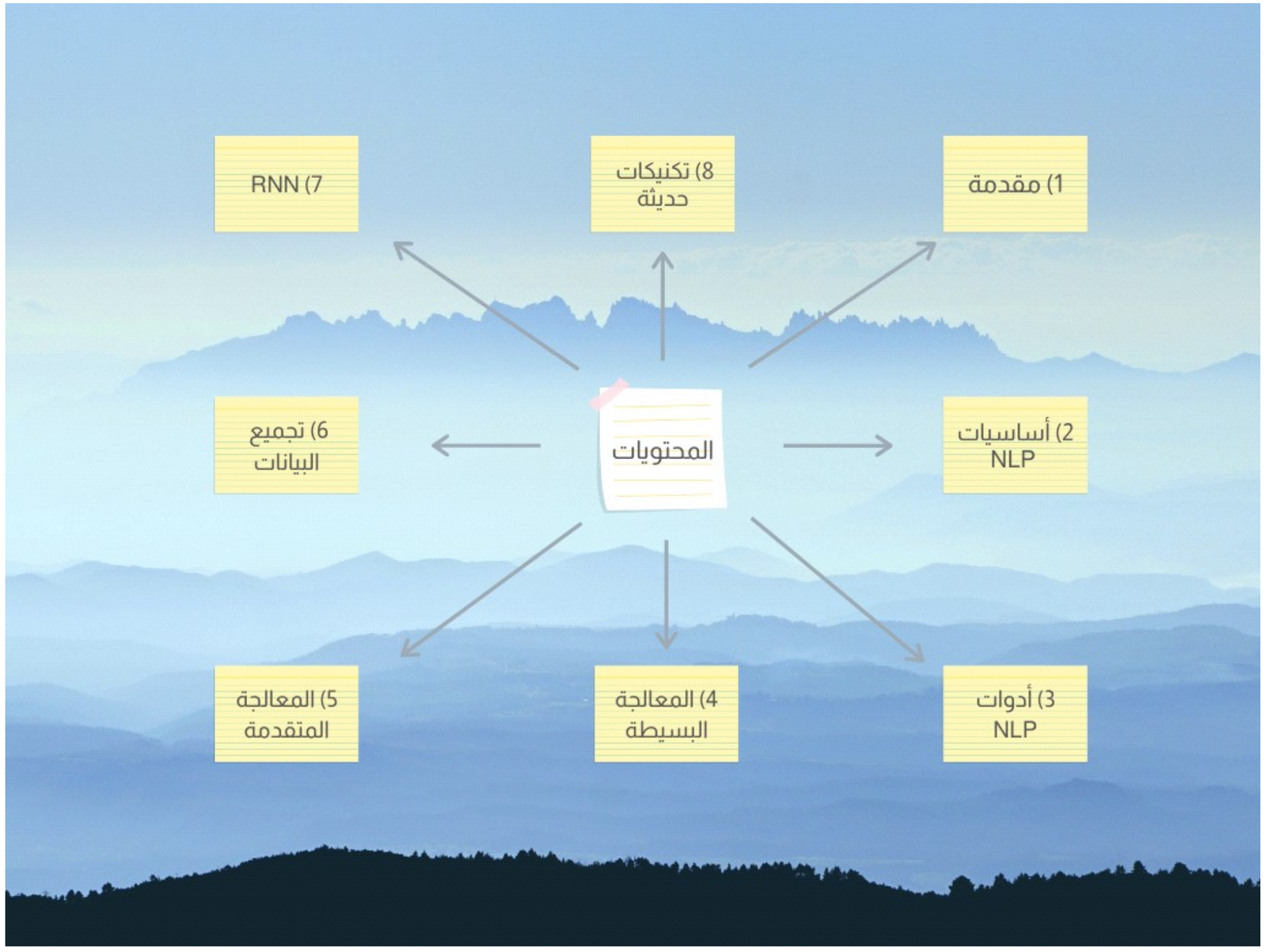
Natural Language Processing المعالجة اللغوية الطبيعية



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

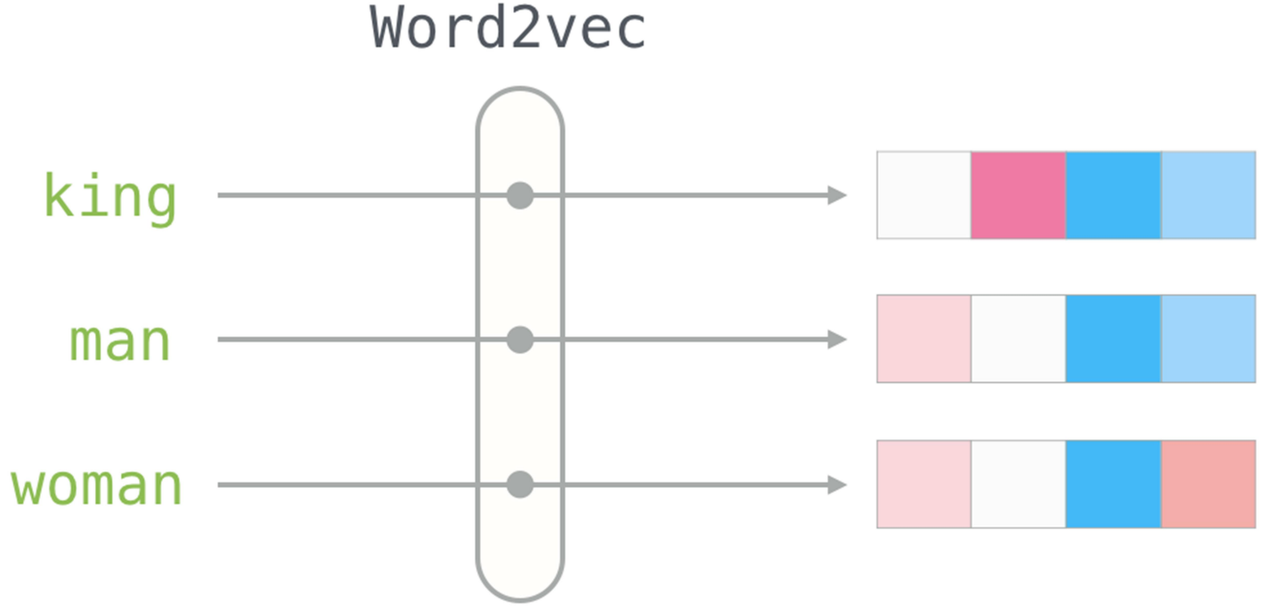
القسم الرابع : المعالجة البسيطة للنصوص

الجزء الرابع : Word2Vec

==========

اداة word2vec هي عبارة عن شبكة عصبية من طبقتين و التي تقوم بمعالجة النصوص. .

يكون المدخل لها هو باقة الكلمات text corpus اما المخرج فهي كمية من المصفوفات الخاصة بالفيتشرز للنص



فهي ببساطة شبكة عصبية , يتم تدريبها علي أساس تضمين الكلمات , وهدفها حساب مدي أهمية و قيمة كل كلمة في الجملة , و من ثم , نقوم باستنتاج الكلمة الباقية

و المهمة الأساسية لأداة word2vec هي عمل تجميع grouping للمصفوفات للكلمات المتشابهة و المتماثلة و المرتبطة معا , وهو ما يتم عبر التشابهات الرياضية لكل كلمة

و هذه التشابهات و التناظرات تشبه (رجل - صبي) == (امرأة – فتاة) .

كذلك انها ستعرف ان هذه الكلمة مفردة و هذه جمع , وهو ما يسهل لاحقا عمل صياغة كاملة لنصوص و معرفة هل المفترض استخدام مفرد ام جمع و هكذا

و ايضا اداة word2vec حينما تاخد كمية كبيرة من البيانات , فلديها المقدرة علي توقع معاني الكلمات , بناء علي موقعها و سياقها . .

-

و منها نوعان أساسيان له هما : CBOW , skip-gram

أولا : تكنيك CBOW

وهو اختصار continuous bag of words , وهو مزيج بين تكنيك bag of words و تكنيك NGram

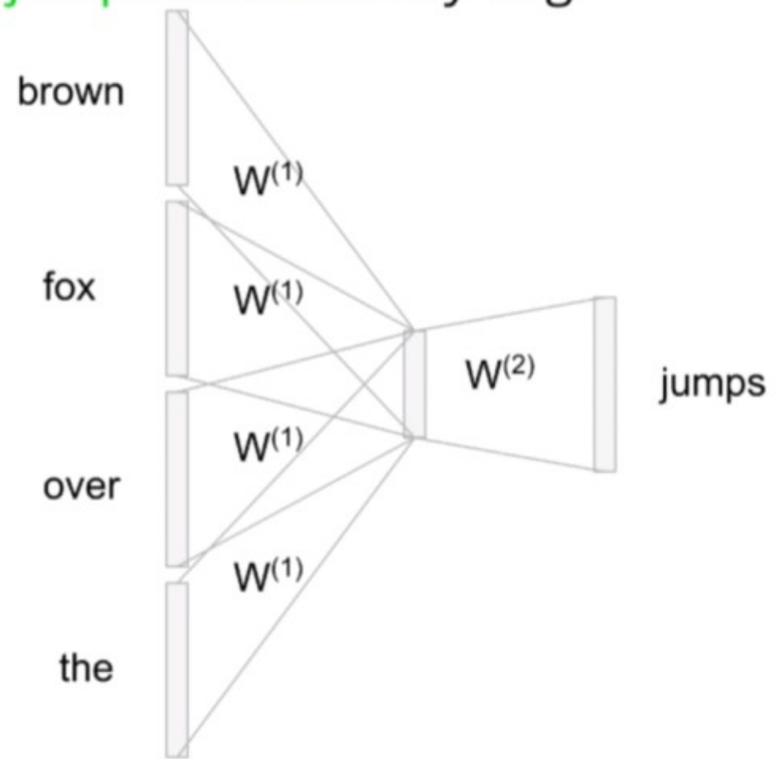
ففكرة BOW تعتمد علي استخدام عدد من الكلمات الموجودة في النص و عمل ارقام 1 و 0 حسب تواجد كل كلمة , وفكرة NGram تعتمد علي استخدام كلمة او اكثر من الكلمات السابقة لاستنتاج الكلمة التالية , و بالتالي عبر استخدام CBOW يمكننا ان نستخدم اكثر من كلمة في نفس الجملة لاستنتاج كلمة محددة

و يمكن تعريفها انها شبكة عصبية , ولكن تستخدم لتوقع ما هي الكلمة الناقصة في جملة معينة (غالبا ما تكون الكلمة الأخيرة) , وذلك عبر حساب المصفوفات الضمنية embedding matrix للكلمات الداخلة , و منها يتم معالجتها للوصول للطبقة التالية في الشبكة , و اخيرا الطبقة النهائية بدالة تفعيل softmax لاختيار الكلمة الناقصة

و يتم اختيار عدد محدد من الكلمات التي سيتم استخدامها لاستنتاج الكلمة المطلوبة , فهنا يتم اختيار رقم 2 , اي كلمتين قبل الكلمة المطلوبة , وكلمتين بعدها , كما نري في المثال

CBOW - continuous bag of words

"The quick brown fox jumps over the lazy dog."



"Context size" could be considered 2 (or 4)

In practice, context size is usually set from 5-10 (on either side)

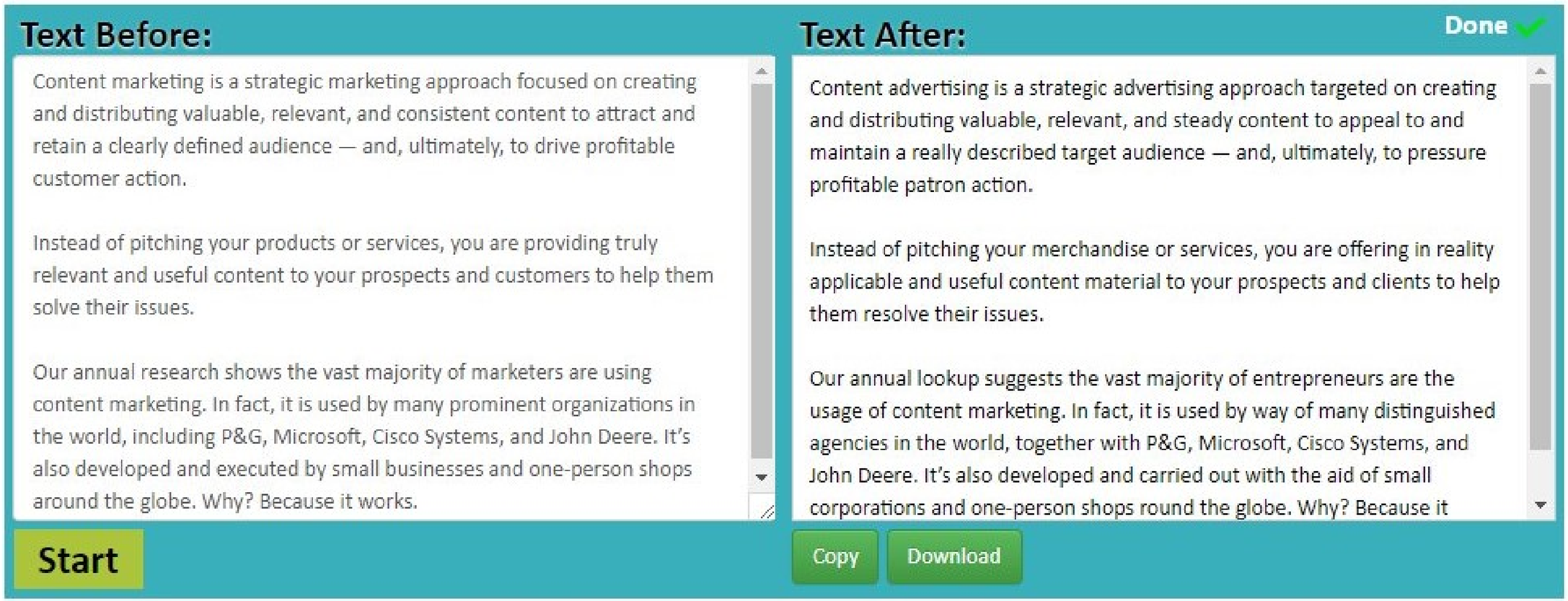
The input weight is $W^{(1)}$ for all input words (same weight used multiple times)

و يمكن زيادة الرقم اكثر من 2 , وغالبا ما يكون من 5 الي 10 , من كلا الاتجاهين

-

و من التطبيقات العملية للـ CBOW هي ما يسمي "تدوير المقال" article spinning

و الذي يقصد به ان نقوم بعمل تغيير في بعض كلمات المقالات , مع الاحتفاظ بالمعني , وذلك لعمل نوع من الاقتباس من بعض المواقع ووضعها في مواقع اخري , دون ان يقوم جوجل بتقليل تصنيف الموقع الثاني علي انه سارق للمحتوي



و من الأدوات التي تقوم بتطبيق article spinning هي الـ CBOW حيث اننا نقوم اولا بحذف بعض الكلمات من المقال , ثم تقوم هي باستنتاج الكلمة المطلوبة باستخدام الكلمات السابقة و التالية

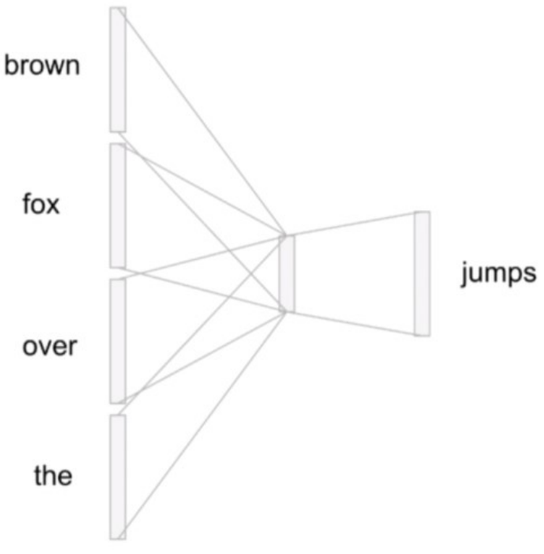
و تكون الفكرة الرياضية للـ CBOW اننا نقوم بما يشبه simple neural network و كان هناك عددا من المدخلات (ضعف الرقم الذي تم تحديده من الاتجاهين) , ثم نقوم بحساب قيم التضمين الخاصة بها, و ايجاد متوسط كل كلمة , ثم نقوم بعمل تدوير لهذه المصفوفة و ضربها في قيم التدريب h و باستخدام softmax يمكننا استنتاج الكلمة التالية

و عبر تدريب الموديل علي مئات الالاف من الكلمات , يمكن حساب قيم مناسبة للـ h ويمكن تطبيق الموديل بنجاح

$$h = \frac{1}{|C|} \sum_{c \in C} W^{(1)} c$$

$$p(y \mid C) = softmax(W^{(2)T} h)$$

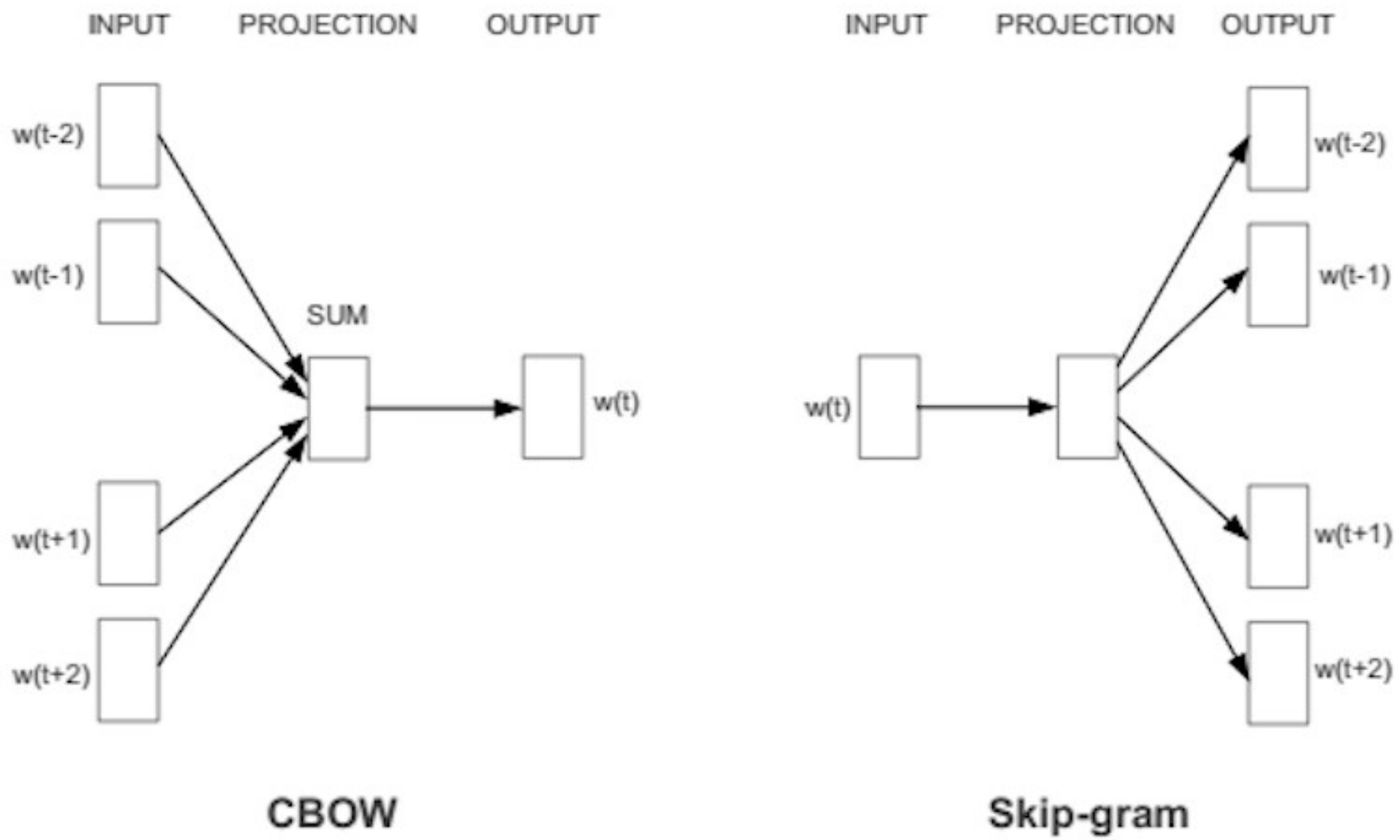
You *could* try to derive the gradients now, but we still have a few more modifications to make



_

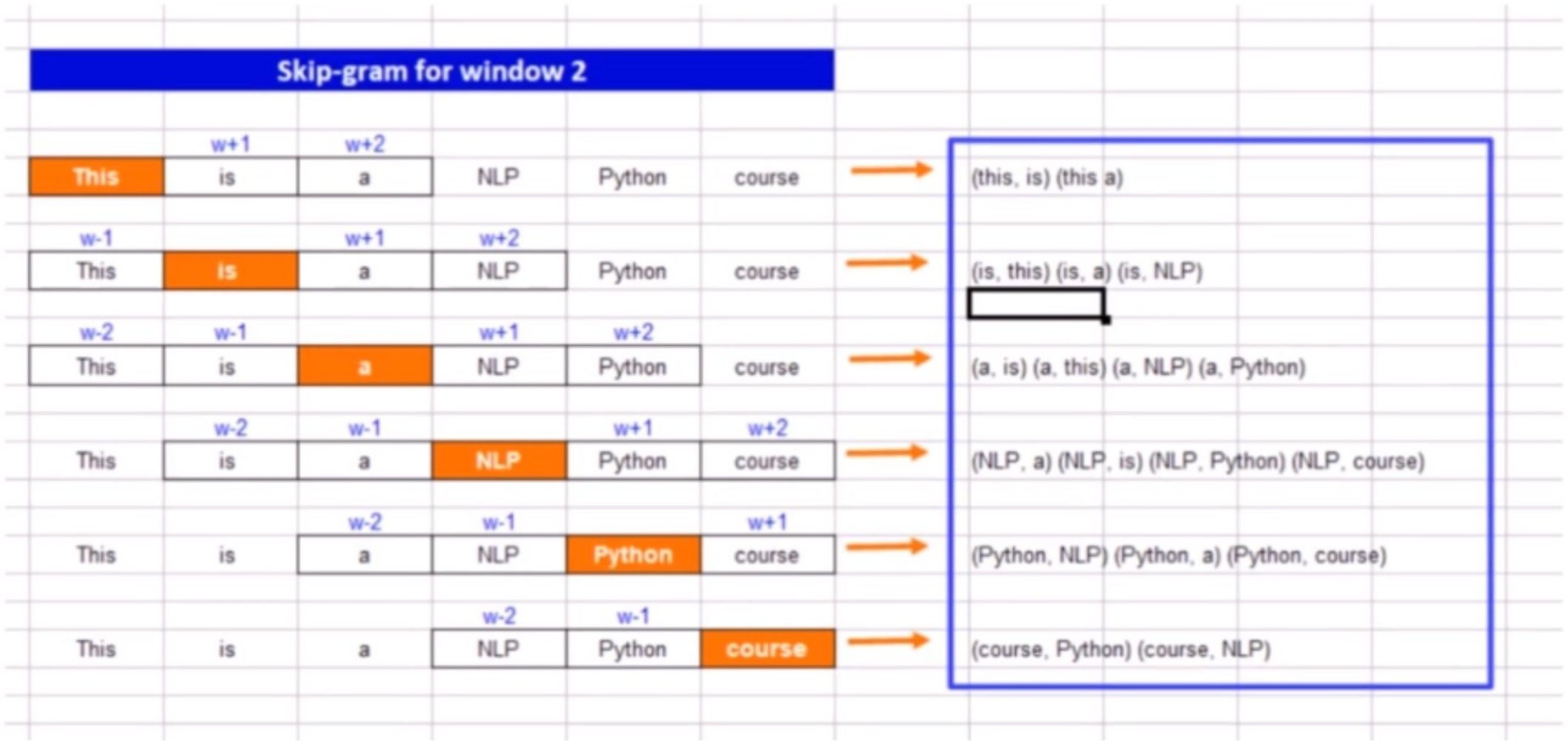
ثانيا : تكنيك SkipGram

إن كان الـ CBOW تقوم باستنتاج كلمة ناقصة عبر تناول كلمات الجملة فإن الـ skip-gram العكس , تستنتج جملة عبر كلمة



و تقوم فكرتها علي حساب العلاقة بين الكلمات و بعضها البعض , في المرة الأولي , تم حساب العلاقة بين الكلمة الاولي و الثانية this, is و بين الاولي و الثالثة this ,a

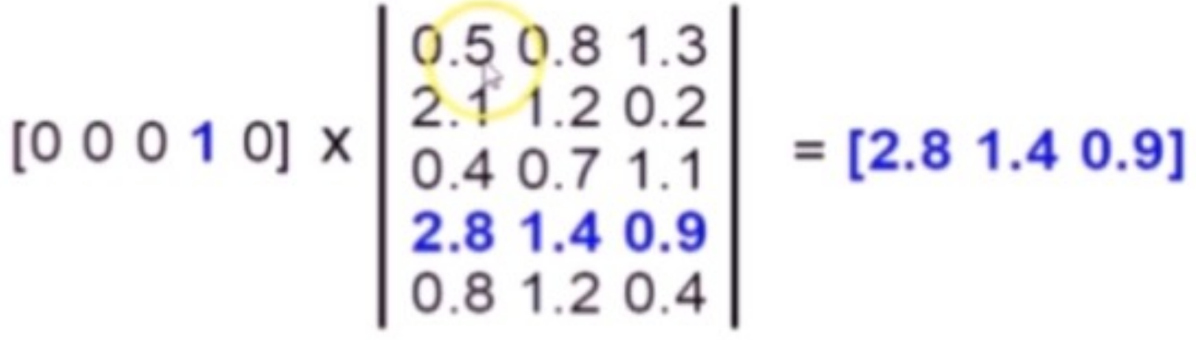
و في المرة الثانية حينما كان التركيز علي كلمة is تم الحساب العلاقة بين الثانية و الاولي ,و الثانية و الثالثة , والثانية و الرابعة



و هكذا , مع ملاحظة أنه يتم تحديد حد أقصي لعلاقة الكلمات مع بعضها البعض , فهنا الحد الاقصي هو 4

و تسمي هنا window 2 لأنه يتم وضع عدد 2 شباك امام و مثلهم في الخلف , و يتم تدريب الشبكة جيدا علي هذه الكلمات , حتي تتمكن من استنتاج الكلمات التالية عبر اعطائها كلمة واحدة

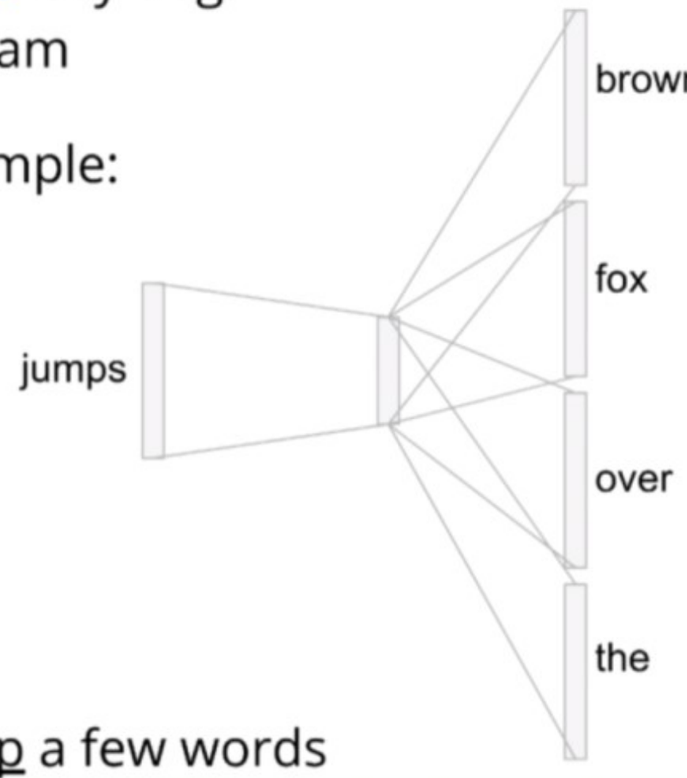
و ما يتم دخوله في الشبكة , هي قيم ال embedding لكل كلمة , ويتم حسابها بسهولة , عبر ايجاد الـ onehotencoder لكل كلمة , وضربها في embedding matrix العامة

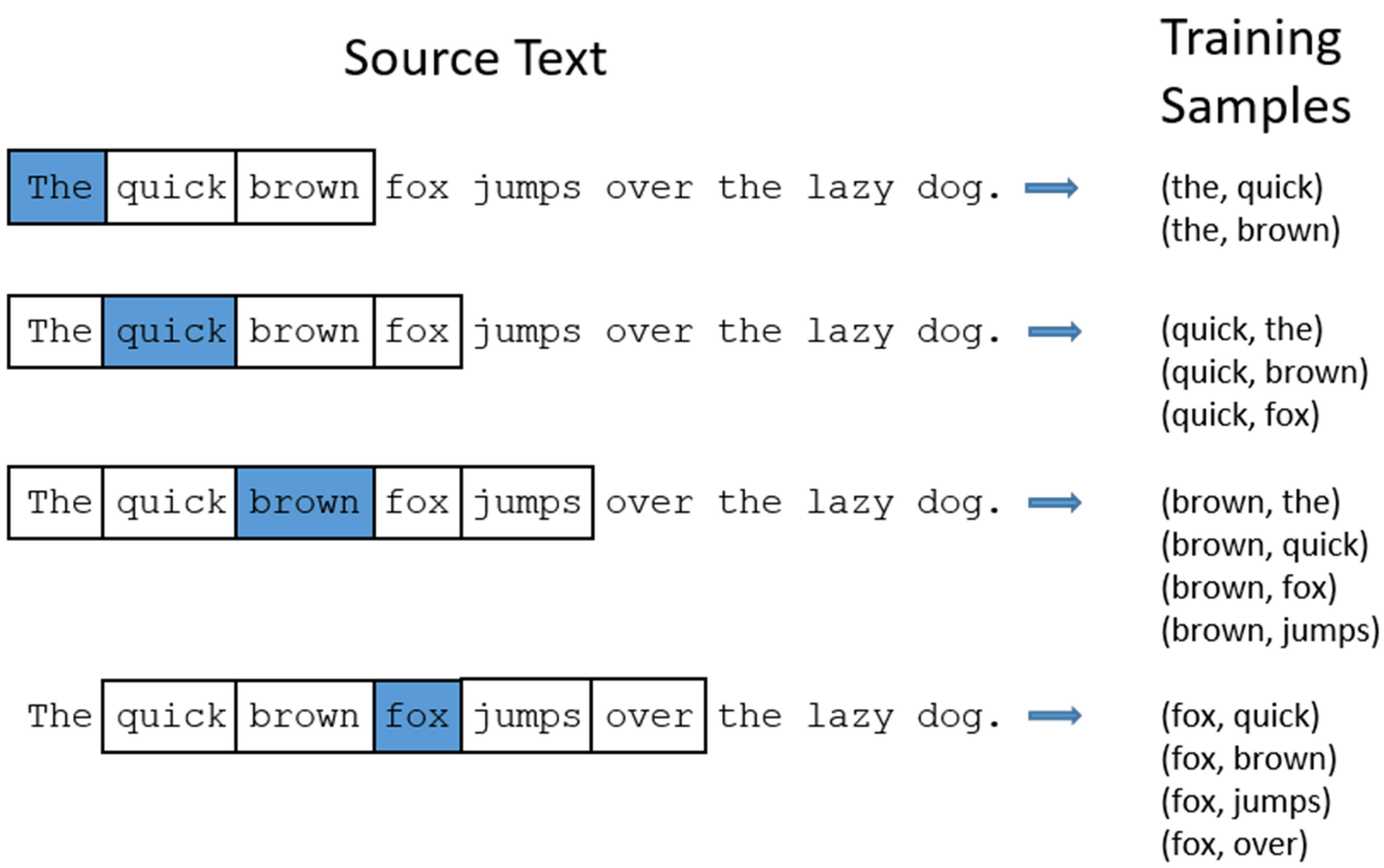


_

و تأتي الفكرة أصلا من فكرة ال bigram ففيها يمكن استخدام كلمة لاستنتاج كلمة تالية لها , اي من كلمة jumps يمكن استنتاج كلمة over , و لكن مع توسيع الفكرة يمكن عمل skipping اي تخطي لعدد من الكلمات بحيث نستنتج الكلمات الثالثة او الرابعة او السابقة , لذا تسمي skipgram

- "The quick brown fox jumps over the lazy dog."
- Helpful to think of it in terms of bigram
- Bigram model gives us 1 training sample:
  jumps → over
- Skipgram gives us 3 additional training samples:
  jumps → brown
  jumps → fox
  jumps → the
- Skipgram: *like* bigram, except we skip a few words





و لكن ما هي الآلية التي سيتم اختيار الكلمات علي اساسها , كيف يمكن عبر معالجة كلمة ان نستنتج كلمة تالية لها ؟

هناك طريقتين , طريقة أقل نجاحا و هي الاختيار الهرمي , وطريقة أكثر نجاحا و هي : العينة السلبية

_

الاختيار الهرمي Hierarchical Softmax

و هي الطريقة الاولي لاختيار الكلمة المناسبة

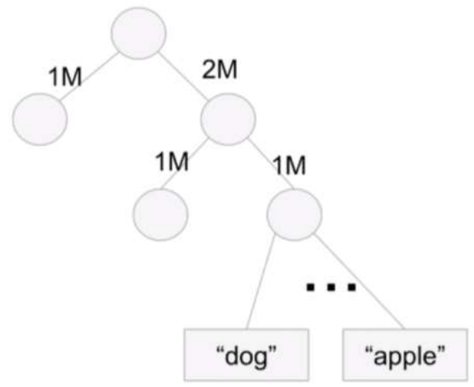
المعضلة تكمن في أن دالة ال softmax ستقوم بالاختيار عبر عدد ضخم من الكلمات , ففي جلوف هناك 400 الف كلمة , و في مكتبة word2vec هناك 3 مليون , وهذا معناه انه اثناء التصنيف ستكون هناك صعوبة جمة في اختيار الكلمة المناسبة , وستكون الدقة دائما قليلة

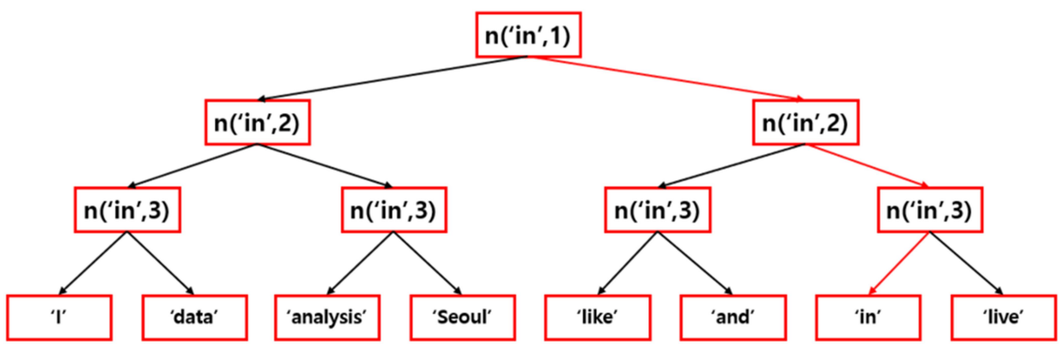
و السبب الاساسي ان دالة السوفت ماكس تعتمد علي مجموعة لوغاريتمات جميع الاصناف , فاذا قمنا بجمع لوغاريتمات 3 مليون كلمة , فسيكون المقام قيمة ضخمة جدا

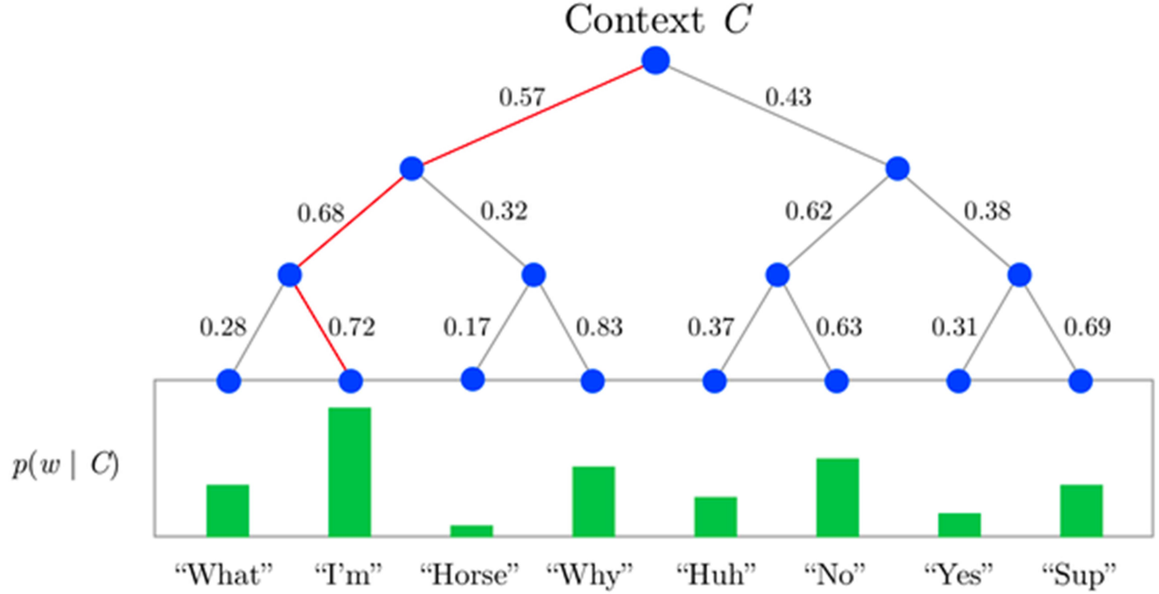
- Calculating softmax is O(V)

$$p(y = j \mid x) = \frac{exp(w_j{}^T x)}{\sum_{k=1}^{V} exp(w_k{}^T x)}$$

لذا فإن فكرة الاختيار الهرمي قائمة علي تقسيم جميع الكلمات الموجودة لدينا علي شكل تقسيم هرمي كامل , بحيث يكون هناك مستويات في الكلمات , بحيث يكون هناك كلمات في المستويات العليا , وأخري في المستويات الاقل و هكذا

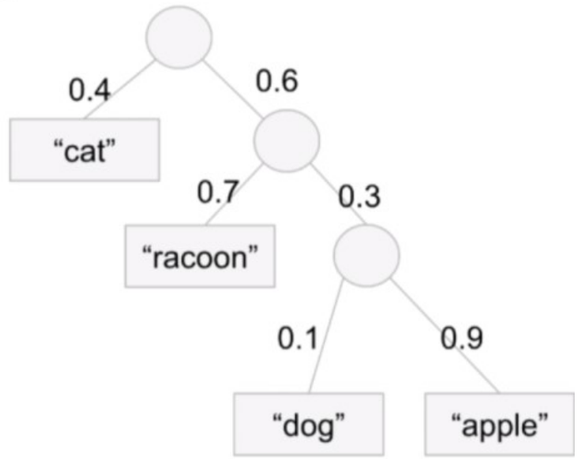






وقتها يكون حساب قيمة احتمالية كلمة معينة , عبر ضرب احتماليات المسارات المستمرة للوصول الي الكلمة , هكذا :

$$p(w \mid input) = \prod_{node \in path\ to\ w} \sigma(v_{node}{}^{T} h_{input})$$



Ex.
p(cat) = 0.4
p(racoon) = 0.6 x 0.7 = 0.42
p(dog) = 0.6 x 0.3 x 0.1 = 0.018
p(apple) = 0.6 x 0.3 x 0.9 = 0.162
Sum = 1, just as softmax should!

مع التاكيد علي ان مجموع كل قيم احتماليات الكلمات تساوي 1

ويتم وضع الكلمات في هذا الترتيب الهرمي عبر عدد من المعادلات الرياضية المختلفة , مثل معادلة هوفمان , و التي تراعي وجود الكلمات الاكثر انتشارا مثل the , a في المستويات العليا ذات الاحتمالية الاكبر , والكلمات النادرة في المستويات الدنيا

_

الطريقة الثانية لاختيار الكلمات , وهي الأكثر سرعة و دقة , وتسمي العينة السلبية negative sampling

و تأتي هذه الفكرة لتجنب عيوب الطريقة السابقة , حيث مشكلة التدريب علي مئات الالاف من الكلمات , وصعوبة الحصول علي كفاءة عالية

و الفكرة تقوم علي أنه بدلا من تدريب الموديل بحيث يكون هناك multi-classification بين الكلمة الصحيحة و مئات الالاف من الكلمات الخاطئة , فإننا نقوم بتحديد أولا بتحديد الكلمة الداخلة و هي jumps ثم تحديد الكلمات الصحيحة و هي (over , the , brown , fox) , وبدلا من تدريبها علي باقي الكلمات الأخرى في اللغة الانجليزية ,فإننا نقول باختيار عدد من الكلمات الأخرى بشكل عشوائي ,فيتم مثلا اختيار اربع كلمات اخري غير صحيحة بشكل عشوائي و لتكن (apple , orange , boat , Tokyo) , ثم تدريب الموديل بطريقة حساب احتمالية وجود الكلمات الصحيحة و زيادتها , ثم حساب احتمالية الكلمات الخطأ و تقليلها

و تكون معادلة الخطأ هي مجموعة لوغاريتمات احتمالية الكلمات الصحيحة , و لوغاريتمات 1 ناقص احتمالية الكلمات الخطأ

لا تنس ان لوغاريتم الـ 1 يساوي صفر , فمعادلة الخطأ ستؤول للصفر حينما تكون احتمالية الكلمات الصحيحة كبيرة و الخاطئة قليلة , وهنا نتمكن من تدريب الموديل علي اختيار الكلمات الصحيحة و الابتعاد عن الخاطئة

و بالنسبة لعدد الكلمات , غالبا ما يتم اختيار من 5 الي 10 كلمات

Input word: jumps
Target words: brown, fox, over, the
Negative samples: apple, orange, boat, tokyo

$$J = logp(brown \mid jumps) + logp(fox \mid jumps) + logp(over \mid jumps) + logp(the \mid jumps) + log[1 - p(apple \mid jumps)] + log[1 - p(orange \mid jumps)] + log[1 - p(boat \mid jumps)] + log[1 - p(tokyo \mid jumps)]$$

و تكون المعادلة العامة :

$$J = \sum_{c \in C} log\sigma({W^{(2)}}_c{}^T W^{(1)}_{in}) + \sum_{n \in N} log[1 - \sigma({W^{(2)}}_n{}^T W^{(1)}_{in})]$$

و هناك فكرة تستخدم أحيانا , فلو كان لدينا كلمة مدخل وهي jumps و لدينا 4 كلمات صحيحة , فبدلا من البحث عن عدد من الكلمات الخاطئة , فيمكن استبدال كلمة المدخل بكلمة واحدة خطأ , وليكن lighthouse , ثم تكون المعادلات الصحيحة هي العلاقة بين كلمة المدخل و الاربع كلمات الصحيحة , وتكون المعادلات الخطأ هي العلاقة بين كلمة المدخل الخاطئة و الاربع كلمات الصحيحة

The quick brown fox jumps over the lazy dog.

The quick brown fox lighthouse over the lazy dog.

+ve samples: jumps → brown, jumps → fox, ...

-ve samples: lighthouse → brown, lighthouse → fox, ...

و حينما نقوم باختيار الكلمات الخاطئة , من الممكن ان يتم هذا بشكل عشوائي , او أن نتبع معايير معينة بحيث تكون الكلمات الخاطئة ليست نادرة تماما , حتي يتمكن الموديل من استبعاد الكلمات المنتشرة

و غالبا ما يتم استخدام هذا القانون , والذي يعتمد علي نسبة تواجد الكلمة اصلا في النص , فلو كانت كلمة نادرة , فستكون قيمة $\tilde{p}(w)$ برقم قليل , وبالتالي قيمة Pdrop ستكون قليلة تقترب من الصفر , والعكس صحيح

$$p_{drop}(w) = 1 - \sqrt{\frac{threshold}{\tilde{p}(w)}}$$

- Typical threshold = 10^{-5}
- Ex. if p(w) = 10^{-5}, then $p_{drop}(w) = 1 - 1 = 0$
- Ex. if p(w) = 0.1, then pdrop(w) = $1 - 10^{-2} = 0.99$

حيث قيمة $\tilde{p}(w)$ تساوي

$$\tilde{p}(w) = \frac{count(w)^{0.75}}{\sum_{w'} count(w')^{0.75}}$$

_

و نري مثالين عمليين مبسطين لفكرة CBOW & Skip-gram في ملفي 4.4.2 & 4.4.1
بالإضافة إلي أن word2vec سيتم استخدامها بشكل افضل مع شرح مكتبة gensim