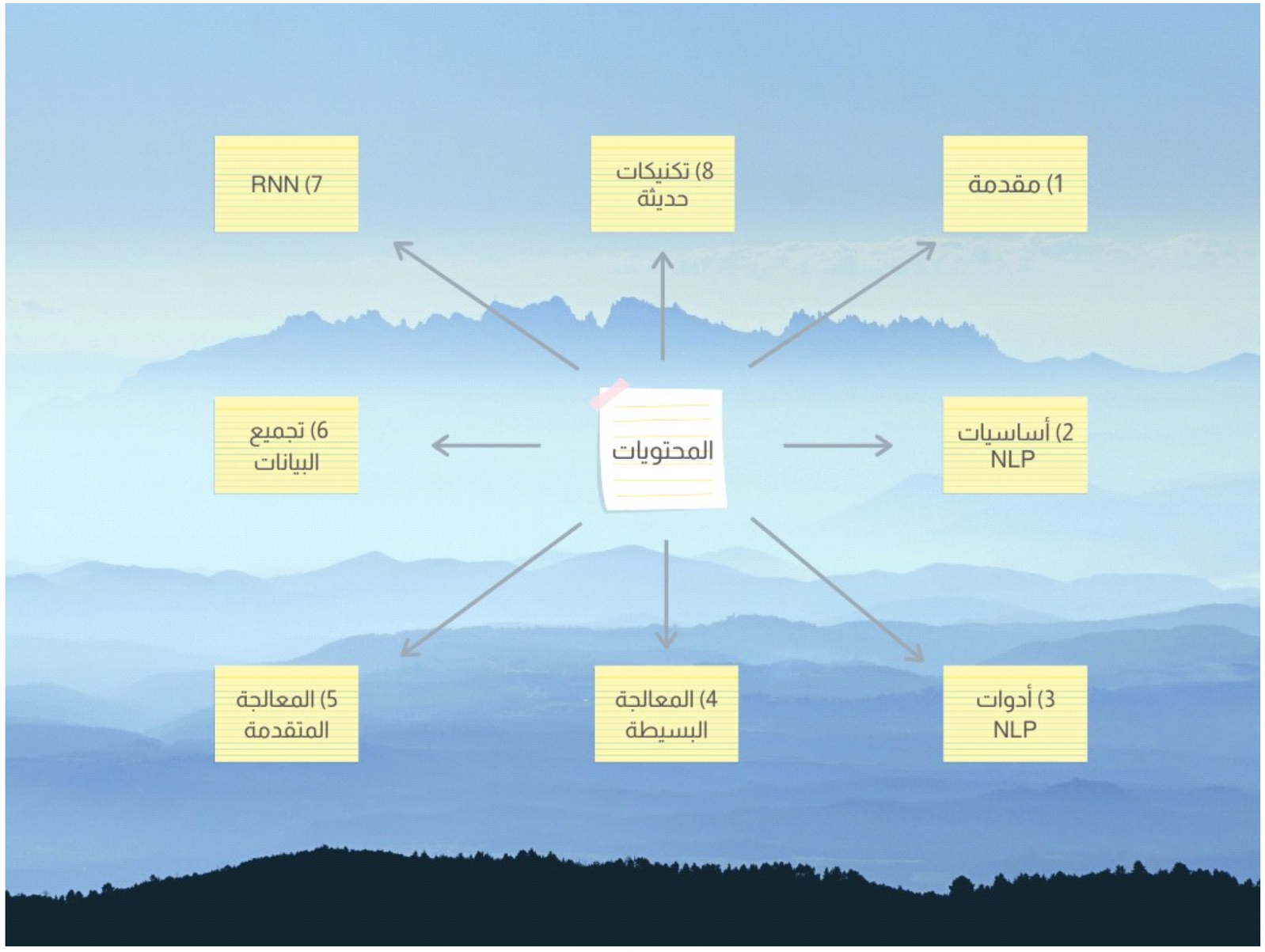
Natural Language Processing المعالجة اللغوية الطبيعية



المحتويات

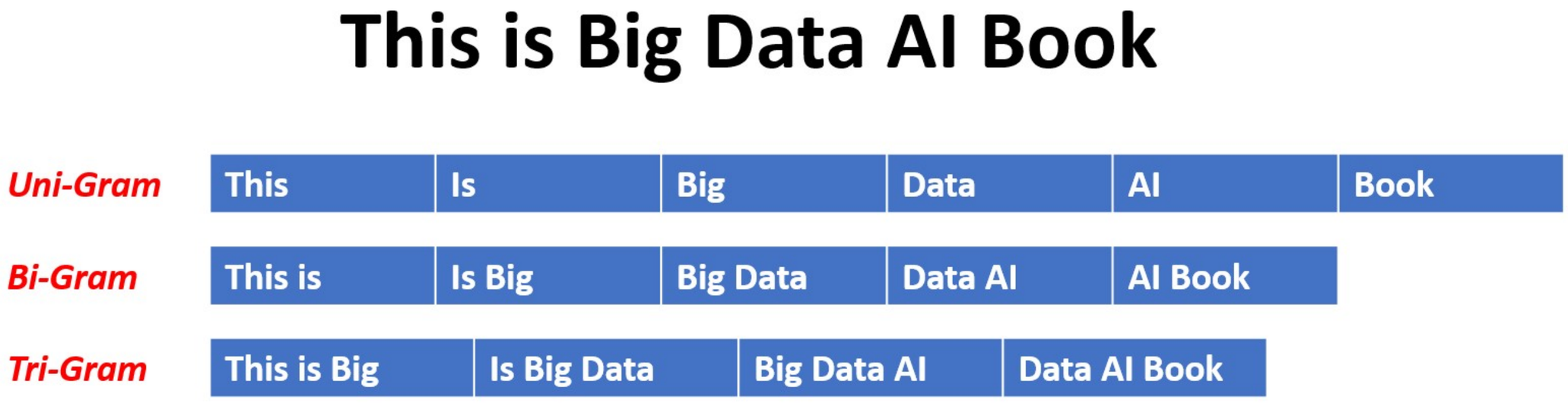
القسم الخامس : المعالجة المتقدمة للنصوص

الجزء الثامن : NGrams

==========

هي فكرة تناول الكلمات الحالية , او السابقة للكلمة المطلوبة بعدد محدد من الكلمات , فحرف N يشير الي عدد معين , وكلمة grams تشير الي الكلمات . .

و بالتالي لدينا ما يسمي unigram لتناول كل كلمة علي حدة , او نظام bigram و الذي يعني التعامل مع كلمتين , او نظام trigram للتعامل مع ثلاث كلمات



Physician note	**"...Patient has evidence of macular degeneration..."**
Unigrams	"patient" "has" "evidence" "of" "macular" "degeneration"
Bigrams	"patient has" "evidence of" "macular degeneration" "has evidence" "of macular"
Trigrams	"patient has evidence" "of macular degeneration" "has evidence of" "evidence of macular"
4-grams	"patient has evidence of" "has evidence of macular" "evidence of macular degeneration"

كما ان العدد مفتوح فهناك 4gram , 5gram و هكذا . .

_

و هي لا تستخدم فقط مع الكلمات , لكن مع الحروف كذلك

Lightweight NLP for Social Media Applications

li	we	tn	or	ia	di	pl	ti
ig	ei	nl	rs	al	ia	li	io
gh	ig	lp	so	lm	aa	ic	on
ht	gh	pf	oc	me	ap	ca	ns
tw	ht	fo	ci	ed	pp	at	

و هي لها العديد من الاستخدامات ,والتي غالبا تعتمد علي انتاج كلمة او حرف محدد مثل :

- انتاج النصوص text generations
- الاجابة علي الاسئلة answering questions
- الرد الآلي chatbot
- التصحيح التلقائي auto correct
- الترجمة الآلية translation

-

اذن كيف يقوم الموديل بالتدرب علي اداة ngrams ؟ (و هنا للاستفادة منها في لغات غير مدرب عليها)

- أولا يتم تناول كمية كبيرة من النصوص
- يتم تحديد الصيغة التي سيتم استخدامها : bigram مثلا, و هي التي تعني انه سيتم استخدام كل كلمتين معا
- يتم احتساب مدي احتمالية تكرار كل كلمة بعد كلمة تالية لها عبر عمل جدول كبير ثنائي الابعاد و التي تكون كالتالي :

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

و القانون يعتمد علي حساب احتمالية وصول كلمة بعد كلمة معينة ,

$$Bigram\ model: \ p(w_t \mid w_{t-1})$$

Ex. p("brown" | "quick") = 0.5, p("the" | "the") = 0
But... how do we find these probabilities?

أي احتمالية استنتاج كلمة تالية لعدد من الكلمات السابقة لها

و يتم حساب هذه القيمة , عبر تحديد الكلمتين المراد حسابهما و ليكن quick + brown , ثم يتم حساب عدد المرات التي جاءت فيهما الكلمتين وراء بعضهما , و ايضا يتم حساب عدد المرات التي تم ذكر فيها الكلمة الاولي , ويتم القسمة :

p("brown" | "quick"):

- How many times does "quick" → "brown" appear in my set of documents?
- How many times does "quick" appear in my set of documents?
- Divide these 2!

$$p(brown \mid quick) = \frac{count(quick \rightarrow brown)}{count(quick)}$$

و فكرة القسمة حتي يتم معرفة عدد المرات التي جاءت فيها الكلمة الثانية وراء الاولي , مقسومة علي عدد المرات الكلي التي جائت فيها الكلمة الاولي دونها , فلو تكررت quick ست مرات , منها ثلاثة تليها brown تكون النسبة 50% , أي 0.5

و اذا اردنا حساب احتمالية مجئ ثلاث كلمات متتالية (هذه ليست trigram) , فتكون بقاعدة السلسلة chain rule

$$p(A \to B \to C) = p(C \mid A \to B)p(B \mid A)p(A)$$

و هي تعني اولا احتمالية مجيء الكلمة الاولي , مضروبة في احتمالية مجيء الكلمة الثانية تالية لها , مضروبة في احتمالية مجيء الكلمة الثالثة اعتمادا علي الاثنين السابقين

و هنا تفاصيلها للـ unigram :

$$p(A) = \frac{count(A)}{corpus\ length}$$

و هنا للـ trigram

$$p(C \mid A, B) = \frac{count(A \rightarrow B \rightarrow C)}{count(A \rightarrow B)}$$

و هنا مع امتداد الكلمات

$$\left| p(A,B,C,D,E) = p(E \mid A,B,C,D)p(D \mid A,B,C)p(C \mid A,B)p(B \mid A)p(A) \right|$$

_

و من مشاكل فكرة NGrams انها تعتمد علي الداتا المتوافرة و ليس علي المنطق

فلو كانت الجمل المتوافرة لدينا في التدريب فيها فقط جملة :

the brown fox jumped quickly over the lazy dog

فلو تعاملنا مع جملة اثناء الاختبار جملة the lazy , و كانت الكلمة التالية في الحقيقة هي turtle فستكون احتمالية حدوثها هي صفر علي الرغم من انها موجودة بشكل حقيقي , لأنه لم يتم التدريب عليها

لذا فإننا نقوم بعمل تعديل في القانون , بحيث نضيف رقم 1 في البسط, و قيمة V و هي عدد كلمات كل النص في المقام , مما يجعلنا نتجنب وجود صفر كاحتمالية , ونزيد من دقة الرقم المحسوب , وتجعل هناك احتمالية لوجود كلمة بعد كلمة , حتي لو لم يتواجدا معا في النص الأصلي , وقتها ستكون قيمتها علي V

$$P_{smooth}(B \mid A) = \frac{count(A \rightarrow B) + 1}{count(A) + V}$$

و هناك ما يسمي markov assumption و الذي ينص علي أن احتمالية وجود كلمة معينة , يعتمد فقط علي وجود الكلمة السابقة لها فحسب , وليس علي كلمات أخري , اي أن

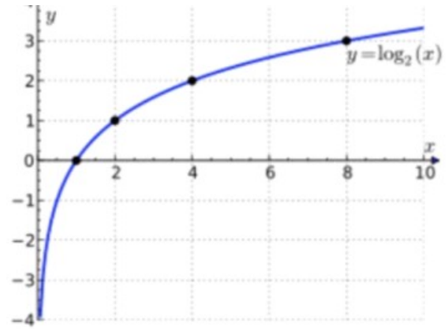
$$p(E \mid A, B, C, D) = p(E \mid D)$$

و هذا كلام غير دقيق تماما , فكثير من الكلمات تعتمد علي كلمات سابقة لها بعدد اكبر من كلمة واحدة

و يجب أن نلحظ أن ضرب الاحتماليات في بعضها البعض سيجعل القيمة النهائية تؤول للصفر , وهذا سيتسبب في مشاكل , لذا يتم استخدام اللوغاريتم , والذي يحول اي رقم بين ال1 و الصفر , الي قم بين 0 و سالب انفينيتي , و بالتالي الارقام الصغيرة ستتحول الي سالب ارقام كبيرة و يمكن مقارنتها بسهولة

Hint

- Use log-probabilities instead
- If A > B, then log(A) > log(B)
- Because log() is a monotonically increasing function
- Try it on your calculator if you don't believe me



$$logp(w_1, \ ..., \ w_T) = logp(w_1) + \sum_{t=2}^{T} logp(w_t \mid w_{t-1})$$

_

في المثال رقم الأول , نري كود لقراءة عدد كبير من الكلمات من ويكيبيديا , ثم عمل bigram لها , ثم قياس score لها

كما نري في المثال الثاني , اسلوب مختلف لمعالجة نفس فكرة ال bigrams و لكن بأسلوب ال logistic regression

و هذا النظام يمكن ان يعمل بنجاح مع اللغة العربية , ولكن بشرط توافر داتا مناسبة

_