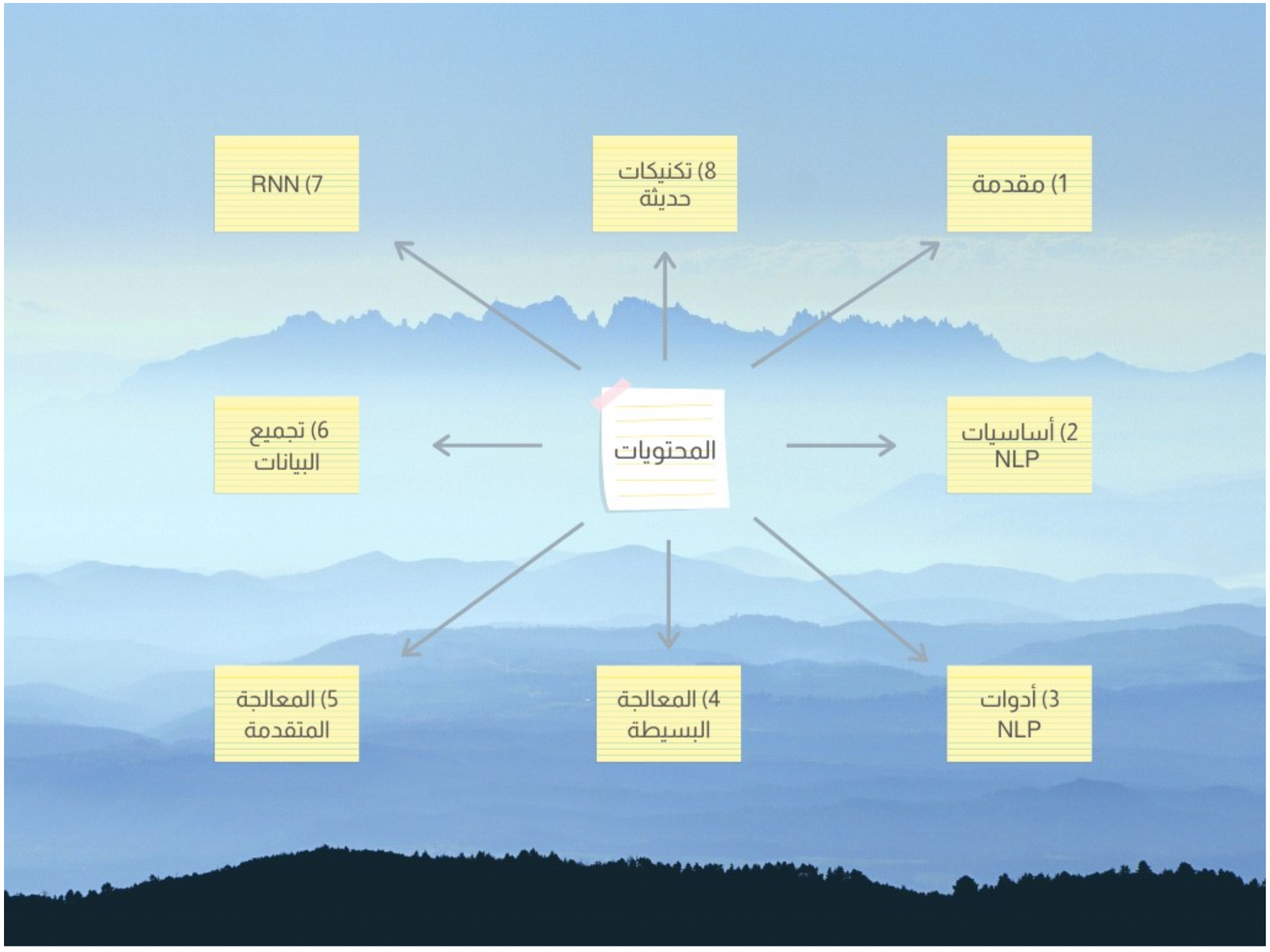
Natural Language Processing المعالجة اللغوية الطبيعية



المحتويات

1) مقدمة	المحتويات	ما هو NLP	تاريخ NLP	العقبات و التحديات	التطبيقات				
2) أساسيات NLP	المكتبات	الملفات النصية	ملفات pdf	البحث في النصوص					
3) أدوات NLP	Tokenization	Sent. Segm.	POS	Stem & Lemm	NER	Stopwords	Matchers	Syntactic Struc.	T.Visualization
4)المعالجة البسيطة	Word Meaning	Word embed	T. Vectors	Word2Vec	BOW	TF-IDF	Text Similarity	Dist. Similarity	
5)المعاجلة المتقدمة	T. Classification	T. Clustering	LDA	NMF	L. Modeling	GloVe	Lexicons	NGrams	T. Generation
	Sent. Analysis	Naïve Bayes	Vader	Auto Correct	Ans. Questions		Summarization & Snippets		
6)تجميع البيانات	Tweet Collecting	Data Scraping	Information Extraction		Information Retrieval		Relative Extraction		Search Engine
RNN (7	Seq to Seq	LSTM	GRU	Rec NN\TNN					
8)تكنيكات حديثة	Word Cloud	CNN	T. Forcing	Attention Model	Transformer	Bert	FastText	Gensim	Chat Bot

القسم الثالث : أدوات NLP

الجزء الثاني : Sentence Segmentation

==========

تقسيم الجمل Sentence Segmentations

وهي الخاصة بتقسيم الكلام كله الي جمل حسب بدايتها و السياق

تقسيم الجمل إلي أجزاء . .

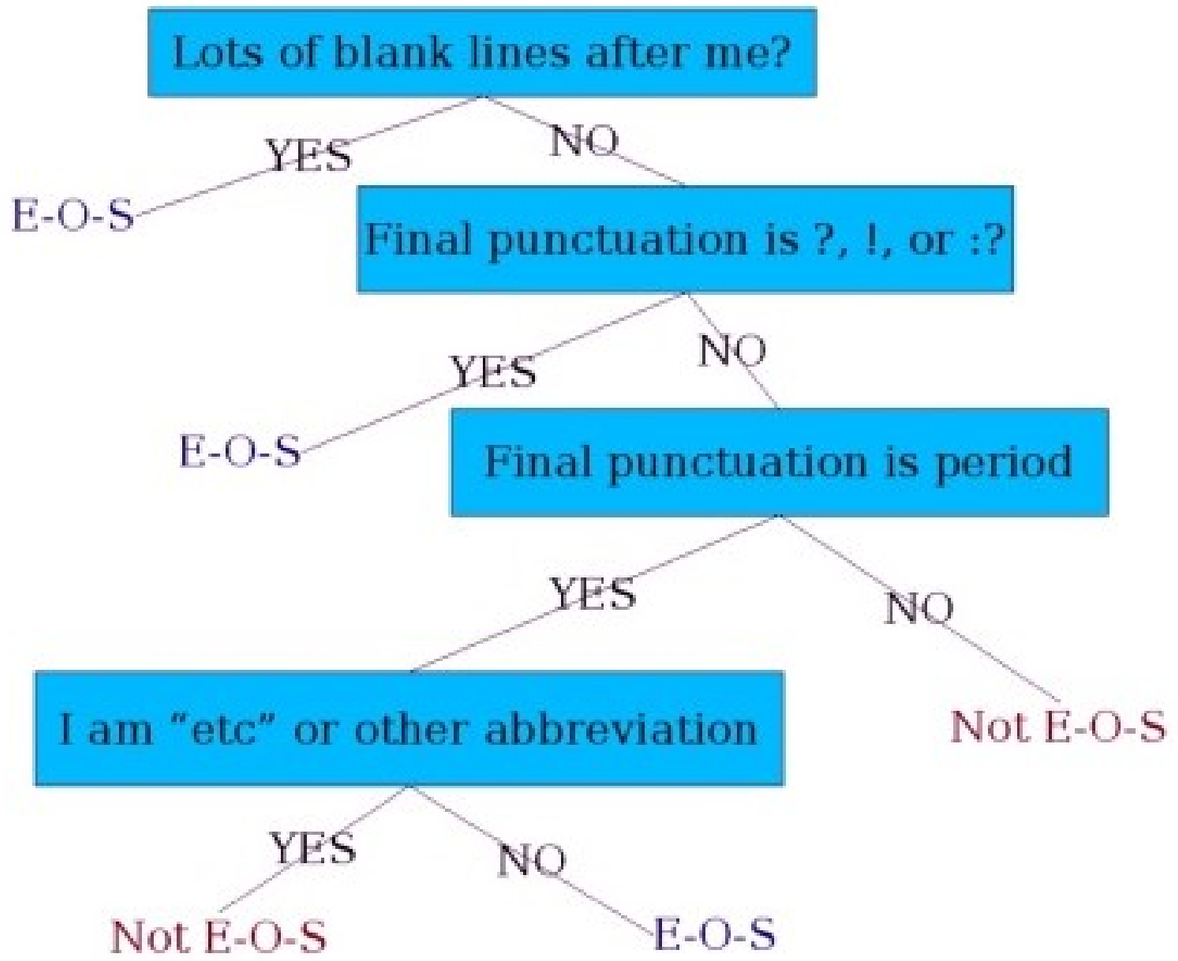
و هي عملية مهمة للغاية , و قد تتم بأحد طريقتين

اما بعلامة واضحة انها لنهاية جملة , مثل ؟ أو ! أو ,

و أحيانا تتم عبر استخدام علامة غير واضحة مثل النقطة . والتي قد يكون لها استخدامات اخري مثل نهاية اختصار .Dr , او ارقام عشرية و هكذا

لذا يتم احيانا استخدام نماذج ML لمعرفة هل هذه الـ . لنهاية الجملة ام لاستخدام آخر

و يمكن أن يتم عمل خوارزم ML بشكل يدوي , لمعرفة هل هذه نهاية الجملة EOS ام لا



كما ان النظر لحالة الحروف قبل و بعد النقطة تدل هل هي EOS ام لا, وايضا طول الكلمة بعدها

- Case of word with ".": Upper, Lower, Cap, Number
- Case of word after ".": Upper, Lower, Cap, Number

- Numeric features
  - Length of word with "."
  - Probability(word with "." occurs at end-of-s)
  - Probability(word after "." occurs at beginning-of-s)

_

و نقوم بتقسيم النص الي جمل , عبر اداة sents التي تقوم بتقسيم الجملة الطويلة الي جمل محددة

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

doc1 = nlp(u'This is the first sentence. This is another sentence. This is the last sentence.')

for sent in doc1.sents:
    print(sent)
```

كذلك يمكن معرفة اذا كانت كلمة محددة هل هي بداية الجملة ام لا هكذا :

```
doc1[6].is_sent_start
```

و اذا اردنا استخدام جملة معينة فلا يصح كتابة هكذا :

```
#print(doc1.sents[0])
```

لانها ليست list, و لكن هكذا

```
list(doc1.sents)[0]
```

او هكذا

```
doc_sents = [sent for sent in doc1.sents]
doc_sents
```

و يمكن فحص كل كلمة علي حدة لتحديد ايهما بداية الجملة هكذا :

```
print(doc_sents[1].start, doc_sents[1].end)
```

```
doc2 = nlp(u'This is a sentence. This is a sentence. This is a sentence.')

for token in doc2:
    print(token.is_sent_start, ' '+token.text)
```

-

كذلك يمكن جعل المكتبة تغير من سلوكها في تقسيم الجمل , فلو كان لدينا جملة هنا

```
doc3 = nlp(u'"Management is doing things right; leadership is doing the right things." -
Peter Drucker')

for sent in doc3.sents:
    print(sent)
```

ستعطينا تقسيم غير مناسب , فنحن نريد ان يتم تقسيمها علي اساس السيمي كولم ;
فيمكن تحديد حرف معين ليقوم التقسيم من خلاله هكذا :

```
def set_custom_boundaries(doc):
    for token in doc[:-1]:
        if token.text == ';':
            doc[token.i+1].is_sent_start = True
    return doc

nlp.add_pipe(set_custom_boundaries, before='parser')
```

```
nlp.pipe_names

doc4 = nlp(u'"Management is doing things right; leadership is doing the right things." -
Peter Drucker')

for sent in doc4.sents:
    print(sent)
```

-

و هنا يتم استخدام اداة sent_tokenize من مكتبة nltk لنفس المهمة

```
from nltk.tokenize import sent_tokenize

EXAMPLE_TEXT = """
Hello Mr. Smith, how are you doing today? The weather is great,
and Python is awesome. The sky is pinkish-blue. You shouldn't eat cardboard.
"""


EXAMPLE_TEXT = '''
```

Thomas Gradgrind, sir. A man of realities. A man of facts and calculations. A man who proceeds upon the principle that two and two are four, and nothing over, and who is not to be talked into allowing for anything over. Thomas Gradgrind, sir—peremptorily Thomas—Thomas Gradgrind. With a rule and a pair of scales, and the multiplication table always in his pocket, sir, ready to weigh and measure any parcel of human nature, and tell you exactly what it comes to. It is a mere question of figures, a case of simple arithmetic. You might hope to get some other nonsensical belief into the head of George Gradgrind, or Augustus Gradgrind, or John Gradgrind, or Joseph Gradgrind (all supposititious, non-existent persons), but into the head of Thomas Gradgrind—no, sir!
In such terms Mr. Gradgrind always mentally introduced himself, whether to his private circle of acquaintance, or to the public in general. In such terms, no doubt, substituting the words ‘boys and girls,’ for ‘sir,’ Thomas Gradgrind now presented Thomas Gradgrind to the little pitchers before him, who were to be filled so full of facts.
Indeed, as he eagerly sparkled at them from the cellarage before mentioned, he seemed a kind of cannon loaded to the muzzle with facts, and prepared to blow them clean out of the regions of childhood at one discharge. He seemed a galvanizing apparatus, too, charged with a grim mechanical substitute for the tender young imaginations that were to be stormed away.
‘Girl number twenty,’ said Mr. Gradgrind, squarely pointing with his square forefinger, ‘I don’t know that girl. Who is that girl?’

```
'''

for s in sent_tokenize(EXAMPLE_TEXT) :
    print(s)
    print('----------------------')
```

أيضا هناك اداة punksentencetokenizer في nltk لتقسيم الجمل

```
from nltk.tokenize import PunktSentenceTokenizer

custom_sent_tokenizer = PunktSentenceTokenizer(EXAMPLE_TEXT)
tokenized = custom_sent_tokenizer.tokenize(EXAMPLE_TEXT)
tokenized[:10]
```

-

لكن أداء هذه الادوات في اللغة العربية ليس علي ما يرام

```
doc1 = nlp('هذه هي الجملة الأولي .. هذه هي الجملة الثانية , والجملة الثالثة')

for sent in doc1.sents:
    print(sent)

print(doc1[2].is_sent_start)
```

```
#print(doc1.sents[0])

list(doc1.sents)[0]

doc_sents = [sent for sent in doc1.sents]
doc_sents

doc2 = nlp(u'This is a sentence. This is a sentence. This is a sentence.')

for token in doc2:
```

```
    print(token.is_sent_start, ' '+token.text)

doc3 = nlp('هذه هي الجملة الأولي .. هذه هي الجملة الثانية , والجملة الثالثة')

for sent in doc3.sents:
    print(sent)

from nltk.tokenize import sent_tokenize

EXAMPLE_TEXT = '''
```

أبو عبد الله محمد بن موسى الخوارزمي عالم رياضيات وفلك وجغرافيا مسلم. يكنى باسم الخوارزمي وأبي جعفر. قيل أنه ولد حوالي 164هـ 781م (وهو غير مؤكد) وقيل أنه توفي بعد 232 هـ أي (بعد 847م). يعتبر من أوائل علماء الرياضيات المسلمين حيث ساهمت أعماله بدور كبير في تقدم الرياضيات في عصره. اتصل بالخليفة العباسي المأمون وعمل في بيت الحكمة في بغداد وكسب ثقة الخليفة إذ ولاه المأمون بيت الحكمة كما عهد إليه برسم خارطة للأرض عمل فيها أكثر من سبعين جغرافيا. قبل وفاته في 850 م/232 هـ

كان الخوارزمي قد ترك العديد من المؤلفات في علوم الرياضيات والفلك والجغرافيا ومن أهمها كتاب المختصر في حساب الجبر والمقابلة الذي يعد أهم كتبه

```
'''
for s in sent_tokenize(EXAMPLE_TEXT) :
    print(s)
    print('----------------------')
EXAMPLE_TEXT = '''
```
يشكل الذكاء الاصطناعي تحديا والهاما لعلم الفلسفة ؛ لزعمه القدرة على إعادة خلق قدرات العقل البشري

وكمارو يحيي الناس
هل هناك حدود لمدى ذكاء الآلات؟ هل هناك فرق جوهري بين الذكاء البشري والذكاء الاصطناعي؟ وهل يمكن أن يكون للآلة عقل ووعي؟ عدد قليل من أهم الإجابات على هذه الأسئلة ترد أدناه.

"آلات الحساب والذكاء "قانون تورنغ
إذا كان الجهاز يعمل بذكاء يضاهي الإنسان، إذافذكائه يماثل ذكاء الإنسان. تفيد نظرية آلان تورنغ أنه، في نهاية المطاف، لا يسعنا إلا أن نحكم على ذكاء الآلة بناء على أدائها. هذه النظرية تشكل أساسا لاختبار تورنغ.

```
'''
for s in sent_tokenize(EXAMPLE_TEXT) :
    print(s)
    print('----------------------')
```