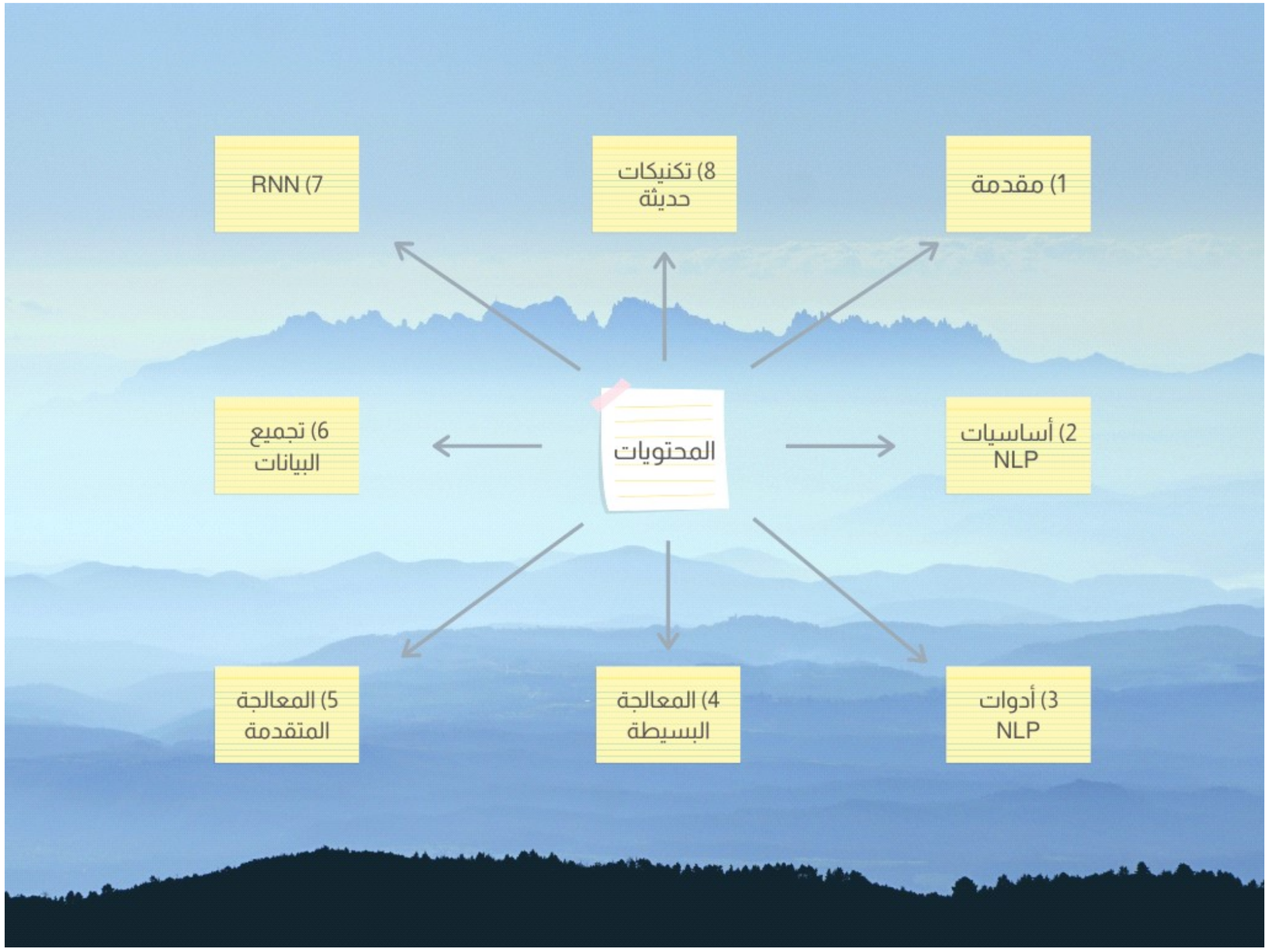
Natural Language Processing المعالجة اللغوية الطبيعية



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	RNN (7
Chat Bot	Gensim	FastText	Bert	Transformer	Attention Model	T. Forcing	CNN	Word Cloud	8)تكنيكات حديثة

القسم الرابع : المعالجة البسيطة للنصوص

الجزء الثاني : Word Embedding

==========

و هي من أهم المصطلحات في علم NLP

و يقصد بها , مصفوفة للكلمات و التي تقوم بتمثيل قيم خاصة لكل كلمة , لتحديد معناها , و لمعرفة مدي تقارب او ابتعاد هذه الكلمة عن باقي الكلمات

و لمعرفة معني هذا الأمر , علينا ان نتخيل مثالا . .

فلو كان لدينا خمس كلمات مختلفة هي (الصبر, رجل , تفاحة , كلب , كتاب)

و نريد عمل علاقات رياضية بينهم , فيمكن أن نقوم بطرح عدد من الأسئلة , و الإجابة عنها لكل كلمة من الكلمات مثل :

- هل هذا الشئ حي ؟
- هل قادر علي التحدث ؟
- هل هو ذكر ؟
- هل هو ملموس ؟
- هل يمكن أكله ؟
- هل يمكن بيعه و شراءه ؟
- هل يتقدم في العمر ؟

و هنا يمكن عمل مصفوفة بسيطة هكذا :

المعيار	الصبر	رجل	تفاحة	كلب	كتاب
هل هذا الشئ حي ؟					
هل قادر علي التحدث ؟					
هل هو ذكر أم انثي ؟					
هل هو ملموس ام شئ معنوي ؟					
هل يمكن أكله ؟					
هل يمكن بيعه و شراءه ؟					
هل يتقدم في العمر ؟					

ستكون الإجابات كالتالي :

المعيار	الصبر	رجل	تفاحة	كلب	كتاب
هل هذا الشئ حي ؟	لا	نعم	نعم	نعم	لا
هل قادر علي التحدث ؟	لا	نعم	لا	نعم	لا
هل هو ذكر ؟	نعم	نعم	لا	نعم	نعم
هل هو ملموس ؟	لا	نعم	نعم	نعم	نعم
هل يمكن أكله ؟	لا	لا	نعم	لا	لا
هل يمكن بيعه و شراءه ؟	لا	لا	نعم	نعم	نعم
هل يتقدم في العمر ؟	لا	نعم	نعم	نعم	لا

ماذا عن الأسئلة التي ليست لها قيمة نعم/لا , بل نسبة معينة , مثلا سؤال : هل هو مهم للإنسان , فقيمة الصبر تختلف عن الكتاب عن التفاحة و هكذا . .

و تضمين الكلمات , معتمدة علي هذا الأساس, و لكن علي مستوي أكبر , فمتوسط المكتبات تتيح لنا قيمة 300 رقم يقوم بوصف كل كلمة بشكل دقيق تماما

و تستخدم هذه الارقام للتعرف علي المعني التقريبي للكلمة المتداولة , و ايضا للمقارنة بين الكلمات , ولمعرفة مدي اقتراب كلمة تفاح من كلمة برتقال , ومدي ابتعاد كلا منهما عن كلمة الصبر

و إنشاء word embedding لا يتم بهذه الطريقة القديمة , ولكن يعتمد علي تدريب الموديل لمعرفة المعني التقريبي لكل كلمة بشكل دقيق بناء علي داتا من ملايين الكلمات

-

و نبدا تحليل الكلمات باستدعاء المكتبة , وكذلك الملف الأكبر للبيانات الذي تم تحميله

```
# python -m spacy download en_core_web_lg
import spacy
nlp = spacy.load('en_core_web_lg')
```

و سنجد هنا ان قائمة الفيكتورز يمكن إظهارها لأي كلمة , فاذا اخترنا مثلا كلمة lion

```
nlp(u'lion').vector
```

سنري انه اظهر قيم كثيرة من الارقام , وهي تمثل عدد 300 فيتشرز لكلمة lion و التي قد تمثل مثلا : اللون , الحجم , الافتراس . . . و كل رقم فيه يترواح حول 1 و -1

```
len(nlp(u'lion').vector)
```

واذا قمنا بعمل فيكتور لجملة كاملة

```
doc = nlp(u'The quick brown fox jumped over the lazy dogs.')

doc.vector
```

سيقوم باظهار 300 رقم ايضا , لانه سياتي بمتوسط ارقام فيكتورز كل الكلمات , وهو استخدام لا معني له غالبا

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

و الخطوة الهامة اننا يمكن حساب التشابه بين الكلمات , فلو اخترنا كلمات ثلاث مثل : lion , cat , pet فيمكن حساب قيم التشابه بينهم بأمر similarity

```
tokens = nlp(u'lion cat pet')

for token1 in tokens:
    for token2 in tokens:
```

```
print(token1.text, token2.text, token1.similarity(token2))
```

فالرقم 1 يدل علي تطابق في التشابه , وتتراوح الارقام بين 0 و 1 و كلما زاد كلما اشار الي قوة الارتباط

و نري ان العلاقة بين cat , pet اكبر من lion, pet لانها منطقية مرتبطة اكثر

ولاحظ انه لا ينظر الي تشابه الحروف فمثلا العلاقة بين الكلمتين ذات الحروف القريبة هي قيمة قليلة لان المعني بعيد

```
nlp(u'lion').similarity(nlp(u'dandelion'))
```

كما أن استخدام نفس الأداة مع الجمل لن يكون لها معني دقيق , فهنا توجد جملتان متعارضتان في المعني ,و علي الرغم من هذا اظهر تشابه قوي بينهما , بسبب اقتراب الكلمات

```
nlp(u'I love school').similarity(nlp(u'I hate school'))
```

```
nlp(u'this file is awesome. I love it').similarity(nlp(u'this file is boring. I hate it'))
```

لذا غالبا ما يستخدم SA لهذا الأمر

ايضا لاحظ ان المعاني المعكوسة ليست بالضرورة قليلة , فكلمتي hate , love قريبة من بعضهم لانهم يستخدمون في استخدامات متشابهة

```
tokens = nlp(u'like love hate')

for token1 in tokens:
    for token2 in tokens:
        print(token1.text, token2.text, token1.similarity(token2))

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

و يمكننا معرفة عدد الكلمات التي يحتويها القاموس لديهم بالكامل وهو :

```
len(nlp.vocab.vectors)
```

و لكل كلمة منهم عدد 300 فيتشرز

```
nlp.vocab.vectors.shape
```

و يمكن اظهار عدد من المعلومات عن كل كلمة مثل :
(has_vector) قيمة بوليان لمعرفة هل في القاموس ام لا
(vector_norm) يعني قيمة اكليديان L2 للارقام ال 300
(is_oov) وهي اختصار out of vocabulary. و هي قيمة بوليان لو كانت الكلمة غير موجودة

```
tokens = nlp(u'dog cat nargle hesham')

for token in tokens:
    print(token.text, token.has_vector, token.vector_norm, token.is_oov)
```

اخيرا يمكننا عمل عمليات حسابية بين الفيكتورز

فهنا نقوم بطرح الملك ناقص رجل زائد امراة , ثم نقوم بتقييم القيمة الجديدة مع عدد من الكلمات لمعرفة اقرب كلمات لها

```
from scipy import spatial
cosine_similarity = lambda x, y: 1 - spatial.distance.cosine(x, y)
king = nlp.vocab['king'].vector
man = nlp.vocab['man'].vector
```

```python
woman = nlp.vocab['woman'].vector
new_vector = king - man + woman
computed_similarities = []
words = ['cat','apple','queen','castle','sea','shell','orange','phone'
        ,'angry','book','white','land','study','crown','prince','dog',
        'great','princess','elizabeth','wow','eat','dead','horrible']
for word in words:
   similarity = cosine_similarity(new_vector,nlp.vocab[word].vector)
   computed_similarities.append((word, similarity))
computed_similarities = sorted(computed_similarities, key=lambda item: -item[1])
for a,b in computed_similarities[:10] :
   print(f'Word {a} , has similarity {b}')
```

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

ماذا عن اللغة العربية ؟

لا تدعم spacy قيمة WE للغة العربية بعد , لكن سنري لاحقا عدد من الموديلز المستخدمة في هذا الأمر , في دروس GloVe و Gensim