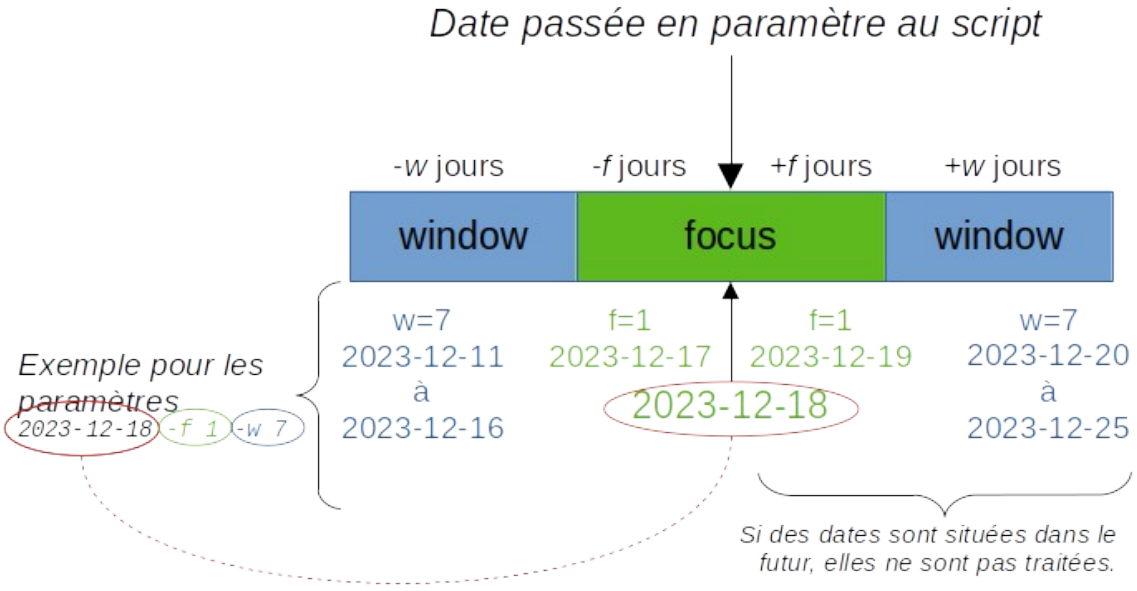
Projet *langages de scripts*

À rendre le 10 janvier 2024

Le projet consiste à créer un script de veille Internet, utilisable en ligne de commande. Vous devrez vous assurer que votre script fonctionne sur un site de votre choix, qui doit être dans une langue supportée par Spacy.

1. On lui donne une date (par exemple 2023-12-18) et un site à surveiller (par exemple lemonde.fr).
2. Il télécharge sur *Web Archive*[1] des versions de ce site pour une fenêtre de ± 7 jours (valeur paramétrable), c'est à dire 15 versions, du 2023-12-11 (inclus) à 2023-12-25 (inclus). Si certaines dates sont dans le futur, elles sont ignorées (pas de téléchargement, pas inclues dans les calculs).
3. Chaque version du site est nettoyée pour ne garder que le contenu textuel.
4. Chaque version est ensuite analysée avec *Spacy* pour récupérer les fréquences absolues de chaque lemme nominal.
5. Ces fréquences sont versées dans deux groupes en fonction de leur date :
   - Les dates ***focus***, celles qu'on veut étudier. C'est la date passée en paramètre au script ± 1 jour (paramétrable). Donc dans notre exemple, ce serait les 17, 18 et 19 décembre.
   - Les dates ***window***, qui correspondent à la date passée au script ± 7 jours (paramétrable), *moins* les dates focus. Donc dans notre exemple, ce serait les 11, 12, 13, 14, 15, 16, 20, 21, 22, 23, 24 et 25 décembre.



1 Sauf la version actuelle, qui est toujours la version *live* du site. Par exemple si on est le **18 décembre 2023**, toutes les versions passées du site seront téléchargées sur *Web Archive* (p. ex. https://web.archive.org/**web/20231217**/lemonde.fr), sauf celle du 18 décembre 2023 qui ira chercher directement lemonde.fr, au lieu de ~~https://web.archive.org/web/**20231218**/lemonde.fr~~.

6. Un score de spécificité est calculé à partir de ces 2 groupes de fréquences pour chaque lemme nominal.
7. Le script affiche enfin :
    - les 10 lemmes nominaux les moins spécifiques
    - les 10 lemmes nominaux les plus spécifiques

Exemple d'utilisation le **19** décembre 2023 :

```
python3 projet.py 2023-12-18 lemonde.fr
En cache: https://lemonde.fr # Version du 20231219
En cache: https://web.archive.org/web/20231218/lemonde.fr
En cache: https://web.archive.org/web/20231217/lemonde.fr
En cache: https://web.archive.org/web/20231216/lemonde.fr
En cache: https://web.archive.org/web/20231215/lemonde.fr
En cache: https://web.archive.org/web/20231214/lemonde.fr
En cache: https://web.archive.org/web/20231213/lemonde.fr
En cache: https://web.archive.org/web/20231212/lemonde.fr
Téléchargement: https://web.archive.org/web/20231211/lemonde.fr
Moins spécifiques
   « -2.7378865096233986
   dubaï -2.1797156973811154
   gouvernement -1.9218427236553874
   cop28 -1.913939650969338
   l’ -1.8524352775047697
   europe -1.6676209313906276
   dollar -1.6676209313906276
   activité -1.6676209313906276
   million -1.6667236555263298
   cantons-de-l’est -1.6667236555263298
Plus spécifiques
   ligue 1.2288309954920136
   scène 1.236799171900868
   aide 1.2763104137917236
   - 1.3344822348542797
   philippe 1.4707144345931493
   territoire 1.8436650201567468
   chanson 1.8436650201567468
   loi 2.4948653579747693
   direct 3.4574861573051576
   projet 3.696793139368558
```

En rouge, la commande. En vert, des logs sur ce qui est téléchargé / cherché dans un cache. En bleu, les résultats. Surligné en jaune, c'est juste un commentaire pour ce sujet d'examen, il ne faut pas l'inclure dans la sortie de votre script.

Notez qu'on n'essaie pas de télécharger les version situées dans le futur (>19 décembre), et que la version du 19 décembre est téléchargée directement depuis le site, pas depuis *Web Archive*.

Paramètres obligatoires du script :

- Une date, au format ISO international YYYY-MM-JJ.
- Un site Web, avec ou sans le protocole (lemonde.fr et https://lemonde.fr).

Paramètres facultatifs :

- -f, --focus <n> : l'empan de la fenêtre *focus*.
- -w, --window <n> : l'empan de la fenêtre *window*.

Exemple avec -f 0 (la partie *focus* sera la date donnée ± 0 jour) et -w 2 (la partie *window* sera la date donnée ± 2 jours, *moins* la partie *focus*), effectué le **19** décembre :

```
python3 projet.py 2023-12-18 lemonde.fr -f 0 -w 2
En cache: https://lemonde.fr  # Version du 20231219
En cache: https://web.archive.org/web/20231218/lemonde.fr
En cache: https://web.archive.org/web/20231217/lemonde.fr
En cache: https://web.archive.org/web/20231216/lemonde.fr
Moins spécifiques
   « -1.592559294751373
   mémorable -1.5228450163853648
   jour -1.1473422381980702
   meilleur -0.879414882565196
   minute -0.7608655090175512
   monde -0.7128780024827371
   entretien -0.591596865620891
   culture -0.5839182781499896
   société -0.5692168089866357
   goût -0.537759275128396
Plus spécifiques
   philippe 1.0379254878859407
   l’ 1.1436185633491927
   projet 1.2748272148185433
   coup 1.4173322288969696
   cmp 1.7281484071276634
   noël 1.8903212428543728
   décembre 1.8903212428543728
   ministre 1.8903212428543728
   cours 2.256873979291057
   immigration 2.3705875463889408
```

Vous pouvez organiser votre programme comme vous voulez (c’est même un des buts du sujet, qui sera évalué), mais vous devez avoir au moins une fonction `get_url(url)` et une fonction `specif(f, F, t, T)`.

Comment bien aborder chaque partie

> Les instructions sont notées avec une barre à gauche, comme ceci. Elles font partie de l’évaluation.

Les conseils sont là pour vous aider, vous faites comme vous voulez. Ils sont notés sans barre à gauche, comme ceci.

1. Paramétrage

> Utilisez la bibliothèque *argparse*. Votre script doit fonctionner avec une URL avec ou sans le protocole : ex. lemonde.fr et https://lemonde.fr .
>
> Votre script doit vérifier que la date est bien une date au format nnnn-nn-nn, que l’URL contient bien un point, et que les paramètres *focus* et *window* (le cas échéant) sont bien des entiers.

Notez qu’une fois votre *parser* construit, vous pouvez le convertir en dictionnaire avec :

- `my_dict = vars(parser.parse_args())`

2. Manipulation de dates

> Ne manipulez pas les dates directement comme des chaînes de caractères ou des entiers ! Utilisez la bibliothèque datetime, en particulier la classe timedelta pour vous déplacer dans le temps.

```
import datetime
date_dt = datetime.date(1999, 12, 31)
my_date = date_dt – datetime.timedelta(days=-1)  # my_date est 1999-12-30
my_date = date_dt – datetime.timedelta(days=1)   # my_date est 2000-01-01
today_dt = datetime.datetime.today()  # Aujourd'hui
print(my_date < datetime.datetime.today())  # Vrai si my_date est dans le passé
my_date.day    # Le jour
my_date.month  # Le mois
my_date.year   # L'année
```

Notez que les numéros de jour, mois et année sont des entiers, mais les téléchargements sur *Web archive* se font avec des dates au format YYYYMMJJ, et qu'il faudra peut-être ajouter des zéros (20240101 et non 202411 pour le 1er janvier 2024).

3. Téléchargement

> Utilisez la bibliothèque *requests*. Les sites populaires sont archivés par *Web Archive* et accessibles par des URLs du type : https://web.archive.org/web/20231218/lemonde.fr où la date est en rouge et l'URL du site en bleu. Vous pouvez inclure le protocole dans le chemin : https://web.archive.org/web/20231218/**https://**lemonde.fr est valide.
>
> Pour cette partie, vous **devez** créer une fonction `get_url(url)`. Aucun téléchargement ne doit avoir lieu en dehors de cette fonction.
>
> Cette fonction doit gérer un cache sous forme d'un sous-dossier *cache*. À chaque appel, elle teste si elle a déjà téléchargé *url* :

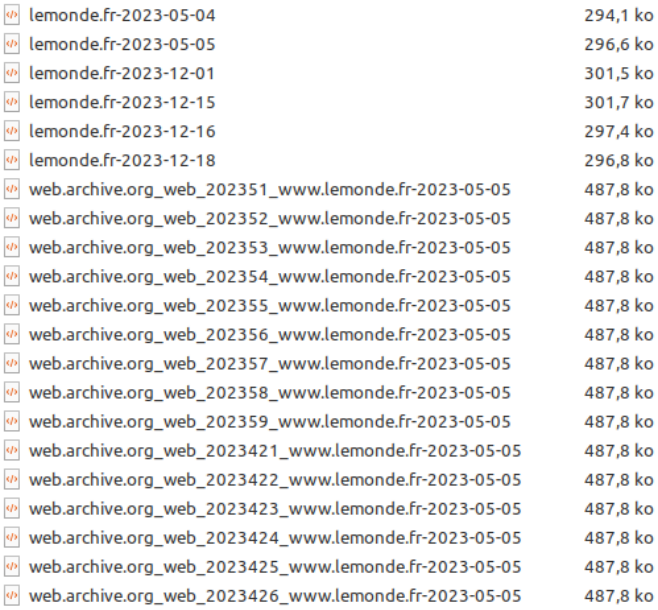
- Si ce n'est pas le cas (= le dossier *cache* ne contient pas de fichier pour *url*), *url* est téléchargé avec la bibliothèque *requests*, sauvegardé dans le cache, puis retourné.
- Si c'est le cas (= le dossier *cache* contient un fichier pour *url*), on ne télécharge rien, mais on retourne à la place le contenu du fichier mis en cache.

> **Important.** Le site *Web Archive* est limité à 15 requêtes par seconde (un requête toutes les 4 secondes). Votre fonction `get_url` devra donc marquer une pause de 4 secondes après chaque téléchargement (pour tous les sites, y compris en dehors de *Web Archive*) :

- import time
  time.sleep(4)

Pour créer votre cache, vous pouvez par exemple créer des fichiers sur le *pattern* url_date, mais faites attention de ne pas inclure de caractères « / » dans vos noms de fichiers, même si l'url passée en paramètre à votre fonction en contient.

Exemple de dossier cache :



> Le script devra afficher une ligne de log à chaque appel à la fonction get_url, qui indiquera « En cache : <url> » si la page a été trouvée dans le cache, ou « Téléchargement : <url> » si il a fallu la télécharger. Voir les exemples en début de sujet.
>
> En cas d'erreur de téléchargement, le script ne devra pas planter, mais afficher un message « Impossible de télécharger <url> ».

4. Nettoyage

> Chaque fichier HTML téléchargé doit être « nettoyé » avant d'être utilisé. Le but est de n'obtenir que le contenu textuel.

N'essayez pas de traiter un fichier HTML comme un fichier XML ; en général ça ne marche pas. Traitez le comme une grosse chaîne de caractère (multiligne). En général, vous pouvez extraire le gros du contenu textuel en faisant des chercher/remplacer à base de regex :

- Tout mettre sur une ligne.
- Enlever les balises *script*, *noscript*, *head* et tout leur contenu. Attention, les balises peuvent avoir des attributs ; n'effacez pas juste tout ce qui est entre <script> et </script> ; il peut y avoir des balises <script untruc="machin"> … </script>. Pensez aussi à utiliser les quantifieurs paresseux ; à ce sujet voir https://stackoverflow.com/questions/2301285/what-do-lazy-and-greedy-mean-in-the-context-of-regular-expressions.
- Enlever tout ce qui se trouve avant `<!-- END WAYBACK TOOLBAR INSERT -->`
- Enlever les commentaires HTML : <!-- … -->.
- Enlever toutes les balises.
- Enlever les espaces multiples.

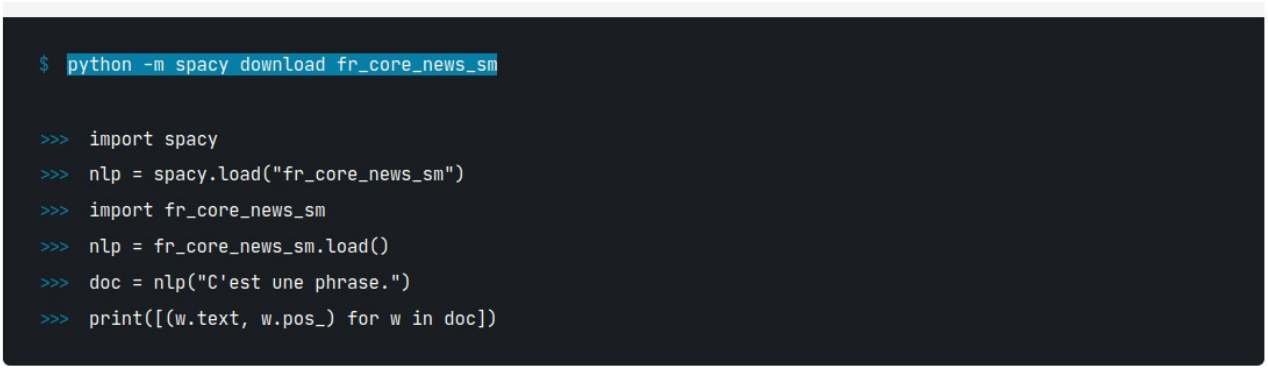
… mais vous aurez peut être à adapter ça en fonction de votre site à surveiller.

5. Analyse *Spacy*

> Indiquez dans un commentaire en début de script quel modèle il faut télécharger. Par exemple :

```
#   Installer le modèle de langue pour Spacy
#     python -m spacy download fr_core_news_sm
```

C’est indiqué au début de la section Quickstart de la doc (surligné en bleu ci-dessous).



```
$ python -m spacy download fr_core_news_sm

>>> import spacy
>>> nlp = spacy.load("fr_core_news_sm")
>>> import fr_core_news_sm
>>> nlp = fr_core_news_sm.load()
>>> doc = nlp("C'est une phrase.")
>>> print([(w.text, w.pos_) for w in doc])
```

Normalement, vous avez juste à reprendre la partie *Quickstart* ici: https://spacy.io/usage/models#quickstart

Attention, le chargement d’un modèle, du genre `nlp = spacy.load("fr_core_news_sm")` est très lent. Effectuez-le une fois pour toute en début de script, et pas dans une boucle !

> Une fois l’analyse faite, vous avez besoin de récupérer les fréquences de chaque lemme nominal (partie du discours NOUN ou PROPN).

6. Calcul de spécificités avec *R*

> On calcule des spécificités de Lafon avec la bibliothèque R *textometry.*
>
> Cette partie implique d’installer R, et le package *textometry* (). Installation du paquet *textometry* dans le dossier courant (chemin « . », paramétrable):
>
> R
> `install.packages('textometry', lib='.')`
> `quit()`
>
> Dans votre script Python, le calcul des spécificités **doit** être effectué par une fonction specif(f, F, t, T) qui retourne le score de spécificité. Cette fonction appellera la commande R spécifiée ci-dessous avec la fonction check_output de la bibliothèque subprocess.

Au sein d’un corpus textuel, le calcul de spécificité sert à déterminer si un mot est significativement plus ou moins fréquent dans un texte donné, par rapport à sa fréquence dans le reste du corpus.

Dans le cadre de ce projet, le texte donné est constitué de la fenêtre ***focus***. Le reste du corpus est constitué de la fenêtre ***window*** (= tous les textes qui ne sont pas dans *focus*). Pour chaque lemme nominal du texte donné (focus), on doit calculer sa spécificté par rapport à se fréquence dans le reste du corpus (window).

Le calcul de spécificité s'effectue avec 4 paramètres :

- partie *focus*
  - ***f*** est la fréquence absolue (= nombre d'occurrences) du mot recherché (= chaque lemme nominal) dans la partie *focus*.
  - ***F*** est la taille totale (= nombre de mots) de la partie *focus*.
- Partie *window*
  - ***t*** est la fréquence absolue (= nombre d'occurrences) du mot recherché dans la partie *window*.
  - ***T*** est la taille totale (= nombre de mots) de la partie *window*.

Notez que F < T, sinon ça plante…

Ci-dessous, vous trouverez un exemple d'utilisation en ligne de commande, sur une seule ligne. Remplacez f, F, t, et T par leurs valeurs respectives. Notez que f est utilisé à deux endroits.

```
R --vanilla -s -e 'library("textometry", lib="."); res <-
specificities.distribution.plot(f,F,t,T); print(res["mode"]); print(res["pfsum"]
[[1]][[f+1]]);'
```

La commande retourne 4 lignes :

1. `$mode`
2. `[1] <mode>`
3. *une ligne vide*
4. `[1] <proba>`

Exemple 1

```
R --vanilla -s -e 'library("textometry", lib="."); res <-
specificities.distribution.plot(34,16176,50,61197); print(res["mode"]);
print(res["pfsum"][[1]][[34+1]]);'
$mode
[1] 13.48105      # Ligne 2, le mode

[1] 9.5692e-10    # Ligne 4, la proba
```

Exemple 2

```
R --vanilla -s -e 'library("textometry", lib="."); res <-
specificities.distribution.plot(203,16176,1181,61197); print(res["mode"]);
print(res["pfsum"][[1]][[203+1]]);'
$mode
[1] 312.4432            # Ligne 2, le mode

[1] -1.996067e-14       # Ligne 4, la proba
```

Traitement de la sortie du script R pour obenir le score de spécificité

Le script R retourne 4 lignes ; on a besoin de récupérer le mode (ligne 2) et la proba (ligne 4). Ces lignes sont toujours précédées d'un [1] dont il ne faut pas tenir compte.

Le score de spécificité est obtenu en calculant :

- si *mode* <= f :
  - abs(log10(abs(*proba*)))
- sinon :
  - - abs(log10(abs(*proba*)))

Ainsi, dans l'exemple 1 (f=34 ; F=16176; t=50 ; T=61997), on a :

- mode = 13.48105 et f =34
  - donc mode ≤ f
    - donc spécificité = abs(log10(abs(*proba*)))
      - soit spécificité = abs(log10(abs(9.5692e-10))) = math.fabs(math.log10(math.fabs(float("9.5692e-10")))) = 9.019124368399561

Dans l'exepmple 2 (f=203 ; F=16176; t=50 ; T=61997), on a :

- mode = 312.4432 et f =203
  - donc mode > f
    - donc spécificité = -abs(log10(abs(*proba*)))
      - soit spécificité = -abs(log10(abs(-1.996067e-14))) = -math.fabs(math.log10(math.fabs(float("-1.996067e-14")))) = -13.699824885272127

Utilisez la bibliothèque *math* pour les calculs (`math.log10(n)` et `math.fabs(n)`). En Python, une chaîne de caractères comme `"-1.996067e-14"` peut se convertir en *float* simplement en faisant `float("-1.996067e-14") .`

Mise en cache

> Le calcul de spécificité prend longtemps. Comme pour les téléchargements, les résultats seront donc mis en cache dans des fichiers placés dans le dossier *cache*. Par exemple :

… les 4 chiffres à la fin de chaque fichier sont les valeurs des paramètres f, F, t et T.

Ce cache devrait grossir de quelques centaines de fichiers à chaque fois qu'on appelle le script avec des paramètres différents. Idéalement, il faudrait le nettoyer régulièrement (de même que le cache de téléchargement utilsé en partie 3), mais ce n'est pas demandé dans le cadre du projet.

> En cas d'erreur lors du calcul, votre script doit afficher « Erreur en calculant la spécif pour <f>, <F>, <t>, <T> ».

En cas de problème pour installer R...

Si vous n’arrivez pas à installer R ou le paquet textometry, vous pouvez utiliser à la place la commande suivante (ici pouf f=1 ; F=10 ;t=100 ;T=1000) ; adaptez la commande suivant vos besoins :

```
wget -q -O - "https://pro.aiakide.net/r-script/?f=1&F=10&t=100&T=1000"
```

Si vous utilisez cette alternative, vous *devez* passer par la commande wget, vous ne pouvez pas juste récupérer l'URL et l'utiliser directement avec le module request de Python. Le but de cette partie du projet est de vous faire interagir avec la ligne de commande depuis votre script Python.

7. Résultats

> Vous afficherez les 10 lemmes les moins spécifiques et les plus spécifiques de la partie *focus* par rapport à la partie *window*. Voir les exemples en début de sujet.

Vous pourrez notamment vous aider de cette discussion pour trier un dictionnaire par ses valeurs :
https://stackoverflow.com/questions/613183/how-do-i-sort-a-dictionary-by-value
… le résultat est une liste de tuples (clé, valeur) triés par valeur.

Si c’est trop compliqué, vous pouvez aussi :

1. afficher tous les scores de spécificité dans le désordre.
2. afficher seulement le lemme le plus spécifique et le lemme le moins spécifique (au lieu des 10 lemmes les plus/moins spécifiques).

… évidemment, ça vous fera moins de points (l’option 2 rapporte plus de points que la 1).

Comment rendre le projet ?

À cette adresse :

https://kdrive.infomaniak.com/app/collaborate/267819/2d51a0a3-6261-45f5-9a13-3c520d86c7ea

Indiquez votre nom et prénom quand le site de dépôt les demandera. Vous pouvez indiquer n’importe quelle adresse mail, même une fausse, il n’y a pas de vérification. Vous pouvez faire plusieurs dépôts.