Une documentation du projet UD et graphes

Mei GAN

1. Les objectifs du projet

L'objectif du projet est de construire un programme qui prend en entrée un fichier ou des fichiers au format UD. Lorsque l'utilisateur saisit deux mots/lemmes/POS, ce programme est capable d'afficher la phrase et les chemins de dépendance entre ces deux entrées.

2. Les données

Les données contiennent des fichiers .csv au format UD. Le source des fichiers .csv est sur le site https://github.com/UniversalDependencies/UD_French-Sequoia. Pour ce projet, deux des leurs fichiers sont adoptés :

Nom du fichier	Description
fr_sequoia-ud-test.conllu.csv	10,048 tokens, 456 phrases
fr_sequoia-ud-train.conllu.csv	50,517 tokens, 2,231 phrases

Un exemple de format UD est ci-dessous :

```
# sent_id = Europar.550_00063
# text = Il est essentiel de considérer les intérêts de toutes les parties.
1   Il  il  PRON    _   Gender=Masc|Number=Sing|Person=3    3   expl:subj   _   _
2   est être    AUX _   Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin   3   cop _   _
3   essentiel   essentiel   ADJ _   Gender=Masc|Number=Sing 0   root    _   _
4   de  de  ADP _   _   5   mark    _   _
5   considérer  considérer  VERB    _   VerbForm=Inf    3   csubj   _   _
6   les le  DET _   Definite=Def|Number=Plur|PronType=Art   7   det _   _
7   intérêts    intérêt NOUN    _   Gender=Masc|Number=Plur 5   obj _   _
8   de  de  ADP _   _   11  case    _   _
9   toutes  tout    ADJ _   Gender=Fem|Number=Plur  11  amod    _   _
10  les le  DET _   Definite=Def|Number=Plur|PronType=Art   11  det _   _
11  parties partie  NOUN    _   Gender=Fem|Number=Plur  7   nmod    _   SpaceAfter=No
12  .   .   PUNCT   _   _   3   punct   _   _
```

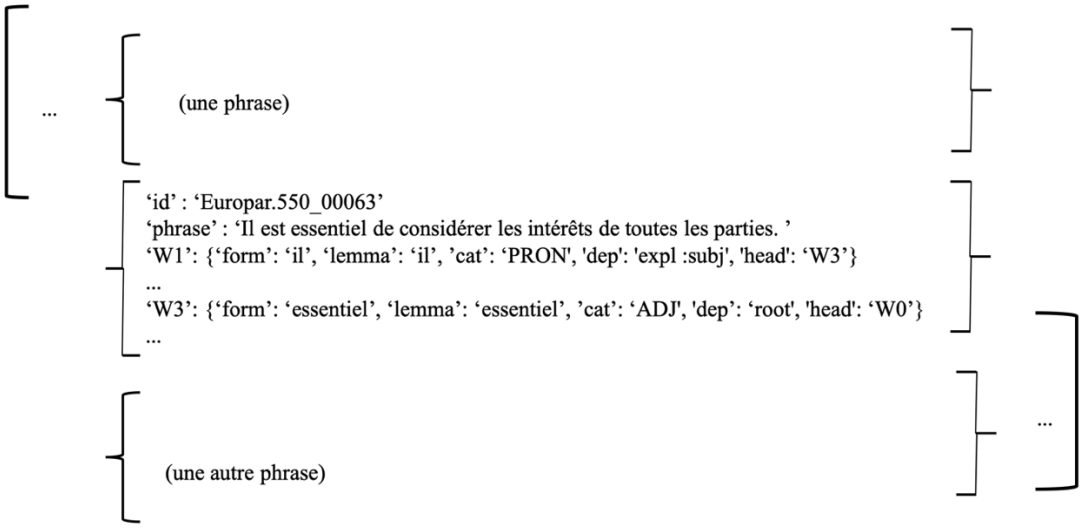
Pour la phrase « Il est essentiel de considérer les intérêts de toutes les parties. », la première colonne est index de chaque mot. La deuxième colonne est le mot. La troisième colonne est le lemme du mot et le quatrième est sa catégorie grammaticale. La sixième colonne présent son genre(masculin/féminin), nombre(pluriel/singulier) et personne (la 1er, 2^{e}, 3^{e} personne). Enfin, la septième colonne et la huitième colonne sont l'index de sa tête et la dépendance entre ce mot et sa tête. Par exemple, pour le premier mot « il », l'index est « 1 », le lemme est « il », sa catégorie grammaticale est un pronom, et « il » est un mot masculin au singulier et la troisième personne. Sa tête de dépendance est l'index 3 (le mot « essentiel ») et la dépendance est « essentiel » – [expl :subj] --> « il », c'est – à – dire que le mot « il » est le sujet du mot « essentiel ». Toutes les données sont dans le dossier « fichiers », on pourrait le modifier en mettant d'autre données au format UD.

3. Les méthodes utilisées

Les méthodes utilisées contiennent deux parties. La première partie est de lire les fichiers au format UD et former un graphe ; la deuxième partie est de trouver le chemin de dépendance de deux mots/lemmes/POS que l'utilisateur saisit.

3.1 Former un graphe

La première étape est de lire tous les fichiers dans le répertoire « fichiers ». Une boucle for a été utilisée pour lire tous les fichiers. Ensuite, une autre boucle for est adoptée pour lire le contenue des fichiers et former une liste des dictionnaires. Dans la liste, chaque phrase est un dictionnaire. Et dans un dictionnaire, 'id', 'phrase' et chaque l'index est les clés, et les valeurs des 'id' et 'phrase' sont un string et les valeurs des index sont aussi un dictionnaire. Le format du graphe est comme ci-dessous :



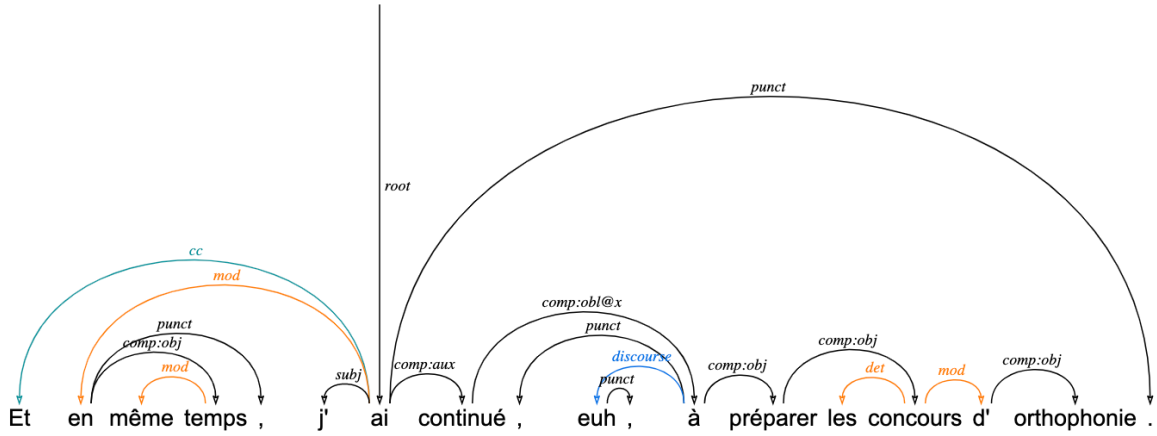
Par exemple, pour la phrase « Il est essentiel de considérer les intérêts de toutes les parties. », dans la dictionnaire, on voit son index(la clé « id ») de la phrase est « Europar.550_00063 » et la phrase comme la valeur de la clé « phrase », et puis pour chaque l'index du mot, la valeur est une dictionnaire qui contient les informations sur sa forme (la clé « form »), son lemme (la clé « lemma »), sa catégorie grammaticale (la clé « cat »), la dépendance (la clé « dep ») et enfin l'index de sa tête (la clé « head »).

Cette partie est réalisée dans la partie **def get_graphe( )** du script python.

3.2 Trouver le chemin de dépendance

Après avoir formé une liste des dictionnaires, l'étape suivante est de trouver le chemin de dépendance. Dans cette partie, on va l'achever par deux étapes : la première étape est de trouver le(s) chemin(s), la deuxième est de trouver leurs dépendances.

La méthode pour trouver le chemin est de trouver leur tête (parent) chaque fois jusqu'à la tête commune. Par exemple, si on a une phrase « Et en même temps, j'ai continué, euh, à préparer les concours d'orthophonie » et l'utilisateur voudrait savoir le chemin de dépendance enter le mot « j' » et « préparer ».



Par conséquence, d'abord, le mot « j' » n'est pas la tête du mot « préparer » et non l'inverse. Donc, il est nécessaire de trouver la tête du mot « j' », ce qui est « ai ». Et même temps, nous trouvons la tête du mot « préparer », ce qui est « à ». Ensuite, le mot « ai » n'est pas la tête du mot « à » et non l'inverse. Puis, nous trouvons la tête du mot « ai », ce qui est lui-même comme « ai » est le root de la phrase, et la tête du mot « à » est « continué ». Jusqu'au maintenant, nous trouvons la tête du mot « continué » est « ai ». Enfin, nous pouvons écrire le chemin de dépendance du mot « j' » et « préparer » : j' <– [subj]- ai – [comp :aux]-> continué – [comp :obl@x]-> à – [comp :obj]-> préparer. La direction des flèches est importante parce qu'elle signifie ce qui est la tête et ce qui est la dépendance.

Cela est réalisé par **def find_path( )** du script python. Premièrement, deux listes sont créées pour les têtes du premier mot et deuxième mot. Chaque fois, une tête sera ajoutée si elle n'est pas la tête commune. Lorsqu'on obtient leur tête commune, on peut former une liste du chemin de l'index du premier mot au dernier mot. Ici, le résultat n'est pas une liste des mots, mais une liste de l'index des mots.

Enfin, lorsque nous avons obtenu la liste de chemin, il est facile de trouver des dépendances et aussi la catégorie (mot/lemme/POS) que l'utilisateur choisit ainsi que l'index de la phrase et la phrase. D'abord, on ajoute le premier mot qui ne change pas. Puis, on trouve le deuxième mot/lemme/POS et l'index de sa tête selon son index. Ensuite, on compare l'index de sa tête avec l'index du premier. Si les deux sont égaux, c'est-à-dire, l'index du permier est la tête du deuxième, le chemin de dépendance sont écrit comme : premier mot -[{dep}]->deuxième mot ; sinon, on écrit à l'inverse : premier mot <-[{dep}]- deuxième mot. Et on le fera pour le reste. Cela est réalisé par **def find_dep_path( )** du script python.

Enfin, nous avons besoin de faire lier les deux parties. D'abord, quand l'utilisateur choisit la catégorie pour chercher (mot/lemme/POS) et il saisit deux entrées. Nous allons chercher si les deux mots existent dans une même phrase, s'ils existent, on ajoute leurs index dans deux listes séparément. Ici, il est possible que la saisie de l'utilisateur existe plusieurs fois dans une même phrase. Donc, on a besoin une liste pour les garder. Après avoir deux listes (une liste de l'index pour la premier saisie et l'autre pour la deuxième saisie). Nous pouvons faire **def find_path( )** et **def find_dep_path( )** pour trouver les chemins. Et à la fin, si le chemin existe, on ajoute toutes les informations sur l'index de la phrase et la phrase et les chemins dans une dictionnaire.

Cela est dans **def find_all_paths( )** du script python

4. Les résultats

Quand l'utilisateur lance le script, d'abord, il doit choisir parmi trois catégories (form/lemma/cat), puis, il peut saisir le premier mot/lemme/cat selon son choix, ensuite le deuxième. Par exemple, quand j'ai choisi « form » et puis j'ai saisi « intérêts » et « toutes », les résultats sont l'index de la phrase et le contenu de la phrase et le chemin de dépendance « intérêts -[nmod]-> parties -[amod]-> toutes ».

```
In [40]: runfile('/Users/meg/Documents/0_Inalco/03_cours/2_Langages_de_script_Plancq/projet2019/
projet_Mei_GAN_TAL_M1/projet_main的副本.py', wdir='/Users/meg/Documents/0_Inalco/03_cours/
2_Langages_de_script_Plancq/projet2019/projet_Mei_GAN_TAL_M1')
Bienvenu au monde de dépendance. D'abord, choisissez la catégorie(form/lemma/cat), ensuite saisissez deux
mots ou lemmes ou POS

Veuillez choisir la catégorie (form/lemma/cat)
form

le permier : intérêts

le deuxieme : toutes

Voici les résultats :

id :  Europar.550_00063
phrase :  Il est essentiel de considérer les intérêts de toutes les parties.
paths : ['intérêts -[nmod]-> parties -[amod]-> toutes']
************************************************************
```

Si nous voulons savoir le chemin de dépendance de deux lemmes « il » et « être », nous les saisissons et les résultats serons ci-dessous. Dans les résultats, il va afficher la forme et le lemme comme « J'/il ».

```
In [41]: runfile('/Users/meg/Documents/0_Inalco/03_cours/2_Langages_de_script_Plancq/projet2019/
projet_Mei_GAN_TAL_M1/projet_main的副本.py', wdir='/Users/meg/Documents/0_Inalco/03_cours/
2_Langages_de_script_Plancq/projet2019/projet_Mei_GAN_TAL_M1')
Bienvenu au monde de dépendance. D'abord, choisissez la catégorie(form/lemma/cat), ensuite saisissez deux
mots ou lemmes ou POS

Veuillez choisir la catégorie (form/lemma/cat)
lemma

le permier : il

le deuxieme : être

Voici les résultats :

id :  Europar.550_00063
phrase :  Il est essentiel de considérer les intérêts de toutes les parties.
paths : ['Il/il <-[root]- essentiel/essentiel -[cop]-> est/être']
************************************************************
id :  Europar.550_00065
phrase :  (EN) J'ai voté en faveur du rapport de Mme Stenzel sur l'initiative EQUAL car je pense qu'il est
essentiel de poursuivre le travail qui a été entamé avec les précédentes composantes NOW, Horizon,
Youthstart et Integra des initiatives ADAPT et Employment.
paths : ["J'/il <-[root]- voté/voter -[conj]-> pense/penser -[ccomp]-> essentiel/essentiel -[cop]-> est/
être", "J'/il <-[root]- voté/voter -[conj]-> pense/penser -[ccomp]-> essentiel/essentiel -[csubj]->
poursuivre/poursuivre -[obj]-> travail/travail -[acl:relcl]-> entamé/entamer -[aux:pass]-> été/être", 'je/il
<-[conj]- pense/penser -[ccomp]-> essentiel/essentiel -[cop]-> est/être', 'je/il <-[conj]- pense/penser -
[ccomp]-> essentiel/essentiel -[csubj]-> poursuivre/poursuivre -[obj]-> travail/travail -[acl:relcl]->
entamé/entamer -[aux:pass]-> été/être', 'il/il <-[ccomp]- essentiel/essentiel -[cop]-> est/être', 'il/il <-
[ccomp]- essentiel/essentiel -[csubj]-> poursuivre/poursuivre -[obj]-> travail/travail -[acl:relcl]->
entamé/entamer -[aux:pass]-> été/être']
************************************************************
```

Parallèlement pour la catégorie POS, les résultats afficherons aussi la forme puis le POS. Par exemple, si on voudrait savoir le chemin de dépendance entre nom (NOUN) et adjectif (ADJ), on peut choisir la catégorie « cat » et puis mettre « NOUN » et « ADJ » comme saisies. Les résultats seront ci-dessous. Il va afficher les index de la phrase et les phrase pour nous indiquer ce qui sont dans quelle phrase, puis il affiche aussi la forme des noms et des adjectifs qui nous permet de savoir les noms et les adjectifs spécifiques.

```
In [10]: runfile('/Users/meg/Documents/0_Inalco/03_cours/2_Langages_de_script_Plancq/projet2019/
projet_Mei_GAN_TAL_M1/projet_main.py', wdir='/Users/meg/Documents/0_Inalco/03_cours/
2_Langages_de_script_Plancq/projet2019/projet_Mei_GAN_TAL_M1')
Bienvenu au monde de dépendance. D'abord, choisissez la catégorie(form/lemma/cat), ensuite saisissez deux
mots ou lemmes ou POS

Veuillez choisir la catégorie (form/lemma/cat)
cat

le permier : NOUN

le deuxieme : ADJ
```

```
id :  frwiki_50.1000_00734
phrase :  Soit, au total, un peu plus de 150 millions de francs ont été versés par les grands groupes :
Bouygues, Eiffage, Spie-Batignolles, Suez et Générale des eaux.
paths : ['total/NOUN <-[root]- versés/VERB -[obl:agent]-> groupes/NOUN -[amod]-> grands/ADJ', 'millions/NOUN
<-[root]- versés/VERB -[obl:agent]-> groupes/NOUN -[amod]-> grands/ADJ', 'francs/NOUN <-[nsubj:pass]-
millions/NOUN <-[root]- versés/VERB -[obl:agent]-> groupes/NOUN -[amod]-> grands/ADJ', 'groupes/NOUN -
[amod]-> grands/ADJ', 'eaux/NOUN <-[conj]- Générale/PROPN <-[appos]- Bouygues/PROPN <-[obl:agent]- groupes/
NOUN -[amod]-> grands/ADJ']
************************************************************
id :  frwiki_50.1000_00736
phrase :  - Michel Roussin, ex-bras droit de Jacques Chirac à la mairie de Paris.
paths : ['ex-bras/NOUN -[amod]-> droit/ADJ', 'mairie/NOUN <-[appos]- ex-bras/NOUN -[amod]-> droit/ADJ']
************************************************************
id :  frwiki_50.1000_00762
phrase :  Le procès en première instance s'est terminé le mercredi 26 octobre 2005.
paths : ['procès/NOUN -[nmod]-> instance/NOUN -[amod]-> première/ADJ', 'instance/NOUN -[amod]-> première/
ADJ', 'mercredi/NOUN <-[root]- terminé/VERB -[nsubj]-> procès/NOUN -[nmod]-> instance/NOUN -[amod]->
première/ADJ', 'octobre/NOUN <-[nmod]- 26/NUM <-[obl:mod]- mercredi/NOUN <-[root]- terminé/VERB -[nsubj]->
procès/NOUN -[nmod]-> instance/NOUN -[amod]-> première/ADJ']
************************************************************
```