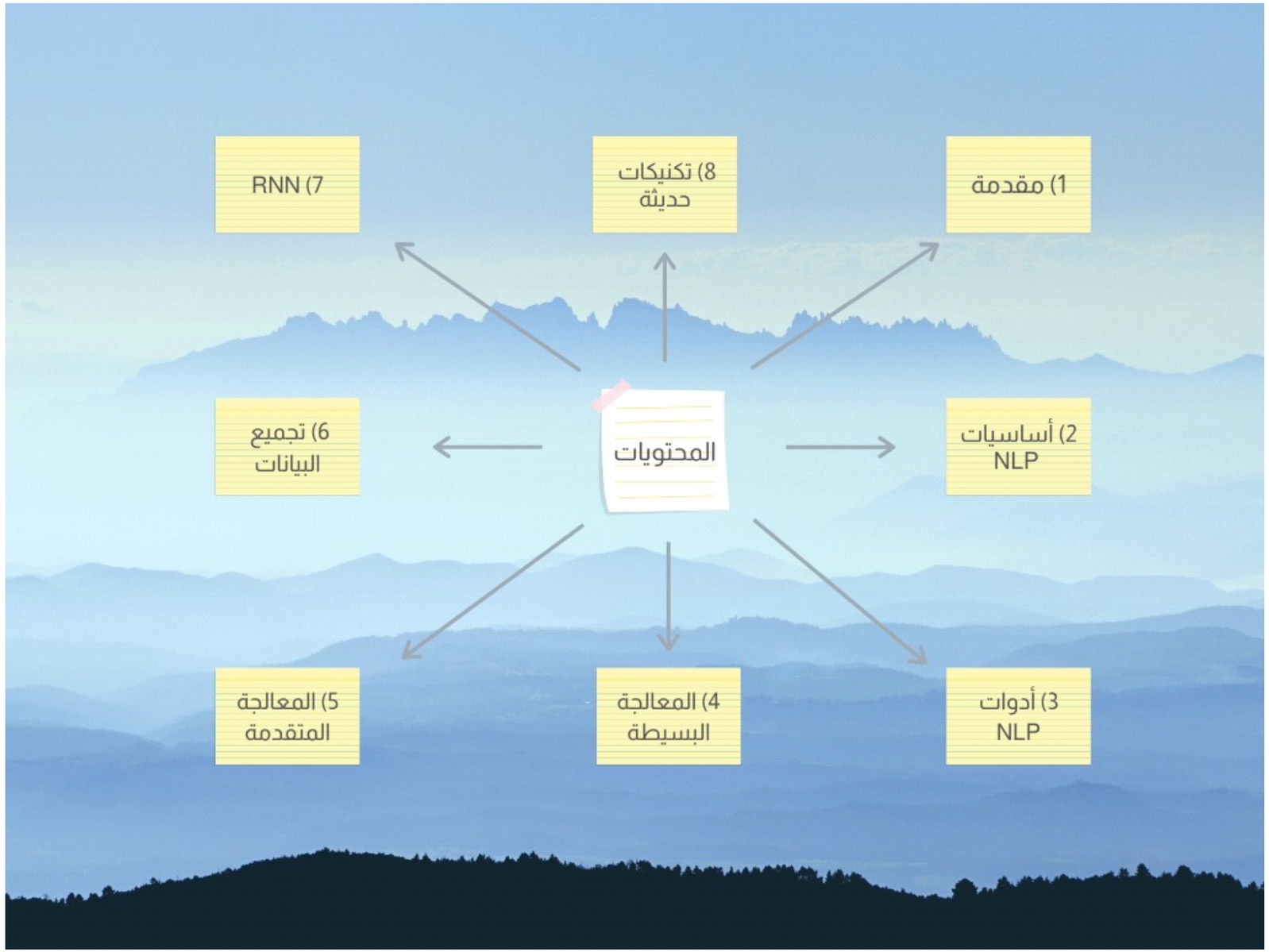
Natural Language Processing المعالجة اللغوية الطبيعية



المحتويات

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

القسم الثالث : أدوات NLP

الجزء السابع : ارتباط الكلمات Matchers

==========

وهي الاداة التي تسمح بربط الكلمات بعضها ببعض , لجعل nlp يدرك اننا نقصد انها بنفس المعني

و هي بالغة الأهمية حينما نبحث عن كلمة معينة يتم كتابتها بأكثر من طريقة (عبدالله , عبد الله , عبد الإله) , او بكلمات بحروف مختلفة لكننا نريد ان نوحدها في كلمة معينة (الذكاء الصناعي , الذكاء الإصطناعي , تعلم الآلة , تعليم الآلة , التعلم العميق)

*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*

و يبدا الامر باستدعاء الاداة المطلوبة من المكتبة

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

فاذا اردنا مثلا ايجاد اي كلمات لـ solar power اذا كانت مكتوبة بكذا صيغة مثل : solar power , solarpower , solar-power

فيتم كتابة 3 نماذج لها , ثم ضمهم معا في امر match

```
from spacy.matcher import Matcher
matcher = Matcher(nlp.vocab)
```

ولا تنس ان امر 'IS_PUNCT': True يعني وجود علامة ترقيم ما بين الكلمتين مثل – او / او * او .

```
pattern1 = [{'LOWER': 'solarpower'}]
pattern2 = [{'LOWER': 'solar'}, {'LOWER': 'power'}]
pattern3 = [{'LOWER': 'solar'}, {'IS_PUNCT': True}, {'LOWER': 'power'}]

matcher.add('SolarPower', None, pattern1, pattern2, pattern3)
```

و هنا في حالة كتابة جملة كبيرة ذكرت فيها الكلمة المطلوبة باكثر من صيغة هكذا

```
doc = nlp(u'The Solar Power industry continues to grow as demand \
for solarpower increases. Solar-power cars are gaining popularity.')
```

و هذا سياتي بكود الكلمة , ثم برقم بدايتها و نهايتها كtokens في الجملة , فيمكن عرض جميع التفاصيل عبر الدالة :

```
found_matches = matcher(doc)
for a,b,c in  found_matches :
    print(f'Word ID {a} , starts at {b} & ends at {c} , and word is {doc[b:c]}')
```

و التي ستاتي بكود الكلمة , ثم الكلمة النهائية , ثم رقم بدايتها و نهايتها , ثم الكلمة الاصلية

كما يمكن حذف باترن معين الامر :

```
matcher.remove('SolarPower')
```

و يمكن جعل علامة الترقيم بين الكلمتين اكثر من علامة ترقيم , فالكود الحالي قد يجد كلمة solar-power لكنه لن يجد solar--power , فيمكن جعل عدد علامات الترقيم مفتوح هكذا :

```
pattern1 = [{'LOWER': 'solarpower'}]
pattern2 = [{'LOWER': 'solar'}, {'IS_PUNCT': True, 'OP':'*'}, {'LOWER': 'power'}]

matcher.add('SolarPower', None, pattern1, pattern2)

pattern1 = [{'LOWER': 'solarpower'}]
pattern2 = [{'LOWER': 'solar'}, {'IS_PUNCT': True, 'OP':'*'}, {'LOWER': 'power'}]
pattern3 = [{'LOWER': 'solarpowered'}]
pattern4 = [{'LOWER': 'solar'}, {'IS_PUNCT': True, 'OP':'*'}, {'LOWER': 'powered'}]
```

حيث ان البارامیتر OP اختصار options و اختيار * للدلالة علي عدد مفتوح من الشئ المطلوب هنا

و هنا استخدامات اخري للـ OP

OP	Description
\!	Negate the pattern, by requiring it to match exactly 0 times
?	Make the pattern optional, by allowing it to match 0 or 1 times
\+	Require the pattern to match 1 or more times
*	Allow the pattern to match zero or more times

_

و هناك ما يسمي الجمل المرتبطة Phrase Matchers وهو تطبيق نفس ما تم شرحه علي ملف كبير و جمل طويلة , عبر كلاس phrasematcherمن مكتبة spacy

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

from spacy.matcher import PhraseMatcher
matcher = PhraseMatcher(nlp.vocab)
```

نقوم بفتح ملف فيه كمية كبيرة من النص من هنا

```
with open('reaganomics.txt') as f:
    doc3 = nlp(f.read())
```

و نقوم باختيار كلمات معينة بحيث يتم وضعها في اطار يجمعهم

```
phrase_list = ['voodoo economics', 'supply-side economics', 'trickle-down economics',
'free-market economics']

phrase_patterns = [nlp(text) for text in phrase_list]
```

```
matcher.add('VoodooEconomics', None, *phrase_patterns)
```

او يمكن استخدام نفس الدالة السابقة مع ملاحظة تم انقاص رقم البداية و زيادة النهاية حتي تظهر الكلمة محاطة بما يسبقها و ما يليها

```
matches = matcher(doc3)

for a,b,c in  matches :
    print(f'Word ID {a} , starts at {b} & ends at {c} , and word is {doc3[b-3:c+3]}')
```

*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*

و يمكن تطبيق نفس الأمر مع اللغة العربية , و نري انه يحقق نتائج ايجابية دقيقة

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


from spacy.matcher import Matcher
matcher = Matcher(nlp.vocab)

pattern1 = [{'LOWER': 'الذكاءالإصطناعي'}]
pattern2 = [{'LOWER': 'الذكاء'}, {'LOWER': 'الإصطناعي'}]
pattern3 = [{'LOWER': 'الذكاء'},  {'LOWER': 'الصناعي'}]

matcher.add('الذكاء الإصطناعي', None, pattern1, pattern2, pattern3)

doc = nlp('''
يتميز الذكاءالإصطناعي أنه يسير بسرعة كبيرة نحو المستقبل

و قد بدأ الكثير من الطلاب في دراسة الذكاء الإصطناعي
```

و تزداد فرص العمل في الذكاء الصناعي في كل مكان

```
")

found_matches = matcher(doc)
for a,b,c in  found_matches :
    print(f'Word ID {a} , starts at {b} & ends at {c} , and word is {doc[b:c]}')

matcher.remove('الذكاء الإصطناعي')

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

from spacy.matcher import PhraseMatcher
matcher = PhraseMatcher(nlp.vocab)


with open('raheeq.txt', encoding="utf8") as f:
    doc3 = nlp(f.read())
phrase_list = ['عبد مناف' , 'بنو المطلب','بني المطلب','قريش', 'بنو هاشم', 'بني هاشم']
```

```
phrase_patterns = [nlp(text) for text in phrase_list]

matcher.add('قريش', None, *phrase_patterns)

matches = matcher(doc3)

for a,b,c in  matches :
    print(f'Word ID {a} , starts at {b} & ends at {c} , and word is {doc3[b-3:c+3]}')
```

*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*