TD 5

Exercices

Terminer le TD 4 :

1. Tokenisation (espaces et signes de ponctuation sont des tokens − transformez les retours à la ligne en espaces avec regex.sub(regex, replacement, str)). Utilisez html.unescape(string) pour résoudre les entités XML.
2. Fonction is_ponct
3. Fonction is_num
4. Fonction is_greek
5. Quels sont les tokens les plus fréquents du texte, hors espaces, ponctuation, nombres, et mots grecs ?
6. Vous êtes en avance ? Essayez le « pour aller plus loin » du TD 4.
7. Fonction is_lemma, is_author et is_normalized_classification
8. Quels sont les tokens les plus fréquents (hors espaces, ponctuation, nombres, et mots grecs) par lemma et par normalized_classification, en ignorant les 100 tokens les plus fréquents du texte ?

Unicode

Unicode − fonctions utiles

```
>>> hex(int('01100000000', 2))
'0x300'

>>> unicodedata.name('\u0300')
'COMBINING GRAVE ACCENT'

>>> unicodedata.name('à')
'COMBINING GRAVE ACCENT'

>>> unicodedata.lookup('LEFT CURLY BRACKET')
'{'

>>> unicodedata.category('A')  # 'L'etter, 'u'ppercase
'Lu'

>>> unicodedata.normalize('NFC', '\u0061\u0301')
'á'

>>> unicodedata.normalize('NFD', '\u00e1')
'á'
```

On peut convertir en notation ASCII pour débuguer :
>>> print(ascii(unicodedata.normalize('NFD', '\u00e1')))
'a\u0301'

Cf. https://fr.wikipedia.org/wiki/Normalisation_Unicode pour + de détails sur la normalisation.

1. NFC is the general common sense form that you should use, `ä` is 1 code point there and that makes sense.
2. NFD is good for certain internal processing - if you want to make accent-insensitive searches or sorting, having your string in NFD makes it much easier and faster. Another usage is making more robust slug titles. These are just the most obvious ones, I am sure there are plenty of more uses.

https://stackoverflow.com/questions/15985888/when-to-use-unicode-normalization-forms-nfc-and-nfd

https://en.wikipedia.org/wiki/Diacritic
https://en.wikipedia.org/wiki/Devanagari

Exercice

Expliquez les résultats du script chars.py .