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 الأول : مفهوم الشبكات المتكررة

==========

نتعرف الان في الكورس الخامس , علي ما يسمي : نموذج التتابع sequence model , ومنه الشبكات المتكررة RNN , وهو المستخدم بكثر في العديد من تطبيقات الـ DL مثل التعرف علي الصوت و غيرها

و يسمي هذا النوع : نموذج التتابع , لأن البيانات التي يتم التعامل معها سواء كـ inputs or outputs هي دالة في الزمن , مثل الملفات الصوتية , او الكلام مكتوب .

و من أمثلة التطبيقات المعتمدة علي نموذج التتابع :

- تحويل الصوت الي كلام :
  - يتلقي ملف صوتي و يترجمه إلي text
- إنتاج الموسيقي :
  - وهي الخوارزميات التي تقوم بانتاج قطع موسيقية كاملة بالشبكات العصبية

- استنتاج التقييم sentiment classification :
  - وهي إمكانية استنتاج تقييم معين للمنتج بناء علي التعليقات, فلو كتب شخص في تعليق كلمة "زفت" فيتم استنتاج تقييم اوتوماتيك لها

- تحليل الـ DNA
  - وهي تكون عبر تحليل الرموز المعقدة لشفرة الـ DNA الي مصطلحات بسيطة

- الترجمة الآلية
  - أي الترجمة من لغة إلي لغة عبر خوارزم محدد

- تحليل فيديو
  - لمتابعة كل اطار من الفيديو و تحليله و التعرف علي الاشخاص به

- التعرف علي الاسماء
  - مثل ان يتم تحليل نصوص كبيرة , وتحديد اي اسماء اشخاص فيه و الاشارة لها

فتطبيقات نموذج التتابع sequence model كثيرة , وتختلف في شكلها و نوعها , فممكن ان تجد ان المدخل هو قيمة تتابعية و المخرج ليس كذلك او العكس , كما ان طول المدخل و المخرج قد يتساوي و قد يختلف

-

و للتعرف علي مفهوم : نموذج التتابع , علينا ان نشرح بالتفصيل أحد تطبيقاته , وليكن التطبيق الاخير المذكور في الفيديو السابق , وهو المسمي : مدي تواجد الاسماء named-entity , وهو الذي يقوم بالبحث في نصوص عديدة , عن اسم اي شخص و الاشارة اليه , لاحظ ان هذا الأمر ليس سهل فهو لا يبحث عن ام شخص معين بآلية search عادية (مثلا يبحث عن اسم دونالد ترامب بالتحديد), ولكن يبحث عن اي اسم حتي لو لم يعرفه الخوارزم , ليكون الخوارزم جاهز لاي اسماء .

و غالبا ما يستخدم في محركات البحث , حتي يقوم اوتوماتيك بتحديد من هم الساسة و الفنانين و رجال الأعمال الذين تم ذكرهم في الاخبار , وسردها بطريقة معينة .

و بالطبع نفس الأمر ينطبق علي اسماء شركات , اسماء دول , اسماء مدن , عملات , كتب , وهكذا .

و من الطرق التقليدية (والقديمة) لتفعيل هذا الأمر , ان يكون المدخل هو نص كلامي , ويكون المخرج هي عبارة عن سلسلة من الارقام 0 او 1 , لكل كلمة , حيث 0 هو ليس اسم , و 1 هو اسم

فمثلا لو كان النص :

Yesterday Mr Donald Trump visited the French president Marconi

ستكون مصفوفة المخرج

0 0 1 1 0 0 0 0 1

و من الممكن ان يكون التصنيف لأكثر من متغير , حيث 1 اسم شخص , 2 اسم مدينة , 3 اسم عملة , 4 اسم كتاب , 0 شئ مختلف

فلو كانت الجملة

yesterday peter went to NYC library to get ML book of Andrew with 5 dollar

فيكون المخرج

0 10 0 2 0 0 0 4 0 0 1 0 0 3

إلا أن هذه الطريقة قديمة نوعا, وحاليا هناط طرق أحدث في تحديد اسم الشخص او المدينة او البلد , ومكانها في النص و تكرارها و هكذا

و لبدء التعامل مع هذا النظام (القديم) , فلو قلنا ان هذه هي جملة المدخل :

Harry Potter and Hermione Granger invented a new spell.

فنقوم بتحديد features لكل كلمة فيهم , و ان يكون رمزها x < t > مع اختيار الاقواس الزاوية هذه لتفريقها عن باقي الاقواس

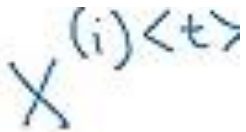
Harry Potter and Hermione Granger invented a new spell.

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ... $x^{<9>}$

كلك في المخرج , سنقوم بتحديد قيم y بنفس الارقام و نفس الاقواس الزاوية للقيم المخرجة

كذلك سنقوم بتحديد Tx للدلالة علي عدد كلمات المدخل (هنا 9) , Ty للدلالة علي عدد المخارج (هنا كذلك 9 لكن في تطبيقات اخري قد لا تساوي عدد المدخل)

ولا تنس أن الاقواس الدائرية تستخدم للدلالة علي عنصر محدد من عناصر العينة , لذا فإن :

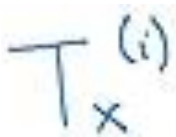


معناها الكلمة رقم t في العنصر رقم i

فلو كانت x(7)<12> فهي معناها الكلمة الثانية عشر في العنصر السابع من عناصر العينة .

ونفس الفكرة في y فلو كان لدينا y(16)<7> تساوي 0 فهذا معناه ان الكلمة السابعة في العينة السادسة عشر هي ليست اسم

و لأن كل عنصر من عناصر العينة قد يحتوي عدد كلمات مختلف , فهناك رمز :



وهو الذي يدل علي عدد كلمات العنصر رقم i

فلو كان لدينا Tx(20) = 16 فهو معناها ان عدد كلمات العينة رقم 20 هي 16 كلمة .

كذلك Ty(14) = 22 معناها ان مصفوفة مخرج العينة 14 هي 22 رقم اصفار ووحايد .

و كل هذه التفاصيل هي في قلب ما يسمي NLP او معالجة اللغات الطبيعية Natural Language Processing

إذن كيف يقوم الخوارزم بقراءة هذه الجملة و التعرف عليها ؟

أول خطوة , هو ان تقوم انت كمبرمج بتكوين قاموس كبير خاص بك , يحتوي علي كل كلمات اللغة الإنجليزية التي قد تستخدمها (ليس بالضروري كل الكلمات بالفعل , ولكن كل الكلمات التي ستستخدمها هنا) .

فبفرض اننا سنقوم باستخدام 10 الاف كلمة مشهورة , فيتم رصهم جميعا في قاموس كبير بحيث يكون لكل كلمة رقم محدد يبدأ من 0 و تزيد :

a	0
aaron	1
.	.
.	.
and	367
.	.
harry	4075
.	.
potter	6830
.	.
zulu	10000

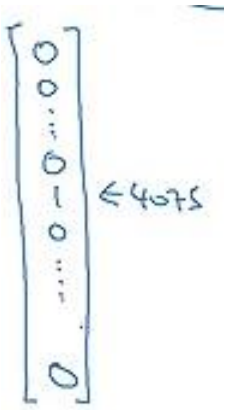
و قد تقوم انت باستخدام قاموس جاهز بالفعل , او قد تقوم بصناعة قاموس خاص بك , في حالة كانت القواميس الموجودة لا تفي باستخداماتك , فمثلا لو كنت تستخدم مصطلحات طبية عديدة و ليست موجودة في القواميس المتوافرة , فيمكنك عمل كود بسيط يبحث في ملايين السطور المتوافرة لديك , لايجاد كل الكلمات و مسح المتشابه , و رصها بالترتيب و إعطائها أرقام , ونفس الفكرة لو كنت ستقوم بعمل قاموس للغة أخري (مثل العربية) .

و أحيانا يتم عمل خانة تسمي <UNK> و هي دلالة عن كلمة unknown و هي تكون للكلمات الغير موجودة في القاموس لديك

ثم يقوم الخوارزم بعمل ما يسمي : مصفوفة 1 الساخنة one-hot vector , وهي المصفوفة ذات العمود الواحد , التي تكون كل قيمها باصفار و ما عدا قيمة واحدة فقط , وتكون هذه القيمة برقم (1) بالتحديد

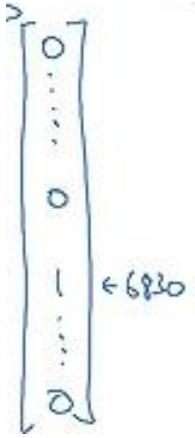
وهي أن يتم عمل مصفوفة عمود واحد , وبها 10 الاف صف (او عدد الصفوف الذي سيتم تحديدة لطول القاموس الخاص بك) , وان تكون كل القيم اصفار , عدا القيمة الموجودة في مكان الكلمة نفسها تكون بواحد

فيقوم الخوارزم باستبدال كلمة harry في اول الجملة بالمصفوفة :

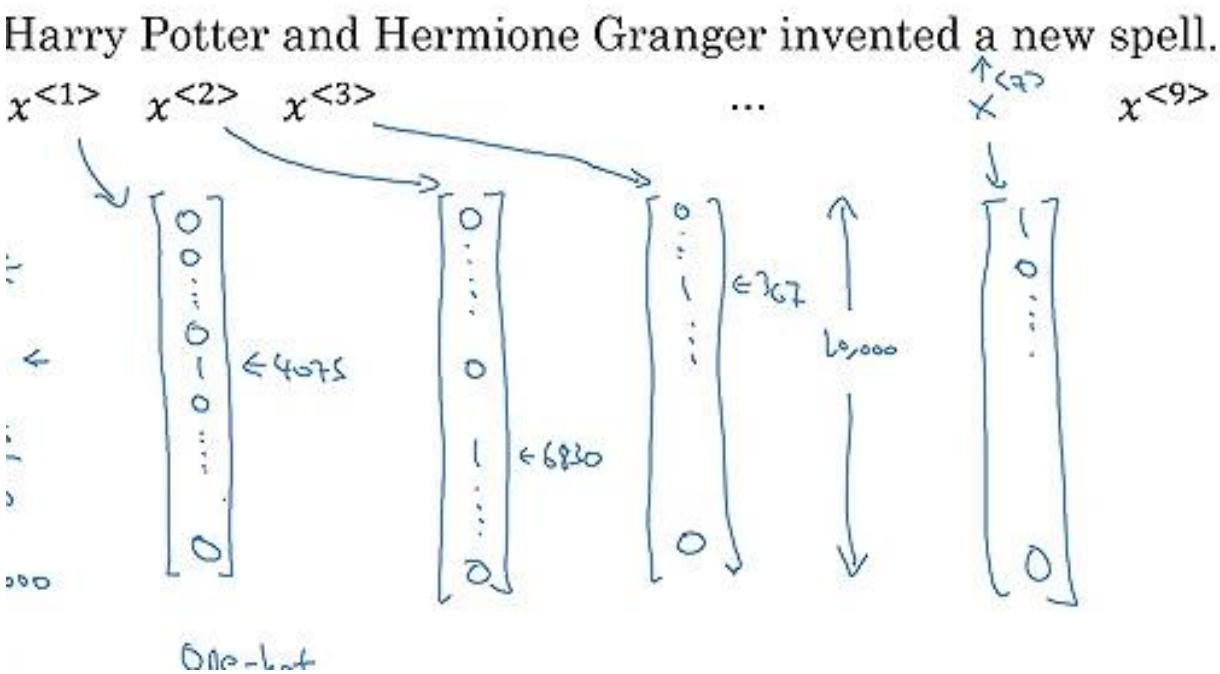


حيث تكون كل القيم اصفار , عدا الصف رقم 4075 تكون بواحد

كذلك تم استبدال كلمة potter بالمصفوفة , التي تكون كلها اصفار عدا الصف الـ 6830 يكون بواحد :



و هكذا في باقي الكلمات :

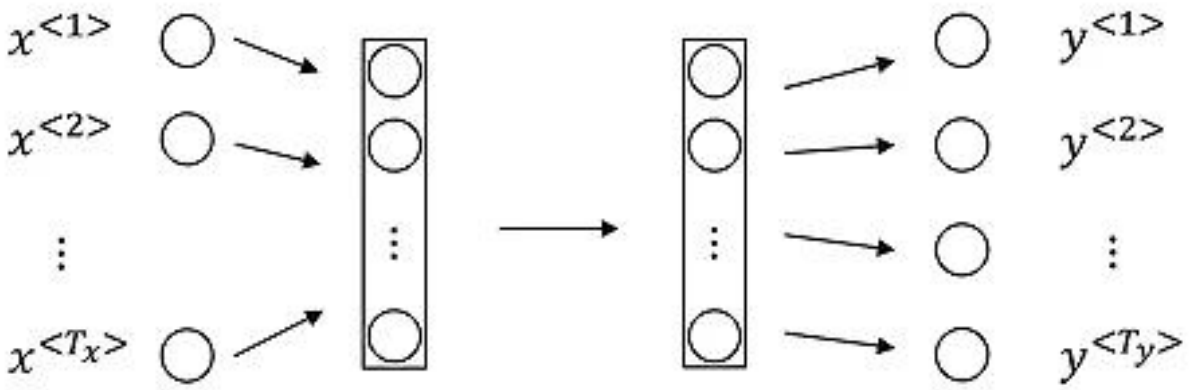


فيكون لدينا تسع مصفوفات (حسب عدد الكلمات الموجودة) , وكل مصفوفة عمود واحد و عشر الاف صف (حسب طول القاموس المستخدم) , و كل القيم في كل المصفوفات اصفار عدا قيم 1 في مواضع محددة , بينما مصفوفة الـ y هي تسع قيم فقط , اصفار او وحايد , ومن هنا يبدأ الخوارزم بالتعلم

-

و للتعرف علي الـ RNN علينا أولا ان نتعرف علي الشكل التقليدي الذي لن يتمكن من معاجلة نموذج التتابع مثل هذا . .

إذا تناولنا المصفوفات الخاصة بكل كلمة من الكلمات , وقمنا بإدخالها في شبكة عصبية تقليدية مثل هذه , للوصول الي قيم y تحدد الاسماء و غيرها , فهي ستكون غير فعالة لسببين هامين . .



أولا أن عناصر العينة ليس لها اطوال ثابتة , فقد تكون الجملة الأولي 10 كلمات , والثانية 15 و الثالثة 6 و هكذا , ولأن كل كلمة لها مصفوفة , فكل عنصر من العناصر سيكون له عدد مصفوفات يتناسب مع عدد الكلمات , و بالتالي سيكون عدد الـ x مختلف من عنصر لآخر , وهو ما يصعب جدا عملية الـ NN العادية .

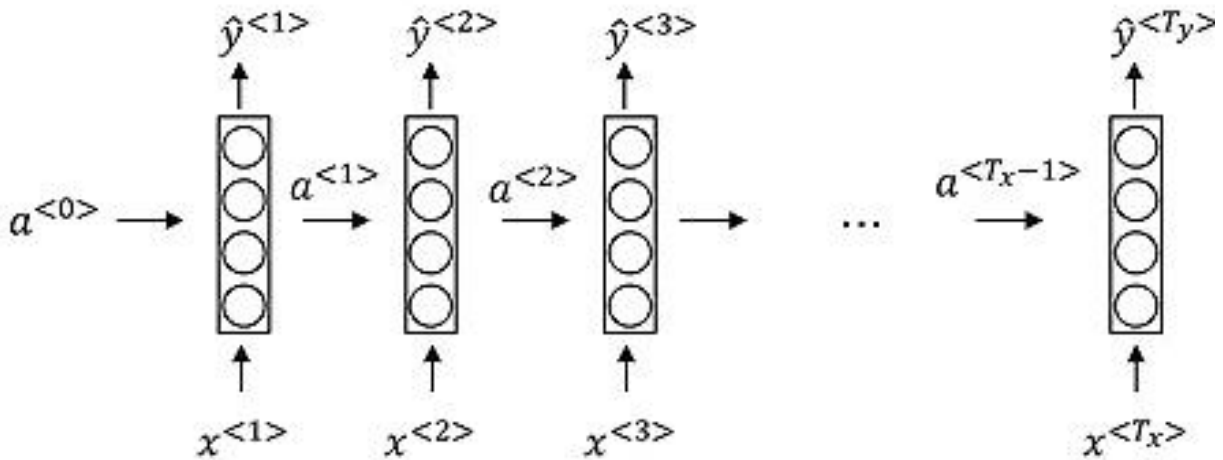
وإذا قمنا بالحساب علي العدد الاقصي للكلمات , فستظهر مشكلة الكفاءة القليلة , والتي ستنتج عن وجود عناصر كثيرة بها فجوات عديدة سيتم ملئها باصفار

السبب الثاني ان اصلا طريقة التعليم غير منطقية , فإذا كان في الجملة الأولي هناك اسماء اشخاص في الكلمة الأولي و الرابعة و السادسة , و في الجملة الثانية في الكلمة السابعة , وفي الجملة الثالثة في الكلمة الخامسة , فترتيب الكلمة ليس منطقيا في تحديد هل هذا اسم ام لا .

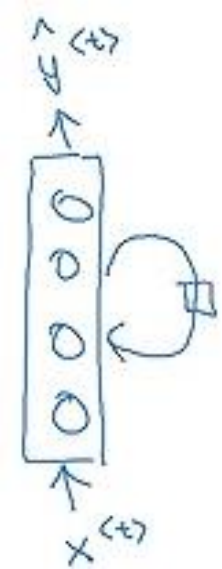
و من هنا كان لابد من اختيار تصميم مختلف للشبكات العصبية , الذي يتلائم مع هذا النوع من البيانات , وهو الشبكات المتكررة RNN

و تعتمد فكرة الـ RNN علي تناول أول كلمة في النص x<1> , وإدخالها في دالة لتحديد هل هي اسم ام لا , و من ثم تقوم الدالة بإخراج رقم y<1> و الذي يحدد هل هي صفر او واحد , و حينما تقوم الشبكة بتناول الكلمة الثانية x<2> , تقوم باستخدام دالة الـ activation الناتجة من معالجة الكلمة الأولي (لا تعتمد عليها 100 % لكن تستخدمها بشكل مساعد او جزئي) , وكأن الدالة التي تقوم بتقييم الكلمة الثانية x<2> ستكون معتمدة بشكل ما علي نتيجة الكلمة الأولي . .

ثم يتكرر الأمر مع معالجة الكلمة الثالثة x<3> , والتي تعتمد علي نتيجة معالجة الكلمة الثانية (والأولي بشكل ما) , وهكذا حتي نصل للكلمة الأخيرة .



وكأن الـ RNN يحدث فيها إعادة استخدام لدالة الـ activation كمدخل في كل مرة , لذا تسمي الشبكة المتكررة , و لذا يتم رسمها بهذا الشكل أحيانا :



و غالبا ما تكون مصفوفة البداية (التي يتم ادخالها لمعالجة أول كلمة) هي مصفوف اصفار , او ارقام عشوائية بسيطة , و تسمي a<0> .

ومن خصائصها الهامة , ان معاملات كل مصفوفة من المصفوفات التي تعالج كل كلمة , هي نفس المعاملات , و تأخذ رمز Wax , كذلك معاملات دالة الـ activation لكل مصفوفات الكلمات تكون هي نفسها و يرمز لها Waa , أيضا معاملات المخرجات y تكون هي هي Wya (و هذا الأمر سنشرحه بالتفصيل بعد قليل)

ومن أهم عيوب الـ RNN أن توقع كل كلمة يعتمد علي الكلمات السابقة لها فقط , لأن معالجة <x5 تعتمد علي الكلمات الاولي و الثانية و الثالثة و الرابعة , ومشكلة هذا الأمر أن فعليا في النصوص قد يكون استنتاج ان هذا اسم شخص يعتمد علي كلمات تالية لها . .

فمثلا في جملة :

I think that Donald Trump at the white house is a reckless president for the US

فحينما ستقوم الـ RNN بمعالجة هذه الجملة , ستقوم بالربط بين كلمتي دونالد ترامب وبين الكلمات السابقة لها , بينما بعدها كلمات مهمة مثل white , house , president , US و هي كلمات دلالية شديدة الأهمية و مرتبط باسم رئيس .

و سيتم معالجة هذا الأمر بشكل ما في ما يسمي الشبكة المتكررة ثنائية الاتجاهات bidirectional recurrent neural network و يرمز لها BRNN

و لتتبع العملية الحسابية في الـ RNN سنجد أن خطوة البداية تكون من مصفوفة <a0 والتي تكون بقيمة اصفار .

ثم لحساب اول اكتيفاشن <a1 تكون المعادلة :

a<1> = g (Waa a<0> + Wax x<1> + ba)

أي أنها مصفوفة Waa (التي قلنا أنها ستكون ثابتة لكل الشبكة) , مضروبة في قيمة a<0> (قيمة الاكتيفاشن السابقة لها وهي حاليا الأصفار) , مجموعة علي حاصل ضرب Wax و هي كذلك ثابتة لكل الشبكة مع قيمة x<1> الخاصة بالكلمة نفسها , ثم معامل انحراف ba وهو كذلك ثابت للشبكة , وكل هذا يدخل في دالة g() والتي غالبا يتم استخدام tanh أو ReLU لها .

ثم ننتقل لحساب قيمة y هات , والتي تكون بالمعادلة :

ŷ<1> = g (Way a<1> + by)

وهي حاصل ضرب Way (كذلك ثابتة للشبكة) في الاكتيفاشن الحالية لها (التي تم حسابها من المعادلة السابقة) , مجموعة علي معامل انحراف ثابت by , وكل هذا يدخل في دالة g() , والتي غالبا تكون سيجمويد او اي نوع من انواع السوفت ماكس , و ذلك لأن حساب ŷ هي عملية تصنيف classification فنريد قيمة 1 , او 0
لا تنس أن دالة g الأولي غير الثانية , حتي ان عدد من الباحثين يكتبون g1() , و g2()

فتكون الصيغة العامة :

$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

كذلك لا تنس أن المصفوفات الثوابت (والتي يقوم الخوارزم بحسابها) هي كلا من :

Wax , Way , Waa , bx , by

كلا من البيهات معروفة , أما قيم الـ w الثلاثة فكلا منها له مكان مرتبط به , ويسمي باسمه

وهنا يجب أن نقوم بعملية حسابية هامة بهدف التبسيط . .

في المعادلة الأولي نري صيغتها غير مريحة و هي :

$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

حيث أن هناك مصفوفتين معاملات Waa , Wax و قيمتين يتم ضربهم فيها و هي , x<t>,a<t-1> فسنقوم حالا بدمجهم بشكل ذكي . .

إذا افترضنا ان مصفوفة a<t-1> هي 100 صف في عمود واحد , فيجب أن تكون مصفوفة Waa هي 100*100 حتي انها حينما تضرب فيها تقوم بانتاج 100 في 1 (وهي القيمة التي ستخرج لنا في a<t>)

كذلك إذا كانت مصفوفة x<t> هي 10 الاف صف في عمود واحد (لا تنس أنها مرتبطة بالقاموس) . فجيب أن تكون مصفوفة Wax هي 100*10000 حتي إذا تم ضربها فيها تكون 100 في 1

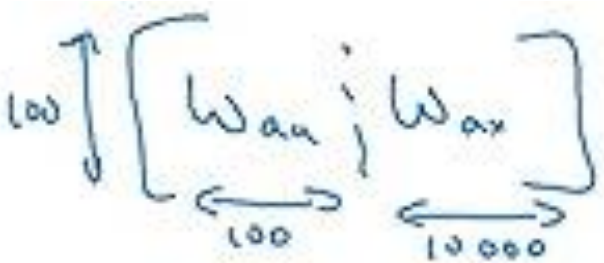
أي أن :

$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

هي :

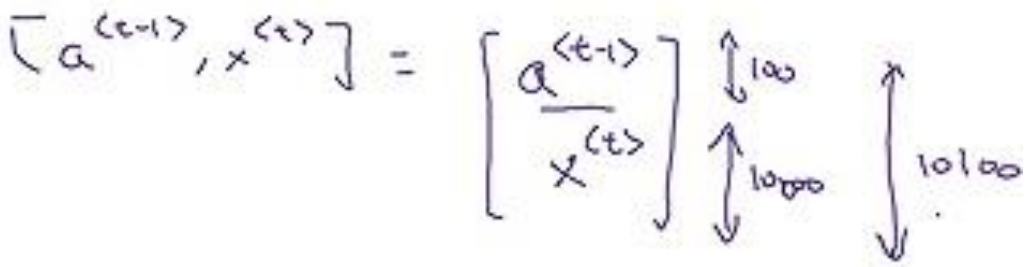
100*1 = 100*100 x 100*1 + 100*10000 x 10000*1 + 100*1

فرغبة في التبسيط , سنقوم بلصق مصفوفتي Waa , Wax بشكل أفقي hstack, بهذا الشكل :



بحيث تكون مصفوفة واحدة , لها 100 صف , ولها 10100 عمود معا , و يكون اسمها Wa

ثم نقوم بلصق مصفوفتي a<t-1> , x<t> بشكل رأسي vstack هكذا :



بحيث تكون المصفوفة الجديدة عمود واحد , و 10100 صف .

فحينما يتم ضرب المصفوفة الجديدة Wa في المصفوفة الجديدة [a<t-1> , x<t>] تكون :

Wa * [a<t-1> , x<t>] = 100*10100 x 10100*1 = 100*1

تكون نفس ابعاد التصور القديم هذا , ونفس قيمه كذلك :

$$W_{aa}a^{<t-1>} + W_{ax}x^{<t>}$$

و بالنسبة للمعادلة الثانية

$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

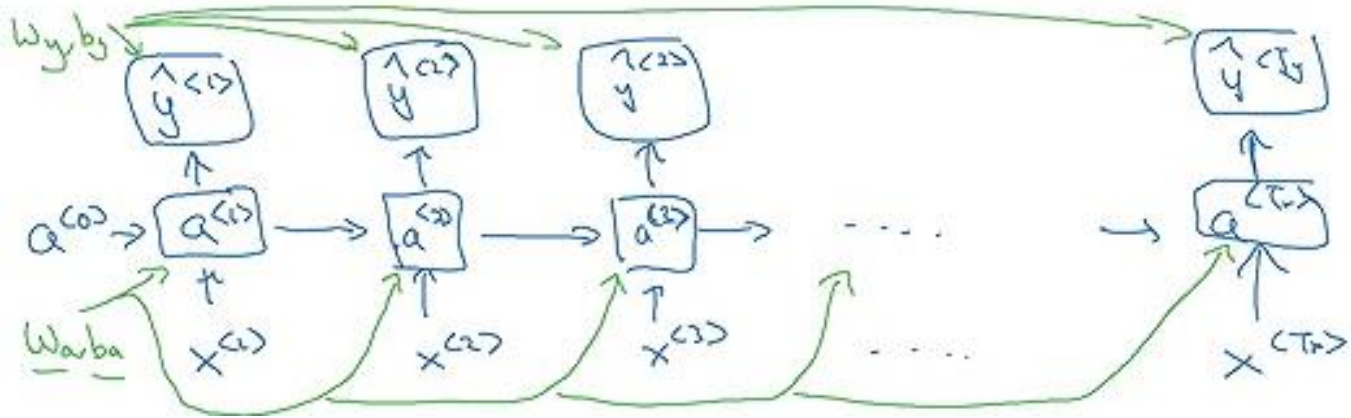
سنقوم فقط بتغيير اسم Wya الي Wy مما يجعل المطلوب الكلي في الخوارزم ايجاد Wa , Wy , ba , by بشكل ابسط .

-

حسنا ماذا عن المسار العكسي للبيانات back propagation ؟

و بالطبع مهمتها الأساسية تحديد قيم الأوزان Wa , Wy , ba , by التي ستقوم بصياغة الشبكة .

نبدأ أولا برسم المسار الأمامي بشكل تفصيلي :



فقيمة a0 , x1 تستخدمان معا لانتاج a1 والتي تساعد x2 في انتاج a2 و هكذا , وكل هذا يتم عبر المعاملات wa,ba

و لانتاج قيمة y1 نحتاج لقيمة a1 و ذلك عبر المعاملات wy,by و التي ستستخدم كذلك في باقي الوايات .

هذا هو المسار الأمامي , فماذا عن معادلة الخطأ loss function ؟

لا تنس أن معادلة الخطأ شديدة الأهمية حتي يتم صياغة المسار الخلفي علي اساس تقليلها . .

لأن هذا الخوارزم يقوم في النهاية بتوقع كل كلمة هل هي اسم ام لا , فستكون معادلة الخطأ شديدة التشابه مع معادلة خطأ التصنيف logistic regression , فتكون المعادلة لكل قيمة y هي :

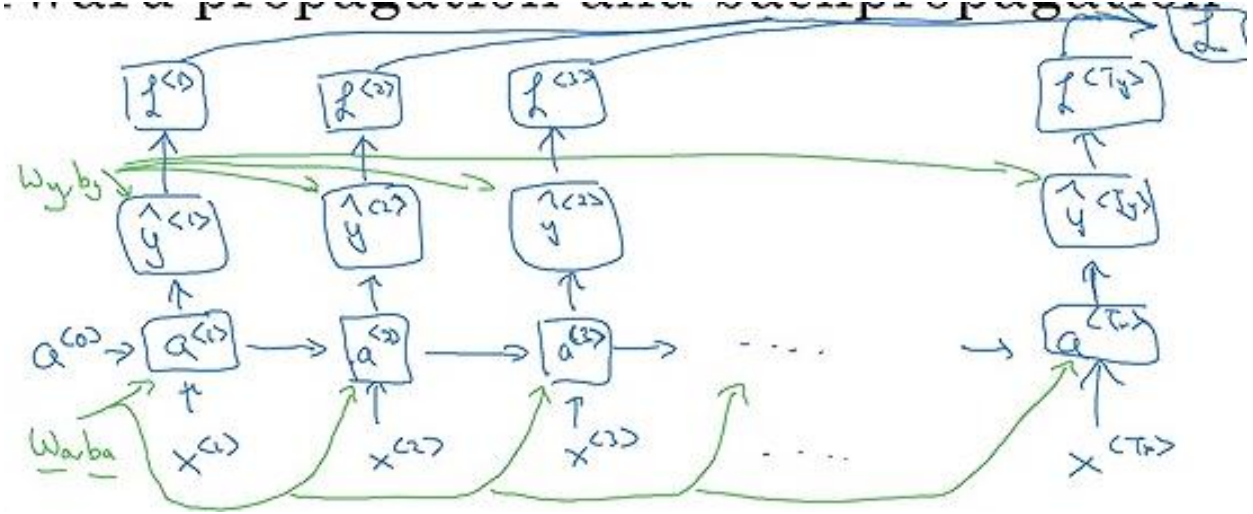
$$\mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{(t)} \log \hat{y}^{(t)} - (1-y^{<t>}) \log (1-\hat{y}^{<t>})$$

و تكون المعادلة الكلي للنص بالكامل , هي مجموع قيم أخطاء كل كلمة :

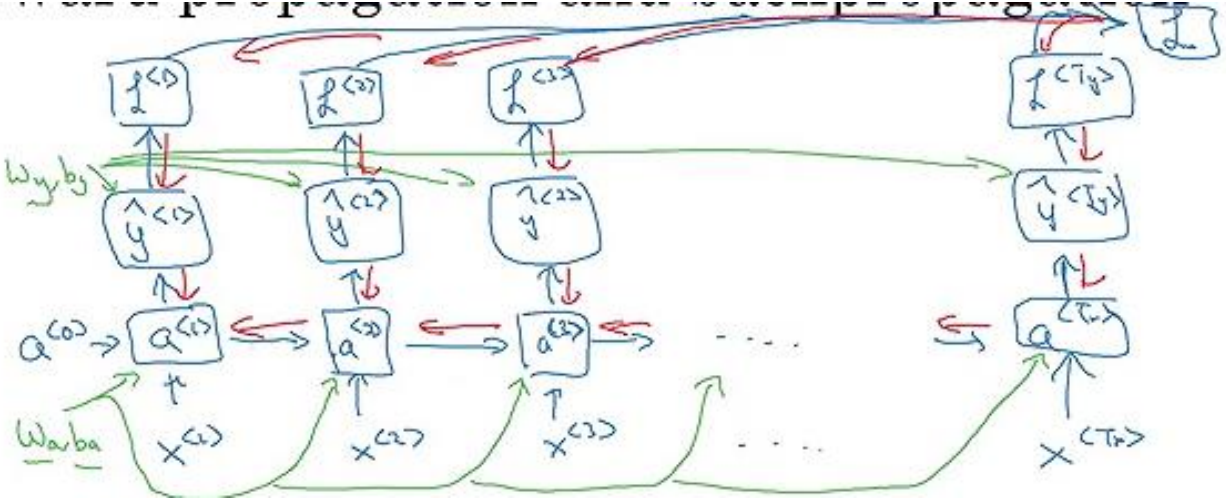
$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_x} \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

حيث السمشن هنا يكون الي Tx وهو عدد كلمات النص

و بها يتم حساب قيمة الخطأ في كل كلمة كالتالي :



و لاستخدام المسار الخلفي back propagation , علينا فقط تتبع الخطوط بشكل عكسي من اليمين الي اليسار كما في الاسهم الحمراء .



والتي تستخدم لتعديل قيم الأوزان الاربعة Wa , Wy , ba , by بشكل مستمر .

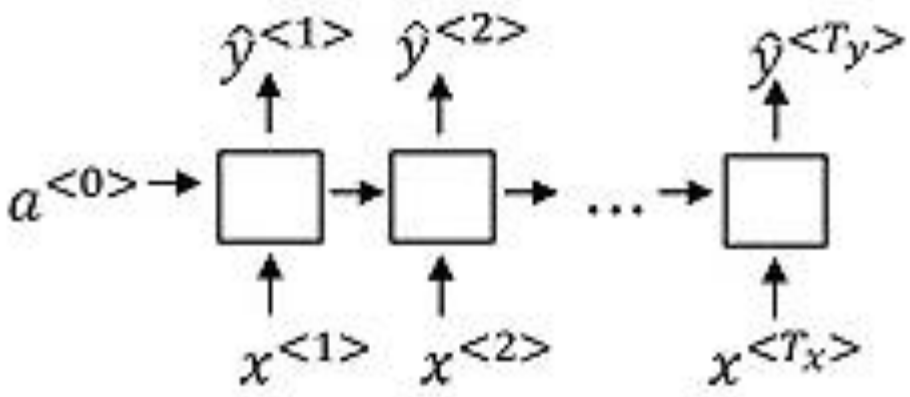
و يسمي المسار الخلفي هنا : back propagation through time و كأنه يمشي في مسار عكس الزمن لأنه ينتقل من الكلمات التالية للسابقة

-

سنتناول الان عدد من أنواع الشبكات المتكررة RNN و التي في عدد منها يكون عدد المدخلات Tx لا يساوي عدد المخرجات Ty

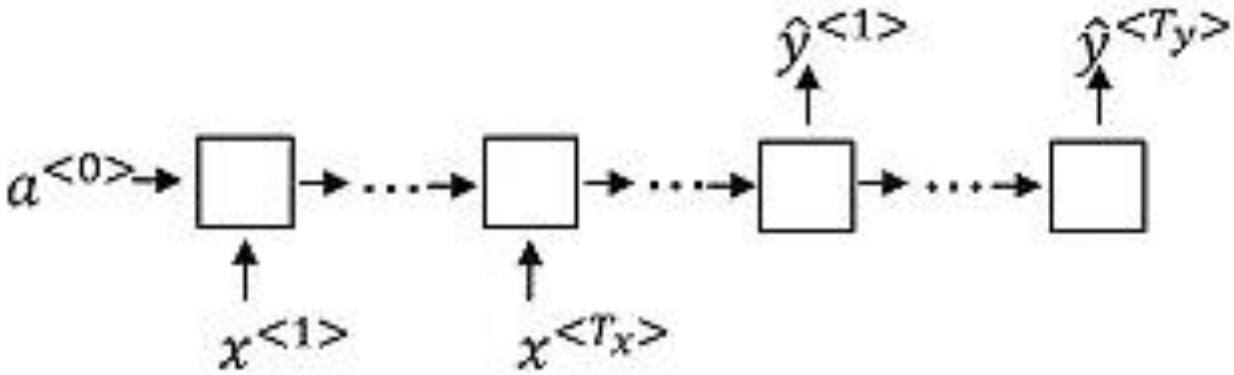
اذا تذكرنا التطبيقات الذي ذكرناها في بداية الملف , سنجد أن هناك نماذج حينما يكون المدخل لا شئ (صناعة موسيقي) , او نماذج حينما يكون المخرج رقم واحد (الكلام يتحول لتقييم) , او نماذج حينما يكون المدخل و المخرج بنفس الرقم (تحديد الاسماء) , او نماذج حينما يكون المدخل و المخرج غير معروفي رقمهم (الترجمة)

فأحد النماذج المستخدمة يسمي many-to-many و الي يكون له العديد من المدخلات يصنع العديد من المخرجات , مثل المثال السالف شرحه في الفيديوهات السابقة (اختيار الاسماء) , حيث أن هناك مدخلات عديدة , وكل منها يكون لها مخرج معين :

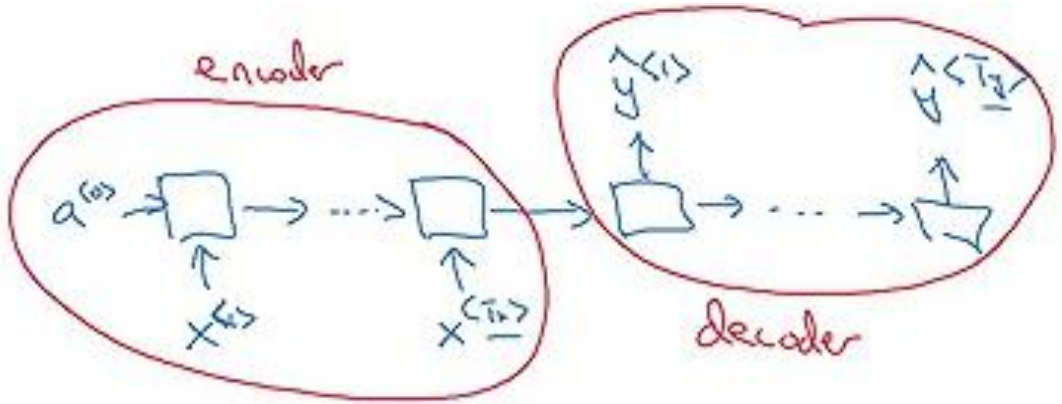


و هناك شكل اخر من نفس النوع , لكن حينما يكون عدد المدخل Tx لا يساوي عدد المخرج Ty و هي مثلا في حالة الترجمة , فلو تم ترجمة جملة فرنسية للانجليزية , فلن تكون عدد الكلمات متساوي و كذلك لن تكون كل كلمة انجليزية صانعة للكلمة الفرنسية بالمقابل .

و تكون الرسمة الخاصة بها :



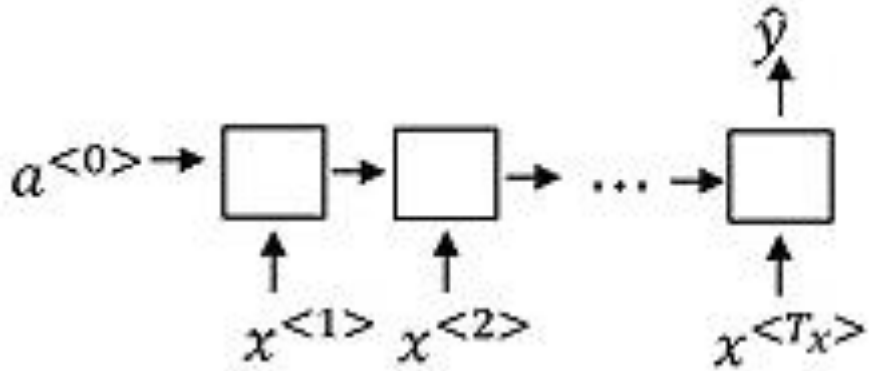
حيث يقوم الخوارزم اولا بتناول كل المدخلات و معالجتها جيدا , ثم يقوم بصناعة المخرجات في النهاية , وكأن الجزء الأول هو القارئ encoder و الجزء الثاني هو الكاتب decoder



نموذج آخر يسمي : many-to-one و الذي يكون له مدخلات عديدة , لكن مخرج واحد .

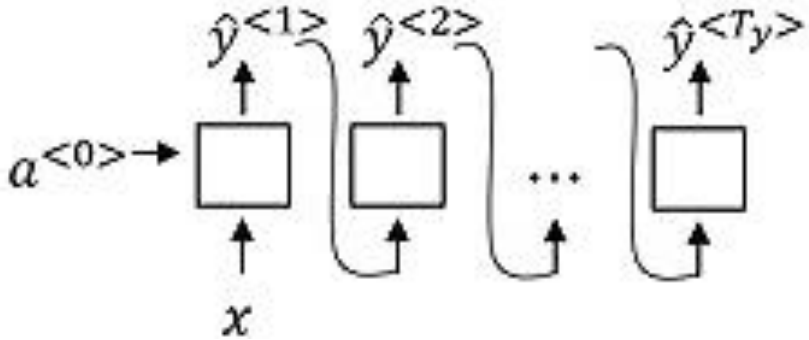
مثل لو كان لدينا خوارزم يقوم بفحص التعليقات علي فيلم معين , لمعرفة تقييم الشخص لها و فالتعليق هو نص له كلمات عديدة , بينا التقييم هو رقم واحد (من 1 الي 10 مثلا)

فلو كان التعليق المكتوب : there is nothing interested in this movie , فتكون الشبكة المتكررة :



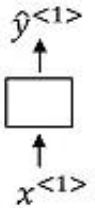
حيث ان لكل كلمة مدخل معين , و بدلا من وجود مخارج لكل كلمة , يكون هناك مخرج واحد في النهاية يعتمد علي كل المداخل .

كما أن هناك one-to-many مثل خوارزميات انتاج الموسيقي , حيث تقوم بإعطائه رقم محدد (نوع الموسيقي المطلوب) , ويقوم هو بإنتاج العديد من المخرجات بهذا الشكل



وفي حالة لم تعطه قيمة مدخل , فسيقوم باختيار رقم عشوائي

و لاحظ أن نوعية الشبكات one-to-one هي الشبكات العادية الـ NN السالف شرحها في السابق , وهي بالطبع ليست RNN لان ليس لها اي نوع من التكرار .



-

سنتناول الان ما يسمي نموذج اللغة language model و الذي هو أحد التطبيقات الهامة في التعامل مع اللغة , ويتم بنائها بالـ RNN .

و للفهم الكامل لمعني language model يمكن أن نقول , أنه قدرة الآلة علي توفع الكلمة التي قلتها صوتيا , عبر فهم المعني الكامل حولها , وهو الذي يكون صعبا حينما تكون الكلمة المنطوقة لها اكثر من spelling

فمثلا لو قال شخص :

the apple and pear salad

أو قال

the apple and pair salad

فمكن الممكن لكمة pear ان تكتب pair لأنها تقريبا نفس النطق , فعلي عاتق الخوارم ان يعلم اي كلمة مقصودة عبر فهم المعاني المحيطة .

ونفس الفكرة في الفارق بين كلمات to , two , too , toe خاصة مع اختلاف لهجة من يقولها .

و يبدأ الأمر عبر تحديد احتمالية probability للكلمة المنطوقة , وسط سياق الجملة . وبالتالي يقوم الخوارزم بتحديد مدي احتمالية صحة كل جملة .

و تكون الخطوة الأولي في بناء الخوارزم , هو إحضار كمية كبيرة من بيانات التدريب corpus of English text و كلمة corpus معناها كمية كبيرة من البيانات .

بعدها نقوم بعملية اسمها tokenization يعني الترميز, وهي عملية ان كل كلمة من كلمات العينة اعمل لها رمز y .

فمثلا جملة :

cats average 15 hours of sleep a day

اعمل لكلمة cats رمز y<1> و كلمة average y<2> , وهكذا .

مع ملاحظة ان حرف a او an او or او & بيكون ليها كمان رمز مستقل , بل ان احيانا علامات الترقيم (, . / * + -) هي كمان يكون لها رمز مستقل .

و أحيانا يتم عمل في نهاية الجملة ما يسمي <EOS> يعني end of sentence توضع في النهاية , حتي يعلم الخوارزم ان الجملة قد انتهت بالفعل , و اذا ما تم استخدامها فهي ايضا تأخذ رمز مستقل

و في حالة ن كان هناك كلمة ليست في القاموس (الكلمات الاكثر استخدام) وقتها يقوم الخوارزم باستبدالها برمز محدد هو <UNK> وهي اختصار unknown للدلالة علي كلمة غير معروفة في القاموس .

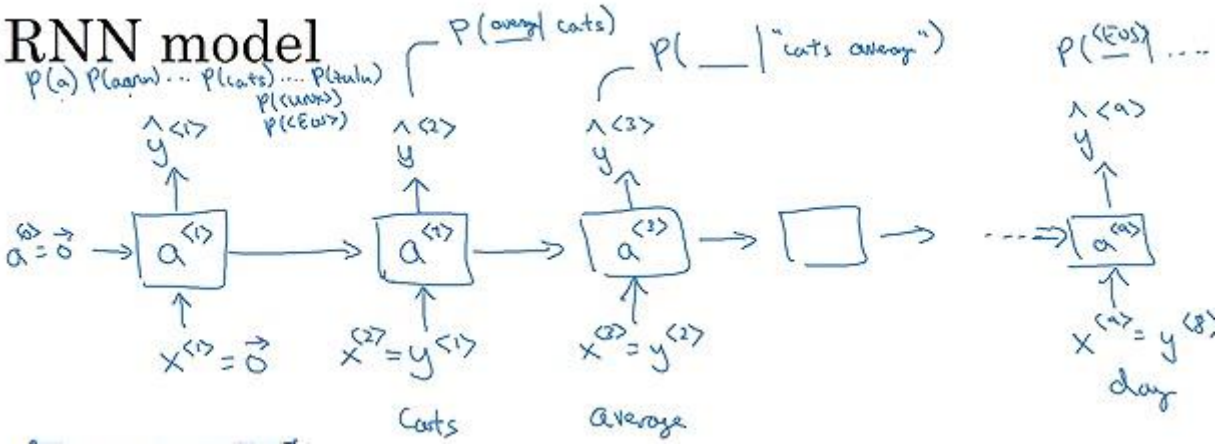
ولتتبع مسار الخوارزم , في الجملة :

cats average 15 hours of sleep a day

في البداية ان المدخل هو a0 و التي هي مصفوفة كلها اصفار , وهي تدخل في الوحدة a<1> والتي أيضا تأخذ مدخل هو : x<1> , ومهمة هذه الوحدة ان تقوم بيجاد احتمالية الكلمة الأولي و هي cats , وهذه الاحتمالية هي ŷ<1> . اي واي هات المتوقعة

ويتم حساب الاحتمالية عبر استخدام softmax به 10 الاف مخرج , وهو كلمات القاموس المستخدم .

وقيمة الإحتمالية تعتمد علي فرصة أن تأتي هذه الكلمة في البداية من قلب الكلمات العشر الاف في القاموس



ثم نذهب للوحدة الثانية , وهي التي كون المدخل فيها هو x<2> والتي هي نفسها قيمة y<1> , وهي الكلمة الأولي بالفعل cats .

لا تنس أن قيمة $\hat{y}$ هي الكلمة المحتملة , بينما y هي القيمة الحقيقية (كلمة cats هنا)

اذن قيمة x<2> هي نفسها y<1> , والتي ستدخل في الوحدة a<1> لتتوقع الكلمة التالية و هي average , ولكن التوقع سيكون بناء علي معلومية الكلمة الأولي cats فسيكون هو التوقع المشروط probability with condition واللذي يكتب هكذا :

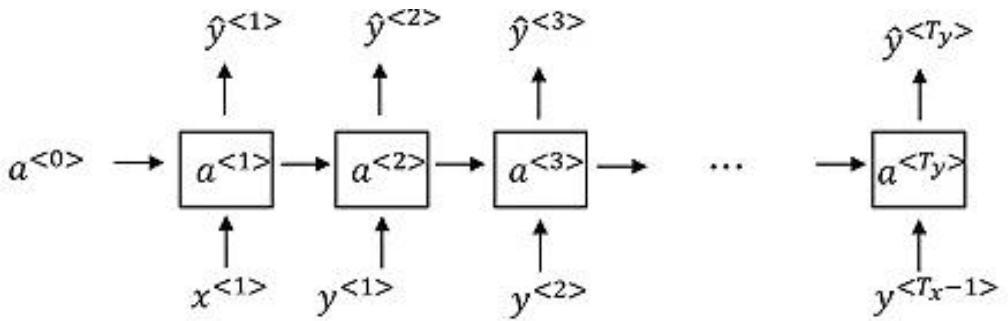
P(average | cats)

و يتم تكرار الأمر في الوحدة التالية , لتوقع الكلمة الثالثة fifteen بمعلومية الكلمتين السابقتين الاولي و الثانية فتكون هكذا :

P(fifteen |average , cats)

ويستمر الأمر حتي نصل للكلمة الأخيرة EOS والتي نريد أن نتوقعها بناء علي الكلمات السابقة كلها . فكأن الخوارزم يتناول الكلمات من اليسار لليمين , و يقوم بتدريب الثيتات عليه .

والصورة الكاملة هنا :



وتكون معادلة الخطأ لكل كلمة علي حدة هي :

$$\mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$$

بينما معادلة الخطأ الكلية تكون مجموع نسب الأخطاء السابقة

$$\mathcal{L} = \sum_t \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

و بالتالي إذا ما اردنا حساب احتمالية كتابة جملة من 3 كلمات هي :

cats average 15

ولأنها احتمالية متراكمة , فتكون الاحتمالية الكلية هي حاصل ضرب الاحتماليات الثلاث , فتكون :

P(y<1> , y<2> , y<3>) = P(y<1>) P(y<2>| y<1>) P(y<3> | y<1> , y<2>)

-

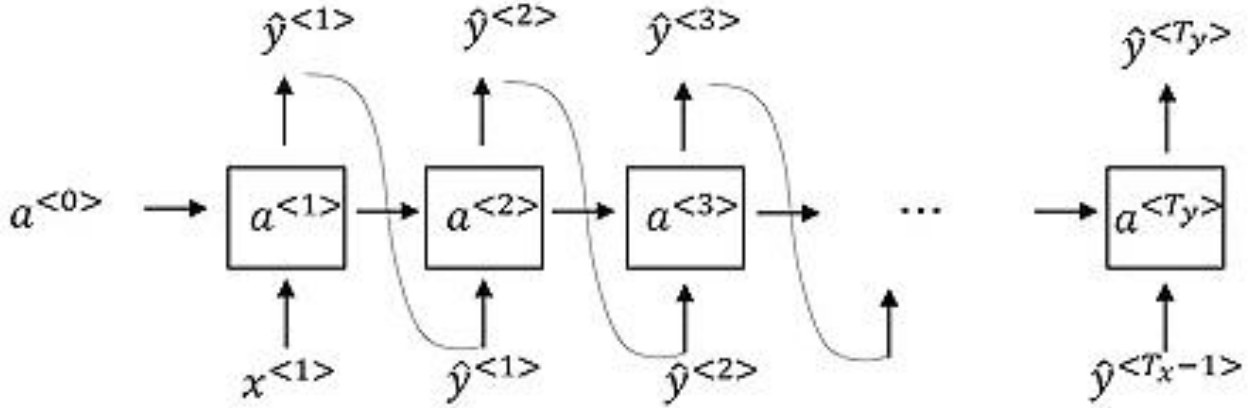
بعدما يتم عمل تدريب للخوارزم , يمكنك التأكدمن مدي كفائته عمر تطبيق ما يسمي متتبعة عينة الرواية sampling novel sequences

والمقصود به انه يمكنتوقع كلمات من لا شئ , بل وكتابة سلسلة كاملة منها علي هيئة مقال او قصة .

و يبدأ الأمر عبر ادخال قيمة a0 الصفرية , ثم <x<1 التي ستكون عشوائية او صفرية , فيقوم الخوارزم (بعد التدريب) باستنتاج الكلمة الأولي , و سيكون الاستنتاج بناء علي ما هي الكلمات المشهورة بانها تاتي اولا , فغالبا سيختار كلمة مثل the , a , an , once , حسب قيمة ŷ

لا تنس أن واي هات هي قيمة واحدة من 10 الاف من السوفت ماكس التي سيختارها من القاموس .

ثم نأخذ هذه الكلمة <ŷ<1, ونقوم بإدخالها في الوحدة الثانية <a<2 والتي ستكون هي بالفعل قيمة <x<2 ليقوم باستنتاج الكلمة التالية بعد بمعلومية الكلمة الأولي the و سيختار كلمة و ليكون girl .



و يتم استمرار العملية ليتم استنتاج الكلمة الثالثة بناء علي الاولي و الثانية , ثم الرابعة بناء علي الكلمات الثلاث الأولي , وهكذا .

و حتي نهاية الجملة ستأتي عبر استنتاج كلمة <EOS> , او يمكن عمل الأمر يدويا عبر تحديد حد معين للكلمات برقم محدد .

و هذا الأمر سيجعل الخوارزم يقوم بصياغة جمل كاملة و علامات ترقيم و كل شئ , سواء كانت رواية او مقال او تحليل او خبر .

كل ما سبق كان يسمي النموذج اللغوي للكلمات word level language model بينما هناك مستوي ادق و اعمق يسمي character level language model وهو الخاص بالتحليل بالحروف .

و يقصد به أنه يتم تحويل الجملة ليس لكلمات , و لكن لحروف , فإذا كانت نفس الجملة :

cats average 15 hours of sleep a day

فيكون حرف c هو واي 1 , وحرف a هو واي 2 و هكذا , فيكون القاموس لا يحتوي علي 10 الاف كلمة , ولكن علي الحروف الخاصة باللغة الانجليزية + الارقام + علامات الترقيم + نفس الحروف كابيتال + رموز هامة مثل #%$

و ميزة هذه الفكرة انها ستتجنب مشكلة الكلمات غير المعلومة , فكل حرف يتم تحليله , وعيبه ان عدد الوحدات اكبر بكثير و هذاسيجعب التحليل بطئ للغاية و بالتالي مكلف .

و في الوقت الحالي يندر استخدام هذا النموذج , لكن قد يتم استخدامه في المستقبل حينما تكون اجهزة الكومبيوتر اسرع .

و لا تنس ان الخوارزم سيقوم بالتنبؤ بجمل , اعتمادا علي قيم ثيتا التي تم تدريبه عليها , وبالتالي لو تم تدريبه علي أدب او اخبار او رياضة او فن او غيره , فسيقوم بكتابة جمل في نفس السياق

هنا نصوص تم التنبؤ بها من نوع الأخبار

News

President enrique peña nieto, announced sench's sulk former coming football langston paring.

"I was not at all surprised," said hich langston.

"Concussion epidemic", to be examined.

The gray football the told some and this has on the uefa icon, should money as.

و هنا من نوع أدب شكسبير

Shakespeare

The mortal moon hath her eclipse in love.

And subject of this thou art another this fold.

When besser be my love to me see sabl's.

For whose are ruse of mine eyes heaves.

-

نتكلم الان هم مشكلة شائعة في الشبكات المتكررة RNN و هي ما يسمي اختفاء الاشتقاقات Vanishing gradients in RNN

و للتعرف علي معناها , تعالي نلقي نظرة علي جملتين :

the cat which ate fish & drink the fresh milk , was brown
the cats which ate fish & drink the fresh milk , were brown

كما هو واضح ففي الجملة الأولي قطة واحدة فتم استخدام was بينما في الجملة الثانية قطط جمع فتم استخدام were .

و المشكلة في أن الفاصل المكاني بين كلا من الكلمتين هنا و هناك كبير , وهذا سيجعل من الصعب علي الخوارزم ان يقوم بالربط بين الكلمة الؤثرة (cat) والكلمة الناتجة (was)

كما أننا في RNN نقوم بعمل backpropagation و بالتالي ستظهر مشكلة اختفاء الاشتقاقات vanishing gradients و هي التي تكلمنا عنها في الكورس الأول , حينما ذكرنا الشبكات شديدة العمق (100 طبقة مثلا) فأثناء عمل المسار الخلفي فالاشتقاقات تتلاشي حينما تصل للطبقات الأولي , مما يجعل تدريب الطبقات الأولي و بالتالي تعديل قيم الثيتات شئ غير سهل .

و لأن فكرة الـ RNN الخاصة بالبناء اللغوي , تعتمد علي ان كل كلمة لها وحدة و بالتالي طبقة , فتأثير الكلمات المتباعدة سيكون ضئيل .

و ليست هذه هي المشكلة الوحيدة في الشبكات العميقة , فكما ان الاشتقاقات ممكن ان تختفي , فهي ممكن ان تزيد بشكل رهيب في ما يسمي مشكلة exploding gradients

لكن المشكلة الثانية (زيادة الاشتقاقات) مقدور عليها ,ويسهل اكتشافها والتي ستظهر القيم بارقام كبيرة او برقم NaN حينما تكون فوق المعتاد , ويمكن ضبطها عبر استخدام تكنيك gradient clipping و ذلك عبر تحديد قيمة قصوي مسموح بها , ثم حينما تزيد القيمة يتم تصغيرها باسلوب النسبة المئوية

لكن مشكلة اختفاء الاشتقاقات أكثر صعوبة في الاكتشاف و في الحل , وهو ما سنري حله في الفيديوهات القادمة

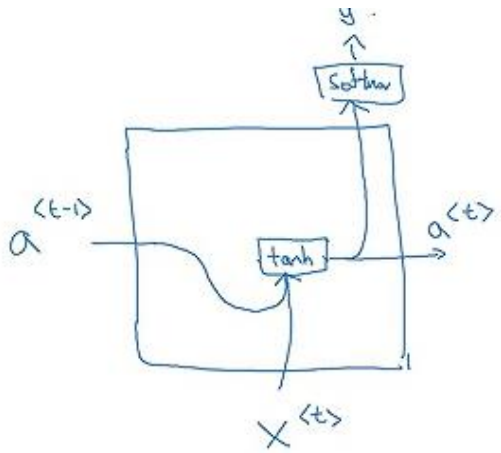
-

نتكلم الآن عن أحد الحلول لمشكلة اختفاء الاشتقاقات , وهي باستخدام تكنيك وحدة البوابة المتكررة Gated recurrent unit و التي باختصار GRU .

واذا تذكرنا المعادلة الخاصة بالـ RNN تكون :

$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

والتي يمكن تصميم رسم بياني مبسط لها وهو :



و علينا الان ان نتعرف علي قيمة جديدة و هي : خلية الذاكرة memory cell و التي يرمز لها بالرمز c .

و في هذا المثال ستكون بالضبط مساوية لدالة الـ activation لذا ستكون هي نفسها قيمة a لكن في تطبيقات أخري سنتناولها ستكون مختلفة . .

إذن ستكون معادلتها :

$$\tilde{c}^{<t>} = \tanh(W_c[\, c^{<t-1>}, x^{<t>}] + b_c)$$

والتي تتطابق مع معادلة a , مع التأكيد اننا نستخدم دالة تانش هنا كدالة g(x) , ايضا لا تنس ان الـ c هنا تسمي c tilde لأن فوقها علامة ~

كما ستتناول قيمة أخري و هي جاما , والتي تم تسميتها بهذا لتشابه حرف الـ G في كلمة gate و هي البوابة , المشتقة من اسم التكنيك GRU

و تكون معادلته :

$$\Gamma_u = \sigma(W_u[\, c^{<t-1>}, x^{<t>}] + b_u)$$

و هنا الدالة سيجمويد بدلا من التانش .

و لأن دالة سيجمويد تخرج لنا قيمة تتراوح بين الصفر و الواحد , فأغلب القيم التي تكون لجاما هنا تكون كبيرة للغاية (1) , او صغيرة للغاية (0) .

وهنا تقوم الـ c بالربط بين الكلمة المطلوبة was و الكلمة البعيدة cat دون الوقوع في مشكلة اختفاء الاشتقاقات , وتستخدم المعادلة :

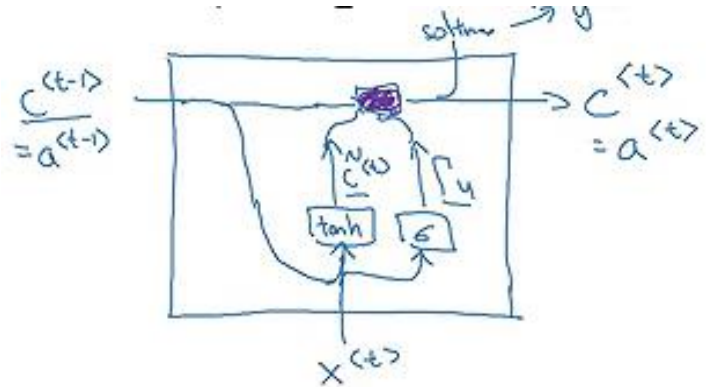
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) + c^{<t-1>}$$

وهي تشبه معادلة التصنيف القديمة

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

و هي تحمل اسم بوابة , لأنها تتحكم في ما قيم ثيتات التي سيتم تعديلها (فتح البوابة) , وما هي القيم التي لن يتم تعديلها (غلق البوابة) .

فكأن في المسار الخلفي للشبكة , لدي كلمة cat ستكون قيمة جاما تساوي 1 , أي أن قيمة c (وهي كأنها ثيتا) سيكون لها قيمة ثيتا المعدلة (سي تيلدا) , بينما في باقي الكلمات ستكون قيمة جاما تساوي صفر , فتكون c عبر المعادلة السابقة لها نفس قيمة c السابقة ولا يتم تعديلها update وهو الشئ المطلوب



و كما هو موضح بالرسم , دخل قيمة c السابقة مع قيمة x لمعادلتي c الحالية بدالة تانش و معادلة جاما بدالة سيجمويد , و ثم تدخل علي المربع الازرق و هي المعادلة الاخيرة المذكورة :

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) + c^{<t-1>}$$

فيكون المخرج هوقيمة y عبر السوفتماكس , وقيمة c الحالية

و غالبا ما يتم إضافة جاما r في المعادلة الأول هكذا :

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

حيث معادلتها هي :

$$\Gamma_r = \sigma(W_r[\, c^{<t-1>}, x^{<t>}] + b_r)$$

-

كما تكلمنا عن GRU كوسيلة لحل مشكلة المسافات البعيدة بين العوامل الهامة , والتي قوانسنها هكذا :

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[\, c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

, نتناول الان وسيلة اخري تسمي الذاكرة طويلة قصيرة المدي long short term memory و التي اختصارها LSTM , وهي أقوي و أكثر كفاءة من GRU

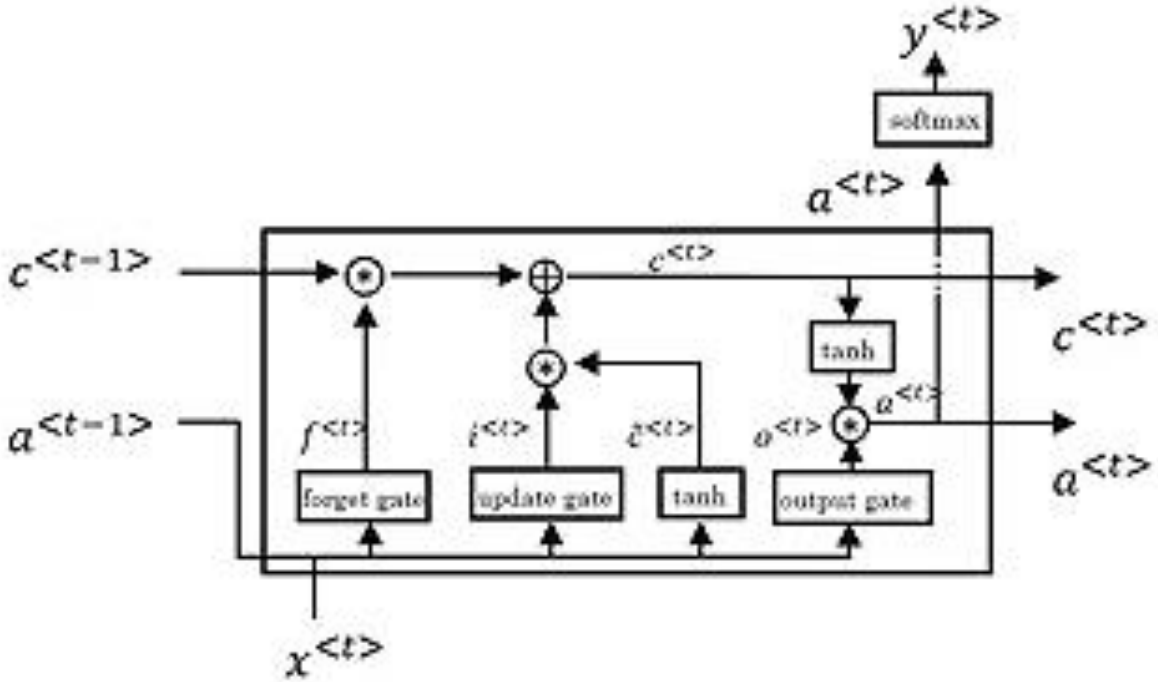
والقوانين الخاصة بها هي :

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$
$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$
$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$
$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$
$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$

حيث أن القيمة u تعبر عن التعديل update بينما القيمة f تعبر عن عدم التعديل و النسيان forget اما القيمة o تعبر عن المخرج output

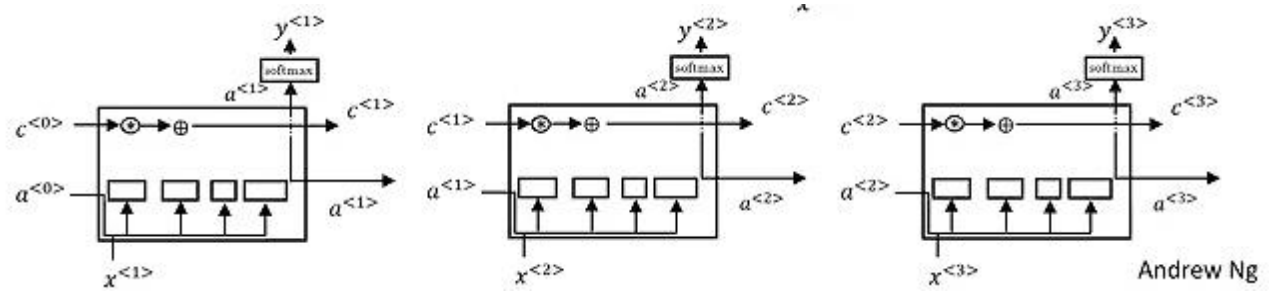
كذلك نلحظ ان قيمة c لا تساوي قيمة a كما في GRU , كما انه تم فصل جاما للتعديل u عن جاما عدم التعديل f

وتكون الرسمة الخاصة بها هكذا :

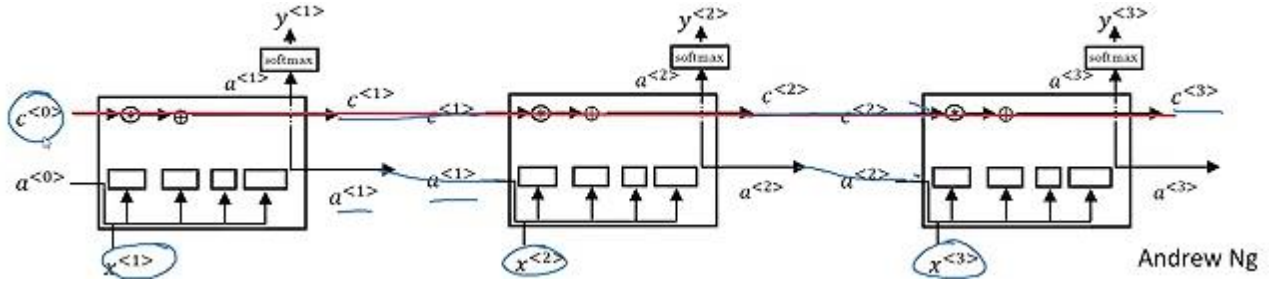


فمن الرسم يتضح ان قيمة a<t-1> (اسفل اليسار) تستخدم لتعديل كلا من update gate , forget gate , output gate كما نري في الجامات الثلاثة في المعادلات , بالإضافة انها تستخدممع تانش لإيجاد (سي تيلدا تي)

وكأن السلسلة كاملة ستكون :



و من عوامل قوة LSTM انه في حالة كان هناك لايا عديدة سيتم نسيانها , فستنتقل قيمة الـ c سريعا من وحدة الي اخري دون تفليلها او غيرها , كما نري في الخط الأحمر



و هي ايضا مثل الـ GRU في فكرة ان عنصر من طبقة معينة (كلمة محددة) قد يقوم بالتأثير علي طبقة اخري بعيدة عنها (كلمة اخري) , دون التأثير علي الباقي بالضرورة .

نصل للسؤال الهام , أيهما افضل الـ GRU أم الـ LSTM ؟

لا توجد إجابة لهذا السؤال , إذ أن كلا منهما له مميزات و عيوب ضد الآخر . .

فالـ GRU لأنه بسيط التكوين , فهو اسرع في التنفيذ , و يتناسب مع الشبكات العميقة , لكن كفائته ليست مثلي , بينما الـ LSTM العكس , فهو أكثر تعقيدا , و ابطئ في التنفيذ , لكنه ذو كفاءه أعلي

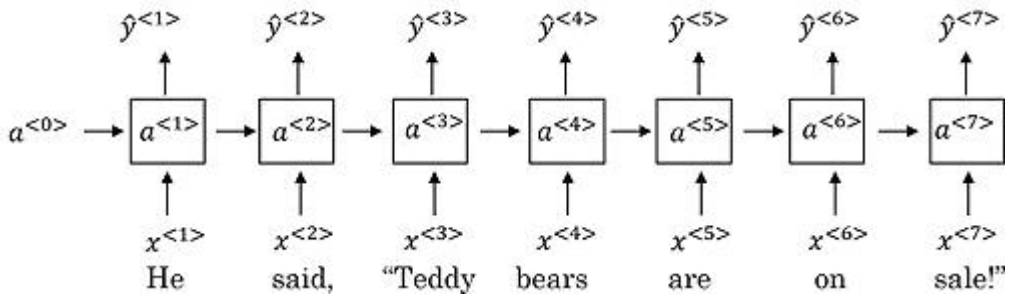
-

نتكلم الآن عن الشبكة المتكررة ذات الإتجاهين Bidirectional RNN . ويطلق عليها BRNN وهي التي تسمح للشبكة بالتعامل مع الكلمات السابقة و التالية معا , في الإتجاهين .

فكما رأينا في مثال سابق, قد يكون لدينا جملتين :

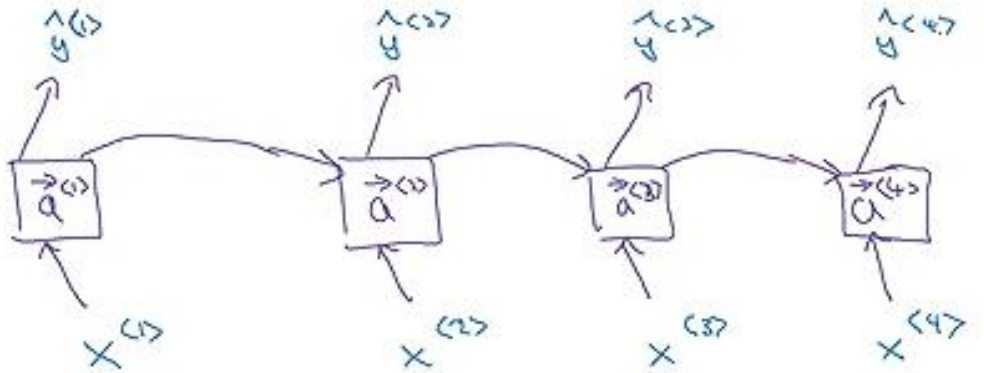
he said : teddy bears are fine
he said : teddy roosevelt was a US president

فالمشكلة ان الكلمات السابقة لكلمة teddy غير كافية لتحديد معناها و بالتالي تحديد الكلمات التالية لها , كما في الرسم :

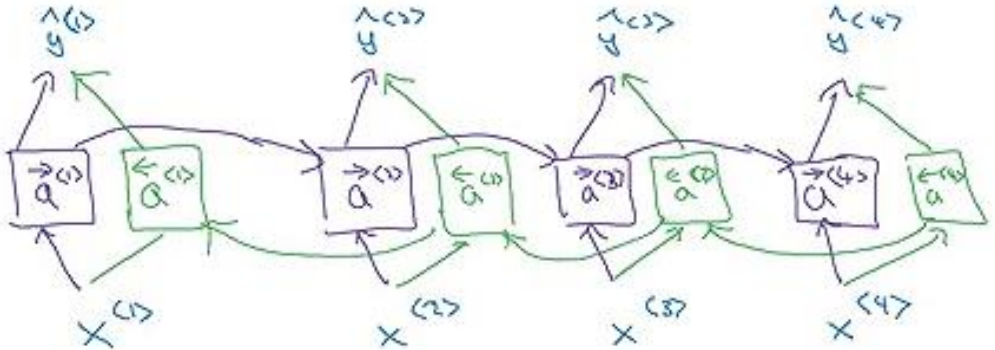


لذا علينا التعامل مع الشبكة ذات الاتجاهين .

فإن كان المسار الطبيعي لشبكة RNN هو هكذا :

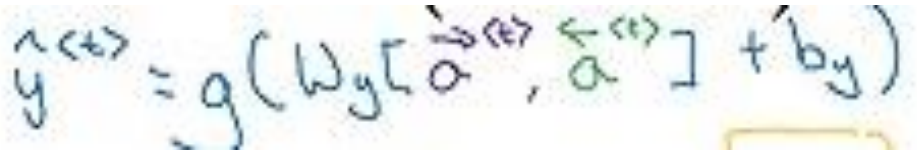


فالمسار العكسي لها سيكون كالتالي :



حيث الوحدات الزرقاء تمثل المسار الأمامي , بينما الوحدات الخضراء تمثل الخلفي . .

وبالتالي ستكون معادلة y هي :



والتي ستجعل قيمة y معتمدة علي كلا من المسار الأمامي و الخلفي معا , و هنا حينما يكون هناك كلمة teddy سيتدرب الخوارزم علي الكلمات السابقة و التالية له

و غالبا ما تكون الوحدات المستخدمة ليس فقط شبكات RNN وحدها بل يكون معها وحدات LSTM او GRU

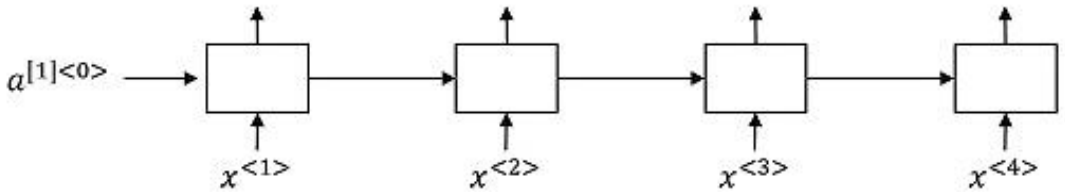
لكن من عيوبها انها تحتاج الي البيانات كاملة لبدء المعالجة , وبالتالي هي لا تتناسب مع البيانات التي تأتي بشكل آني و متواصل , مثل تحويل الصوت القادم الي نصوص

-

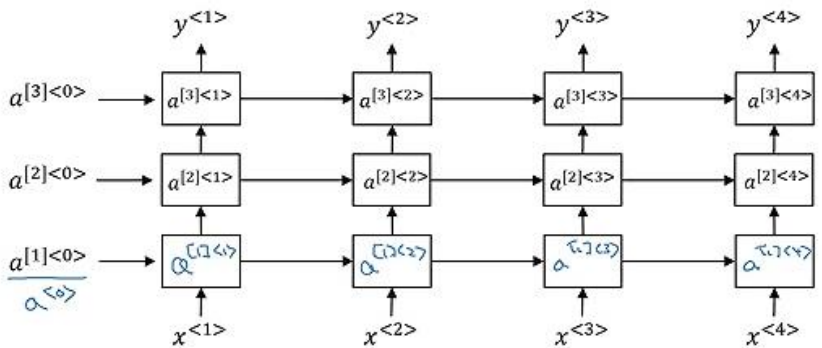
في نهاية الأسبوع الأول , سنتكلم عن الشبكات المتكررة العميقة Deep RNN, والتي تتعلق بناء انواع مختلفة من النماذج التي رأيناها معا .

فكل نوع من الأنواع السابق ذكرها يمكن ان يعمل وحده جيدا , لكن ماذا عن ربطهم معا ؟

فإذا كانت هناك طبقة محددة من الـ RNN هكذا :



فدمجهم مع يمكن أن يكون هكذا :



فتكون كل قيمة تعتمد علي قيم مجاورة او سابقة لها . .

فمثلا القيمة a[2]<3> وهي الخاصة بالكلمة الثالثة في العينة الثانية . فستكون معادلتها :

$$a^{[2]<3>} = g\left(W_a^{[2]}\left[a^{[2]<2>}, a^{[1]<3>}\right] + b_a^{[2]}\right)$$

حيث ان لها مدخلين هما a[2]<2> وهي الكلمة السابقة لها في نفس العينة , و a[1]<3> وهي نفس ترتيب الكلمة , لكن في العينة السابقة

مع التأكيد علي أن كل طبقة سيكون لها معاملات w , b ثابتة لكل الوحدات في نفس الطبقة , لكن ستختلف بالطبع مع طبقة تالية

و لأن RNN هي معقدة بطبيعتها , فنقوم بتجنب ربط عدد كبير من الطبقات معا لتجنب صعوبة الحساب , لذا يتم احيانا بناء شبكة RNN ثلاثية الابعاد, حيث تكون هذه الطبقات الثلاثة وراءها 3 طبقات اخري , ثم 3 طبقات أخري و هكذا , مما يسهل الحساب نوعا .

و قد تكون هذه الوحدات RNN عادية , و قد تكون ملحق بها GRU or LTSM لكن نادرا ما يتم تكوين سلسلة طويلة من الـ BRNN لصعوبتها الكبيرة

-

نهاية الاسبوع الأول