التعلم العميق Deep Learning

- **الدرس الأول : التعلم العميق و الشبكات العصبية**
  - الأسبوع الأول : مقدمة للتعلم العميق
  - الأسبوع الثاني : أساسيات الشبكات العصبية
  - الأسبوع الثالث : الشبكات العصبية المجوفة
  - الاسبوع الرابع : الشبكات العصبية العميقة
- **الدرس الثاني : تطوير الشبكات العميقة : المعاملات العليا**
  - الأسبوع الأول : السمات العملية للتعلم العميق
  - الأسبوع الثاني : الحصول علي القيم المثالية
  - الأسبوع الثالث : ضبط قيم الشبكات العميقة
- **الدرس الثالث : هيكلية مشاريع الـ ML**
  - الأسبوع الأول : استراتيجيات الـ ML - 1
  - الأسبوع الثاني : استراتيجيات الـ ML - 2

- **الدرس الرابع : الشبكات العصبية الملتفة CNN**
  - الأسبوع الأول : أساسيات الشبكات العصبية الملتفة
  - الأسبوع الثاني : حالات عملية من الشبكات العصبية الملتفة
  - الأسبوع الثالث : التعرف علي الأشياء
  - الاسبوع الرابع : التعرف علي الوجه
- **الدرس الخامس : الشبكات العصبية المتكررة RNN**
  - الأسبوع الأول : مفهوم الشبكات العصبية المتكررة
  - الأسبوع الثاني : المعالجة اللغوية الطبيعية NLP
  - الأسبوع الثالث : نماذج التتابع

درس 5: الشبكات العصبية المتكررة RNN

الأسبوع الثاني : المعالجة اللغوية الطبيعية NLP

==========

استخدام المعالجة اللغوية الطبيعية Natural language processing والتي تختصر NLP مع التعلم العميق هو من التطبيقات الهامة , والتي تستخدم الكلمات كمصفوفات رأسية vectors , ويمكنك ان تقوم بتدريب الشبكة المتكررة RNN عليها , لتستخدم في العديد من التطبيقات , مثل تحليل المشاعر في الكلام (معرفة هل الكلام سلبي ام ايجابي) , التعرف علي الأسماء , او الترجمة الأوتوماتيكية

-

بعدما تعلمنا من الاسبوع الماضي كيفية التعامل مع شبكات الـ RNN و ادوات GRU , LSTM , فسنستخدم هذه الأشياء في تطبيقات هامة في التعامل مع الكلمات , ومنها ما يسمي تضمين الكلمات word embedding .

و يقصد بها قدرة الخوارزم علي معرفة الكلمات المناسبة لبعضها البعض , وما الذي يتم استخدامه حينما يتم اختيار كلمة معينة , واختيار كلمات الذكور و الاناث دون تداخل , وكذلك تجنب اي استخدامات غير مناسبة مع العرق او الدين او النوع .

وكل هذا يعتمد أولا علي فهم : عرض الكلمات word representation

و لفهم هذا المصطلح , علينا ان نتذكر فكرة مصفوفة الكلمات .

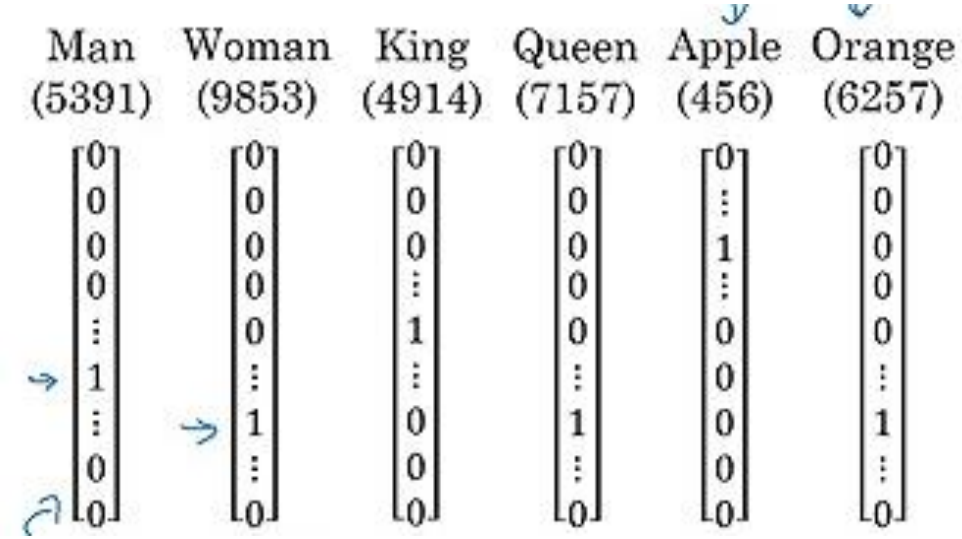
ذكرنا أن هناك قاموس بكل الكلمات , وليكن يحتوي علي 10 الاف كلمة , فإذا كانت كلمة man لها ترتيب رقم 5391 في القاموس , فستكون المصفوفة الخاصة بها هي :

Man
(5391)

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

و هي ما تسمي مصفوفة الـ one-hot , لذا يقال عن هذه المصفوفة ان اسمها O5391 حيث حرف الـ O يشير الي one-hot و الرقم هو ترتيب رقم 1 في المصفوفة .

و نفس الحال في باقي المصفوفات :



فكل مصفوفة فيهم يكون رقمها O ثم ترتيب رقم 1 فيها .

و المشكلة من هذه المصفوفات , ان الشبكة لن تتمكن من معرفة الكلمات المرتبطة ببعضها البعض عبر الارقام , فالشبكة لن تتمكن من معرفة ان كلمتي , king queen هما اقرب لبعضهما من كلمتي apple , queen

و حتي لو كان هناك جملة هي :

I love eating orange or ----------

فلن يتمكن الخوارزم من الربط بين البرتقال و التفاح لاختيارها هنا . و ستكون فرصة اختيار كلمة apple مساوية لكلمة king و هي بالطبع غير سليمة .

ولأن هذه الفكرة غير مناسبة , فسنقوم باستخدام فكرة أخري تسمي عرض الخصائص featurized representation

وتقوم الفكرة علي , عمل جدول ضخم الأعمدة فيه هي كل الكلمات الموجودة في القاموس , أما الصفوف فهي جميع الخصائص التي يمكن أن تتواجد في اي شئ في الدنيا

فقد يكون الجدول مثل هذا :

	Man 5391	Woman 9853	King 4914	Queen 7157	Apple 456	Orange 6257
Gender						
Royal						
Age						
Food						

Size						
Color						
Cost						

علي أن تكون الأعمدة فيها 10 الاف عمود , بينما الصفوف يها كل الخصائص الممكنة لأي كلمة , وقد تكون مثلا 300 خاصية .

و جميع النتائج تكون بين الـ 1 و الـ -1 , و تكون برقم يتناسب معها , فمثلا الكلمة الأولي :

	Man 5391	Woman 9853	King 4914	Queen 7157	Apple 456	Orange 6257
Gender	-1					
Royal	0.01					
Age	0.03					
Food	0.09					
Size	..					
Color	..					
Cost	..					

وباقي الكلمات قد تكون :

	Man 5391	Woman 9853	King 4914	Queen 7157	Apple 456	Orange 6257
Gender	-1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97
Size	..	..	..	..	..	..
Color	..	..	..	..	..	..
Cost	..	..	..	..	..	..

و بالتالي بدلا من أن يكون لـ vector الخاص بكلمة man هو :

0
0
0
1
0
0

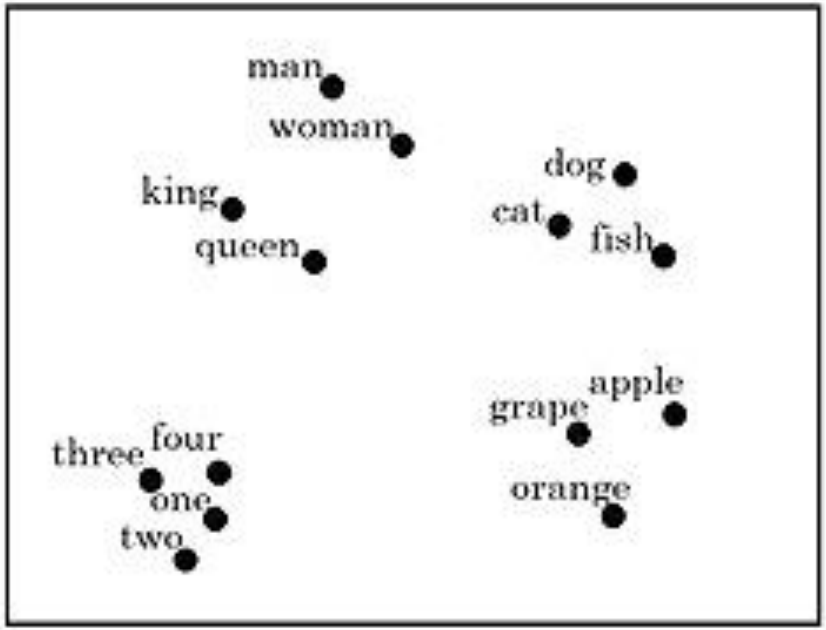
حيث رقم 1 ترتيبه 5391 , سيكون الفيكتور هو :

-1
0.01
0.03
0.09
..
..

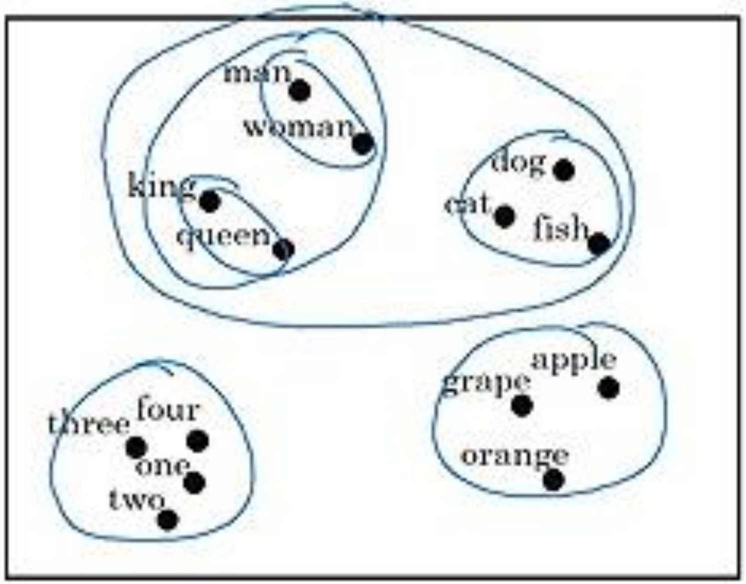
فيكون طول الفيكتور 300 رقم (بناء علي عدد الخصائص المستخدم) . و تكون هذه الأرقام أكثر تعبيرا عن مضمون الكلمة نفسه , وتقوم بمساعدة الخوارزم علي الربط بين الكلمات المتشابهة . لأن كلا من الفيكتور الخاص بالكلمات المتشابهة سيكون له ارقام قريبة من بعضها البعض .

ووقتها يسمي الفيكتور بدلا من O5391 سيكون e5391 و حرف الـ e دلالة علي embedding

ثم يمكن رسمها بيانيا كمخطط من بعدين هكذا :



ثم استخدام أحد وسائل التقسيم مثل t-SNE والتي تقوم بتقسيمها لمجموعات متقاربة من بعضها بحيث تجمع man , woman معا , ثم king , queen , ثم تجمع بين الاربعة معا , ثم تجمعهم كلهم مع باقي الحيوانات علي اعتبار انهم كائنات حية و هكذا


man
woman
king
queen
dog
cat
fish
apple
grape
orange
four
three
one
two

-

سنقوم الآن بتطبيق فكرة جدول الخصائص , لمعرفة كيفية استخدامها جيدا .

فإذا كان لدينا خوارزم لمعرفة اذا كانت الكلمة هي اسم شخص ام لا , وكان لدينا جملة في عينو التدريب :

Sally Johnson is an apple farmer

و قمنا بإعلام الخوارزم ان الكلمتين الأولي و الثانية قيمة y تساوي 1 (اسماء اشخاص) بينما قيمة y للباقي هي صفر (ليسو اشخاص)

فإذا قمنا في عينة الاختبار بإعطائه مثال وهو :

Robert Lin is an orange farmer

فيكون سهلا علي الخوارزم معرفة ان كلمتي orange & apple هما من نفس النوع و هما قريبتان من بعضهما , وبالتالي قد يتعرف علي ان اول كلمتين هما اسمين .

والميزة في طريقة جدول الخصائص , انه حتي الكلمات نادرة الاستخدام و التي قد لا تتواجد في عينة التدريب, ستكون معروفة .

فلو كان هناك جملة :

Robert Lin is a durian cultivator

الـ durian هي فاكهة غير منتشرة وموجودة أكثر في جنوب شرق اسيا , وكلمة cultivator هي بمعني مزارع لكنها غير شائعة , فميزة فكرة جدول الخصائص انه سيتمكن من معرفة ان كلمة durian هي من نفس نوعية orange & apple حتي لو لم يرها من قبل في التدريب

و هناك ثلاث خطوات أساسية لتطبيق هذا التكنيك :

1. القيام بتعلم تضمين الكلمات word embedding learn و الذي سيكون عبر التعامل مع كميات هائل من الكلمات , والتي قد تصل لـ 100 مليار كلمة , و التعلم منها لمعرفة خصائص كل كلمة (وقد نستغني عن هذه الخطوة في حالة تواجد تعلم مسبق و تحميله و العمل به)
2. تطبيق هذا التعلم في عينة التدريب في كمية قليلة من الكلمات , وليكن 100 الف
3. عمل تعديل في التعلم في الخطوة الأولي , من نتيجة تطبيق الخطوة الثانية , و هذه غالبا تتم اذا ما كانت العينة في الخطوة الثانية كبيرة .

-

كما رأينا في فكرة تضامن الكلمات word embedding و جدول الخصائص , ان لها عدد كبير من المميزات , و منها ما يسمي مسببات التشابه analogy reasoning وهي التي تختص بإيجاد المساحات المشتركة بين الكلمات بعضها البعض .

و لفهم مني التشابه , دعنا نلقي نظرة علي الجدول السابق شرحه :

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

لو كان لدينا سؤال معين عن : علاقة الرجل بالمرأة و مثل علاقة الملك بمن ؟

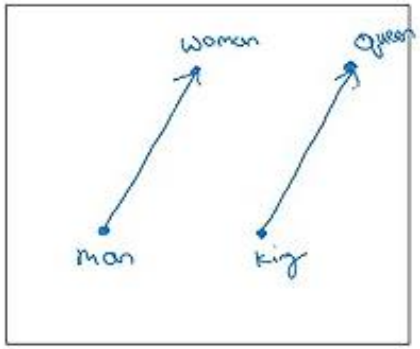
$$e_{man} - e_{woman} \approx e_{king} - e_?$$

فالطريقة الممكن استخدامها , عبر طرح مصفوفتين الرجل و المرأة معا في البداية , فتكون مصفوفة الرجل – مصفوفة المرأة ستساوي تقريبا :

2
0
0
0

ثم عمل تجربة طرح مصفوفة الملك من باقي المصفوفات , اختيار من يكون الفارق بينهما هونفس الفارق , وستكون الأرقام التقريبية مع الفارق بين الملك و الملكة

وبالتالي لو تم رسم النقاط الأربعة في جراف من بعدين , و تم إيصال الفيكتور من man to woman , ثم من king to queen هتجد ان الفيكتور الواصل بينهم (رياضيا يمثل الطرح) , قيمة متشابهة هنا و هنا



و هذا يتم بسهولة في بايثون , بإيجاد دالة تأتي بالتشابه الأكبر كالتالي :

find word w , where max sim(**e**w , **e**king – **e**man + **e**woman)

فقيمة الملك + الرجل – المرأة , ستكون قريبة جدا من الكلمة المطلوبة و هي الملكة .

ومعادلة التشابه تسمي معادلة cosine و تكون قيمتها الرياضية :

$$Sim(u , v) = \frac{uT\ v}{|| u ||\ \ ||v||}$$

و السبب في تسميتها بالـ cos أن هذا القانون يحسب قيمة الـ cos بين متجهين , كما أن لو كمان المتجهين متطابقين (الزاوية 0) تكون الـ cos تساوي 1 , لو كانو متعامدين (الزاوية 90) يكون الـ cos يساوي 9 , ولو كانو متضادين (الزاوية 180) تقوم القيمة سالب 1 , وهو ما يتوافق مع معني التشابهة similarity

و من الأمثلة المتنوعة لدالة التشابه :

Man – woman ===== boy – girl
Ottawa – Canada ===== Nairobi – Kenya
Big- bigger ===== Tall – Taller
EN – Japan ===== Ruple-Russia

-

حينما يتم تدريب خوارزم خاص بالتحليل اللغوي , وبالتحديد في مجال تضمين الكلمات word embedding تظهر لنا ما يسمي مصفوفة التضمين embedding matrix , فما هي . .

كنا ذكرنا بالفعل أنه يجب تحضير مصفوفة ضخمة عن كل الكلمات المستخدمة , وان يكون لكل كلمة عدد كبير من الخصائص , مثل هذه :

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

و عدد أعمدة هذه المصفوفة هي نفس عدد الكلمات في القاموس , وليكن 10 الاف , بينما عدد صفوفها هي عدد الخصائص التي تم تحديدها , وليكن 300 خاصية .

و هذه المصفوفة تسمي E بالكابيتال , وتكون عامة لكل الكلمات .

و اذا تذكرنا مصفوفة الـ one-hot و الخاصة بكل كلمة , والتي فيها مثلا لكلمة orange يكون هناك عمود طويل طوله 10 الاف صف بعمود واحد , ويكون كل الارقام اصفار عدا ترتيب محدد يتناسب مع الكلمة , وليكن هنا 6257 يكون فيه رقم 1 , ويكتب هكذا : O_{6257}

و المهم أنه لم تم ضرب المصفوفتين معا تكون كالتالي :

$$E \times O_{6257}$$

نلحظ أن المصفوفة الأولي هي 300 صف في 10 الاف عمود , بينما الثانية 10 الاف صف في عمود واحد , فيكون الناتج 300 صف في عمود واحد .

و رياضيا , سيكون هذا العمود هي كل قيم الخصائص الخاصة بالـ orange لانه مع ضرب E في O_{6257} ستجعل الاصفار كل القيم بأصفار , عدا العمود الخاص بالكلمة وحدها .

و هذه المصفوفة الناتجة تسمي , embedding matrix و يرمز لها بالرمز e سمول , وتكون ملحقة بالرقم , فمثلا : e_{6257}

و رياضيا لاختصار الوقت , بدلا من ضرب المصفوفتين معا و استغراق وقت طويل فيها , ممكن ان يتم استخدام دالة معينة لاستقطاع العمود من E الخاص بالرقم المطلوب

ومن الممكن ان يتم افتراض جميع قيم المصفوفة E الكبري بارقام عشوائية , ثم عمل تدريب للخوارزم علي عدد هائل من الكلمات , حتي نصل للقيم المثلي فيها

-

إذن كيف يتم التدريب ؟

إذا ما كان لدينا جملة مثل هذه , ونريد توقع الكلمة القادمة :

I	want	a	glass	of	orange	_____.

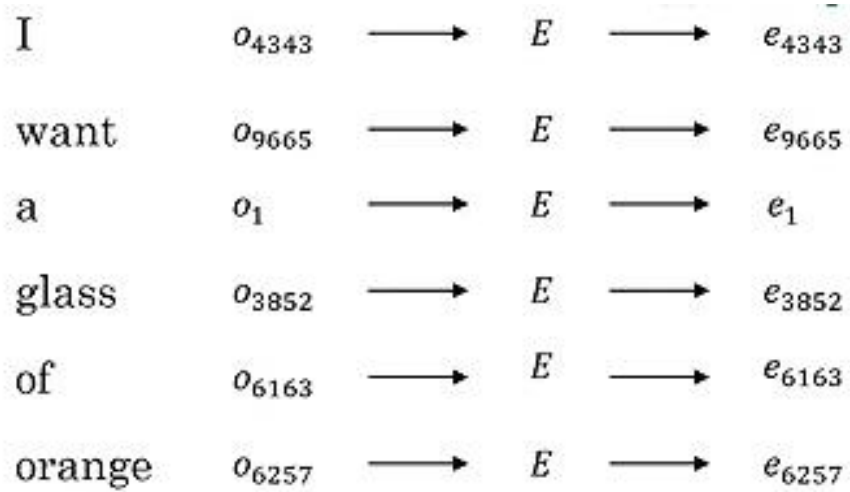
حيث لكل كلمة فيهم رقم محدد , فتكون :

I	want	a	glass	of	orange	_____.
4343	9665	1	3852	6163	6257	

فيبدأ الأمر , بإيجاد مصفوفات one-hot لكل كلمة علي حدة :

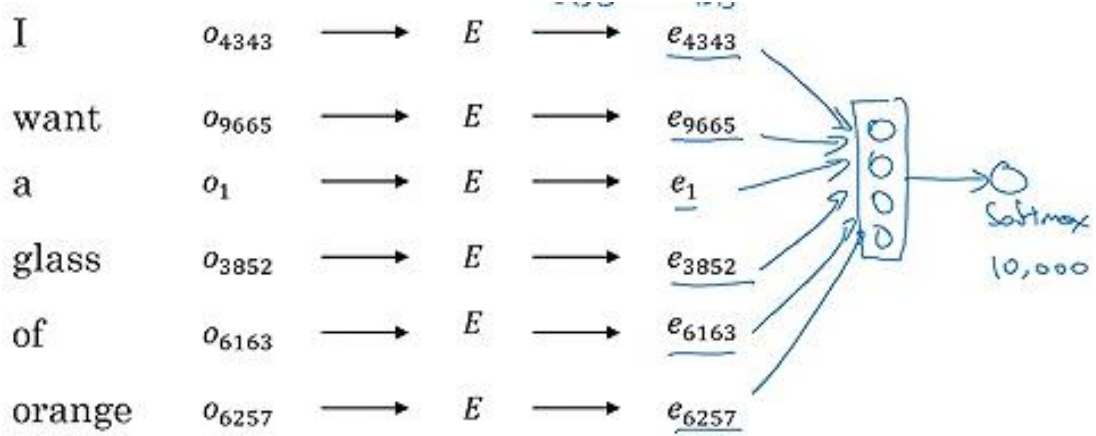
I	o_{4343}
want	o_{9665}
a	o_1
glass	o_{3852}
of	o_{6163}
orange	o_{6257}

ثم ضرب المصفوفة الكبري E في فيكتور الـ one-hot لكلا منهم , لإيجاد الـ embedding matrix لكل كلمة هكذا :



بحيث تكون كل embedding matrix لكل كلمة e مصفوفة 300 صف في عمود واحد .

ثم نتناول هذه المصفوفات و ندخلها معا في شبكة عصبية (طبقة واحدة أو أكثر) و منها نخرج لـ softmax لقوم بالاختيار من وسط 10 الاف اختيار (عدد كلمات القاموس) هكذا :



و عبر بناء هذا الخوارزم , وتدريبه جيدا مع عشرات الالاف من النماذج , سيقوم بتحديد قيم لثيتات الخاصة بالشبكة و السوفت ماكس , لتوقع اي كلمة قادمة .

و غالبا ما يتم تحديد عدد معين من الكلمات ليتم علي اساسها توقع الكلمة التالية , فمثلا انت تقوم بتحديد أنني ساستخدم فقط 4 كلمات سابقة للكلمة الفارغة لاحدد هذه الكلمة , فتكون في هذا المثال :

a	glass	of	orange	_____.
1	3852	6163	6257	

و أهمية هذا التحديد , هو التمكن من بناء الخوارزم بشكل صحيح , لأن عدد المعاملات w , b التي ستكون في الشبكة العصبية معتمدة علي عدد المدخلات inputs , وغير منطقي ان تكون مفتوحة

فإذا كان لدينا 4 كلمات سيتم استخدامهم , ولكل كلمة 300 رقم , فيكون عدد المدخلات هو 1200 , ويتم رصهم فوق بعضهم البعض , فيكون هناك 1200 ثيتا سيتم تدريبها في الشبكة المطلوبة .
و في حين هذا التصور هو الشكل الرسمي او المثالي , فهناك تصورات أخري قد تكون أكثر بساطة , وفينفس الوقت بنفس الفعالية الخاصة بها .

فمثلا لو كان لدي جملة مثل :

I want a glass of orange --------- to go alone with my cereal

وبفرض كانت كلمة juice هي المجهولة و المطلوب استنتاجها , فالشكل الاصلي ان نختار الكلمات الأربعة الأخيرة هكذا :

a glass of orange ---------

إلا أنه يمكن عمل المزيد من الدقة (والوقت) باختيار كلمات اربع قبل و كلمات اربع بعد مثل :

a glass of orange --------- to go alone with

و قم يتم اختيار كلمة واحدة فقط قبلها :

orange ---------

وقد يتم اختيار كلمة محددة قبلها , ليس بالضرورة السابقة لها , مثل كلمة glass

* **** * glass ** ****** ---------

وهو التكنيك المسمي skip gram والذي سنتكلم عنه لاحقا

-

و الفكرة هنا تسمي word2vec و القائمة علي المبدأ الذي سبق ذكره وهو : skip gram

و تقوم الفكرة علي اختيار الكلمة التي سنستدل بها علي الكلمة الناقصة , و تسمي كلمة المضمون context و الكلمة الناقصة نفسها و نسمها كلمة الهدف target .

ثم نتناول الجملة و ليكن :

I want a glass of orange juice to go alone with my cereal

لاحظ ان كلمات المضمون و الهدف قد تختلف , فقد تكون قريبة من بعضها او تالية بعيدة عن بعضها البعض , لكن غالبا ما تكون في نطاق 10 كلمات , اذ نادرا ما تؤثر كلمات في مسافة ابعد من هذا

فيتم إيجاد مصفوفة المضمون embedding matrix الخاصة بكل كلمة , وهي التي لها رمز e مثل **e**6257 , وهي التي تأتي من ضرب مصفوفة الخصائص E في مصفوفة الـ one-hot .

ثم يتم حساب قيمة P(t | c) أي حساب احتمالية كلمة الهدف t اعتمادا علي معلومية كلمة المضمون c , وتكون بالقانون :

$$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$$

نبدأ بشرحه جزء جزء .

أولا علي اليسار القيمة :

$$p(t|c)$$

هي احتمالية الحصول علي الكلمة الهدف t بناء علي معلومية الكلمة المضمون c

بالنسبة للبسط :

$$e^{\theta_t^T e_c}$$

فحرف e الاسفل هو الـ exponential اي 2.71 , و الأس الخاص به هو ثيتا الخاصة بالكلمة الهدف t بعد عمل مقلوب لها T مضروبة في ec وهي مصفوفة المضمون embedding matrix الخاصة بالكلمة المضمون c

بينما المقام

$$\sum_{j=1}^{10,000} e^{\theta_j^T e_c}$$

هو مجموع كل expo علي حدة للكلمات العشر الاف , حيث كل واحدة expo لها أس هو الثيتا الخاصة بها مقلوبة T مضروبة في ec وهي الكلمة المضمون

بالنسبة للثيتا t هي الأوزان الخاصة بكل كلمة في الجملة المعطاة للخوارزم

والفكرة تقوم علي قسمة بسط علي مقام حيث البسط يوضح مدي ارتباط كلمة المضمون بكلمة الهدف , بينما المقام هو مجموع كل العلاقات بين كلمة المضمون و باقي الكلمات في اللغة .

أما معادلة الخطأ المطلوب تقليلها هي :

$$L(\hat{y}, y) = -\sum_{i=1}^{10,000} y_i \log \hat{y}_i$$

مع ملاحظة ان قيمة y هنا هي الناتج , وهي مصفوفة الـ one-hot الخاصة بالكلمة الناتجة .

وهذا التكنيك يسمي skip gram , لأنه قد يتخطي عدد من الكلمات قبل اختيار الكلمة المناسبة , فليس شرط ان تكون تالية او سابقة لها

و من عيوب هذه الطريقة الوقت المطلوب لها , حيث ان المقام يحتاج لعشر الاف عملية حسابية , و تكون المشكلة اعمق اذا ما كان القاموس يحتوي علي عدد ضحم و ليكن مليون كلمة .

فيكون أحد الحلول استخدام التصنيف الهرمي hierarchical softmax classifier , وهو الذي يجعل الأمر اسرع .

ويبدأ عبر جعل الكلمات العشر الاف في قسمين كل قسم 5 الاف , فيقوم الخوارزم بفحص هل هي في القسم الاول ام الثاني , فإذا كان في الاول يقوم بفحص هل هي من الـ 2500 الاول ام الثانية , وهكذا حتي نصل للكلمة المطلوبة , في عد اقل من العمليات , وبوقت اسرع .

و لا يشترط ان يكون التقسيم متوازن تماما او متماثل symmetric بل يفضل في الاساس ان يتم وضع الكلمات الاكثر شيوعا (the , of , who , he , she) في الطبقات العليا , والكلمات الاقل شيوعا تحت , حتي نوفر الوقت في التعامل مع الكلمات الكثيرة الاستخدام

أخيرا , ماذا عن اختيار كلمة المضمون c

معروف ان كلمة الهدف t تكون علي بعد 10 كلمات بحد اقصي من كلمة المضمون , لكن كيف يتم اختيار كلمة المضمون c نفسها ؟

ينصح دائما بتجنب التعامل مع الكلمات الاكثر شيوعا (he , she , of , a , an , and) والتركيز فقط علي الكلمات النادرة نوعا , وذلك لأن الكلمات الاكثر شيوعا هي موجودة في كل انوع النصوص , ولن يكون لها دور حقيقي في تمييز الكلمات , او معرفة الكلمة الناقصة , بينما الكلمات النادرة نوعا هي ما تحددها بدقة

-

كنا قد رأينا في التكنيك السابق skip gram أنه فعال في توقع الكلمات , لكن يعيبه البطئ الشديد , سنتعرف الآن علي تكنيك العينة السلبية negative sample و الذي يعطي نفس الكفاءة و في وقت أقل .

و تقوم الفكرة علي عمل جدول , يضم ثلاث عواميد , الأول هي كلمة المضمون context التي سنتعامل معها , الثاني ستكون كلمات سنقوم باختبار مدي ارتباطها بكلمة المضمون , والثالث هو نتيجة هذا الارتباط

وتكون الكلمة الأولي في الـ word هي الكلمة المرتبطة بكلمة المحتوي , لذا تكون قيمة الـ target تساوي 1 , بينما يتم اختيار عدد اخر من الكلمات بشكل عشوائي من القاموس , بحيث تكون قيمة الـ target تساوي 0 , هكذا

Context	Target	y
Orange	juice	1
Orange	king	0
Orange	book	0
Orange	the	0
Orange	of	0

و يكون عدد الكلمات العشوائية هو k و الذي يكون من 5 الي 20 كلمة في العينات ذات العدد القليل , بينما من 2 – 5 في الأعداد الكبيرة .

ثم يتم إعطاء هذه البيانات للخوارزم ليتدرب عليها , علي اعتبار ان أول عمودين هما المدخلات x بينما العمود الثالث هو المخرج y

ويسمي هذا التكنيك العينة السلبية , لأننا نقوم باختيار كلمات غير مرتبطة بكلمة المضمون وذات ارتباط سلبي بها

و هنا تكون المعادلة الخاصة بالاحتمالية :

$$P(y=1 \mid c,t) = \sigma\ (\Theta_t^T\ \mathbf{e}c)$$

وهو الذي يعني أننا نأتي بثيتا تي , ويتم عمل مقلوب لها , ثم ضربها في **e**c ثم عمل سيجمويد لكل هذا .

و الميزة في هذا التكنيك عن السابق , ان عدد البيانات التي يتم تدريب الخوارزم عليها يكون عدد قليل وهو k+1 (اي عدد العناصر الغير سليمة + 1 وهو العنصر السليم)

فبدلا من جعل الخوارزم يتدرب علي 10 الاف كلمة بوقت رهيب , اجعله يتدرب علي كلمات عشوائية غير مرتبطة به , هو ما يجعل الوقت اقل

و لاختيار الكلمات العشوائية , يتم استخدام صيغة معقدة نوعا لكن مؤثرة وهي :

$$P(\omega_i) = \frac{f(\omega_i)^{3/4}}{\sum_{j=0}^{10,000} f(\omega_j)^{3/4}}$$

-

نتعلم الان تكنيك مختلف في نفس مجال NLP و هوا ما يسمي تكنيك GloVe , وعلي الرغم من ان انشاره و شهرته اقل من تكنيك word2vec إلا انه ابسط و له استخدامات عديدة

و اسمه GloVe هو اختصار :

Global Vector for word representation

و تبدأ الفكرة بالتعرف علي مصطلح X_{ij}

و تعريفه هو عدد المرات التي ظهرت فيها الكلمة i في نطاق الكلمة j , وكلمة في نطاق المقصود بها في خلال العدد المحدد من الكلمات و ليكن 10 كلمات .

كأن الكلمة i تعبر عن الهدف t بينما الكلمة j هي المضمون c , وهو ما يجعل هذا التكنيك مبنيعلي نفس اساس التكنيك السابق , وهو إيجاد العلاقة بين كلمتي المضمون و الهدف .

و غالبا ما يكون Xij هي نفسها Xji , حيث انه لم تم استبدال مكاني كلمتي الهدف و المضمون ستكون القيمة نفسها .

و يكون الهدف هو تقليل القيمة بين ثيتا i (مقلوب) مضروبة في ej , وبين لوج قيمة Xij , لان اقتراب القيمتين من بعضهما يعني ان الكلمتين مرتبطان ببعضيهما

$$\left(\theta_i^T e_j - \log X_{ij}\right)^2$$

ويتم عمل الصيغة الكاملة :

$$\text{minimize} \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij})\left(\theta_i^T e_j - \log X_{ij}\right)^2$$

مع اعتباران علامتي sum في البداية لمسح جميع الكلمات و علاقاتها بباقي كلمات القاموس .

أما الجزء الخاص بالدالة f(Xij) فله مهمتين .

الأولي انه سيساوي صفر في حالة كانت Xij تساوي صفر ,وقيمة Xij تساوي صفر معاناها عدم وجود اي كلمة هدف في نطاق كلمة المضمون .

و أهمية جعل الـ f(Xij) تساوي صفر لتجنب المشكلة الرياضية الناتجة من حساب log للصفر , والذي سيساوي سالب مالانهاية , فوجود الصفر في البداية يجعل القيمة بصفر و يجنبنا المشكلة الرياضية .

كما ان الدور الثاني بها متعلق بتحديد اوزان للكلمات حسب اهميتها و انتشارها في اللغة الانجليزية , فكلمات منتشرة مثل (he , she , a , of , and) يكون لها اوزان كبيرة , بينما الكلمات النادرة يكون لها اوزان اقل , وهو ما يجعل الخوارزم اكثر دقة في التعامل معها

أخيرا يتم وضع معامل bias لكلا من i , j والذي سيكون هكذا :

$$\text{minimize } \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij})(\theta_i^T e_j + b_i - b_j' - \log X_{ij})^2$$

-

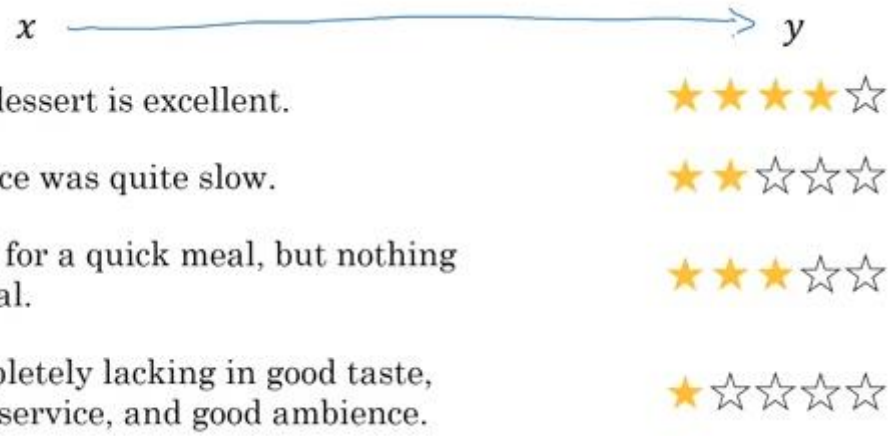
نتعرف الان علي تطبيق هام من تطبيقات NLP وهو ما يسمي تصنيف الانطباع sentiment classification

كلمة sentiment تعني مشاعر , و يقصد بها هنا : التعرف علي انبطاعات و آراء كاتب التعليق , من كتابته .

فلو كان هناك الاف التعليقات علي منتج معين او فيلم ما , فنريد عمل ترجمة لها لارقام , لمعرفة كاتب هذا التعليق يقصد تقييمه بكم من 5 مثلا . او علي الأقل هل هو مؤيد لو رافض له .

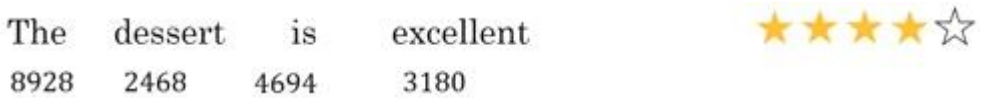
كما ان له استخدامات سياسية , في تناول اعداد هائلة من التويتات , وتحليلها لمعرفة نسبة كم منها مؤيد لترامب و كم منها رافض له .

و هنا امثله لها :



و أكبر تحدي في التعامل معها , أننا لا نملك كمية كبيرة من بيانات التدريب لتدريب الخوارزم عليها , فيندر ان تجد اكثر من 10 الاف تعليق عن كتاب او مطعم معين , و هذا رقم متواضح يصعب الإعتماد عليه .

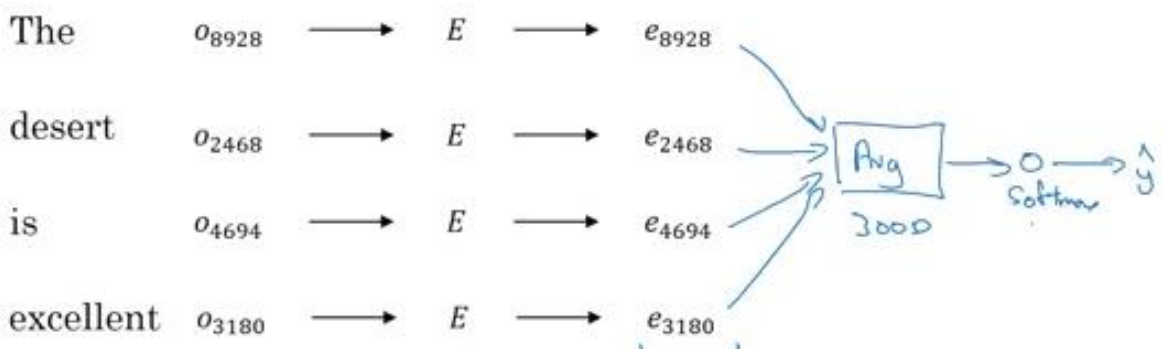
و نبدأ بتناول التعليق المكتوب كمدخل x بينما عدد النجوم هو المخرج y هكذا :



و تكون الخطوة الأولي بحساب صفوفة المضمون embedding matrix لكل كلمة (حاصل ضرب الـ E في مصفوفة one-hot)

فيكون لدينا 4 مصفوفات (عدد 4 معتمد علي عدد الكلمات) و كل واحدة عمود واحد مع 300 صف (عدد الخصائص)

ثم نقوم بإيجاد مجموع هذه الارقام , او متوسطها , وإدخالها في شبكة عصبية , لتخرج لنا في سوفت ماكس بقيمة من 5 قيم



وكأن هذا النظام مع تدريبه , يقوم بتعليم الخوارزم ان انتشار كلمات مثل كذا و كذا , يعني نتيجة كذا , فيتمكن من قراءتها بسهولة و تقييمها

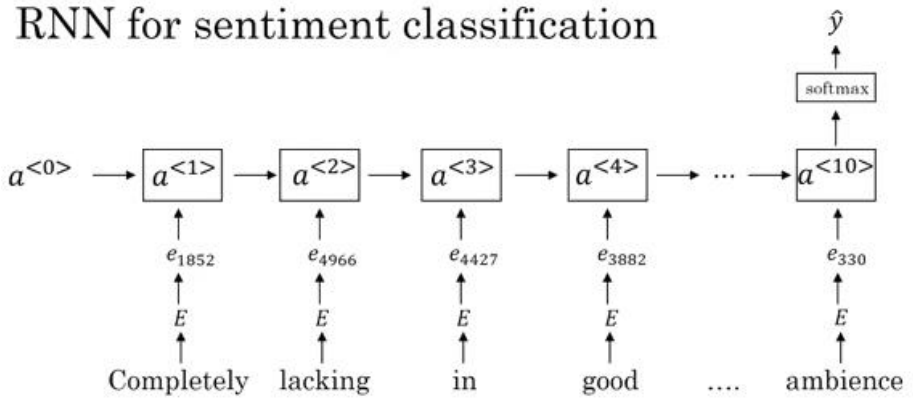
و من أهم عيوب هذه الطريقة , انه يمكن خداعها بسهولة , فمثلا جملة مثل :

Completely lacking in good taste,
good service, and good ambience.

من الواضح ان التعليق سلبي تماما , إلا أن انتشار كلمة good , taste , service سيجعل الخوارزم يميل في كفة التقييم العالي , مع أن كلمة lacking قد قلبت المعني .

فمشكلة ايجاد المجموع او المتوسط انه يفحص عدد وجود كلمة ما , بغض النظر عن معناها الإجمالي .

فيكون الحل هنا هو استخدام RNN من نوعية many-to-one حينما نقوم بتغذيتها بكل الكلمات , وتقوم بإخراج مخرج واحد مثل هذا :



وميزة الـRNN هنا ان كل الكلمات تتداخل معا في اتخاذ القرار , و هنا سيتعلم الخوارزم ان not good شئ سلبي , وأن lack of good taste تقلب المعني , وذلك لأن طبيعة الـ RNN تجعل المخرج يعتمد علي جميع المداخل , وليس علي متوسط مثل التكنيك السابق .

و الميزة هنا انه حتي لو تم استخدام كلمة سلبية غير منتشرة مثلا (absence of good taste) حتي لو لم يتم استخدامها اثناء التدريب , فاقتراب معناها من معني lack سيجعل الخوارزم يفهم المقصود .

و الخوارزم سيفهم انها قريبة منها اعتمادا علي مصفوفة المضمون embedding matrix و التي تشتمل علي الخصائص .

-

الموضوع الأخير هذا الأسبوع , متعلق بعلاج الأخطاء العنصرية في المعالجة اللغوية .

و المقصود بها تجنب اي تصنيف او اختيار , بما يدعم فكرة التفرقة العنصرية , او الصورة النمطية الخاصة بالنوع , او العرق , او الدين , او التوجه الجنسي .

فحديثنا الان ليس عن خطأ الاختيار انه غير صحيح , لكن عن الاخلاقيات الواجب مراعاتها للآلة حينما تقوم بصياغة الجمل .

كمثال بسيط , لو قلنا أن :

رجل == امرأة يساوي ملك== ملكة

فإذا ذكرنا أن : رجل == مبرمج يساوي امرأة == ؟

حينما قامو بتدريب الخوارزم اكتشفو نتائج مخيفة , ان اغلب الاستنتاجات كانت لاشياء مثل : ربة منزل , عاطلة .

كذلك . .

أب == طبيب يساوي أم == ؟

كانت النتيجة : ممرضة . .

و المشكلة هنا ليست مشكلة تكنيكال , لكن أن البيانات التي يتم تدريب الخوارزم عليها هو فيه الكثير من العنصرية , والأزمة أن الكلام الحقيقي الموجود في كل مكان فيه الكثير من العنصرية , ولن نستطيع اختراع بيانات للتدريب من الهواء .

و سنتناول الان فقط مشكلة العنصرية مع النوع : ذكر و أنثي , ولكن فكرة الحل كون نفسها مع اي نوع أخر

و لعلاج المشكلة , نبدأ بأولا باستعراض أماكن عدد من الكلمات المستخدمة و التي تداخل بشكل ما في المشكلة مثل :

doctor , babysitter , boy , girl , he , she , grandfather , grandmother

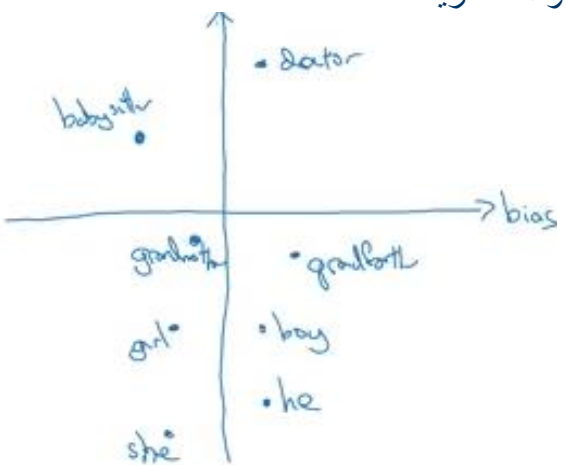
و نري أن هذه الكلمات قد يتم وضعها هكذا , بناء علي التدريب من كمية كبيرة من البيانات , والتي قد يكون بها بعض العنصرية :

و مصدر المشكلة رياضيا , انه مع مقارنة she—he تكون المقارنة قريبة جدا من doctor—babysitter وهو ما يجعل الخوارزم يتخذ قرارات عنصرية , ونريد ان نقوم بتغيير هذا الأمر

و يكون حل المشكلة في خطوات ثلاث .

الأولي : في تحديد اتجاه الإنحراف (العنصرية)

فيكون المحور الأفقي هنا هو محور الإنحراف , لأنه يتحول من اليسار إلي اليمين لما يتماشي مع اختلاف الجنس , بينما المحور الرأسي هو اللا انحراف , لأن الفارق بين كلمتي she , girl ليس متعلق بالإنحراف او العنصرية



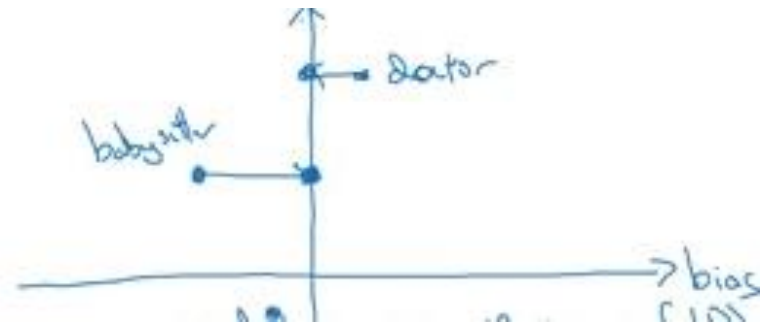
تأتي الخطوة الثانية : في التفريق بين نوعين من الكلمات .

الكلمات الي يكون تعريفها معتمد أساسا علي الفارق في النوع , مثل grandmother , فهي في الأساس مبنية علي أنها أنثي , كذلك she و كذلك : girl و wife

وكلمات أخري تعريفها لا يعتمد علي النوع , مثل doctor , programmer , babysitter , nurse , فهذه الكلمات ليس لها علاقة بالنوع من الأساس .

فنمسك كلمات النوع الثاني , و نقوم بتحريكها حتي تسقط علي المحور الرأسي , وذلك حتي يكون ليس لها تأثير مع اختلاف النوع

فتتتحرك كلمة doctor من اليمين لليسار و كلمة babysitter من اليسار لليمين هكذا

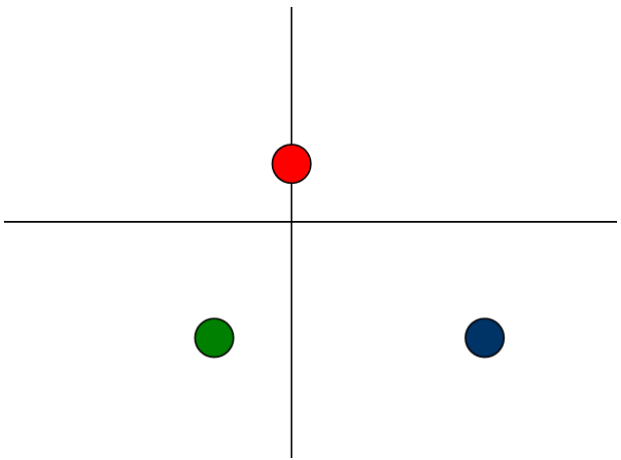


ثم تأتي الخطوة الثالثة وهي تناول الكلمات من النوع الأول (المعتمدة في تعريفها علي النوع) وتطبيق: تساوي النوعين في المسافة .

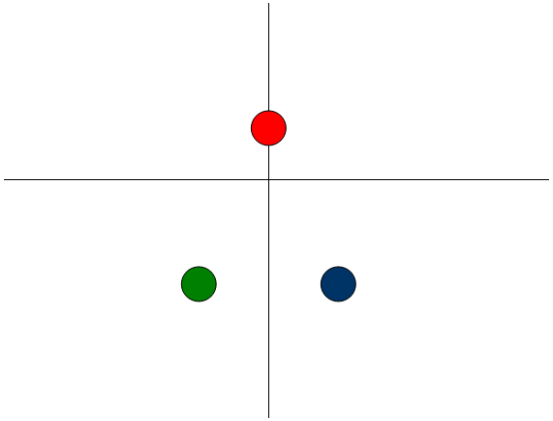
لو كانت كلمة he هي الزرقاء , و كلمة she هي الخضراء , وتم بالفعل تغيير مكان كلمة babysitter الحمراء من اليسار حتي تقع علي المحور الرأسي (لتجنب تصنيفها) .

فحتي مع هذا الأمر , لازالت كلمة she اقرب لكلمة babysitter من كلمة he و هو ما يجعل اختيارها اقرب للأنثي منها الي الذكر , فالمشكلة قائمة .

و لن نستطيع جعل كلا من he , she علي المحور الراسي لان هذا سيحدث خلل في التعامل اللغوي من الاساس , اذ ان التفريق في النوع بينهم شئ هام



فيكون الحل , ضبط مكانيهما ,بحيث يكون كلا منهما علي نفس المسافة من المحور الرأسي , مع الإبقاء أن هذا يمين و هذا يسار , وهو ما يجعل المسافة بين كلا من he , she (الكلمات المعتمدة علي اختلاف النوع) هي نفسها من كلمة لا تعتمد علي اختلاف النوع



و يتم تكرار هذه المساواة في المسافة بين كل كلمات مناظرة لبعضها مثل :

uncle , aunt
niece , nephew
daughter , son
sister brother

و هكذا
و أخيرا , يمكننا تحديد ما هي الكلمات الواجب تمييزها (boy , girl) من الكلمات الممنوع تمييزها (doctor) عبر مصنف يقوم بالبحث في جميع الكلمات ,
وسنجد أن أغلب الكلمات غير مطلوب تصنيفها

-