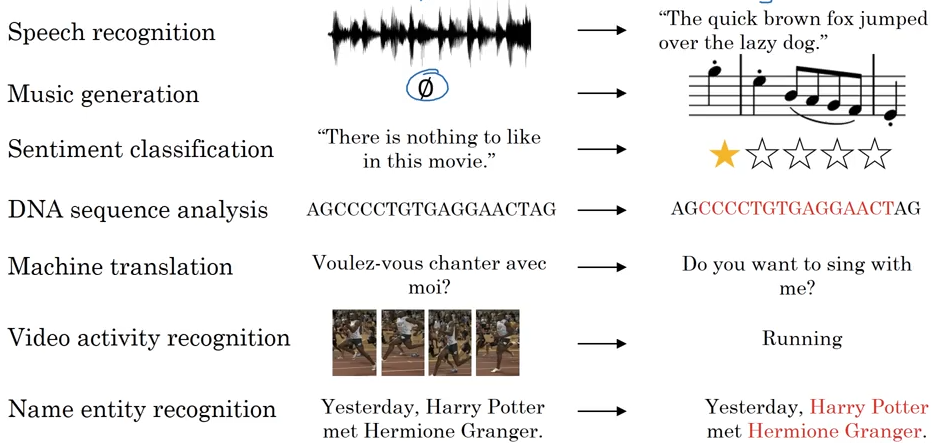
**Recurrent Neural Networks (RNN)**

Estos modelos han transformado reconocimiento del habla, procesamiento natural del lenguaje y otras áreas.

**¿Por qué modelos secuenciales?**

Primero veamos unos ejemplos de datos secuenciales a través de un modelo supervisado de aprendizaje.



Por ejemplo, reconocimiento de audio transcribe un audio a oraciones. Generación musical puede crear música (partituras), clasificación sentimental puede asignar un rating a una oración que rankea una película por ejemplo, análisis secuencial de ADN se encarga de ver que parte de la cadena corresponde a una proteína, traducción es otra de las posibilidades, también se puede a partir de una secuencia de imágenes determinar la acción que se está realizando o determinar a partir de una oración los sujetos de la misma.

**Notación**

A modo de motivación, supongamos que tenemos el siguiente ejemplo



Como salida del mismo querríamos saber cuáles de esas palabras son caracteres de Harry Potter por ejemplo entonces la salida esperada seria la siguiente, un uno si la palabra corresponde con el objetivo y un 0 si no.



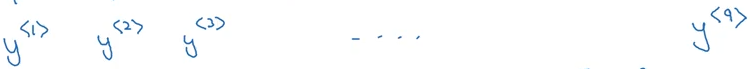
Dado que la entrada posee 9 palabras, la salida posee nueve números, 1 o 0.

Introduciendo terminología, a las entradas las escribiremos como

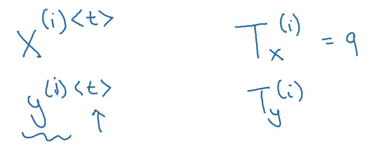


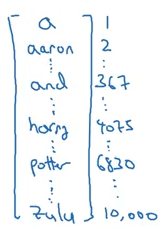
Y a la longitud de este vector como .

De la misma manera para la salida



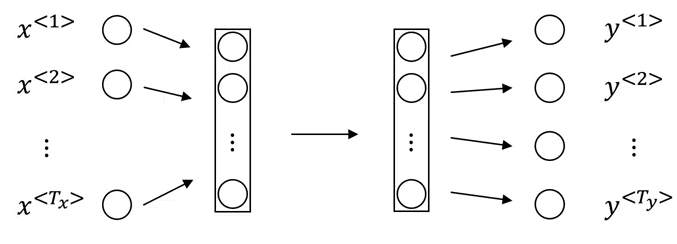
Con . Si hay varias entradas y por ende salidas esto se representara como sigue





Veamos ahora como representar palabras individuales en una oración. Lo primero que debemos hacer es definir un vocabulario o diccionario. Esto es una lista de palabras que se utilizaran en las representaciones. Un ejemplo de esto se muestra en la figura de la derecha junto con las posiciones que ocupan en el array. En este caso son 10mil palabras, pero usualmente se utilizan entre 30 y 50mil, aunque algunas aplicaciones llegan a tener hasta 100mil palabras. Dicho esto, una manera de representar palabras en una oración seria a través del método one-hot. El cual es un array de 0 o 1s del tamaño del vocabulario que indica con un uno la posición de la palabra en el vocabulario. Por ejemplo, si la palabra es Harry habrá un 1 en la posición 4075 del vector. Por último, en el caso de que haya una palabra que no esté en el vocabulario, hay un espacio reservado denominado <unkw> de unkown o desconocida.

**¿Por qué no utilizar una red neuronal estándar?**

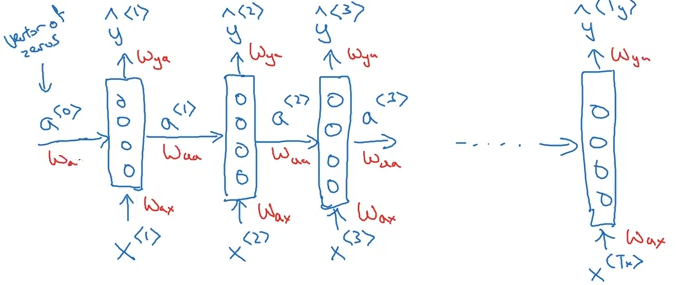


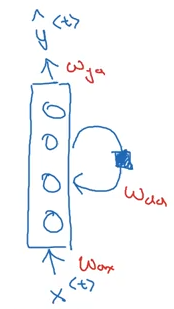
Haciendo uso del problema explicado anteriormente, dada una oración podemos ir viendo palabra por palabra (Cada una de las 9 de la oración de ejemplo; con viendo nos referimos a que sea un input de la red) y analizar la salida que será un 0 o 1 de acuerdo a si la misma representa un nombre de persona. Pero esto posee dos problemas principales:

* Las entradas y salidas pueden ser de distintas longitudes para distintos ejemplos.
* No se comparten características aprendidas a lo largo de las distintas posiciones del texto; Por ejemplo, si Harry aparece en la primera posición de la oración y se aprende que es un nombre de persona y luego aparece en alguna otra parte entonces la red deberá aprender todo de nuevo y no se hizo uso de lo aprendido previamente.

Dado estos problemas utilizaremos **redes neuronales recurrentes (RNN)**. A continuación, veremos que son.

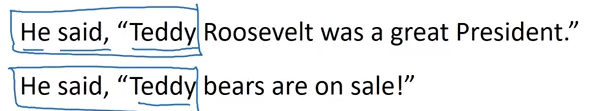
La primera palabra se alimentará con una capa de una red neuronal tratando de predecir si la misma es un nombre de persona o no. Similarmente con la segunda palabra, pero la activación es alimentada del proceso anterior. Lo mismo es hecho para el resto de las palabras que componen la oración. El diagrama de esta red neuronal recurrente se muestra en la figura inferior.



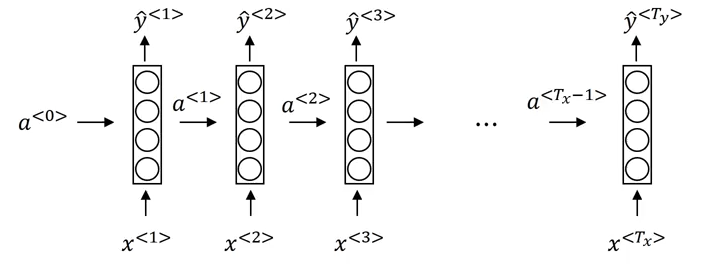


Esta misma red se suele simplificar en términos de la notación como sigue. Habrá, a su vez, un conjunto de parámetros asociados para cada time step que gobiernan las conexiones desde la entrada a la red neuronal oculta.

Una de las limitaciones de esta red es que no usa información de la oración posterior. Es decir, en el ejemplo siguiente no es posible determinar si la palabra Teddy forma parte de un nombre o no a partir de las primeras tres letras, mientras que si se leyera la oración completa se sabría que en el primer caso si forma parte y en el segundo no. Las redes neuronales que se encargan de este tema se llaman **Bidirectional Recurrent Neural Networks (BRNN)**.



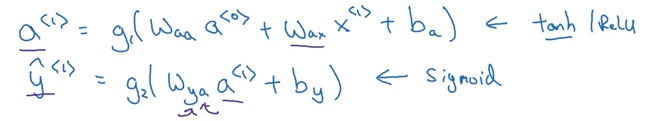
A partir de una imagen limpia de una RNN, veamos los cálculos que hace la misma. Empecemos con los cálculos de forward propagation

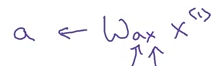




Como se dijo anteriormente la primer activación es un vector de ceros.

Mientras que la siguiente activación y la primera salida se calculan como sigue, donde g es una función de activación. Como dice la imagen, para las activaciones las funciones pueden ser tanh o ReLUs, mientras que, para las salidas, funciones sigmoides.



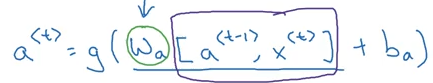
En cuanto a la terminología utilizada para los pesos w, con dos subíndices, el primero hace referencia a la variable que se quiere calcular, mientras que el segundo a la variable sobre la cual es multiplicada.

De manera general, las funciones de activación y la salida se calculan como sigue

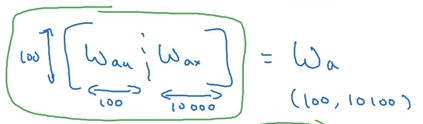




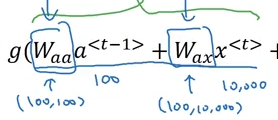
Pero antes de continuar simplifiquemos la notación de las mismas. Comprimamos los primeros dos términos de la primera ecuación.



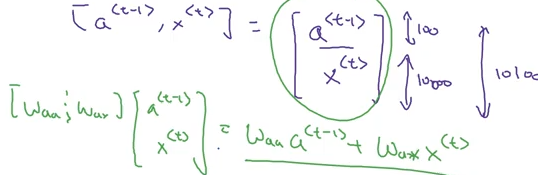
En la ecuación superior se introdujeron dos términos. Las dos matrices W se concatenaron en una sola de la siguiente manera.



Poniendo unas dimensiones la matriz, si la activación a tiempo t-1 posee dimensión 100 y la entrada a tiempo t 10mil, entonces Waa debe tener dimensión (100,100) y Wax (100,10mil). Uniendo las dos matrices se obtendrá una matriz Wa de dimensión (100,10100).



Por otra parte la terminología de corchetes introducida para multiplicar Wa es la siguiente donde los términos simplificados se muestran en rojo.



Por otra parte, la ecuación para el cálculo de la salida se computa de la siguiente manera



**Backpropagation a través del tiempo**

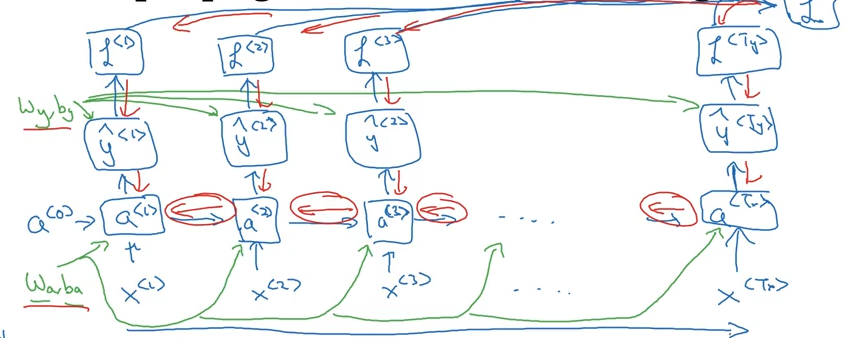
Para calcular el backpropagation, hay que calcular la función de coste. La misma se calcula a tiempo t como la regresión logística usual.



Mientras que la función de coste total es la suma de estas pérdidas.



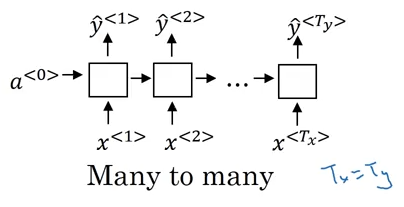
El cálculo de backpropagation (y la función de costo) se muestra en la figura siguiente en rojo. Luego del cálculo de cada salida se calcula la función de coste correspondiente a tiempo t. Calculadas todas estas se modifican los parámetros a través de backpropagation utilizando derivadas de los parámetros correspondientes, como es usual.



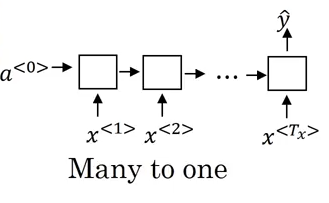
**Diferentes tipos de RNNs**

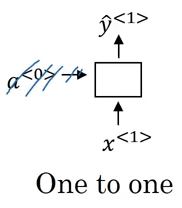
A continuación veremos otros tipos de RNNs que incluyen tener un número de entradas distintos al número de salidas (Tx distinto a Ty). Un ejemplo de esto puede ser cuando se mostraron ejemplos de datos secuenciales, al realizar traducciones, para generación musical o clasificación sentimental (rankeo de películas a partir del review).

La red neuronal vista previamente donde se cumple que Tx = Ty se la llama Many-to-many.

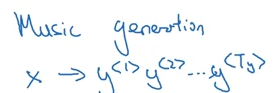
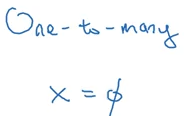


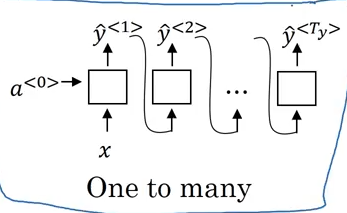
Mientras que en el caso de clasificación sentimental donde se requiere una salida, el tipo de red utilizado es Many-to-one.



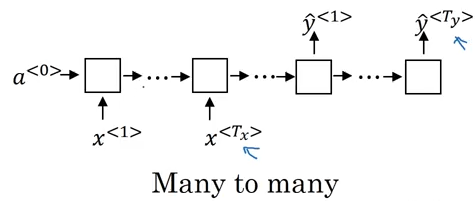
Por una cuestión de completitud también se tiene la red one-to-one que sería la red neuronal estándar, explicada en los dos primeros cursos de la especialización.

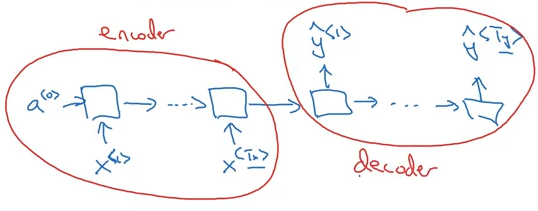
Por otra parte, la generación musical se puede realizar con redes del tipo one-to-many



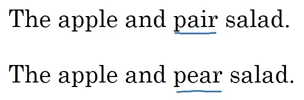
Todavía queda ver un caso particular de many-to-many en por ejemplo, traducciones donde la entrada puede tener distinta longitud que la salida.



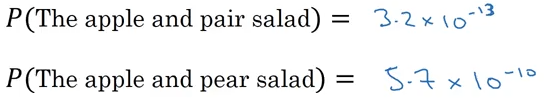


**Modelado del lenguaje y generación secuencial**

Dada por ejemplo una oración que suena similar a otra en reconocimiento del habla. Para determinar si se dijo una frase o la otra, se suele utilizar el contexto de la frase teniendo en cuenta que una es más probable que la otra. En el ejemplo siguiente, ¿se dijo pair o pear? Suenan ambas de manera muy similar.



Un modelo de lenguaje interpretara que lo más probable es que se haya dicho la segunda oración dado que es más probable que la primera.

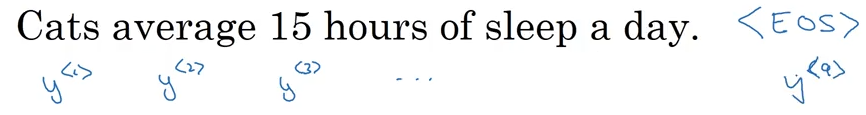


Es decir, lo que el modelado del lenguaje hace es determinar la cantidad

Pero, ¿cómo se modela un idioma con una RNN para determinar estas probabilidades?

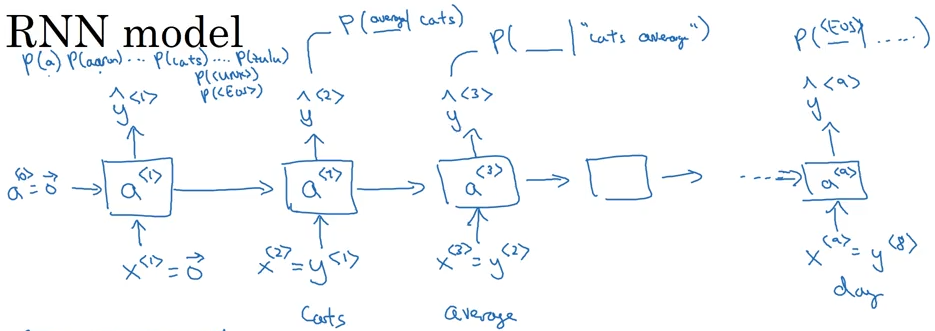
Training set: un texto grande de texto en el idioma.

Dada una oración que se quiere medir la probabilidad



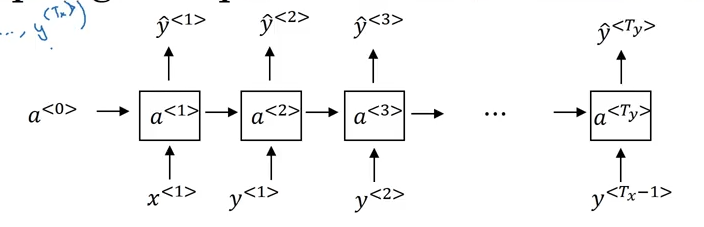
Donde se puede agregar un end-of-sentence al final. Si hay una palabra que no está presente en el vocabulario, se puede reemplazar por el token <unk>.





**Obteniendo oraciones nuevas (Sampling novel sequences)**

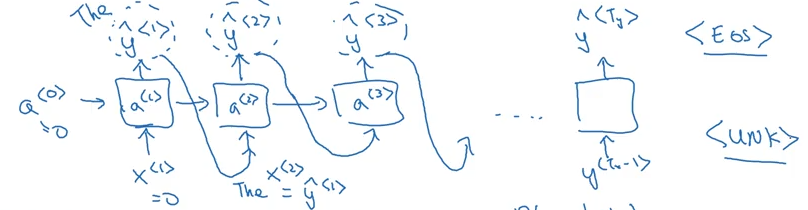
Dada la siguiente RNN, lo que queremos hacer es samplear oraciones originales a partir de una distribución de probabilidad



Aunque en realidad, la red neuronal presentada anteriormente se utiliza para entrenar la red. Para samplear se utiliza una modificación de la misma. De manera aleatoria se hace como muestra la figura inferior. Seteando a0 y x1 a 0, se calcula una palabra aleatoria con la cual empezar la oración o se elige una en particular



Luego esta alimenta el paso siguiente de la red para determinar la siguiente palabra y así sucesivamente. Esta iteración termina hasta que se llegue a un EOS o hasta que uno decida si se quiere un número determinado de palabras. Por otra parte, si se obtiene un UNK, se puede volver a samplear hasta obtener una palabra distinta.



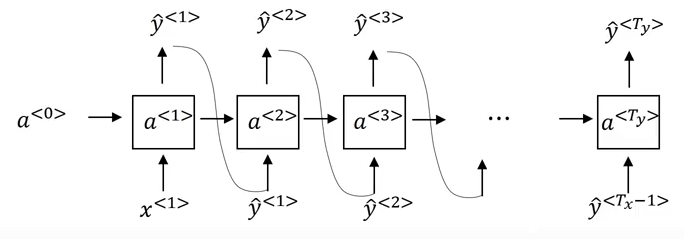
**Modelo de lenguaje a nivel carácter**

En lugar de construir un vocabulario de palabras como se hizo anteriormente

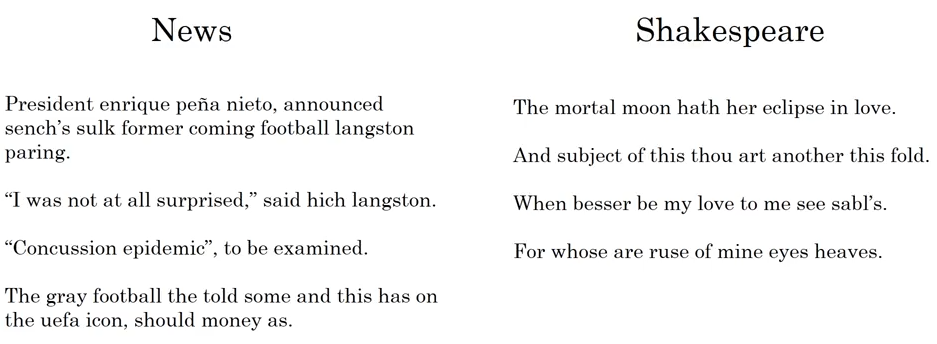


Se puede construir un vocabulario de caracteres.

De esta manera usando un modelo de este tipo para la oración anterior (“Cats average 15 hours…”) se obtendría que c=y1,a =y2, etc. Usando modelado de lenguaje a nivel caracteres no deberíamos preocuparnos por UNK. A la palabra Mau se le asignaría una probabilidad distinta de cero. Mientras que si no estuviera esta palabra en el vocabulario en el modelado a nivel de palabras, a esta se le asignaría un UNK. Una de las mayores desventajas de este modelo es que se terminan con oraciones mucho más largas. Muchas oraciones en ingles poseen entre 10 y 20 palabras, pero tendrán decenas de caracteres. Entonces el modelo a nivel carácter podría no capturar el rango de dependencias a largo alcance. A su vez, son computacionalmente más caras de entrenar.



Un ejemplo de modelado de lenguaje de nivel carácter se muestra a continuación. La primer columna fue obtenida a partir de artículos de noticias mientras que la segunda de textos de Shakespeare.



**Gradientes desvanecientes (vanishing gradients) con RNNs**

Dependencias de largo alcance en oraciones. Si dice gato el verbo es singular, si dice gatos, debería ir plural.

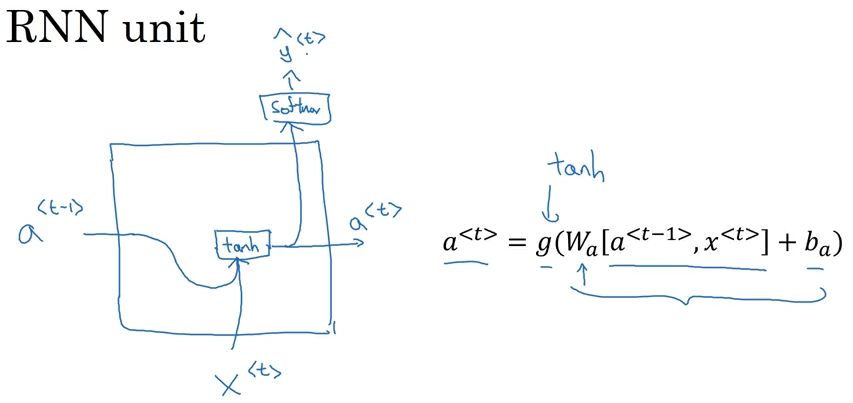


La RNN que vimos hasta ahora tiene una influencia local. Es decir, las palabras se ven afectadas principalmente por las palabras cercanas.

Recordemos que también se puede ver el efecto de gradientes que “explotan” al ver que hay parámetros con NaNs, aplicando gradient clipping, re escalar vectores de gradientes para que no haya overflow. En el caso de vanishing gradient debemos introducir **Gated Recurrent Units (GRU)** para introducir dependencias de largo alcance.

**Gated Recurrent Units (GRU)**

Recordemos que una Unidad de RNN viene definida de la siguiente manera, y usemos la misma para explicar el concepto siguiente.



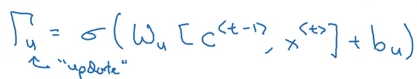
*[El siguiente tema esta basado en los siguientes trabajos:* [*Cho, Kyunghyun, et al. "On the properties of neural machine translation: Encoder-decoder approaches." arXiv preprint arXiv:1409.1259 (2014)*](https://arxiv.org/abs/1409.1259) *y* [*Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).*](https://arxiv.org/abs/1412.3555)*]*

GRU units van a tener una nueva variable, la célula de memoria, C. a tiempo t, esta poseerá el valor, por ahora, a la activación.

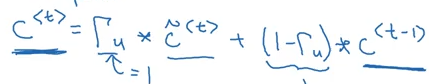
En cada paso temporal, consideraremos reemplazar C con un valor C tilde,



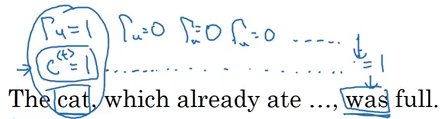
La idea importante de GRU es que se considerara actualizar o no el memory cell de acuerdo a otra variable gamma



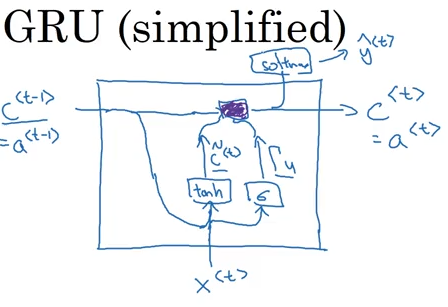
Por lo cual esta actualización viene dada de la siguiente manera



Mediante el ejemplo de la oración que estamos viendo esto se ve de la siguiente manera



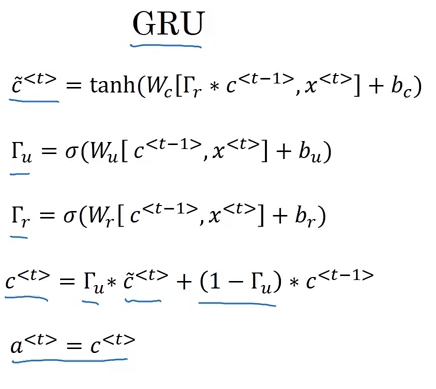
Un esquema simplificado de esta compuerta se muestra a continuación



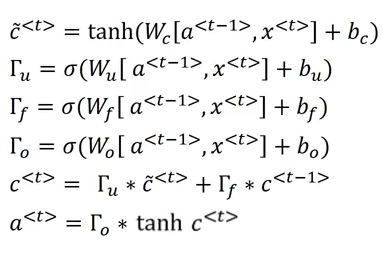
Haciendo un resumen de las ecuaciones y reescribiendo c tilde con Γr (relevancia) llegamos a las GRU completas (Full GRU)

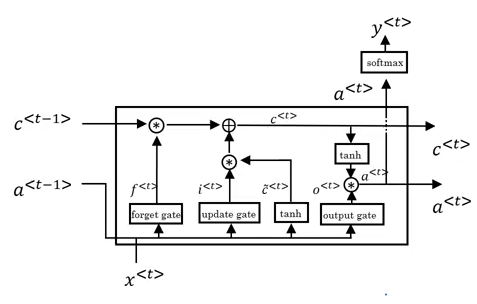


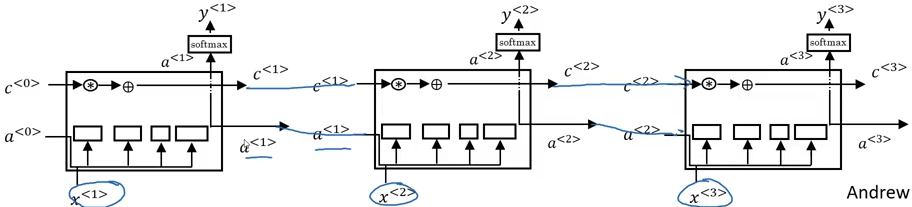
**Long Short Term Memory Unit (LSTM)**



*[Este texto esta basado en la siguiente investigacion:* [*Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.*](https://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735)*]*

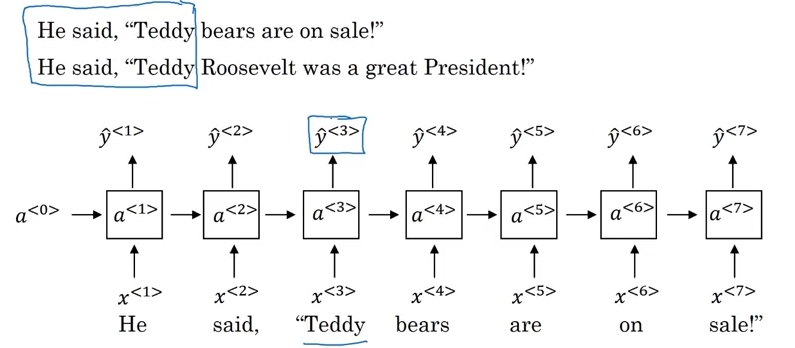




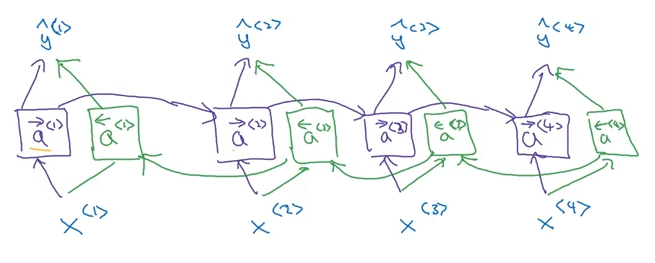


Para construir modelos más poderosos faltan introducir dos conceptos. Uno es RNNs bidireccionales que permite en un punto del tiempo tomar información de antes y después en la secuencia. Por otra parte, Deep RNNs se verán luego.

A modo de motivación, no hay manera de saber si Teddy es un nombre de persona o no conociendo las primeras tres palabras, independientemente de si las unidades de la RNN son GRU o LSTM.



Para construir una red bidireccional, se insertan activaciones que mueven el flujo de información en sentido derecha-izquierda, que a modo de distinguirlas con las previas se les inserta una flecha horizontal indicando el sentido de flujo de la información. Entonces el grafico resultante es uno acíclico. La salida queda determinada por ambas activaciones, una que tiene en cuenta la información pasada y otra que tiene en cuenta la futura. Notemos que todos los cálculos son realizados en forward propagation.



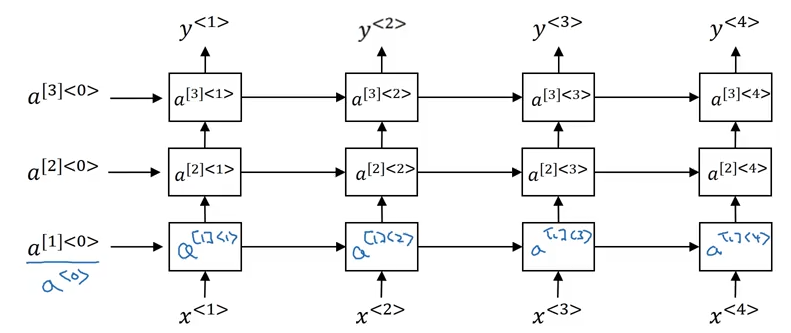
El cálculo de la salida puede expresarse entonces como



Como desventaja, este método necesita tener toda la secuencia determinada para poder procesar la información. Por ejemplo, en speech recognition habría que esperar a que la persona termine de hablar.

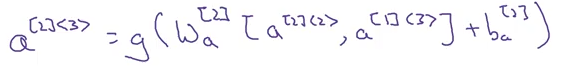
**Deep RNNs**

Para lograr procesamientos más completos de datos secuenciales a veces es necesario acumular varias unidades de RNNs para determinar una salida en particular.



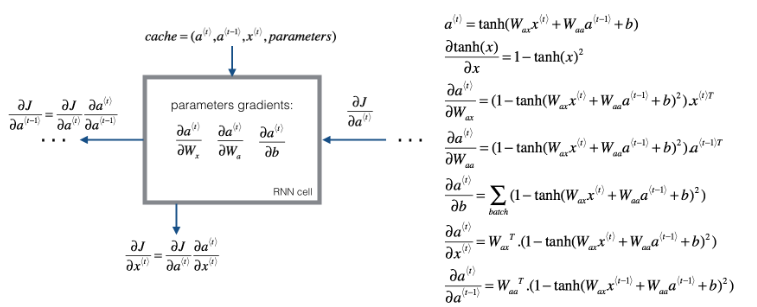
Donde se introdujo una nueva notación a las activaciones donde l significa en número de capas y t el tiempo. En el caso de la imagen superior se muestra una RNN profunda con tres capas.

A modo de ejemplo, a23 se calcula como sigue



Aparte de este método se pueden agregar redes ocultas que no estén conectadas entre si horizontalmente.

A modo de completitud, las ecuaciones correspondientes a backpropagation son las siguientes



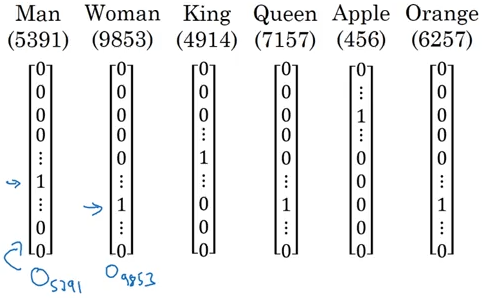
**Week 2: Natural Language Processing & Word Embeddings**

#### **Introduction to Word Embeddings**

Es una manera de entender relaciones entre palabras, como hombre-mujer, rey-reina. Esto nos permitirá desarrollar aplicaciones de NLP.

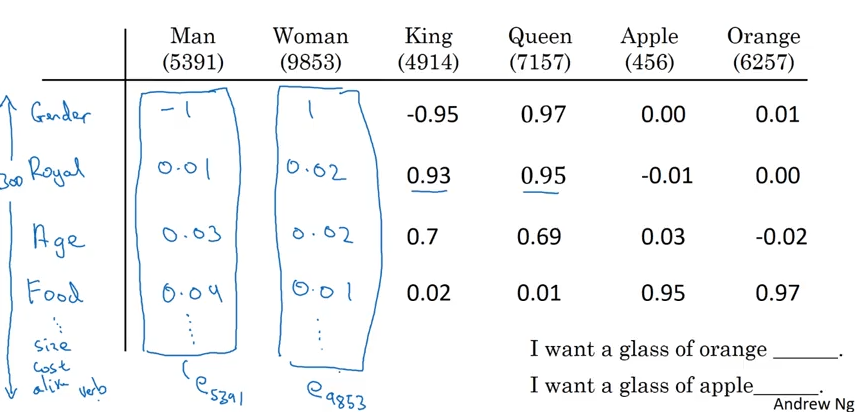
**Representación de palabras**

Palabras representadas por un vocabulario y el método de one-hot vector. Las desventajas de este método es que las palabras son tratadas por separado y no existe una correlación entre las mismas.



Para esto se les asignan características a las palabras, como género, royal, edad, etc. con números entre -1 y 1, donde cercanos a uno indican una correlación fuerte y un 0 no relacionado entre si.

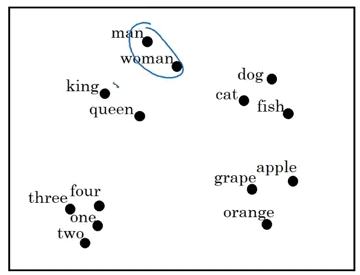
Dado que para manzana y naranja comparten prácticamente valores similares para las características. Esto da a pensar que, si queremos completar las oraciones mostradas abajo, entonces si terminan con dos palabras que son similares, es muy probable que se completen con la misma, en este caso juice (jugo).



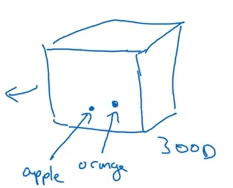
**Visualización de word embedding usando t-SNE**

Esta seccion esta basada en el siguiente trabajo: [Maaten, Laurens van der, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of machine learning research 9.Nov (2008): 2579-2605.](http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf)

La técnica llamada t-SNE permite visualizar datos en un numero alto de dimensiones al asignarle a cada punto un punto del espacio 2D o 3D.

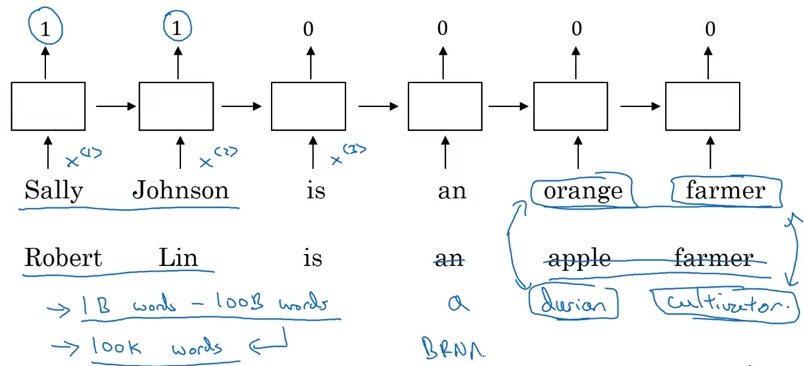


Vemos en la figura de la izquierda que hombre-mujer tienden a estar agrupados juntos. Lo mismo sucede para rey-reina o animales. Esto mismo sucede para números o fruta. A su vez los seres vivientes suelen estar agrupados más cerca entre si que aquellos que no lo son.



**Usando word embedding**

Veamos cómo usar los word embeddings para aplicaciones en NLP. Por ejemplo, si sabemos que Sally Johnson es el nombre un orange farmer, entonces se podría determinar que Robert Lin es el nombre de un apple farmer debido a los features representation. Pero si en lugar de ser apple farmer es durian cultivator, identificar el nombre ya es más complicado porque es probable que no se pueda utilizar feature representation para esto. Una manera de solucionar este inconveniente podría ser el de utilizar una base de datos ya entrenada de 1B-100B de palabras en lugar de la que uno podría disponer más pequeña de 100k. Esto sería Transfer Learning usando word embeddings.



**Transfer learning y word embeddings**

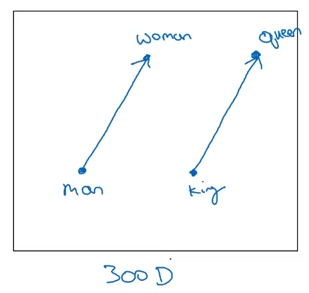
* Aprender word embeddings desde un texto largo de billones de palabras o descargar un word embedding online pre-entrenado.
* También se pueden transferir word embeddings a nuevas tareas donde se tienen sets de entrenamiento muchos más chicos. En lugar de utilizar un one-hot vector de 10mil posiciones se puede utilizar un vector denso de 300D.
* Opcional: continuar ajustando word embeddings con nueva información.

Word embedding ha sido útil para tareas donde se tienen base de datos pequeñas.

**Propiedades de los word embeddings**

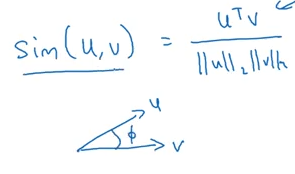
Si sabemos que hombre-mujer están relacionados, ¿entonces se puede determinar con que palabra se corresponde rey, por ejemplo? Una manera de determinar esto es restando las columnas de embeddings y ver que son similares. Esto fue mencionado por primera vez en el trabajo: [Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig. "Linguistic regularities in continuous space word representations." Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2013.](http://www.aclweb.org/anthology/N13-1090)





Queremos encontrar ela palabra w que maximiza la similitud entre



**¿Qué función de similitud se utiliza?**

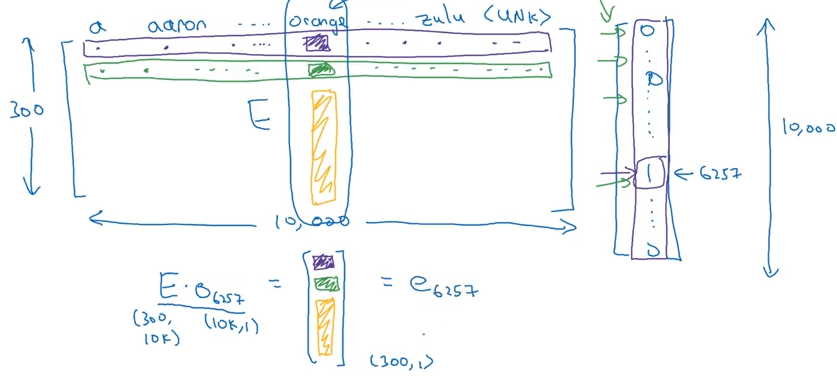
**Cosine similarity**

Correr un word embedding sobre un texto de gran longitud puede provocar que se obtengan relación entre las mismas como las siguientes

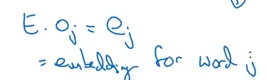


**Embedding Matrix**

Cuando entrenamos una red para obtener un word embedding lo que se hace es obtener una matriz embebida. Esta matriz posee dimensión igual a #palabras en el vocabulario \* # de features. Al multiplicarla por cualquier one-hot vector se obtiene un vector de features correspondiente a la palabra a la cual el one-hot se corresponde.



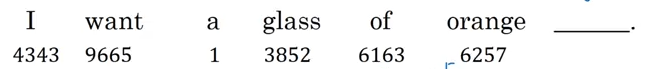
De manera general, se obtiene entonces



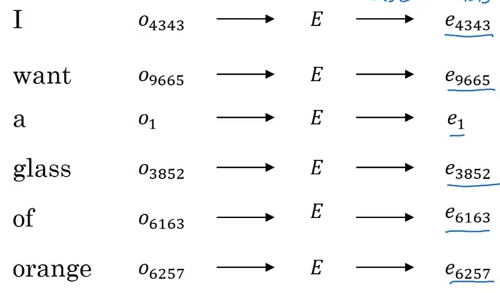
**Neural language model**

Este modelo fue desarrollado a partir del trabajo: [Bengio, Yoshua, et al. "A neural probabilistic language model." Journal of machine learning research 3.Feb (2003): 1137-1155.](http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf)

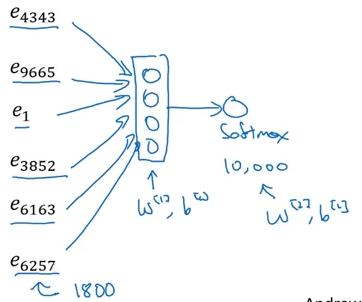
Modelo para predecir la siguiente palabra en la secuencia. Por ejemplo, dada la siguiente oración, queremos saber cómo continuarla.



Primero creamos los vectores de embedding

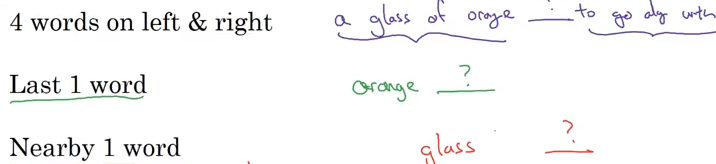


Estos vectores son la entrada de una red neuronal conectada a un Softmax del tamaño del vocabulario prediciendo la palabra necesaria. Si cada vector embebido posee un tamaño de 300, entonces 300\*6=1800, será el tamaño de la entrada. El esquema de esta red se representa en la red siguiente



También podemos tratar de predecir la palabra utilizado únicamente una parte de la oración, lo cual disminuiría el número de entradas y parámetros de la red.

Veamos otro contexto. Definiendo contexto de acuerdo a las palabras que anteceden/prosiguen a la palabra a determinar. Por ejemplo, podría ser las 4 palabras previas, las 4 a la izquierda y las 4 a la derecha, la palabra anterior o, algo que funciona bastante bien, 1 palabra cercana, llamado esto skip-gram.

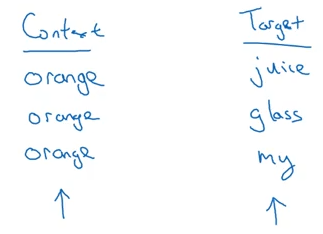


**Modelo Word2Vec**

Este modelo fue desarrollado en el siguiente paper: [Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).](https://arxiv.org/abs/1301.3781)

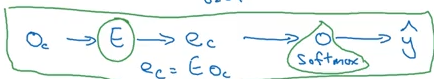
**Skip-gram model**

Este modelo crea un par de datos supervisados de la siguiente manera: una palabra que marca el contexto se determina aleatoriamente. Por otra parte, la palabra de target/objetivo se crea al azar también hasta un cierto número de palabras al azar hacia adelante o hacia atrás del contexto. Dada la oración “*I want a glass of orange juice to go along with my cereal*”, unos ejemplos. De pares de embeddings son:

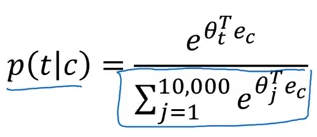


La idea del paper es aprender buenos word embeddings, no solucionar un problema de aprendizaje supervisado correctamente.

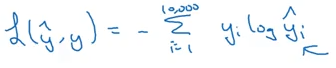
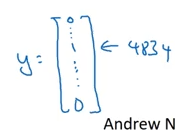
Resumiendo, dado un vocabulario de 10k palabras, la red neuronal será de la siguiente manera



Donde Softmax viene dada por

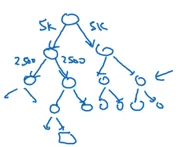


Donde theta se corresponde con los parámetros asociados con el el target t. El costo se calcula como



Donde y se corresponde con el one-hot vector del target.



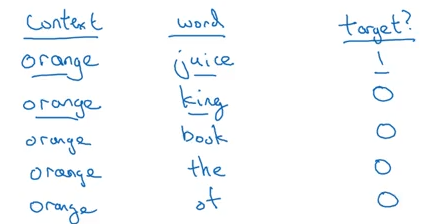
Aunque, este modelo presenta un inconveniente al calcular Softmax. El denominador requiere hacer una suma sobre el vocabulario completo. Con 10k palabras no hay problema, pero como un vocabulario más grande la sumatoria se vuelve costosa computacionalmente. Una solución a esto podría ser utilizar un clasificador Softmax Hierarchical (jerarquico) que nos dice primero si el target esta en las primeras 5k palabras o en las ultimas 5jerárquicocesivamente. En la práctica las palabras más comunes están al principio del árbol, mientras que las menos comunes se encuentran en la parte más profunda del mismo.

¿Cómo se samplea el contexto c? Dado esto el target simplemente se samplea dentro de una ventana de esta palabra. Si sampleamos de manera uniforme lo más probable es que aparezcan palabras frecuentes todo el tiempo como the,of,a,and,to,… En lugar de esto hay algunas técnicas heurísticas que se pueden utilizar.

**Negative sampling**

Este trabajo fue presnetado en el siguiente paper: [Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.](https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf)

El gran problema de skip-gram es lo que tarda softmax. Este algoritmo trata de que este proceso sea mas eficiente.

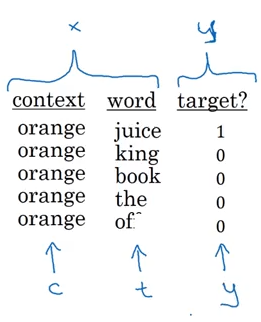


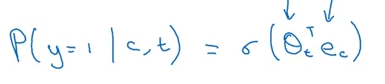
La idea es crear un metodo supervisado mas eficiente. La idea es samplear un contexto y un target y asociar si las miasmas estan asociadas o no, es decir si son un par valido de contexto- target.

Dado un par de palabras, ¿estas poseen target 1 o 0? Asi se genera el training set. Si consideramos k como el numero de repeticiones por palabras (filas) en las que repetimos cada contexto. Se suele utilizar k entre 5 y 20 para datasets pequeño y k entre 2 y 5 para datasets grandes.

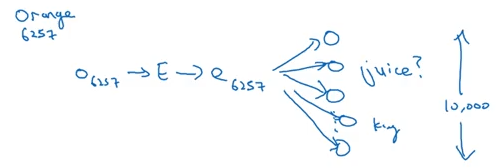
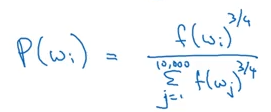
Describamos el modelo de aprendizaje supervisado para aprender a mapear desde x a y. .

El training set vendra dado por

 Para estimar la probabilidad de que y = 1, definimos



En lugar de entrenar el softmax con 10k palabras vamos a entrenarlas con 5 (k +1, k # de negativos)



Pero,¿ como seleccionamos los ejemplos negativos? Se toma un valor heurisitico. Un valor inversamente proporcional (con una potencia) al numero de veces que aparece.

**GloVe (Global vectors for word representation) word vectors**

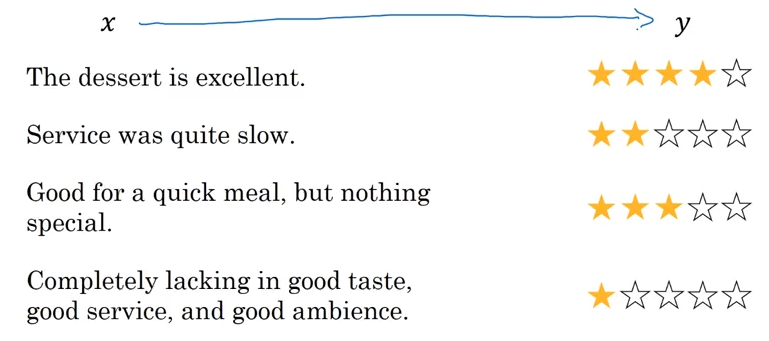
Este trabajo fue obtenido del siguiente trabajo: [Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.](http://www.aclweb.org/anthology/D14-1162)

Antes: pares de palabras cercanas. En cambio glove mide el numero de veces que la palabra i aparece en el contexto de j, X\_ij.

El modelo glove pretende minimizar la siguiente funcion de coste donde la terminologia parece confusa pero w transpuesta por w tilde es lo que nosotros definimos previamente como theta transpuesta por el vector de embedding e. Por otra parte, f(xij) se define por el hecho de que si Xij es cero entonces el log del mismo diverge y, para evitar esto, f se define igual a cero en esa situacion. Para mayor informacion respecto a formas funcionales de f, redirigirse al paper.

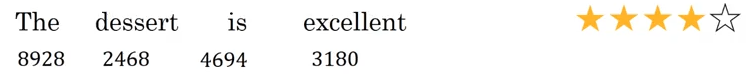
Sentiment Classification

Veamos como aplicar lo que vimos hasta ahora. Un ejmplo de clasificacion sentimental podria ser como sigue

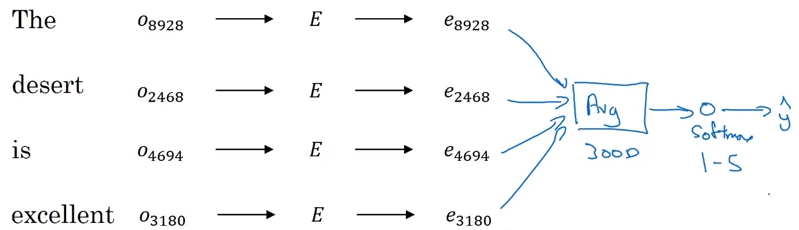


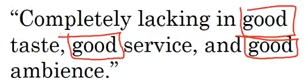
El problema con clasificacion sentimental es que no se cuenta con un data set tan grande. Los training sets para este tipo de tareas son de entre 10mil a tal vez 100mil palabras.

Un modelo de clasificacion sentimental simple seria como sigue. Dado el siguiente review



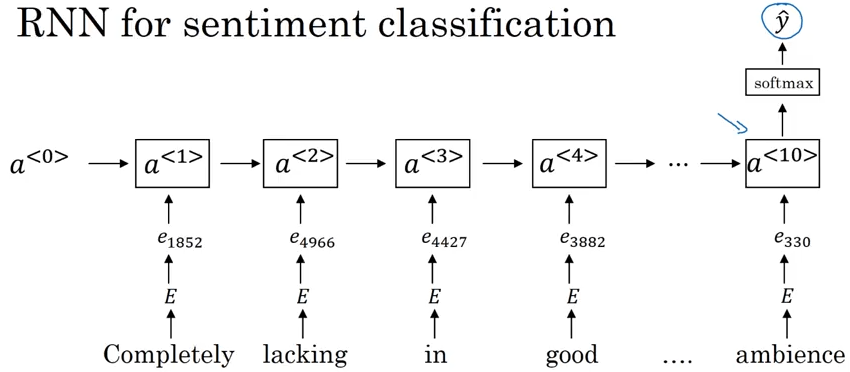
Se puede agarrar cada one-hot vector de la palabra, la matriz E de embedding entrenada sobre 100B de palabras, y asi obtener el vector de embedding, el cual se alimenta a una red neuronal que promedia los valores de rating y se obtiene la salida de softmax entre 1 y 5 que corresponden al rating.



Lo malo de este alforitmo es que ignora el orden de las palabras entonces podemos tener un review negativo con muchas palabras positivas y se obtendran resultados erroneos.

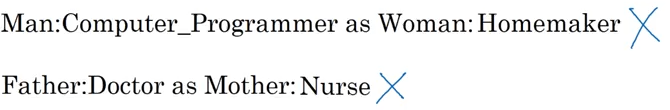
RNN para clasificacion sentimental

Un ejmemplo de many-to-one RNN, se muestra a continuacion el cual hara un review mejor que el metodo presentado anteriormente.



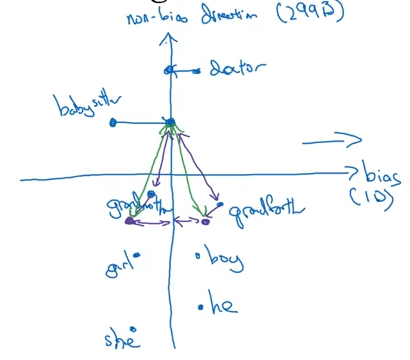
Eliminando el bias de word embeddings

A contnuacion se mostraran tecnicas para eliminar el bias de word embeddings. Con bias no nos estamos refiriendo a bias-variance, sino al bias de genero, etnicidad, etc. Para entender esto basta con mirar el titulo del paper que trató este tema: “Man is to Computer Programmer as Woman is to Homemaker?”.



Este tema fue tratado en el paper que se muestra a continuacion: [Bolukbasi, Tolga, et al. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." Advances in Neural Information Processing Systems. 2016.](http://papers.nips.cc/paper/6228-man-is-to-computer-programmer-as-woman-is-to-homemaker-debiasing-word-embeddings.pdf)

Veamos com oeliminar el bias con un caso particular de bias de genero. Esto necesita 3 pasos.

El primero seria identificar la direccion del bias. Por ejemplo restar los pares de palabras de aquellas que estan relacionadas, por ejemplo he-she, male-female y promediar.

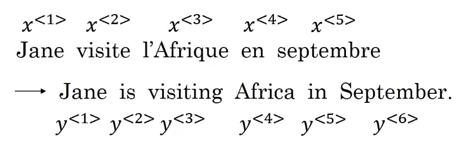
Segundo, cada palabra que no este definida en los paress anteriores se debe proyectar para eliminar el bias. Esto se conoce como neutralizar. Por ejemplo como sucede en el caso de babysitter y doctor.

Tercero y ultimo, equilibrar los pares. Por ejemplo teniendo en cuenta, grandmother y grandfather o girl-boy, para el caso de babysitter esta mas cerca del lado femenino que masculino, entonces para eliminar este bias lo que se suele utilizar es hacer que los pares esten equidistantes la dimension sobre la cual no deberia haber bias.

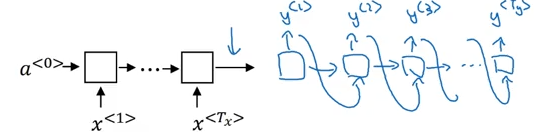
Week 3: Varuius Sequence to Sequence Arquitechtures

Sequence to Sequence model

Supongamos que queremos traducir una oracion de frances a ingles. Las ideas que se van a presentar fueron obtenidas del paper [Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.](http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf) Y [Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).](https://arxiv.org/abs/1406.1078)

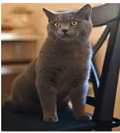


Consideremos primero una parte de una red, llamada encoder con entradas dadas por la oracion en frances conectada la salida a otra red llamada decoder cuya salida es la oracion en ingles. Este metodo ha demostrado funcionar relativamnete bien.



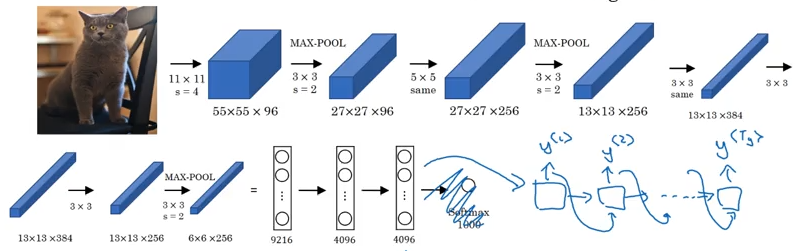
A su vez, se puede utilizar Image Captioning usando RNN. Esto es, dada una imagen describirla. Esto fue obtenido de los trabajos

* [Mao, Junhua, et al. "Deep captioning with multimodal recurrent neural networks (m-rnn)." arXiv preprint arXiv:1412.6632 (2014).](https://arxiv.org/abs/1412.6632)
* [Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.](https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Vinyals_Show_and_Tell_2015_CVPR_paper.pdf)
* [Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.](https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Karpathy_Deep_Visual-Semantic_Alignments_2015_CVPR_paper.pdf)

Por ejemplo, dada la siguiente imagen querriamos obtener una oracion como la mostrada.

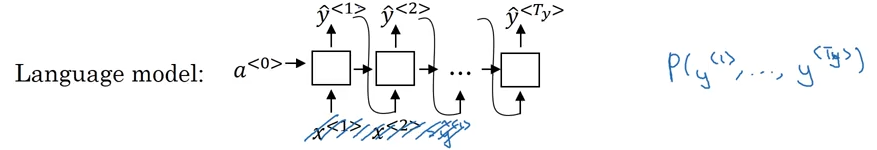


Para ello alimentamos la iamgen de un gato a una arquitectura del tipo AlexNet, pero en lugar de la funcion softmax final, lo que hacemos es conectarle un decoder como vimso anteriormente.

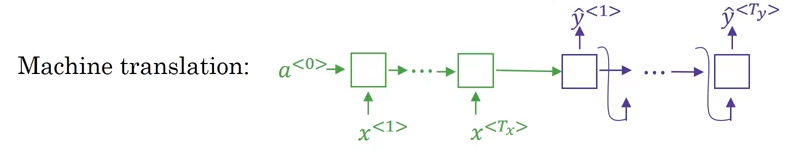


Veamos ahora como poder determinar en el caso de realizar una traduccion, hay una diferencia entre generar texto como vimos antes, que es una oracion traducida aleatoriamente elegido, sino que queremos la mejor traduccion.

Basicamente una traduccion se puede pensar como construir un modelo de lenguaje condicional. Esto es, un language model como vimos se generaba como sigue dad una distribucion de probabilidad.



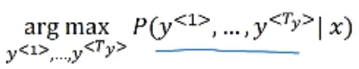
Por otra parte la traduccion viene dada por la estructura siguiente



Donde se puede mirar que la parte purpura coincide con la del modelo de lenguaje, entonces ambos son similares pero en lugar de empezar con un vector de ceros, se empieza con un decorder. Es por ello que se la llama a la traduccion modelo de lenguaje condicional, porque la distribucion de propbabilidad que genera esta basada en una oracion de entrada en otro idioma, en este caso, frances.



Entonces, ¿Cómo encontramos la traduccion mas adecuada? No queremos samplear oraciones al azar. Podemos encontrarnos con oraciones que esten realativamente bien pero otras que no.



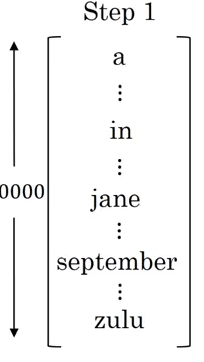
Lo que queremos hacer es encontrar la oracion que maximice la probabilidad condicional. El algoritmo que hace esto se lalma BeamSearch.

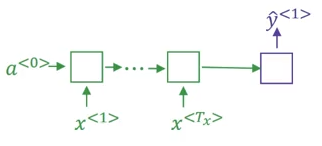
Pero, ¿Por qué no utilizar greedy search? Este algoritmo es similar al anterior. Lo que hace es tratar de encontrar la mejor primer palabra, luego la mejor segunda y asi sucesivamente. Considerando que el algoritmo eligio ‘Jane is’ como las primeras dos palabras de la traduccion lo mas probable es que para la tercera elija going en lugar de visiting dado que es mas usual en el ingles, y acabaria eligiendo una con probabilidad mayor que es como queremos pero que no tiene mucho sentido dado que la primera en este caso es mas adecuada.



Otra desventaja de esto es que dado un vocabulario de 10mi lpalabras, si se quiere formar una oracion de 10 letras acabariamos sampleando 10mil^10 muestras diferentes lo cual es un numero bastante grande. Es por ello que un algoritmo de busqueda aproximada es recomendado en estos casos.

**Algoritmo Beam Search**

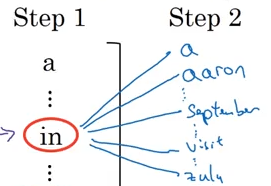
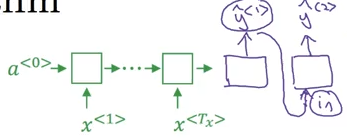






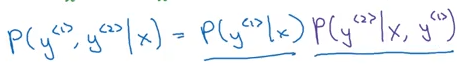
Este algortimo lo que hace es traquear primero, dado como input la oracion a traducir las mejores B palabras, esto es las B palabras con mayor probabilidad como primera palabra de la traduccion. Este parametro B se lo llama Beam Width, que puede valer por ejemplo B=3. Un caso, podria ser que haya elegido In, Jane y September como memjores tres palabras.

Como segundo paso, hay que elegir la segunda palabra, para cada una de las anteriores.



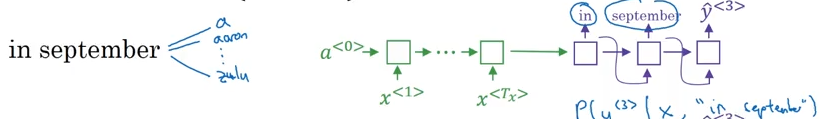
Entonces, lo que se elige es la mayor probabilidad para esa la palabra in. 

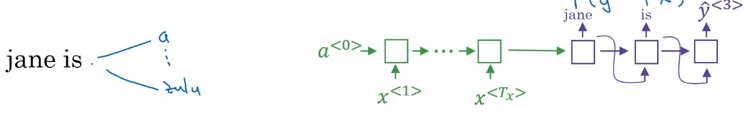
Pero el objetivo final del segundo paso seria calcular la probabilidad del par entre la primera y segunda palabras sea mayor

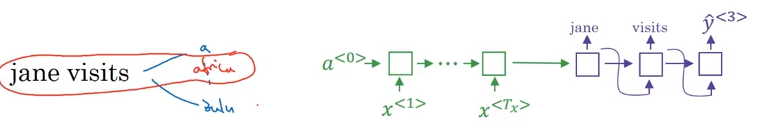


En total deberiamos testear en este paso, 30mil=3\*10mil palabras. Si continuamos con B=3, podriamos tener por ejemplo in-september, jane-is y jane-visits, como los pares mas probables, esto descartaria september como primera palabra.

Similarmente esto se puede realizar para obtener la tercer palabra.





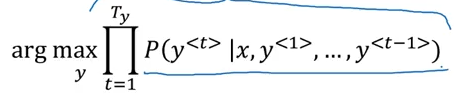


Afortunadamente esto terminaria hasta que lo mas probable sea un EOS.

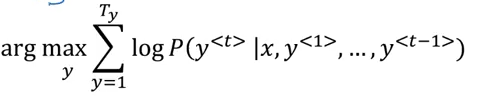


Refinamientos al algoritmo de Beam Search

* Normalizacion de longitud: como vimso anteriormente la oracion final elegida va a ser la que maximice las probabilidades condicionales



Pero dado que estas son un numero pequeño, terminamos teniendo un underflow. Para evitar esto se maximiza en su lugar el logaritmo de la productoria lo que resulta en calcular el maximo de



Resulta que hay que hacer una pequeña modifcacion al realizar la traduccion dado que el algoritmo suele preferir oraciones cortas, dado que mientras mas palabras se agreguen mas chica sera la probabilidad. Para ello se multiplica por un factor

Donde alfa = 0.7 elegido heuristicamente. Si alfa es 1 tenemos una normalizacion total y si es uno no hay normalizacion.

* Otras cuestiones a discutir: el tamaño del ancho del beam (B). si es mas grande se obtendran mejores resultados, pero sera mas lento. Por el contrario, si es pequeño se obtendran peores resultados pero sera mas rapido. Tipicos valores de B son 10, 100, mil y 3mil. Como problema, este modelo de Beam Search no esta garantizado que encuentre el maximo a diferencia de los algorimos no aproximados como BFS o DFS.

**Analisis de errores con Beam Search**

Ahora veremos como detectar si el beam search algorithm o el RNN el que posee errores. Para esto considewremos el siguiente ejemplo, dada una entrada consideremos una salida y\* que escribiria una persona y yhat la obtenida por el algoritmo.

 Entonces si queremos mirar si el error proviene del RNN (encoder-decoder) o del Beam Search, podriamos empezar aumentando el parámetro de busqueda B. Pero aumentar este valor, como asi tambien obtener mas training data puden no llegar a ser una solucion. ¿Cómo sabemos entonces si el error proviene del algoritmo de beam search?

Lo que se puede hacer es considerar las probabilidades de error P(y\*|x) y P(y\_hat|x). Comparemos ambas, pueden suceder dos cosas

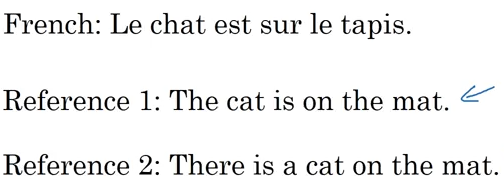
* Primer caso: que P(y\*|x) > P(y\_hat|x). En este caso beam search eligio y\_hat, pero y\* posee una probabilidad mas alta. Entonces beam search posee el problema.
* Segundo caso: P(y\*|x) <= P(y\_hat|x). y\* es una mejor traduccion que y\_hat. Pero el RNN predijo que P(y\*|x) < P(y\_hat|x). La conclusion es que el modelo de RNN es fallido.

Este proceso se puede repetir seguidamente con diferentes ejemplos del devset y tratar de definir en que proporcion los errores son mas probables por beam search o RNN model. En el caso de que el problema venga del beam search se puede tratar de aumentar el parametro B, si proviene del RNN podriamos tratar de agregar regularizacion, agregar mas training data o probar una arquitectura distinta, etc.

**Bleu Score**

Veamos ahora el caso en el que una oracion en frances tiene varias traducciones en ingles posibles. Esta seccion esta basada en el siguiente trabajo: [Papineni, Kishore, et al. "BLEU: a method for automatic evaluation of machine translation." Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002.](https://aclanthology.info/pdf/P/P02/P02-1040.pdf) Donde Bleu significa Bilingual evaluation understudy.

Dada una traduccion hecha por el algoritmo lo que realiza es un score de cuan buena es la traduccion.

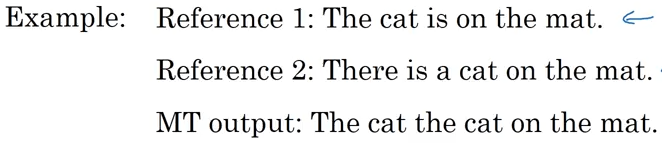


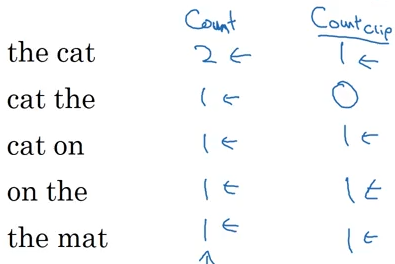
Esto comienza fijandose si cada una de las palabras generadas por el algortimo figuran en la traduccion humana a traves de definir un score llamado precision. Consideremos el siguiente caso extremo en el que la tradccion del algoritmo esta dado unicamente por una palabra repetida



Dada que hay 7 palabras en la oracion y cada una de estas palabras aparece en ambas referencias (1 o 2) entonces la precision daria Precision 7/7. Por lo cual no es un buen metodo. La precision modificada que es la que se utiliza tiene en cuenta no cada palabra sino las veces en las que aparece en la oracion de referencia como maximo. Entonces esto da como resultado 2/7.

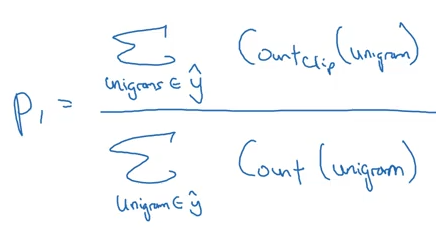
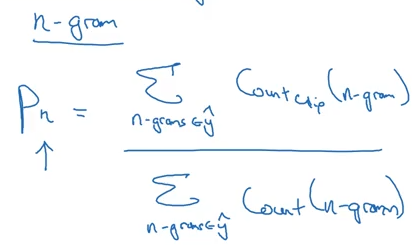
Continuando con otro ejemplo un poco mejor, analicemos el ejemplo del bleu score en bigramas, esto es pares de palabras.



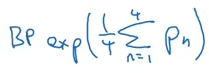
Para analizar el score podemos seleccionar pares de palabras. Entonces los bigramas del MT se listan a continuacion. (de manera similar se pueden tomar triadas de palabras llamado trigramas), cuantas veces aparecen cada uno, y cuantas veces aparecen en cada referencia como maximo (count\_clip).

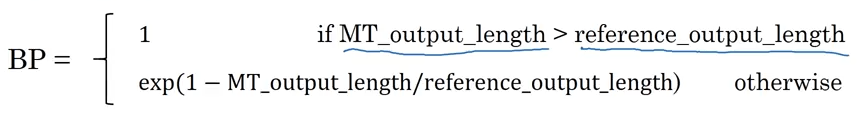
De esta manera nuestra la precision modificada equivale al #count\_clip / # count = (1+0+1+1+1) / (2+1+1+1+1) = 4/6.

Ahora formalicemos esto un poco mas, primero sobre unigramas. Si definimos a la traduccion de la computadora como y\_hat. De manera general para n-gramas la definicion de la preciison viene dada por la figura de la derecha.



A su vez se puede definir el Bleu Score combinado midiendo p1,p2,p3 y p4, donde pn es el bleu score para n-gramas como el promedio de las mismas exponenciada y multiplicada por un factor de penalizacion llamadao Brevity penalty (BP). Done esta penalidad posee una forma funcional particular.



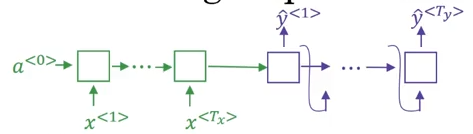


En resumen, Bleu Score es una metrica de un unico numero que permite decidir la calidad de la traduccion de un texto.

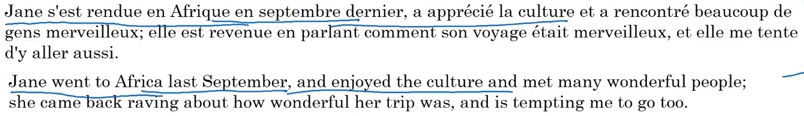
**Attention model**

Este modelo se desarrollo en el trabajo: [Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).](https://arxiv.org/abs/1409.0473)

Basicamnete hasta ahora habiamos visto las redes neuronales consistiendo en una parte del tipo decoder y otra encoder, lo que vamos a ver ahora es un modelo mejorado de esto.

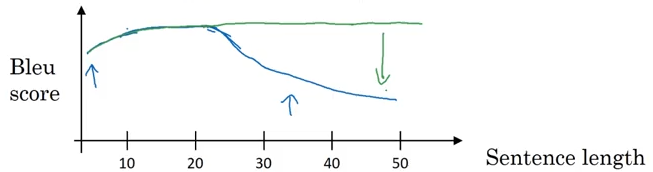


Veamos primero a que se debio este cambio, considerando largas secuencias de texto.



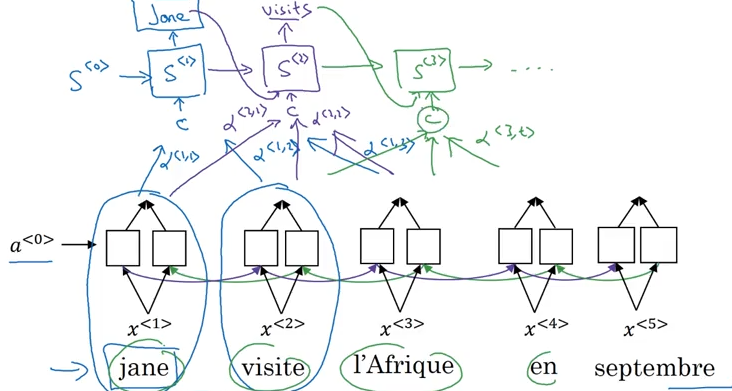
Principalmente el modelo de encoder-decoder toma la secuencia completa de texto y la traduce, mientras que un humano suele hacer esto por tramos de oraciones.

A su vez, se puede notar que el bleu score no funciona correctamente para secuencias mayores a 30 palabras (linea azul de la figura inferior), mientras que el modelo de atencion al ir mirando oraciones mas cortas posee un score mayor para oraciones mas largas.

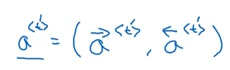


Ilustremos las ideas del modelo con una oracion corta auqnue fue pensado para oraciones mas largas.

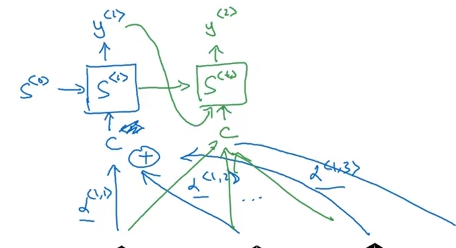
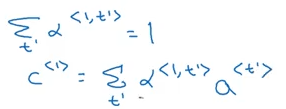
Consideremos un BRNN (bidirectional RNN), el primer paso para generar una salida es considerar que si se traduce la palabra Jane, cuantas palabras se tuvieron en cuenta para generarla? No es necesario que se mire por completo toda la oracion sino tal vez las mas cercanas nomas. Es por eso que se definen unos parametros de peso alfa que seran de entrada a la red neuronal que generara la oracion en ingles.



De manera similar para generar la segunda palabra se generaran una serie de parametros de peso y asi sucesivamente con el resto de las palabras.

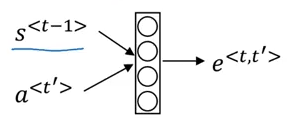
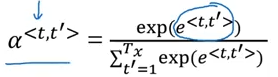
Veamos como funciona este modelo formalmente. Primero definamos a las activaciones forward y backward como una unica activacion evaluada a tiempo t’.

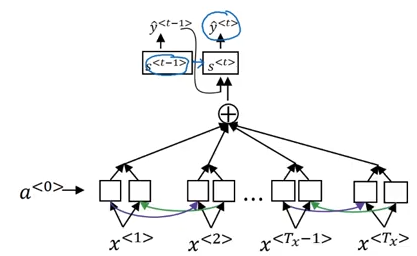
Por otra parte para generar la traduccion, usamos una RNN que NO es bidireccional.

La nueva palabra generada dependera del estado incial S0 y de los vectores de contexto C que son una suma pesada de los pesos alfa con las activaciones.

Lo que queda por ver es como calcular los pesos de atencion alfa. Esto fue obtenido del siguiente trabajo: [Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International conference on machine learning. 2015.](http://proceedings.mlr.press/v37/xuc15.pdf)





Una manera de determinar cuanto tienen que valer los parametros alfa es entrenando una red neuronal.

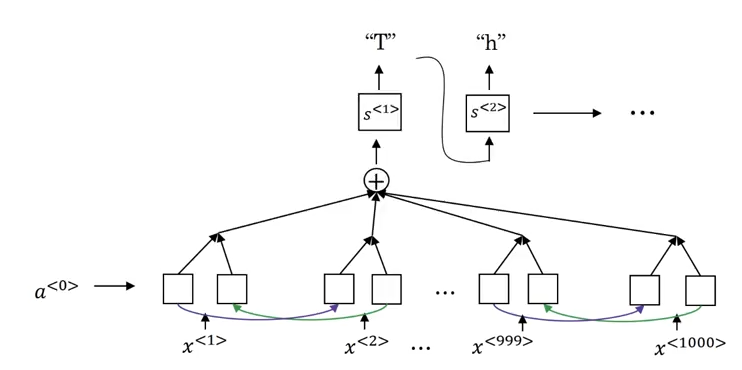
Una dsventaja de este algoritmo es que el timepo de computo va como Tx\*Ty, donde Tx y Ty son el numero de palabras de entrada y salida, respectivamente.

**Reconocimiento del habla (Speech recognition)**

Un ejemplo de esto es transcribir un audio a texto. El oido humano no procesa ondas sin procesar, pero posee estructuras que miden la cantidad de intesniddad a diferentes frecuentas. Esto puede verse omo un espectrograma, que es un grafico de frecuencias en funcio ndel tiempo donde los colores indican la cantidad de energia.



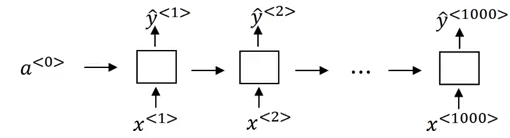
¿Cómo construimos un modelo de atencion para reconocimiento de audio?



Costo CTC para reconocimiento de sonido

Esta idea fue obtenida del trbaajo: [Graves, Alex, et al. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks." Proceedings of the 23rd international conference on Machine learning. ACM, 2006.](https://mediatum.ub.tum.de/doc/1292048/file.pdf) Donde CTC significa Connectionist temporal classfication.

Supongamos que el audio clip decia “the quick brown fox”. Si el audio dura 10 seg con un sampleo de 10Hz entonces el audio posee mil caracteres. A su vez, se puede considerar el tamaño dew la entrada igual que el de la salida.



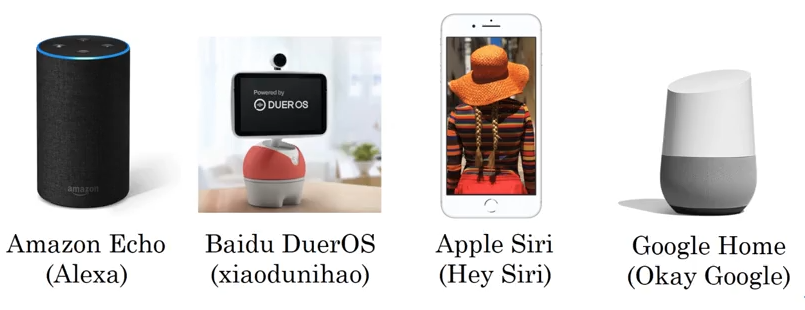
Pero es probable qiue la entrada no necesite ni tenga ese mismo numero de caracteres que la entrada. Un ejemplo podria ser el que sigue



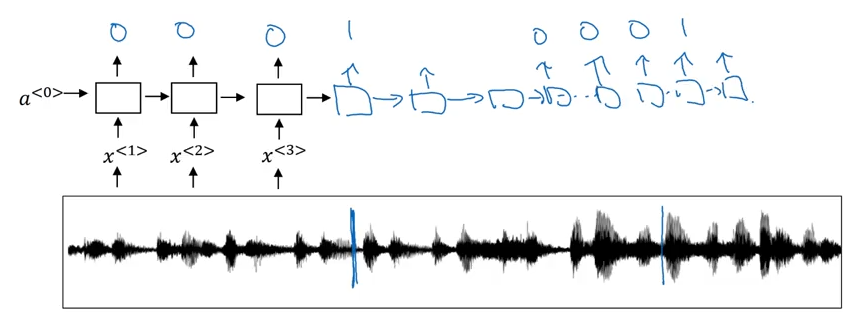
Que seria el comienzo de la oracion “the quick..”. Una regla basica que se podria aplicar a continuacion seria la de colapsar caracteres repetidos no separados por un espacio en blanco. Notemos que tenemos un espacio como un carácter adicional en este caso.

A dia de hoy construir sistema de reconocimiento de audio a nivel de produccion requiere una base de datos muy grande. Ahora veremos como realizar un sistema de deteccion de palabras clave que requiere una cantidad razonable menor de datos.

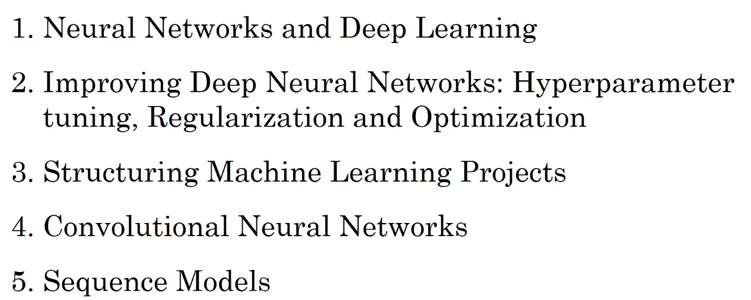
Ejemplos de sistemas que funcionan con una palabra para triggerear la deteccion de sonido son los siguiente.



Veamos un ejemplo de algoritmo que podemos utilizar para word trigger detection. En este caso se posee una deteccion de trigger que es cero en el caso en el que se diga una palabra clave como ‘Okay Google’ y unos luego. Como desventaja de esto es el numero de ceros que posee.



**Conclusion**



*“we wrap up this sequence of courses as we wrap up this specialization I hope that you will find ways to use these ideas to further your career to pursue your dreams but perhaps most important to do whatever you think is the best work you can do our humanity the world today has challenges but with the power of a on power of deep learning I think we can make it a much better place and now that you have this superpower I hope you will use it to go out there and make life better for yourself but also for other people”.* Andrew Ng.