictionnaire à compiler.
output : fichier compilé
Le dictionnaire compilé.
options
–help ou -h : switch
affiche l'aide
-k ou –kind : énumération {token, multiword}
le type de dictionnaire en entrée. token : chaque entrée représente un mot. multiword : chaque entrée représente une suite de mots.
-i ou –input-encoding : string
définit l'encodage du fichier d'entrée. Prioritaire sur la valeur de –encoding (défaut : –encoding).
–log ou -l : string
définit le niveau de log : info, warn ou critical (défaut : critical).
–log-file : fichier
le fichier où écrire le log (défaut : sortie terminal).

5.10 decompile

description
Décompile (désérialise) un fichier dictionnaire. Cela permet alors de modifier la ressource (changement d'encodage, ajout / suppression / modification d'entrées).
arguments
input : fichier compilé
Le dictionnaire compilé.
output : fichier dictionnaire
Le dictionnaire décompilé.

options

–help ou -h : switch
affiche l'aide

–input-encoding : string
définit l'encodage du fichier d'entrée. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–output-encoding : string
définit l'encodage du fichier de sortie. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–encoding : string
définit l'encodage du fichier d'entrée et de sortie. Si un encodage est fourni pour un fichier, cette valeur est surchargée (défaut : UTF-8).

–log ou -l : string
définit le niveau de log : info, warn ou critical (défaut : critical).

–log-file : fichier
le fichier où écrire le log (défaut : sortie terminal).

5.11 tagger

description

Il s'agit du module principal de SEM. Il permet d'effectuer une chaîne de traitements sur un fichier. Ces traitements correspondent à des modules ou à des annotations faites à l'aide de Wapiti. Les modules à enchaîner et l'ordre dans lequel cet echaînement s'effectue est donné par un fichier de configuration xml appelé fichier de configuration maître.

arguments

master : fichier xml
le fichier de configuration maître. Définit le séquencage des traitements à effectuer.

input_file : fichier
le fichier d'entrée. Peut être soit un fichier de texte brut soit un fichier vectorisé.

options

–help ou -h : switch
affiche l'aide

–output-directory ou -o : dossier
le répertoire où les fichiers temporaires vont être créés (défaut : dossier courant).

5.12 gui

description

arguments

options

–help ou -h : switch
affiche l'aide

–input-encoding : string
définit l'encodage du fichier d'entrée. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–output-encoding : string
définit l'encodage du fichier de sortie. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–encoding : string
définit l'encodage du fichier d'entrée et de sortie. Si un encodage est fourni pour un fichier, cette valeur est surchargée (défaut : UTF-8).

–log ou -l : string
définit le niveau de log : info, warn ou critical (défaut : critical).

–log-file : fichier
le fichier où écrire le log (défaut : sortie terminal).

5.13 annotation_gui

description

arguments

options

–help ou -h : switch
affiche l'aide

–input-encoding : string
définit l'encodage du fichier d'entrée. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–output-encoding : string
définit l'encodage du fichier de sortie. Prioritaire sur la valeur de –encoding (défaut : –encoding).

–encoding : string
définit l'encodage du fichier d'entrée et de sortie. Si un encodage est fourni pour un fichier, cette valeur est surchargée (défaut : UTF-8).

–log ou -l : string
définit le niveau de log : info, warn ou critical (défaut : critical).

–log-file : fichier
le fichier où écrire le log (défaut : sortie terminal).

6 Fichiers de configuration

6.1 Pour le module enrich

Le fichier de configuration du module enrich permet d'ajouter des informations à un fichier vectorisé. Il décrit d'abord les entrées présentes puis les informations à ajouter.

Le fichier d'enrichissement est un fichier XML de type de document "enrich". Il a trois parties : une "entries" qui définit les entrées déjà présentes dans le fichier, une "endogenous" qui permet d'extraire des informations à partir des données présentes et une "exogenous" qui permet d'intégrer des dictionnaires.

Chaque entrée (qu'elle soit déjà présente ou ajoutée) doit être nommée (via l'attribut *"name"*) et deux entrées différentes ne peuvent pas avoir le même nom. En gras sont les différentes "sections", en italique les différentes informations. Un exemple de fichier de configuration pour le module enrich est illustré dans la figure 1.

```
<? xml version="1.0" encoding="UTF-8" ?>
<information>
  <entries>
    <before>
      <entry name="word" />
      <entry name="POS" />
    </before>
    <after>
      <entry name="NE" mode="train" />
    </after>
  </entries>
  <features>
    <nullary name="lower" action="lower" display="no" />
    <unary name="starts-with-upper" action="isUpper">0</unary>
    <dictionary name="title" action="token" path="title.txt" entry="lower" />
    <find name="VerbForward" action="forward" return_entry="word">
      <regexp action="check" entry="POS">^V</regexp>
    </find>
  </features>
</information>
```

FIGURE 1 – exemples de fichier de génération de features de SEM, il est utilisé par le module enrich. Il permet de rajouter des descripteurs qui seront alors utilisés par les algorithmes par apprentissage automatique.

6.2 Pour le module tagger

Le fichier de configuration du module tagger est appelé le fichier de configuration maître. Il permet de définir une séquence de traitements (modules) ainsi que des options globales aux différents modules qui seront lancés les uns après les autres.

Le fichier maître est un fichier XML de type de document "master". Il a deux parties : une "pipeline" qui est une séquences de modules et une "options" qui permet de définir les options globales. Un exemple de pipeline est illustré dans la figure 2.

```
<? xml version="1.0" encoding="UTF-8" ?>
<master>
  <pipeline>
    <segmentation name="fr" />
    <enrich config="pos.xml" />
    <label model="models/POS" field="POS" />
    <clean_info to-keep="word,POS" />
    <label model="models/chunking" field="chunking" />
    <enrich config="NER.xml" />
    <label model="models/NER" field="NER" />
    <clean_info to-keep="word,POS,chunking,NER" />
    <export format="html" pos="POS" chunking="chunking"
        ner="NER" lang="fr" lang_style="default.css" />
  </pipeline>

  <options>
    <encoding input-encoding="utf-8" output-encoding="utf-8" />
    <log log_level="INFO" />
    <export format="html" pos="POS" ner="NER" lang="fr" lang_style="default.css" />
  </options>
</master>
```

FIGURE 2 – Spécification d'un pipeline de SEM. Les pipelines permettent de définir une séquence de traitements ainsi que certaines options globales.

Références

[Abeillé et al.2003] Abeillé, A., Clément, L., and Toussenel, F. (2003). Building a treebank for french. In Abeillé, A., editor, *Treebanks*. Kluwer, Dordrecht.

[Clément et al.2004] Clément, L., Sagot, B., and Lang, B. (2004). Morphology based automatic acquisition of large-coverage lexica. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004*. European Language Resources Association.

[Crabbé and Candito2008] Crabbé, B. and Candito, M. H. (2008"). Expériences d'analyse syntaxique statistique du français. In *Actes de TALN'08*.

[Lavergne et al.2010] Lavergne, T., Cappé, O., and Yvon, F. (2010). Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 504–513. Association for Computational Linguistics.

[Sagot et al.2012] Sagot, B., Richard, M., and Stern, R. (2012). Annotation référentielle du corpus arboré de paris 7 en entités nommées. In *Actes de la 19e conférence sur le Traitement Automatique des Langues Naturelles*, pages 535–542, Grenoble, France. Association pour le Traitement Automatique des Langues.

[Tellier et al.2012a] Tellier, I., Duchier, D., Eshkol, I., Courmet, A., and Martinet, M. (2012a). Apprentissage automatique d'un chunker pour le français. In *Actes de TALN'12, papier court (poster)*.

[Tellier et al.2012b] Tellier, I., Dupont, Y., and Courmet, A. (2012b). Un segmenteur-étiqueteur et un chunker pour le français. In *Actes de TALN'12, session démo*.