ENTRENAMIENTO DE MODELOS DE LENGUAJE CONEXIONISTAS CON GRANDES VOCABULARIOS

Francisco Zamora[*]

Universitat Jaume I
Dep. de Llenguatges i Sistemes Informàtics
fzamora@guest.uji.es

Salvador España, Jorge Gorbe, María José Castro

Universidad Politécnica de Valencia
Dep. de Sistemas Informáticos y Computación
{sespana, jgorbe, mcastro}@dsic.upv.es

RESUMEN

Las redes neuronales artificiales (RNAs) demuestran su utilidad en ámbitos muy diversos, especialmente en reconocimiento de formas. En particular, los modelos conexionistas resultan especialmente interesantes para el modelado de lenguaje como alternativa a los modelos de tipo estadístico (por ejemplo, n-gramas estadísticos).

No obstante, los modelos de lenguaje conexionistas basados en RNAs presentan algunos problemas de entrenamiento conforme crece el vocabulario del lenguaje a modelar, debido al aumento en el número de parámetros de las redes. El presente trabajo aborda el problema de entrenamiento de RNAs grandes a través de una herramienta que hemos desarrollado, *April*, que implementa eficientemente el entrenamiento de las RNAs.

Keywords: redes neuronales, modelado de lenguaje, n-gramas, grandes vocabularios, entrenamiento eficiente

1. INTRODUCCIÓN

El objetivo de un modelo de lenguaje en un sistema de reconocimiento estadístico de formas consiste básicamente en estimar la probabilidad de una secuencia de palabras $W = w_1, w_2, \ldots, w_{|W|}$. Los modelos de lenguaje de tipo estadístico calculan la probabilidad a priori $P(W)$ realizando la siguiente descomposición:

$$P(W) = \prod_{i=1}^{|W|} P(w_i|w_1^{i-1}) \quad (1)$$

Los modelos de n-gramas [8] simplifican la probabilidad de aparición de una palabra, dadas todas las anteriores, a una historia que sólo considera las $n-1$ palabras anteriores:

$$P(W) = \prod_{i=1}^{|W|} P(w_i|w_1^{i-1}) \approx \prod_{i=1}^{|W|} P(w_i|w_{i-n+1}^{i-1}) \quad (2)$$

Esta probabilidad se puede estimar a partir de un corpus, por lo que se puede obtener de forma automática sin necesidad de introducir conocimiento de tipo deductivo de la lengua para la cual se quiere obtener el modelo de lenguaje. Una de las dificultades de estos modelos consiste en asignar probabilidades a combinaciones de palabras que no hayan aparecido en el corpus. Este problema es inevitable, por lo que es necesario usar métodos de suavizado [5].

Dada la gran utilidad de estos modelos, diversos autores se han planteado su modelización mediante RNAs [9, 13, 2, 4, 7, 10]. Este nuevo paradigma presenta ciertas ventajas e inconvenientes. Entre sus ventajas destacamos el hecho de que un modelo de n-gramas calculado mediante RNAs no necesita aplicar ningún tipo de suavizado explícitamente, puesto que las RNAs interpolan la función que aproximan ante entradas desconocidas.

2. PROBLEMAS ENTRENANDO RNAs GRANDES

El mayor problema de los n-gramas conexionistas se encuentra en la gran cantidad de parámetros a estimar, debido al tamaño de las redes que pueden llegar a aparecer. Si queremos calcular un modelo de n-gramas, la red necesitará tener $(n-1)m$ unidades de entrada, siendo m el número de unidades necesarias para codificar una palabra del corpus. Esto hace que la codificación del vocabulario tenga una importancia decisiva en el coste del cálculo de los modelos. En diversos trabajos previos se han utilizado diversas técnicas para codificar el vocabulario de la tarea [12].

La codificación más sencilla de todas es la *local*, que consiste en usar como representación de cada palabra un vector con tantas componentes como palabras tiene el vocabulario. Este vector tendrá todas las componentes a 0 menos una de ellas que estará a 1 y que indica qué palabra del vocabulario representa dicho vector.

Por otro lado existe la codificación *distribuida*, que consiste en utilizar vectores con un tamaño inferior a la talla del vocabulario, de manera que se pueden decidir formas muy diversas de codificar las palabras. Una forma automática de obtener esta codificación es mediante RNAs como la de la figura 1, donde se utilizan redes que tienen tantas entradas y salidas como palabras del vocabu-

[*]Este trabajo ha sido parcialmente financiado por el proyecto de la Generalitat Valenciana GV06/302.

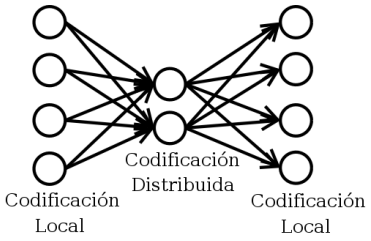

Figura 1. *Topología de red para calcular automáticamente la codificación distribuida de un vocabulario, en este caso con 4 palabras de vocabulario y 2 unidades de codificación.*

lario (Ω), y una capa oculta relativamente pequeña, típicamente $\log_2 \Omega$. Es aconsejable disponer un par de capas ocultas más, una entre la entrada y la capa oculta original, y otra entre esta última y la salida[3]. La red se entrena situando en la entrada y en la salida la codificación local de cada una de las palabras, de manera que el patrón de entrada y de salida deseada es el mismo. Una vez está entrenada la red, la activación de la capa oculta con cada una de las palabras del vocabulario sirve como codificación distribuida.

Otro de los factores decisivos en el entrenamiento de redes con muchos parámetros es el alto coste computacional del algoritmo de entrenamiento. Para poder reducir el tiempo de los entrenamientos hemos desarrollado la herramienta *April*, con una implementación muy eficiente del algoritmo de retropropagación del error con *momentum* [14] y con un gran número de utilidades que permiten manipular conjuntos de datos, haciendo mucho más fácil la obtención de los pares de patrones entrada/salida de las RNAs, además de conseguir un uso reducido de memoria. Más adelante comentaremos el coste temporal de los experimentos, y cómo esta herramienta nos ha ayudado a reducirlo.

3. MODELO DE N-GRAMAS CONEXIONISTAS

Para poder calcular n-gramas mediante RNAs se construyen redes con una topología muy específica. Primero hay que decidir la forma de codificación del vocabulario, y después el número de unidades que tendrá la capa oculta. Construiremos una red con $(n-1)m$ entradas, donde n es el valor del n-grama y m el número de unidades necesarias para codificar una palabra, de manera que la entrada de la red para la posición j de la secuencia W será $h_j = w_{j-n+1}, \ldots, w_{j-2}, w_{j-1}$, codificando cada w_i con m unidades. La salida de la red tendrá tantas unidades como palabras tenga el vocabulario (Ω), y la capa de salida calculará la probabilidad a posteriori de *todas* las palabras del vocabulario:

$$P(w_j = i|h_j) \qquad \forall i = 1, \ldots, \Omega \qquad (3)$$

3.1. Topología de las redes

Hemos señalado algunos detalles de la topología general de las redes en lo que respecta a las entradas y salidas. Vamos a examinar más características, y algunos problemas de la codificación de las palabras.

La codificación distribuida puede ser obtenida de formas muy diversas, presentando algunos métodos más problemas de entrenamiento que otros. Así, por ejemplo, la técnica de la figura 1 converge muy lentamente para ciertos tamaños de entrada. Pero es posible introducir una capa extra en la red completa, como se ve en la figura 2, con la que se calculan los n-gramas, de manera que esta capa extra realice la codificación. La entrada de la red será la codificación local de las $n-1$ anteriores palabras de la frase, y la salida de la red la probabilidad a posteriori de *todas* las palabras del vocabulario (expresión 3). Por tanto, ahora tendremos una red que tendrá $(n-1)\Omega$ unidades de entrada. Se añadirá una nueva capa de unidades que no tendrán conexiones todas a todas. Esta nueva capa estará dividida en conjuntos de N_p unidades, de manera que cada conjunto se conectará con Ω unidades de entrada, y tendremos $n-1$ conjuntos, de manera que el conjunto i estará asociado con las unidades de entrada que representen a la palabra i*-ésima*. A esta capa la denominaremos capa de *proyección*, y es la capa que realizará la codificación distribuida del vocabulario. Para reducir el número de parámetros y que la codificación de una palabra sea siempre la misma, independientemente de la posición de aparición en la entrada de la red, se ligan las conexiones entre los conjuntos de la capa de proyección [10]. Sin esta ligadura de conexiones, la convergencia de los entrenamientos no está asegurada, ya que aparece una cantidad de parámetros excesiva [12]. La ligadura de pesos en las redes es una nueva característica que hemos tenido que añadir a la herramienta *April* para poder llevar a cabo estos experimentos [3] [1].

4. EXPERIMENTACIÓN

Los experimentos se han realizado utilizando una partición del corpus Dihana [1], utilizando la herramienta *April*, desarrollada en nuestro grupo de investigación, para entrenar los modelos neuronales. Los parámetros utilizados para realizar los entrenamientos se muestran en la tabla 1, los detalles de la partición del corpus en conjuntos de entrenamiento, validación y test aparecen en la tabla 2.

Se ha realizado un entrenamiento de las RNAs durante 500 épocas, guardando la red con menor MSE en el conjunto de validación. La figura 3 muestra un ejemplo de la evolución del MSE durante las primeras 40 épocas en el entrenamiento de la red para 4-gramas. El tiempo empleado por ciclo en este entrenamiento ha sido de aproximadamente $18{,}9$ segundos. Este tiempo incluye el coste de

[1] Para ligar los pesos, primero se ajustan los pesos mediante el algoritmo de retroprogación del error, y después, cada conjunto de pesos ligados entre sí se sustituye por la media del conjunto.

Parámetro	Valor
Factor de aprendizaje (η):	0,005 calculado en pruebas preliminares
Momentum (μ):	0,002 calculado en pruebas preliminares
Weight decay (λ):	0,00003 calculado en pruebas preliminares
Función de activación de la capa oculta (f_o):	$tanh$
Función de activación de la capa de salida (f_s):	$softmax$
Función de error (MSE):	*error cuadrático medio*
Tamaño de una palabra en la entrada (Ω):	430 unidades
Tamaño de capa de proyección (N_p):	10 unidades
Tamaño de la capa oculta (N_o):	512 unidades
Valor n de los n-gramas:	$2, 3, 4$

Tabla 1. *Parámetros de los entrenamientos.*

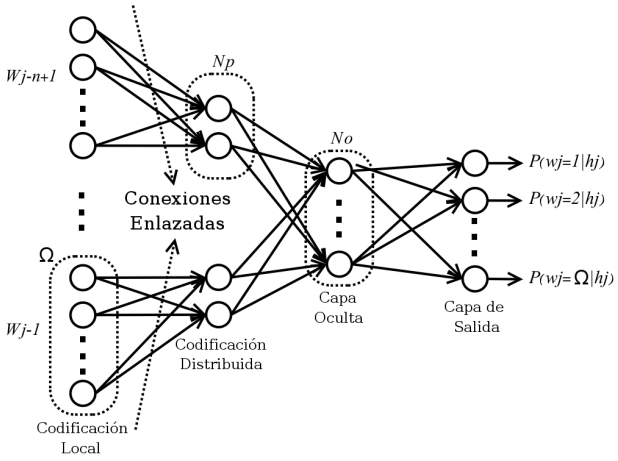

Figura 2. *Topología de red para calcular n-gramas conexionistas.*

	Palabras	Frases	Patrones
Entrenamiento:	3695	460	4152
Validación:	774	100	874
Test:	3185	427	3610

Tabla 2. *Datos relativos a la partición del corpus Dihana utilizada en los experimentos, con un vocabulario de 430 palabras.*

un ciclo completo de las muestras de entrenamiento y de validación, y está muy por debajo de los tiempos que podríamos obtener con herramientas como *SNNS* [15], que en experimentos similares [14] ha llegado a ser hasta 5 veces más lenta. Además, la herramienta *April* ha facilitado también el coste de la preparación de los entrenamientos gracias a las distintas utilidades que posé para manipular este tipo de datos y crear patrones de entrada/salida que hacen un uso reducido de memoria.

Una vez entrenada la red, hemos calculado la perplejidad con el conjunto de test. Los modelos estadísticos de n-gramas han sido obtenidos mediante el *SRI Language Modeling Toolkit* [11] para poder comparar las perplejidades, utilizando la opción que admite proporcionar todo el vocabulario del corpus. Hemos utilizado los conjuntos de entrenamiento y validación para calcular los parámetros mediante el *SRI*.

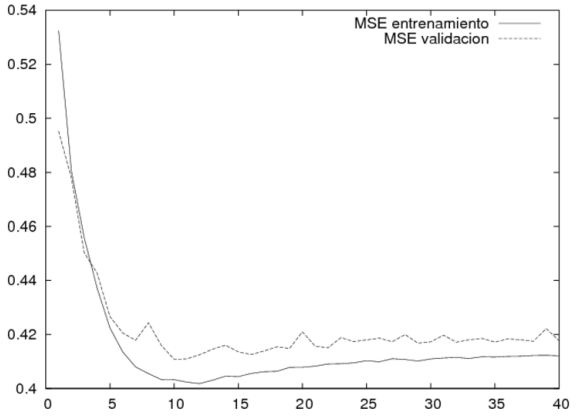

Figura 3. *Detalle de la gráfica de la evolución del MSE durante el entrenamiento del modelo de 4-gramas. La mejor red fue encontrada en la época 10.*

5. CONCLUSIONES

Podemos ver en la tabla 3 que la perplejidad del modelo conexionista es menor en la mayoría de los casos. A pesar de eso, debemos ser precavidos puesto que no

Tipo de modelo	2-gramas	3-gramas	4-gramas
n-gramas conexionistas	26,81	18,96	15,22
n-gramas estadísticos (SRI)	22,35	20,19	20,62

Tabla 3. *Resultados de perplejidad para la tarea Dihana con n-gramas conexionistas y estadísticos, calculados con el SRI. Tarea Dihana.*

existe una relación estricta entre las perplejidades y los resultados generados por un sistema de reconocimiento de habla [6]. Por ello nos planteamos como trabajo futuro la utilización de este modelo en un sistema de reconocimiento del habla para realizar mediciones de las tasas de error reales.

Para lograr reducir el coste temporal de los experiments hemos necesitado emplear un gran esfuerzo en implementar algoritmos y estruturas de datos eficientes. Después de todo este esfuerzo ha surgido la herramienta *April* con la que hemos realizado este trabajo.

6. BIBLIOGRAFÍA

[1] José-Miguel Benedí et al. Design and acquisition of a telephone spontaneous speech dialogue corpus in Spanish: DIHANA. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 1636–1639. European Language Resources Association, 2006.

[2] Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. A neural probabilistic language model. In *NIPS*, pages 932–938, 2000.

[3] C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1996.

[4] María José Castro and Federico Prat. New Directions in Connectionist Language Modeling. In *Computational Methods in Neural Modeling*, volume 2686 of *Lecture Notes in Computer Science*, pages 598–605. Springer-Verlag, 2003.

[5] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 310–318, 1996.

[6] P. Clarkson and A. Robinson. The applicability of adaptive language modelling for the broadcast news task. In Proceedings International Conference on Spoken Language Processing, 1998.

[7] A. Emami and F. Jelinek. Exact Training of a Neural Syntactic Language Model. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 245–248, 2004.

[8] F. Jelinek. Statistical methods for speech recognition. *MIT PRess, Cambridge, MA, USA*, 1997.

[9] M. Nakamura and K. Shikano. A study of English word category prediction based on neural networks. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 731–734, 1989.

[10] Holger Schwenk and Jean-Luc Gauvain. Training neural network language models on very large corpora. *HLT/EMNLP*, pages 201–208, 2005.

[11] A. Stolcke. Srilm – an extensible language modeling toolkit. In Proceedings International Conference on Spoken Language Processing., 2002.

[12] Salvador Tortajada Velert and María José Castro Bleda. Diferentes aproximaciones a la codificación del vocabulario en modelado de lenguaje conexionista. *En Actas de las IV Jornadas de Tecnologías del Habla*, 2006.

[13] W. Xu and A. Rudnicky. Can Artificial Neural Networkds learn Language Models? *Proceedings of the 6th International Conference in Spoken Language Processing*, 2000.

[14] Francisco Zamora Martínez. Implementación eficiente del algoritmo de retropropagación del error con *momentum* para redes hacia delante generales. *Proyecto Final de Carrera.* Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, 2005.

[15] Andreas Zell, Niels Mache, Ralf Huebner, Michael Schmalzl, Tilman Sommer, and Thomas Korb. SNNS: Stuttgart Neural Network Simulator. User manual Version 4.1. Technical report, Stuttgart, 1995.