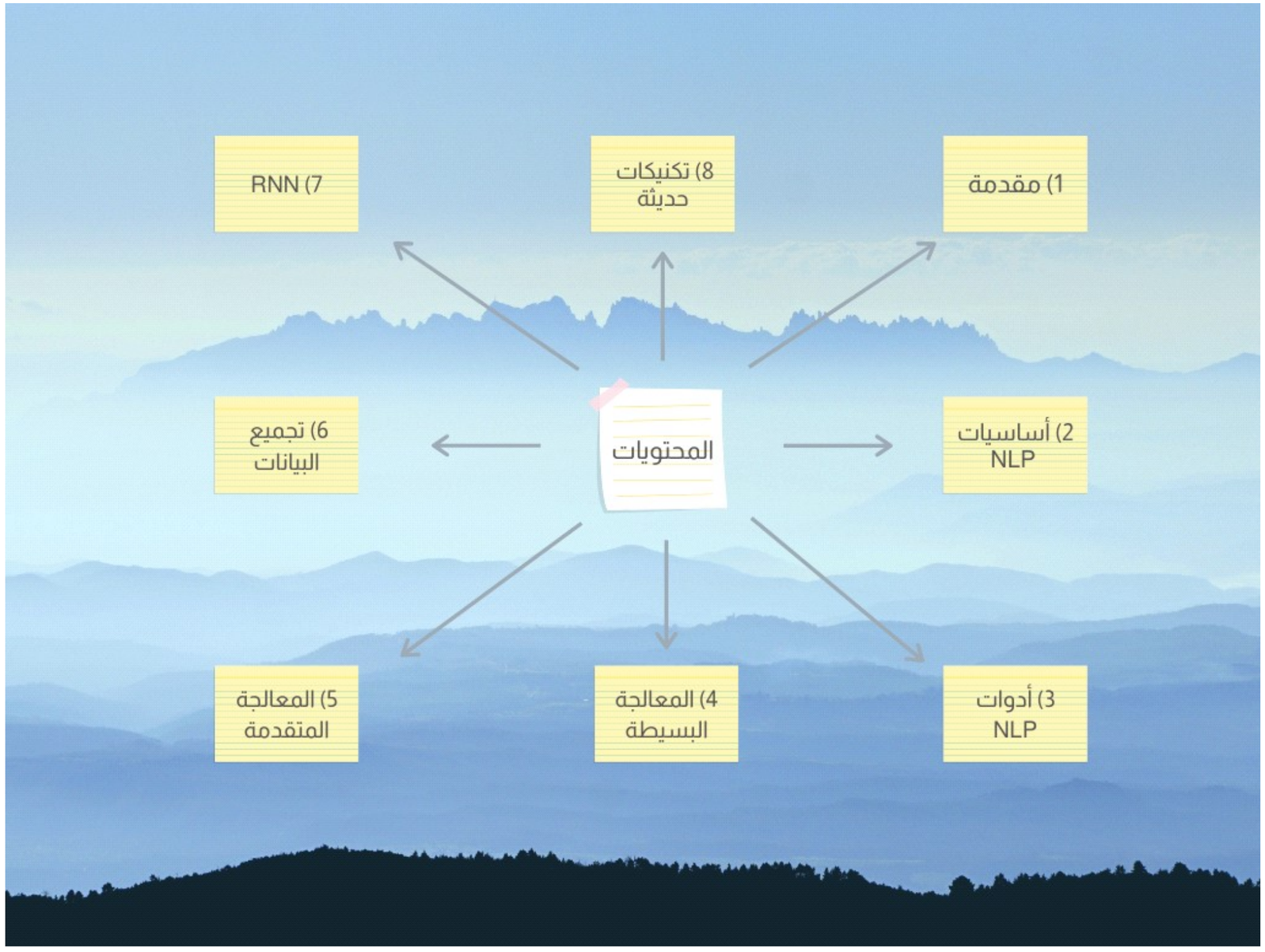
Natural Language Processing المعالجة اللغوية الطبيعية



المحتويات

القسم الثالث : أدوات NLP

الجزء الرابع : Stemming & Lemmatization

==========

مصادر الكلمات Stemming & Lemmatization

وهي الاداة التي تسمح بتجريد اي كلمة من جميع الاضافة التي فيها , و العودة للمصدر الاصلي لها

و هي معتمدة علي فكرة إرجاع الكلمة الي اصلها , وحذف جميع الاضافات عليها سواء في البداية او النهاية , فكلمات مثل (playing , player , plays , played) كلها تعود لكلمة play

و هي مفيدة بشكل كبير في لتعرف علي معني الكلمات , وكذلك في ضم جميع الكلمات ذات اصل واحد الي نفس الكلمة , فلو تكررت كلمات كلها تدور في فلك كلمة win بأشكال كثيرة , فيجب حساب عدد الكلي لها بعد ارجاعها لأصلها

فكلمات مثل (كتاب , كتب , مكتبة , كاتب , كتبة , مكتوب , كتاتيب) , كلها من الاصل (كتب) , ونفس الامر في اغلب اللغات

كذلك جملتي : (I was taking a ride in the car.) (I was riding in the car.) بنفس المعني حتي لو اختلفت الكلمات

لكن احيانا تفشل اداة stemming في ايجاد جذر الكلمة , وتقوم بحذف حرف متحرك في النهاية , بينما هو حرف اصلي since→ sinc

و مكتبة سباسي لا تدعم خاصة الـ stemming لانها تدعم خاصية مشابهة و هي lemmatization لذا لنتعامل معها من مكتبة NLTK

والـ lemmatization هي مشابهة للـ stemming في الفكرة لكنها اكثر قوة و فعالية , فهي لا تكتفي بازالة الحروف الزائدة في الكلمات , ولكن بالبحث في معناها و اساسها , فكلمات been , was , اصلها هو be , و هكذا

كما انه يراعي المعني في الجملة فكلمة meeting قد يكون اصلها meet لو كانت فعل مضارع , وقد يكون اصلها هو نفسه meeting في حالة كانت اسم و ليس فعل (بمعني اجتماع)

_

وتقوم الأداة بإرجاع كل كلمة إلي اصلها , مع حذف اي اضافات عليها سواء سابقة او لاحقة , وهي ما تتم في خطوتي stemming & lemmatization

فنبدأ بخطوة الـ normalization و هي تشمل حذف النقاط الاضافية لتتحول كلمة مثل U.S.A الي USA

و كذلك مراعاة حرف s الجمع و الكابيتال و السمول لدي البحث

Need to “normalize” terms

- Information Retrieval: indexed text & query terms must have same form.
  - We want to match ***U.S.A.*** and ***USA***

We implicitly define equivalence classes of terms

- e.g., deleting periods in a term

Alternative: asymmetric expansion:

- Enter: ***window*** Search: ***window, windows***
- Enter: ***windows*** Search: ***Windows, windows, window***
- Enter: ***Windows*** Search: ***Windows***

Potentially more powerful, but less efficient

الخطوة التالية هي case folding و التي تشمل عمل جميع الكلمات small مع استثناءات , فكلمة FED تعني Federal Reserve System اما fed فهي ماضي feed

كذلك sail و SAIL او us و US

Applications like IR: reduce all letters to lower case

- Since users tend to use lower case
- Possible exception: upper case in mid-sentence?
  - e.g., ***General Motors***
  - ***Fed*** vs. ***fed***
  - ***SAIL*** vs. ***sail***

For sentiment analysis, MT, Information extraction

- Case is helpful (***US*** versus ***us*** is important)

يليها lemmatization و هي ارجاع الكلمة لاصلها :

Reduce inflections or variant forms to base form

- *am, are, is → be*
- *car, cars, car's, cars' → car*

the boy's cars are different colors → the boy car be different color

Lemmatization: have to find correct dictionary headword form

Machine translation

- Spanish quiero ('I want'), quieres ('you want') same lemma as querer 'want'

مع معلومية ان اصل الكلمة يسمي stem بينما الاضافات اسمي affixes

_

و باستخدام خوارزم porter يمكن تطبيق هذا الامر علي بعض الكلمات , منها ما ينجح و منها ما يفشل

Step 1a

sses	→	ss	caresses	→	caress
ies	→	i	ponies	→	poni
ss	→	ss	caress	→	caress
s	→	ø	cats	→	cat

Step 1b

(*v*)ing	→	ø	walking	→	walk
			sing	→	sing
(*v*)ed	→	ø	plastered	→	plaster

Step 2 (for long stems)

ational	→	ate	relational	→	relate
izer	→	ize	digitizer	→	digitize
ator	→	ate	operator	→	operate
…					

Step 3 (for longer stems)

al	→	ø	revival	→	reviv
able	→	ø	adjustable	→	adjust
ate	→	ø	activate	→	activ

و يجب الاهتمام بنوع الكلمات ,فكلمة sing فيها ال ing من اساس الكلمة بينما walking فيها ing زائدة

```
(*v*)ing → ø  walking    → walk
              sing       → sing
```

و نري ان بعض الكلمات بها ing اساسية و اخري مضافة :

```
(*v*)ing → ø  walking    → walk
              sing       → sing
```

```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep 'ing$' | sort | uniq -c | sort –nr
```

```
   1312 King              548 being
    548 being             541 nothing
    541 nothing           152 something
    388 king              145 coming
    375 bring             130 morning
    358 thing             122 having
    307 ring              120 living
    152 something         117 loving
    145 coming            116 Being
    130 morning           102 going
```

```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep '[aeiou].*ing$' | sort | uniq -c | sort –nr
```

```
   37
```

كما ان الموضوع يكون اصعب في عدد من اللغات ذات الاضافات العديدة مثل التركية و الالمانية

Some languages requires complex morpheme segmentation

- Turkish
- Uygarlastiramadiklarimizdanmissinizcasina
- `(behaving) as if you are among those whom we could not civilize'
- Uygar `civilized' + las `become'

  + tir `cause' + ama `not able'
  + dik `past' + lar 'plural'
  + imiz 'p1pl' + dan 'abl'
  + mis 'past' + siniz '2pl' + casina 'as if'

_

و نبدأ باستدعاء المكتبة و عمل الاوبجكت

```
import nltk

from nltk.stem.porter import *
p_stemmer = PorterStemmer()
```

و هنا نقوم باعطائه كلمات مختلفة و متشابهة ,و نقوم باستخدام الميثود stem لتحليلها

```
words = ['run','runner','running','ran','runs','easily','fairly']
for word in words:
    print(word+' --> '+p_stemmer.stem(word))
```

مع ملاحظة ان كل النتائج ليست دقيقة , فلا يوجد شئ اسمه easili or fairli

-

و يمكن استخدام snowball هكذا

```
from nltk.stem.snowball import SnowballStemmer
s_stemmer = SnowballStemmer(language='english')

words = ['run','runner','running','ran','runs','easily','fairly']

for word in words:
    print(word+' --> '+s_stemmer.stem(word))
```

فتكون النتيجة افضل

```
words = ['generous','generation','generously','generate']

for word in words:
    print(word+' --> '+s_stemmer.stem(word))
```

و هنا نستخدم nltk

```
from nltk.stem import PorterStemmer , LancasterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

ps = PorterStemmer()
ls =  LancasterStemmer()
```

مع الكلمات

```
words = ["is","was","be","been","are","were"]
```

نستخدم النوع الاول

```
for w in words:
    print(ps.stem(w))
```

و الثاني

```
for w in words:
    print(ls.stem(w))
```

مثال اخر

```
words = ["book","booking","booked","books","booker","bookstore"]

for w in words:
    print(ps.stem(w))


for w in words:
    print(ls.stem(w))
```

مثال اخر

```
sentence = 'had you booked the air booking yet ? if not try to book it ASAP since
booking will be out of books'

words = word_tokenize(sentence)

for w in words:
    print(ps.stem(w))
```

مثال اخر

```
words = word_tokenize(sentence)
```

```
for w in words:
    print(ls.stem(w))
```

عرض تفاصيل اكثر

```
word_list = ["friend", "friendship", "friends",
"friendships","stabil","destabilize","misunderstanding","railroad","moonlight","football"]
print("{0:20}{1:20}{2:20}".format("Word","Porter Stemmer","lancaster Stemmer"))
for word in word_list:
    print("{0:20}{1:20}{2:20}".format(word,ps.stem(word),ls.stem(word)))
```

```
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
```

الان مع استخدام الـ lemmatization

```
import spacy
nlp = spacy.load('en_core_web_sm')

doc1 = nlp(u"I am a runner running in a race because I love to run since I ran today")

for token in doc1:
    print(token.text, '\t', token.pos_, '\t', token.lemma, '\t', token.lemma_)
```

```
def show_lemmas(text):
    for token in text:
        print(f'{token.text:{12}} {token.pos_:{6}} {token.lemma:<{22}} {token.lemma_}')

doc2 = nlp(u"I saw eighteen mice today!")

show_lemmas(doc2)

doc3 = nlp(u"I am meeting him tomorrow at the meeting.")

show_lemmas(doc3)
```

و مع مكتبة nltk

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

words = ["cats","cacti","radii","feet","speech",'runner']
```

```
for word in words :
    print(lemmatizer.lemmatize(word))
```

و يتم تحديد الكلمة هل اي اسم ام فعل

```
print(lemmatizer.lemmatize("meeting", "n"))
print(lemmatizer.lemmatize("meeting",'v'))

import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

sentence = "He was running and eating at same time. He has bad habit of swimming
after playing long hours in the Sun."
punctuations="?:!.,;"
sentence_words = nltk.word_tokenize(sentence)
for word in sentence_words:
    if word in punctuations:
        sentence_words.remove(word)

sentence_words
```

```
print("{0:20}{1:20}".format("Word","Lemma"))
for word in sentence_words:
    print ("{0:20}{1:20}".format(word,wordnet_lemmatizer.lemmatize(word)))

for word in sentence_words:
    print ("{0:20}{1:20}".format(word,wordnet_lemmatizer.lemmatize(word, pos="v")))

words = ["is","was","be","been","are","were"]

for word in words :
    print(lemmatizer.lemmatize(word))

words = ["is","was","be","been","are","were"]
for word in words :
    print(lemmatizer.lemmatize(word,'v'))
words = ["feet","radii","men","children","carpenter","fighter"]
for word in words :
    print(lemmatizer.lemmatize(word,'n'))
```

و ايضا ادائها في اللغة العربية ضعيف للغاية