PROGRAMA DE MASTER
"Inteligencia Artificial, Reconocimiento de Formas e Imágen Digital"

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia

Tesis de Master:
Modelos Conexionistas para el Procesado del Lenguaje Natural

Francisco Zamora Martínez
fzamora@dsic.upv.es

Directora: María José Castro Bleda

14 de noviembre del 2008

Índice general

1. Introducción **9**

2. Modelos conexionistas para el PLN **11**
2.1. Modelos de lenguaje conexionistas 11
2.1.1. Modelo estadístico 11
2.1.2. Modelo conexionista 12
2.1.3. Reestimación de grafos de palabras 13
2.1.4. Parsing conexionista de grafos basado en Viterbi 13
2.1.5. Aplicación en traducción automática 15
2.1.6. Aplicacion al reconocimiento automático del habla 16
2.1.7. Aplicación a tareas de reconocimiento de escritura 17
2.2. Etiquetado PoS con modelos conexionistas 17
2.2.1. Modelo probabilista 17
2.2.2. Modelo conexionista 18
2.2.3. Generación de grafos de etiquetas para procesar frases de test 19

3. Experimentos **23**
3.1. Herramientas utilizadas 23
3.2. Tareas de traducción automática 24
3.2.1. Corpus 24
3.2.2. Experimentos con MLCs 24
3.2.3. Experimentos con modelos de TAE 24
3.2.4. Evaluación de los modelos 25
3.2.5. Resultados 25
3.3. Tareas de etiquetado PoS 26
3.3.1. Corpus 26
3.3.2. Entrenamiento de los modelos conexionistas de etiquetado 27
3.3.3. Entrenamiento de los MLCs 28
3.3.4. Evaluación de los modelos 28
3.3.5. Resultados 28
3.4. Tareas de reconocimiento automático del habla 29
3.4.1. Corpus 29
3.4.2. Entrenamiento de MLCs 30
3.4.3. Experimentos con MOMs 31
3.4.4. Experimentos con MOMs Híbridos 32
3.4.5. Resultados 34

4. Conclusiones **37**

5. Trabajos futuros **39**
5.1. Tarea de etiquetado PoS 39
5.2. Tarea de traducción automática estadística 39
5.3. Tarea de reconocimiento automático del habla 40

Referencias **41**

Índice de tablas

3.1. *Estadísticas de la partición de entrenamiento del corpus NIST'08 (después de extraer* 30 000 *frases al azar).* 25
3.2. *Comparativa de resultados de perplejidad con el corpus de validación para diferentes modelos de lenguaje* 25
3.3. *Resultados comparativos para la tarea de traducción Árabe-Inglés* 26
3.4. *Particiones del Penn TreeBank utilizadas para entrenamiento, validación y test. El número de tokens es el número total de palabras en cada partición.* 27
3.5. *Resultados de clasificación con la partición de valicación para el barrido de contextos. Estos resultados acotan de inferiormente el error que podemos llegar a conseguir con el método de evaluación basado en GADs.* Unknown *es el error de etiquetar las palabras desconocidas, y* KnownAmb *es el error de etiquetado en las palabras ambiguas conocidas.* 27
3.6. *Perplejidad de los MLCs de etiquetas para la tarea de etiquetad PoS.* 28
3.7. *Resultados de etiquetado PoS con la partición de validación.* KnownAmb *es el error de desambiguación para las palabras ambiguas conocidas.* Unknown *al error de desambiguación de las palabras ambiguas desconocidas.* Total *el error total de desambiguación sobre todo el conjunto de palabras de la partición (ambiguas y no ambiguas).* 29
3.8. *Resultados con la partición de test. Otras aproximacions: SVMs (Support Vector Machines), MT (Machine Translation techniques), TnT (Statistic Models), RANN (Recurrent Neural Networks).* 29
3.9. *Estadísticas de la parte del WSJ usada para modelos de lenguaje y de la parte usada en acústica. En el caso de modelos de lenguaje el número de tokens hace referencia al número de palabras, y en el caso de acústica hace referencia al número de tramas acústicas (vectores de parámetros) de cada partición.* 30
3.10. *Perplejidad de los modelos de lenguaje utilizados en los experimentos con la partición de Test.* 32
3.11. *Mejores resultados de los barridos de GSF con la partición Nov92 con los MOMs clásicos. Se indica el WER obtenido y con que valor de GSF.* 32
3.12. *Barrido GSF de MOMs Híbridos con la partición Nov92.* 34
3.13. *Resultados con la partición Nov92 de la mejor configuración de cada uno de los modelos.* 34
3.14. *Resultados con la partición de test de la mejor configuración de cada uno de los modelos.* 34

Índice de figuras

1.1. *Red neuronal para el cálculo de la codificación distribuida de un conjunto de palabras codificadas de forma local.* 10

2.1. *Topología de red neuronal para calcular n-gramas conexionistas (3-gramas en el ejemplo).* 12
2.2. *Fragmento de un grafo de palabras resultante del primer paso del sistema desacoplado, listo para ser procesado y evaluado por los modelos conexionistas, siguiendo el algoritmo descrito en la sección 2.1.4. El grafo tiene un tamaño mucho más reducido que los grafos utilizados en el sistema de verdad. El nodo cuadrado es el estado inicial.* 14
2.3. *Topología de red neuronal para etiquetado PoS.* 18
2.4. *Fragmento parcial de un grafo generado para etiquetar "The ruling follows a host ..." usando un modelo con 2 etiquetas de pasado y 1 de futuro, organizado por etapas, de manera que tiene 6 etapas, 1 por cada palabra a etiquetar más otra etapa adicional que hace de nodo inicial del grafo. La etiqueta de los nodos tiene este formato:* $< l_{i-2} l_{i-1} > l_i < l_{i+1} >$*. Se han omitido las probabilides de los arcos.* 20

3.1. *Evolución del MSE durante el entrenamiento del Modelo de Lenguaje Conexionista de 3-gramas con el corpus WSJ.* 31
3.2. *Gráficas del barrido de GSF con la partición Nov92 para ajustar el reconocimiento, combinando el score del modelo de lenguaje usado en HTK con el de* `April`*.* . . 33
3.3. *Gráficas del barrido de GSF con la partición Nov92 para ajustar el reconocimiento, utilizando de los grafos de palabras de HTK únicamente el score acústico.* . . 33

Capítulo 1

Introducción

El Procesado del Lenguaje Natural (PLN) es un campo muy importante en todas las tareas de Reconocimiento de Formas. Incluso cuando las tareas no están estrictamente relacionadas con el lenguaje natural, las técnicas empleadas en este campo, como pueden ser los modelos de lenguaje, pueden ser de gran utilidad para ayudar al reconocimiento o clasificación de patrones. Por ejemplo, cuando se trata de reconocer secuencias, podemos utilizar modelos de lenguaje para guiar mejor el reconocimiento. O también, en problemas de clasificación, podemos tener un modelo de lenguaje para cada clase, y clasificar las muestras nuevas en función de su entropía con cada uno de los modelos.

Sin embargo todavía ha sido poco explorado el uso de técnicas conexionistas (redes neuronales, ya sean tipo hacia delante o recurrentes) para resolver problemas típicos de este campo de investigación.

La *maldición de la dimensionalidad* hace acto de presencia cuando se trata de tareas de PLN con tamaños de vocabulario ($|\Omega|$) grandes. En concreto, las redes neuronales que se aplican a estos problemas tienen una gran cantidad de parámetros, que aumenta con $|\Omega|$, repercutiendo en la experimentación con problemas de convergencia, y con problemas de coste temporal.

Codificación del vocabulario

Típicamente la representación más utilizada para las palabras en problemas de este campo ha sido la que se conoce como *codificación local*. En esta representación cada palabra se codifica con un vector de tamaño $|\Omega|$, de manera que tienen $|\Omega| - 1$ componentes a 0 y una única componente a 1, que será la componente que identifica a la palabra. Este tipo de codificación es muy dispersa, con lo que complica la capacidad de convergencia de los algoritmos. Para evitar la dispersión de los datos se puede utilizar la *codificación distribuida* [TC06], que consiste en utilizar vectores de tamaño $d << |\Omega|$, donde se puede permitir cualquier tipo de combinación de valores en las componentes de los vectores, siempre y cuando dos palabras diferentes no tengan el mismo vector.

Coste temporal de los experimentos

El entrenamiento de estos modelos necesita también de cantidades muy grandes de datos para poder estimar bien los parámetros. Esto repercute en que el entrenamiento puede pasar de durar unos pocos minutos (para corpus pequeñitos, con $|\Omega| = 100$ o $|\Omega| = 200$) a durar varíos días o incluso meses. En esta situación es necesario tener algoritmos de entrenamiento sofisticados y muy eficientes [EZCG07]. De la misma manera, el reconocimiento se vuelve también muy pesado. Por ejemplo, en un reconcedor de voz, podemos usar un modelo conexionista como

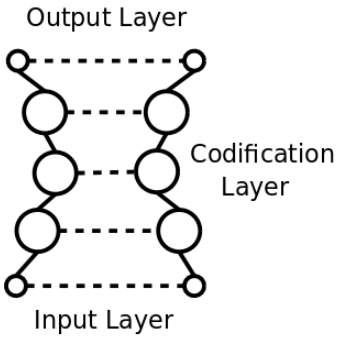

Figura 1.1: *Red neuronal para el cálculo de la codificación distribuida de un conjunto de palabras codificadas de forma local.*

modelo de lenguaje para guiar la búsqueda. Hacer esto de manera acoplada tiene un coste tan grande que se vuelve inviable.

Los primeros trabajos que aparecieron en torno al PLN mediante técnicas conexionistas empezaron por atacar toda esta problemática. Para solucionar el problema de la codificación de las palabras algunos autores han utilizado redes neuronales autoasociativas [SIMY99, Bis95] (mapas autoasociativos). Con ellas se pretende calcular de manera automática una codificación distribuida de las palabras. Habría muchas maneras de hacer esto, incluso se podrían utilizar técnicas como el PCA ("Principal Components Analisys" o Analisis de Componentes Principales). Pero los mapas autoasociativos pueden hacer una discriminación no lineal que a priori parece más interesante que la lineal de métodos como el PCA. En la figura 1.1 se puede ver la topología típica de una red neuronal cuando se usa como mapa autoasociativo. La idea es que tanto la entrada de la red como la salida deseada sean iguales, y que además sean la codificación local de las palabras. Cuando la red ya ha sido entrenada, se puede extraer la codificación local de la capa central, la capa de codificación, evaluando la red con todas las palabras, y extrayendo de cada evaluación la activación de las neuronas de esta capa. Se pueden probar diversas activaciones para las neuronas, pero lo más clásico es que la capa central tenga activación lineal, y el resto activación *sigmoide* o *tangente hiperbólica*.

Partiendo de estas ideas, aparecen los primeros trabajos [BDV00, BB00] donde se muestra una forma novedosa de entrenar los modelos conexionistas al mismo tiempo que se calcula la codificación distribuida de las palabras, todo con el mismo modelo. Estos trabajos han sido ampliados en [Ben, SG05, Sch07, XR00, RSC07, SCjF06, SDG06].

En una línea paralela podríamos situar a los trabajos de [TCP05, Sch94, POF01, NS89, MVUG02, ABCK02, GM04], donde se utilizan modelos conexionistas para llevar a cabo etiquetado Part-of-Speech (etiquetado PoS). En esta aplicación los problemas citados anteriormente desaparecen, ya que estamos hablando de tareas donde el vocabulario de palabras puede ser muy grande, pero el vocabulario de etiquetas es realmente muy pequeño. Esto mejora la convergencia y permite probar una mayor diversidad de modelos.

Capítulo 2

Modelos conexionistas para el PLN

En esta sección vamos a diferenciar dos modelos que, en su esencia, siguen las mismas ideas, pero permiten abordar tareas muy diferentes. Por un lado, tenemos los modelos de lenguaje conexionistas (MLCs), y por otro los modelos de etiquetado PoS conexionistas. La mayor diferencia entre ambos va a ser que en el caso de los segundos podremos utilizar la información del contexto futuro[1] para realizar el cálculo de probabilidades.

2.1. Modelos de lenguaje conexionistas

En este apartado la investigación ha partido de las ideas de [BDV00, Ben, Sch07] para atacar el problema del tamaño de los modelos conexionistas y de la codificación de las palabras. Vamos a ver primero una definición del modelo clásico estadístico, en este caso n-gramas estadísticos, para más tarde presentar el modelo de lenguaje conexionista (MLC).

2.1.1. Modelo estadístico

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:

$$p(W) = \prod_{i=1}^{|W|} p(w_i|w_1^{i-1}) \tag{2.1}$$

Los modelos de n-gramas [Jel97] simplifican la probabilidad de aparición de una palabra, dadas todas las anteriores, a una historia que sólo considera las $n-1$ palabras anteriores:

$$p(W) = \prod_{i=1}^{|W|} p(w_i|w_1^{i-1}) \approx \prod_{i=1}^{|W|} p(w_i|w_{i-n+1}^{i-1}) \tag{2.2}$$

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

[1] En un procesado de la frase de izquierda a derecha, el contexto futuro serán las $|f|$ unidades lingüísticas que siguen a la que actualmente está siendo procesada.

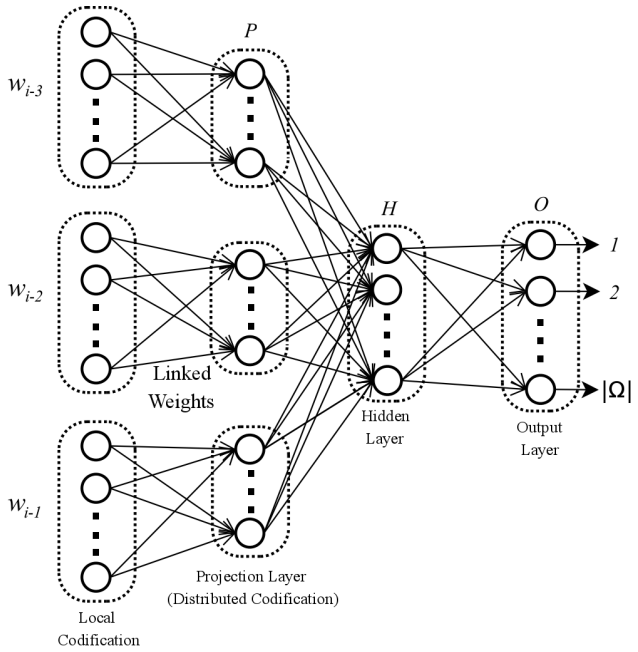

Figura 2.1: *Topología de red neuronal para calcular n-gramas conexionistas (3-gramas en el ejemplo).*

corpus. Este problema es inevitable, por lo que es necesario usar métodos de suavizado [CG96]. Precisamente los problemas del suavizado es una de las características que se pretenden atacar con los MLCs, dada la capacidad de generalización de las redes neuronales.

2.1.2. Modelo conexionista

Los n-gramas han demostrado ser de gran utilidad en el reconocimiento de formas, por lo que se plantea la posibilidad de modelizarlos mediante técnicas que permitan superar algunos de sus inconvenientes. Las redes neuronales no necesitan la aplicación de ningún tipo de suavizado explícito, puesto que interpolan la función que aproximan ante entradas desconocidas.

Para poder calcular n-gramas mediante redes neuronales se construyen redes con una topología muy específica. Primero hay que decidir la 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 ($|\Omega|$), y la capa de salida calculará la probabilidad a posteriori de *todas* las palabras del vocabulario:

$$p(w_j = i|h_j) \qquad \forall i = 1, \ldots, |\Omega| \tag{2.3}$$

El vocabulario se codifica de forma local[2], *pero* para reducir el número de parámetros a calcular se introduce una nueva capa, denominada capa de proyección, en la que cada palabra codificada de forma local es codificada de forma distribuida, y además las conexiones entre las unidades de entrada y esta capa están ligadas [SG05] para cada palabra codificada, de manera

[2] Un uno en la unidad que representa a la palabra de entrada y el resto de unidades a 0.

que reducimos el número de parámetros en un factor de $\frac{(n-2)(|\Omega|+1)P}{n-1}$, siendo n el valor del n-grama, $|\Omega|$ el tamaño del vocabulario y P el tamaño de la capa de proyección (véase la figura 2.1). Teniendo en cuenta que el valor de n en la mayoría de los casos está entre 3 y 4, y que $|\Omega|$ puede llegar a valores entre 1 000 y 20 000 palabras, con una capa de proyección entre $P = (n-1)64$ y $P = (n-1)128$ unidades, estamos hablando en términos absolutos de entre $\frac{(3-2)(1\,000+1)\ (3-1)64}{3-1} = 64\,064$, y $\frac{(4-2)(20\,000+1)\ (4-1)128}{4-1} = 5\,120\,256$ conexiones (parámetros) menos.

Una vez este tipo de redes neuronales han sido entrenadas, es posible reducir el coste de evaluar la red eliminando la capa de proyección. El proceso consiste en precalcular, para cada palabra del vocabulario, la codificación distribuida que se obtiene con la capa de proyección, y generar una tabla auxiliar que asocia esta codificación a cada palabra. El resto de la red se manteniene exactamente igual. Por lo tanto la única capa que se ve modificada es la capa de entrada, que ahora en lugar de recibir una codificación local de las palabras, recibe una codificación distribuida que depende de la tabla auxiliar. De esta forma se puede reducir hasta casi la mitad el número de conexiones en las redes. Sin embargo, la codificación distribuida se ha aprendido de forma conjunta con las probabilidades de aparición de los n-gramas, con lo que es de esperar que la codificación haya sido capaz de observar patrones y haya agrupado a las palabras por su parecido.

2.1.3. Reestimación de grafos de palabras

Los MLCs tienen un gran incoveniente, el cual impide en muchas casos hacer uso de técnicas conexionistas cuando el corpus o la talla de los problemas es muy grande. El inconveniente es el coste computacional necesario para, una vez entrenado el modelo, utilizarlo en el proceso de reconocimiento. Este problema sólo se presenta con redes muy grandes, por ejemplo, los MOMs híbridados con redes neuronales, donde se sustituyen las mixturas de gaussianas por redes neuronales, están siendo utilizados por reconocedores de voz comerciales, sin tener problemas de rendimiento [GMA]. Incluso son más rápidas de evaluar que las mixturas de gaussianas. Sin embargo la tarea de utilizar de forma acoplada los MLCs es todavía inviable.

Esto nos lleva a plantear sistemas desacoplados, en los que el núcleo del proceso de reconocimiento sea ejecutado por modelos que permitan una rápida ejecución, y con capacidad de procesar cantidades muy grandes de datos, y generar con estos datos grafos de palabras o listas de N-best relativamente pequeños, comparados con la cantidad de estados procesados por el algoritmo de Viterbi del núcleo del reconocedor. El modelo conexionista se situaría en un segundo paso, de manera que se utilizaría el modelo para reestimar probabilidades en el grafo de palabras, combinar esta reestimación con las probabilidades acústicas del grafo de palabras (o incluso con la probabilidad del modelo de lenguaje utilizado para generar el grafo), y definitivamente buscar el camino de mayor probabilidad. En la figura 2.2 se puede ver un ejemplo de grafo de palabras resultado del primer paso. Este grafo puede ser obtenido tanto para tareas de reconocimiento de voz/escritura, como para tareas de traducción.

2.1.4. Parsing conexionista de grafos basado en Viterbi

Para poder llevar a cabo los experimentos de MLCs se ha desarrollado un algoritmo que, dado un MLC y un grafo de palabras, es capaz de buscar el camino que maximiza la probabilidad combinada del grafo con el modelo de lenguaje.

Este algoritmo está basado en un interfaz genérico para parsers de grafos que fue presentado en el artículo [BMM07]. Siguiendo este esquema, el grafo es presentado al algoritmo de forma secuencial, siguiendo un orden topológico de los vértices y un orden de las aristas en función

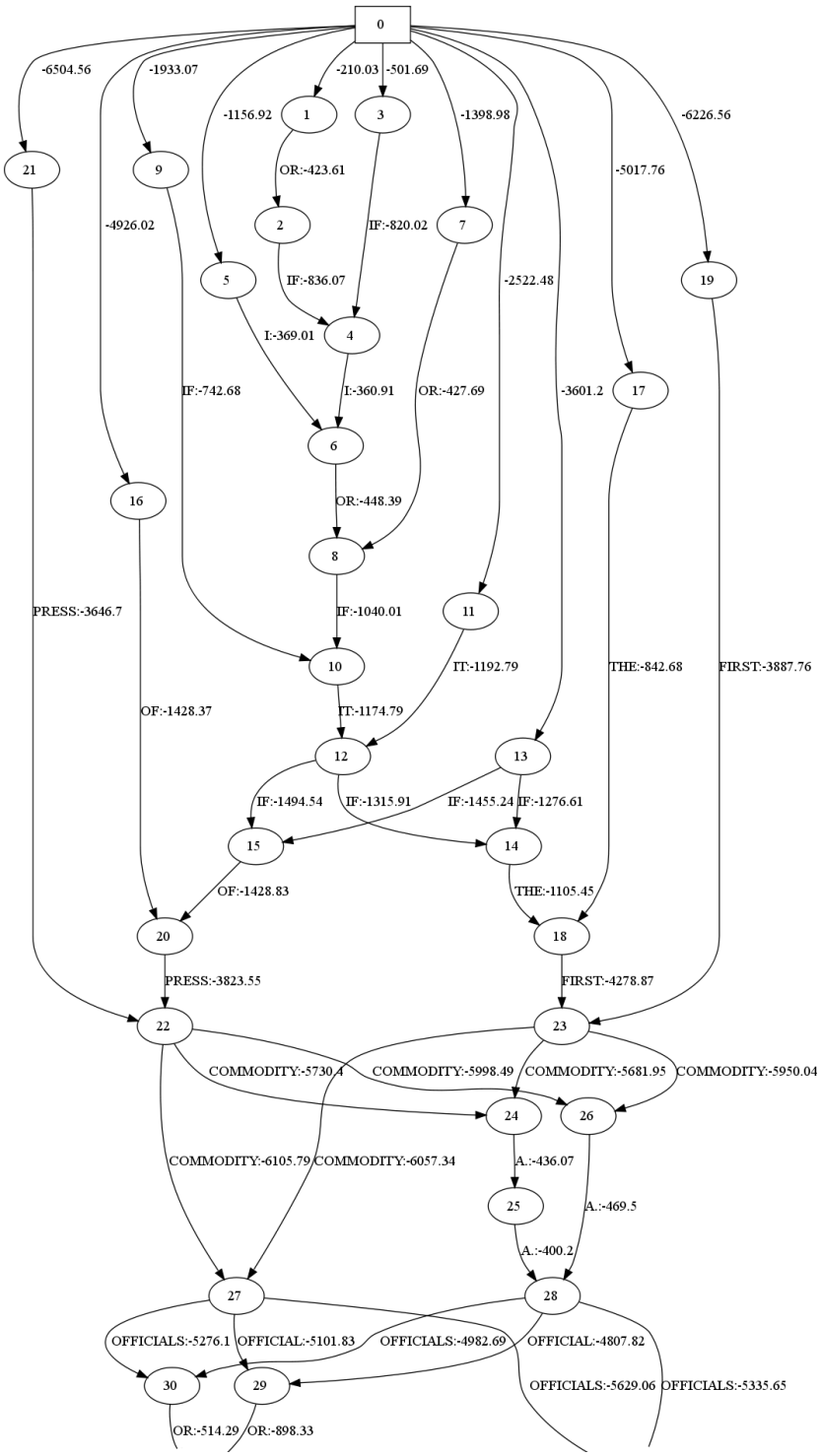

Figura 2.2: *Fragmento de un grafo de palabras resultante del primer paso del sistema desacoplado, listo para ser procesado y evaluado por los modelos conexionistas, siguiendo el algoritmo descrito en la sección 2.1.4. El grafo tiene un tamaño mucho más reducido que los grafos utilizados en el sistema de verdad. El nodo cuadrado es el estado inicial.*

del vértice al que inciden. Este protocolo permite la ejecución del algoritmo de manera online, aunque en este caso lo utilizaremos como si fuera un algoritmo offline.

Siguiendo este interfaz, el parser conexionista implementa un algoritmo de Viterbi basado en listas de estados activos:

- Asocia a cada vértice del grafo una lista de estados activos.
- Para cada arista, el algoritmo extrae uno a uno los estados activos de la lista del vértice origen.
- Aplica la probabilidad asociada a la arista junto con la del MLC utilizando la palabra asociada a esa arista y las $n-1$ últimas palabras de la historia[3], y combina este resultado con la probabilidad que tuviera antes el estado.
- Busca, en la lista de estados activos del vértice destino aquel estado que está identificado por las $n-2$ últimas palabras de la historia del estado activo actual más la nueva palabra de la transición, y, si lo encuentra, maximiza su probabilidad según si la que acaba de calcular es mejor o peor que la que tenía, y si no está, lo crea con la probabilidad que acaba de calcular.

Cuando ya ha terminado de procesar todas las aristas, se extrae el mejor estado activo de los vértices finales del grafo, y se recupera su historia (1-best). Esa historia es la secuencia que ofrece como resultado este proceso.

2.1.5. Aplicación en traducción automática

Abordaremos la traducción automática estadística (TAE). Esta metodología está basada en el principio de traducir una frase origen s en una lengua determinada, en una frase destino t en otra lengua diferente. El problema se formula en términos de la equación (2.4) y puede reformularse como la selección de la traducción de mayor probabilidad de un conjunto de frases objetivo (2.5).

$$\hat{t} = \arg\max_t p(t \mid s) = \tag{2.4}$$

$$= \arg\max_t p(s \mid t) \cdot p(t). \tag{2.5}$$

Esta descomposición se realiza de acuerdo con la regla de Bayes, y el primer trabajo que apareció utilizando esta aproximación fue [BCP+90]. Los sistemas modernos de TAE operan con unidades bilíngües extraídas de corpus paralelos basados en el alienamiento entre las palabras del texto de ambas lenguas. Una modificación de los sitemas de TAE consiste en calcular la probabilidad a posteriori en base a una combinación log-lineal de una serie de características extraídas de varios modelos. Usando esta aproximación es posible combinar M características para la determinación de la hipótesis de traducción como se muestra en la ecuación (2.6):

$$\hat{t} = \arg\max_t \sum_{m=1}^{M} \lambda_m h_m(t, s) \tag{2.6}$$

donde h_m hace referencia a los distintos modelos utilizados (modelo de traducción bilíngüe, modelo de lenguaje para la lengua origen, modelo de lenguaje de la lengua destino, modelo de distorsión para el reordenamiento, etc.).

[3] n se refiere al valor del n-grama, y la historia es la secuencia de palabras que han sido procesadas hasta llegar al estado activo.

La idea es introducir el MLC como modelo de lenguaje de la lengua destino, y hacerlo añadiendo una característica nueva al modelo log-lineal. Nuestro sistema va a ser desacoplado, con lo que en un primer paso se procesa con un traductor base cada una de las frases origen, obteniendo como salida para cada frase una lista de las N-best. Más tarde, se calcula una nueva característica de la combinación log-lineal utilizando el MLC con cada una de las frases de la lista. Finalmente se vuelve a lanzar la búsqueda de la mejor traducción usando el modelo log-lineal, ciñiéndose únicamente a las N frases de la lista de las N-best.

2.1.6. Aplicacion al reconocimiento automático del habla

El objetivo del reconocimiento automático del habla (RAH) es transcribir una secuencia de sonidos en una secuencia de palabras. La forma clásica de abordar el problema es mediante una aproximación estadística [Jel97]. En estos términos se define como un proceso de búsqueda que pretende encontrar una secuencia de palabras $\hat{Y} = y_1, y_2, \ldots, y_{|\hat{Y}|}$, pertenecientes a un cierto vocabulario $\Omega = \{v^1, v^2, \ldots, v^{|\Omega|}\}$, que maximice la probabilidad a posteriori de Y dada una secuencia acústica $X = x_1, x_2, \ldots, x_{|X|}$,

$$\hat{Y} = \arg\max_{Y \in \Omega^+} \; p(Y|X) = \arg\max_{Y \in \Omega^+} \; p(X|Y)p(Y), \tag{2.7}$$

donde $p(X|Y)$ es la probabilidad condicional de observar la secuencia acústica X cuando se pronuncia la secuencia de palabras Y, y $p(Y)$ es la probabilidad a priori de la secuencia de palabras Y. El primer término es calculado mediante los denominados modelos acústicos, que, en este caso, serán Modelos Ocultos de Markov (MOMs), y el segundo término es calculado a través del modelo de lenguaje. Clásicamente, el modelo de lenguaje es un modelo de n-gramas estadísticos, pero en este trabajo se abordará el problema con los MLCs presentados en esta sección.

Volveremos a tener un sistema desacoplado, de manera que primero lanzaremos el reconocedor con un modelo de lenguaje estándar, extrayendo del proceso de reconocimiento en lugar de la mejor Y, las N-best o un grafo de palabras que contenga a las N-best. En un segundo paso se realizará un nuevo proceso de parsing de esa lista de N-best, o del grafo de palabras, para buscar la mejor transcripción, guiando en este segundo paso al proceso de búsqueda mediante un MLC. Para este segundo paso volveremos a utilizar el algoritmo de parsing conexionista de la sección 2.1.4.

Reconocimiento basado en Modelos Ocultos de Markov

Clásicamente el RAH ha sido abordado mediante Modelos Ocultos de Markov (MOMs). La idea es entrenar un modelo para cada uno de los fonemas/sonidos que queremos reconocer, de manera que asociamos como transcripciones fonéticas de cada palabra un conjunto de secuencias de estas unidades básicas. Estas unidades, a su vez, están compuestas por estados, dando lugar a un proceso Markoviano. Los estados tienen transiciones entre ellos y cada transición tiene una determinada probabilidad. Estos estados tienen la característica de que no los podemos observar en la secuencia analizada, sin embargo tienen la capacidad de emitir un cierto símbolo o vector de características (estados diferentes pueden emitir el mismo símbolo, con lo que sería imposible diferenciarlos). Estas emisiones tienen asociada también una cierta probabilidad, que depende de los vectores de características extraídos a partir de la señal acústica. El proceso de reconocimiento se realiza concatenando palabras formadas por modelos de este tipo, y al final del todo recuperar el camino de mayor probabilidad.

En este esquema, lo más extendido es calcular la probabilidad de emisión mediante mixturas de Gaussianas. Sin embargo, en este trabajo también se abordará la posibilidad de tener una red neuronal capaz de calcular las probabilidades de esas emisiones, dando lugar a lo que se conoce como Modelos Ocultos de Markov Híbridos [BW90, MB95, Cas97].

2.1.7. Aplicación a tareas de reconocimiento de escritura

La metodología empleada para tareas de reconocimiento de escritura RES es exactamente la misma que para RAH. Pasaremos directamente a abordar las dos formas de utilizar modelos de lenguaje en este tipo de tareas:

- *Modelos de lenguaje para grafemas*: calculan la probabilidad de la frase, no a partir de su descomposición en palabras, sino, a partir de su descomposición en grafemas. De esta forma reducimos la complejidad de los MLCs, ya que el número de grafemas posibles a codificar en la entrada de las redes es en todos los casos un valor pequeño, de unas pocas decenas, mientras que en el caso de las palabras este número puede llegar fácilmente a miles (o millares). Con este tipo de modelo de lenguaje se podrían abordar tareas con vocabulario abierto. Podrían generarse secuencias de grafemas que formarían palabras que no fueron encontradas en la partición de entrenamiento del corpus, y que por tanto no aparecen en los diccionarios de la tarea.

- *Modelos de lenguaje de palabras*: consiste en utilizar el modelo de lenguaje del mismo modo que en la sección 2.1.6, solo que en este caso el reconocedor en lugar de parametrizar voz, para después decodificarla en una grafo de palabras con las posibles soluciones reconocidas, lo que hará será parametrizar imágenes con la escritura, generando el grafo de palabras del mismo modo, y por tanto evaluando este grafo mediante el MLC de la misma manera que en RAH.

2.2. Etiquetado PoS con modelos conexionistas

En este campo las redes neuronales han sido muy poco exploradas. En los últimos años algunos autores han probado modelos como las máquinas de soporte vectorial (SVMs) [GM03, GM04], redes neuronales recurrentes [POF01], redes neuronales hacia delante [TCP05, VI90, Sch94, Rat96, ABCK02], obteniendo resultados competitivos frente a los MOMs [PM03] o los modelos estadísticos [Bra00].

2.2.1. Modelo probabilista

Las principales aproximaciones para tareas de etiquetado PoS están basadas en modelos estocásticos. Desde este punto de vista, el etiquetado PoS se define como un problema de maximización. Dado $\mathcal{T} = \{t_1, t_2, \ldots, t_{|\mathcal{T}|}\}$ un vocabulario de etiquetas PoS y dado $\Omega = \{v^1, v^2, \ldots, v^{|\Omega|}\}$ el vocabulario de la tarea, el objetivo es encontrar la secuencia de etiquetas $\hat{L} = l_1, l_2, \ldots, l_{|L|}$, con $l_i \in \mathcal{T}$, que maximice la probabilidad asociada con la secuencia de palabras $W = w_1, w_2, \ldots, w_{|W|}$, con $w_i \in \Omega$:

$$\hat{L} = \arg\max_{L} \; p(L|W). \tag{2.8}$$

Usando el Teorema de Bayes, la expresión (2.8) se convierte en la expresión (2.9):

$$\hat{L} = \arg\max_{L} \; \frac{p(W|L)p(L)}{p(W)}. \tag{2.9}$$

Dado que la probabilidad $p(W)$ es una constante, puede ser ignorada en el proceso de maximización, reduciendo el problema a:

$$\hat{L} = \arg\max_{L} \; p(W|L)p(L). \tag{2.10}$$

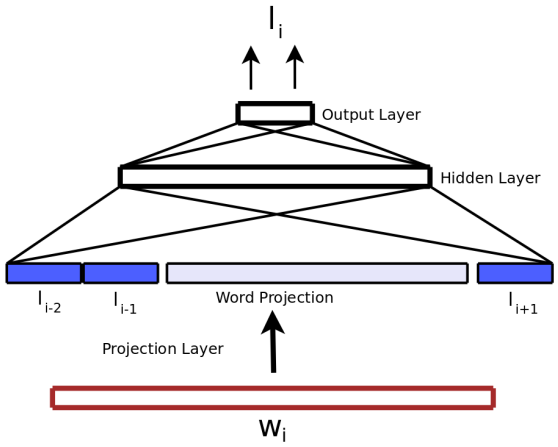

Figura 2.3: *Topología de red neuronal para etiquetado PoS.*

El cálculo de estos parámetros precisa de una gran cantidad de cálculo, con lo que algunas asunciones son necesarias para simplificarlos. En este caso, se asume que las palabras son independientes unas respecto a otras, y una palabra sólo depende de su etiqueta propia. De esta manera obtenemos las probabilidades *léxicas* del modelo:

$$p(W|L) \approx \prod_{i=1}^{n} p(w_i|l_i). \tag{2.11}$$

Ahora cabe establecer la probabilidad de una etiqueta como dependiente de sus etiquetas predecesoras. Considerando 2-gramas, la probabilidad *contextual* del modelo queda:

$$p(L) \approx \prod_{i=1}^{n} P(l_i|l_{i-1}), \tag{2.12}$$

si son 3-gramas:

$$P(L) \approx \prod_{i=1}^{n} P(l_i|l_{i-1}, l_{i-2}), \tag{2.13}$$

Con estas premisas, un modelo probabilísta típico, siguiendo las expresiones (2.10), (2.11) y (2.12), quedaría como:

$$\hat{L} = \arg\max_{L} \; P(L|W) \approx \arg\max_{L} \; \prod_{i=1}^{n} P(w_i|l_i)P(l_i|l_{i-1}), \tag{2.14}$$

considerando que $P(l_1|l_0) = 1$.

Este modelo tiene ciertas limitaciones. Una de ellas es la incapacidad de calcular relaciones a larga distancia. Otra que la información contextual sólo incorpora información del contexto izquierdo (pasado), desechando la información que aporta el contexto derecho (futuro).

2.2.2. Modelo conexionista

En este modelo la información léxica y la contextual es utilizada para predecir qué etiqueta PoS le corresponde a cada palabra ambigua. La principal diferencia entre este modelo y el probabilista es la introducción del contexto derecho (futuro) en el cálculo realizado por el modelo, y la

capacidad de las redes para ampliar el tamaño del contexto (pudiendo así capturar información a larga distancia) perdiendo poca precisión. Por lo tanto, la entrada de la red neuronal consiste en el conjunto de etiquetas que acompañan a la palabra que se quiere desambiguar, además de la ella misma:

$$f(w_i, context) = l_i, \tag{2.15}$$

con "context" igual a la secuencia de etiquetas $l_{i-|p|}, l_{i-|p|+1}, \ldots l_{i-1}, l_{i+1}, \ldots l_{i+|f|-1} l_{i+|f|}$, donde $|p|$ es la longitud del contexto pasado y $|f|$ la longitud del contexto futuro. La palabra ambigua w_i se codifica de forma local, i.e. la unidad que representa la palabra es activada mientras el resto no. Los pesos de la red actúan como parámetros de la función f.

En una primera aproximación se puede utilizar una red con una sola capa oculta, *pero* le añadiremos una nueva capa de neuronas de tal forma que se conecte únicamente con la codificación local de la palabra w_i, y después con la siguiente capa oculta. Esto servirá para calcular una codificación distribuida de la palabra w_i (véase figura 2.3), siguiendo de nuevo las ideas de [BDV00, BB00].

2.2.3. Generación de grafos de etiquetas para procesar frases de test

Para poder evaluar los modelos presentados de una forma correcta, se ha desarrollado un algoritmo basado en Viterbi, para grafos acíclicos dirigidos (GADs), que además del modelo etiquetador, también utiliza un modelo de lenguaje de etiquetas. Este modelo de lenguaje puede ser tanto conexionista como estadístico.

El algoritmo consta de dos pasos:

1. Primero la generación de un grafo de etiquetas a partir de la secuencia de palabras W, una tabla donde buscar las posibles etiquetas válidas para cada palabra (T), un modelo entrenado para etiquetar las palabras a partir de su contexto m_{PoS} (serviría cualquier tipo de clasificador, en este caso usaremos redes neuronales), y el tamaño de los contextos del modelo anterior ($|p|$ para el pasado y $|f|$ para el futuro). Con toda esta información se genera un grafo de etiquetas con probabilidades y etiquetas en los arcos. Se puede ver un ejemplo de este tipo de grafo en la figura 2.4. Las probabilidades de los arcos del grafo de etiquetas se calculan mediante la ecuación 2.15.

2. Una vez generado el grafo, se utiliza un modelo de lenguaje de etiquetas m_{lm} que, mediante el algoritmo de parsing conexionista de la sección 2.1.4, extrae del grafo la secuencia de etiquetas más probable, combinando la probabilidad de cada arco con la probabilidad calculada por el modelo m_{lm}.

Como salida, el algoritmo nos da la secuencia de etiquetas $\hat{L} = l_1, l_2, \ldots, l_{|\hat{L}|}$ que es el camino de mayor probabilidad en el grafo de etiquetas.

Generación del grafo de etiquetas

Lo primero que genera el algoritmo es un nodo vacío inicial, que con transiciones λ nos lleva a los verdaderos estados iniciales del grafo. De esa manera el grafo tiene un único nodo inicial. Cada uno de estos estados iniciales se genera de la siguiente manera:

- Se extraen las posibles etiquetas para las $|f|$ primeras palabras de la secuencia W: $T[w_1], T[w_2], \ldots, T[w_{|f|}]$.
- Se calcula el producto cartesiano de estas etiquetas: $iniciales = \{T[w_1] \times T[w_2] \times \ldots \times T[w_{|f|}]\}$.

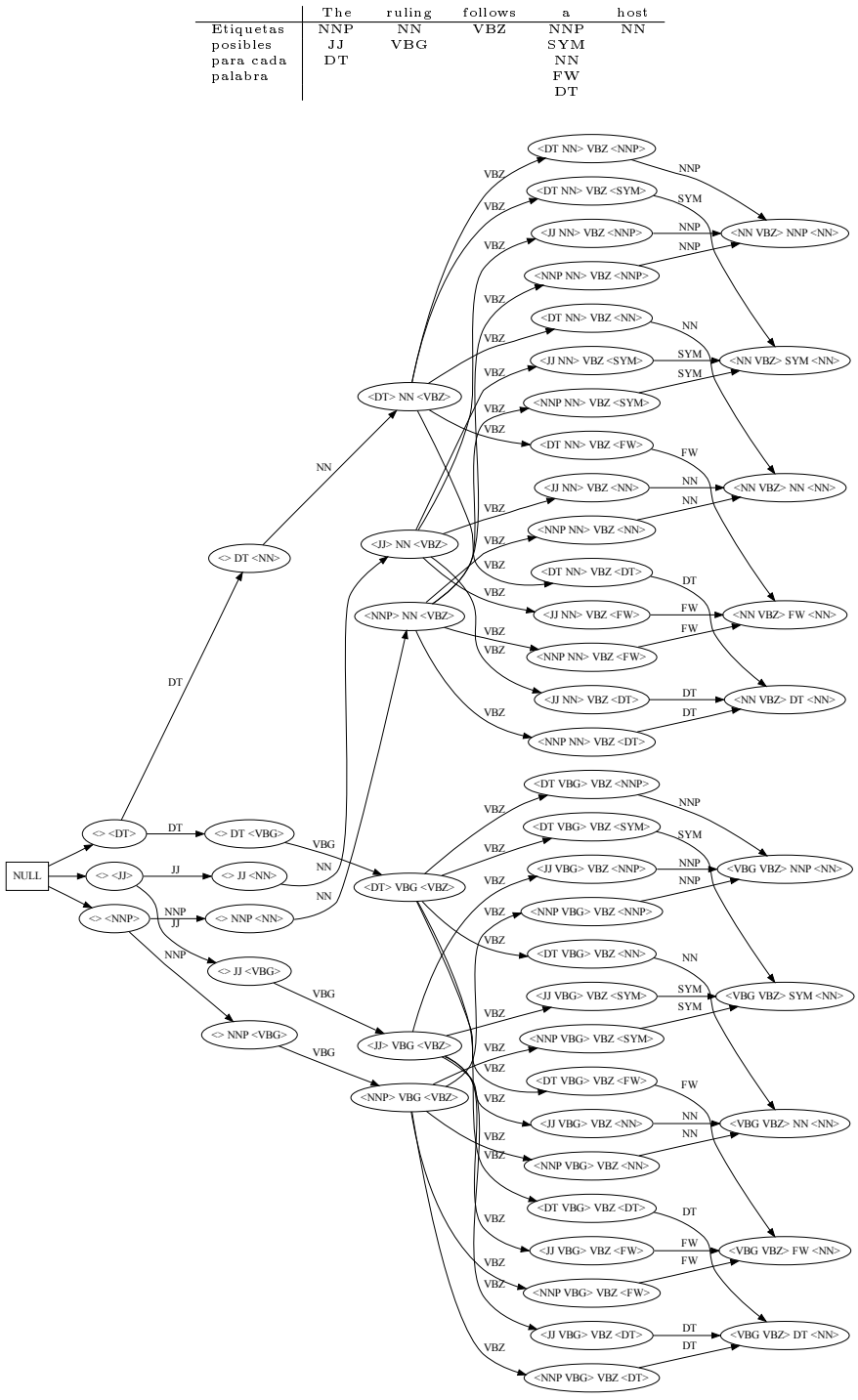

Figura 2.4: *Fragmento parcial de un grafo generado para etiquetar "The ruling follows a host ..." usando un modelo con* 2 *etiquetas de pasado y* 1 *de futuro, organizado por etapas, de manera que tiene* 6 *etapas,* 1 *por cada palabra a etiquetar más otra etapa adicional que hace de nodo inicial del grafo. La etiqueta de los nodos tiene este formato:* $< l_{i-2} l_{i-1} > l_i < l_{i+1} >$. *Se han omitido las probabilides de los arcos.*

- Se genera un nodo por cada elemento del conjunto *iniciales*. Cada nodo se etiqueta con el "context cue" como contexto pasado. La etiqueta de la palabra actual también será el "context cue". El contexto futuro contendrá las etiquetas $l_1, l_2, \ldots, l_{|f|}$.

Ahora se entra en un bucle que itera sobre $i = 1, \ldots, |W|$. En cada iteración procesa todos los nodos que se generaron en la iteración anterior, y genera los nodos que se procesarán en la iteración siguiente:

- Se extrae un nodo e mientras la lista de la iteración anterior no esté vacía.
- Se desplazan las etiquetas asociadas al nodo e una posición a la izquierda, de manera que la etiqueta del extremo se pierde, dejando un lugar vacío en el extremo derecho.
- Se extraen las $|T[w_{|f|+i}]|$ posibles etiquetas futuras para la posición $l_{i+|f|}$, que es la que ha quedado libre.
- Por cada etiqueta futura de la lista $T[w_{|f|+i}]$ se genera de nuevo la cadena de etiquetas que identifica a un nodo.
- Se busca esta cadena en la lista de estados para la siguiente iteración, y si no existe, se inserta un nodo nuevo con esta cadena como identificador.
- Se crea una arista entre el nodo e y los $|T[w_{|f|+i}]|$ que le siguen. Los arcos tendrán la etiqueta l_i y la probabilidad p, que será la componente correspondiente a la etiqueta l_i del vector de probabilidades calculado con m_{PoS}:

  $p = m_{PoS}(w_i, \text{context})[l_i]$

Cuando termine el bucle, tendremos generado el grafo de etiquetas que queríamos, con probabilidades relacionadas con el etiquetado de cada palabra en cada una de las transiciones del GAD. Estará todo listo para utilizar el parser conexionista junto con un MLC de etiquetas y extraer el camino de mayor probabilidad. Este camino será la secuencia de etiquetas $\hat{L}$ que buscábamos.

Capítulo 3

Experimentos

En este capítulo se van a presentar todos los experimentos realizados con MLCs y con modelos conexionistas para etiquetas PoS. Para cada tarea se presentará primero el corpus, después la experimentación necesaria para entrenar los modelos, y por último los resultados de evaluar los modelos para diferentes configuraciones de los mismos.

3.1. Herramientas utilizadas

En el desarrollo de los experimentos se han utilizado 4 herramientas:

- `April` [EZCG07]: herramienta que estamos desarrollando en el grupo de investigación para realizar tareas de reconocimiento de formas y entrenamiento de redes neuronales. Posee una gran cantidad de utilidades para trabajar con conjuntos de datos, procesado de imágenes [ZEC07], preproceso y extracción de características de texto manuscrito [GEZC08], preproceso y parametrización de voz, entrenamiento de MOMs Híbridos, diversos algoritmos de parsing, entre ellos diversas versiones de Viterbi especializadas en léxicos en forma de árbol [ECZG07], parsers de n-gramas estadísticos y conexionistas y una implementación del CYK [CS70, You67, Kas65] basada en grafos palabras [BMM07] entre otras. Implementa poda basada en histograma para acelerar la búsqueda con los algoritmos de Viterbi.

  Todos los experimentos con redes neuronales (MLCs, MOMs Híbridos, etiquetado PoS,...) han sido realizados con esta herramienta. También ha sido utilizada para realizar la reestimación de los grafos de palabras generados a partir de MOMs clásicos, mediante el algoritmo de parsing conexionista de la sección 2.1.4.

- `SRILM` [Sto02]: ésta es una herramienta para trabajar con n-gramas estadísticos. Permite aplicar una gran cantidad de suavizados distintos. En estos experimentos se utilizará en todos los casos la opción de vocabulario cerrado.

- `HTK toolkit` [WLO+95, WY93]: herramienta para el entrenamiento de MOMs. Permite entrenar modelos con cualquier tipo de topología, y con gaussianas para calcular las probabilidades de las emisiones de cada estado de los modelos. Permite tanto reconocer y obtener la mejor transcripción como extraer del proceso de reconocimiento una lista de N-best o un grafo de palabras que contenga a dicha lista. Será utilizada para entrenar MOMs y generar grafos de palabras que puedan ser reestimados más tarde por otras herramientas.

- `Marie` [CMdG05]: herramienta de traducción automática basada en n-gramas y que permite utilizar como función objetivo una combinación log-lineal de características. Ha sido

desarrollada en la Universidad Politécnica de Cataluña. Será la herramienta utilizada para las tareas de traducción. Implementa tanto búsqueda en haz como poda basada en histograma para acelerar el proceso de búsqueda.

3.2. Tareas de traducción automática

En esta línea se han publicado dos trabajos [KFZ+08a, KFZ+08b], uno en el LREC del 2008 y otro en el ICTAI del mismo año. Ambos trabajos han sido realizados en colaboración con la Universidad Politècnica de Catalunya (UPC). Presentaremos en esta sección el trabajo del LREC, por ser el más completo de los dos debido al tamaño del corpus y del vocabulario.

3.2.1. Corpus

Los experimentos se han realizado sobre una extracción de 30 000 frases del corpus NIST'07, de traducción del Árabe al Inglés. Se proporcionan particiones de validación y de test, extraídas de 4 traducciones de referencia, que poseen 489 y 500 frases cada una. El MLC ha sido entrenado con las 30 000 frases de entrenamiento del corpus, y se ha utilizado el corpus de validación para evitar el sobreentrenamiento y comparar los modelos de lenguaje. En la tabla 3.1 se pueden ver algunas características del corpus.

3.2.2. Experimentos con MLCs

Se ha realizado un evaluación exhaustiva de diversas configuraciones de MLCs. La función de activación utilizada para todas las capas ha sido la *tangente hiperbólica*, salvo para la capa de salida que ha estado formada por unidades con función de activación *softmax*. La mejor configuración que se ha encontrado ha sido con una capa de proyección con 32 unidades por palabra, es decir $P = (n-1)32$, siendo n el valor del n-grama, y 64 unidades ocultas. El factor de aprendizaje ha sido fijado a 0.001 y el *momentum* a 0.002. Se ha fijado un factor de caída para estos parámetros de manera que en cada iteración se multiplica su valor por 0.999. También se ha usado el parámetro *weight decay* con un valor de 10^{-7}, y un factor de caída de 0.99. Se han utilizado dos tallas de vocabulario. La primera aquella que incluye las palabras que tienen una frecuencia de aparición mayor o igual que $k = 12$, obteniendo $|\Omega| = 4\,398$ palabras. Y la segunda para $k = 10$, con $|\Omega| = 4\,908$.

El entrenamiento se ha realizado con la herramienta `April`, utlizando el mismo corpus que se utilizó para entrenar los modelos de lenguaje del baseline del traductor. Se han generado MLCs para 3-gramas, 4-gramas y después se han combinado en una mixtura estos dos modelos (3- 4-gramas).

Los coeficientes de la mixtura han sido ajustados utilizando la partición de validación, siguiendo la ecuación 3.1 para combinarlos. El mejor resultado se obtuvo con $\alpha = 0.5$, lo que quiere decir que ambos modelos son combinados de manera equitativa.

$$p(w_1 \dots w_{|W|}) \approx \prod_{i=1}^{|W|} \alpha\, p(w_i|w_{i-2}w_{i-1}) + (1-\alpha)\, p(w_i|w_{i-3}w_{i-2}w_{i-1}) \tag{3.1}$$

3.2.3. Experimentos con modelos de TAE

Como modelo de TAE se ha utilizado el sistema basado en n-gramas [CMdG04, CMdG05] que utilizan en la UPC. Este modelo sigue una aproximación que combina de manera log-lineal todas las características extraídas del proceso de traducción. Para realizar esta combinación se

	Árabe		Inglés	
No. de frases	30.00	K	30.00	K
No. de palabras	839.59	K	936.39	K
Longitud media de frase	27.99		31.22	
$\|\Omega\|$	43.39	K	33.07	K

Tabla 3.1: *Estadísticas de la partición de entrenamiento del corpus NIST'08 (después de extraer* 30 000 *frases al azar).*

	3-gram		4-gram		3-4-grams mixture	
k	10	12	10	12	10	12
SRI LM	106.55	103.17	111.49	108.19	–	–
NN LM	97.75	93.14	98.11	92.32	94.89	89.73

Tabla 3.2: *Comparativa de resultados de perplejidad con el corpus de validación para diferentes modelos de lenguaje*

utiliza la aplicación `Marie` [CMdG05], que implementa el algoritmo de búsqueda. Esta parte de la experimentación fue realizada en la UPC, los detalles del sistema de traducción están publicados en [KFZ+08b, KFZ+08a].

3.2.4. Evaluación de los modelos

Se han generado listas de 1 000-best por cada frase a traducir, de manera que el proceso de evaluación ha consistido en reestimar con todas las configuraciones de la experimentación cada una de las frases de las listas de 1 000-best, añadiendo a la combinación log-lineal esta nueva característica. Para evitar problemas de optimización, se han utilizado diferentes puntos iniciales en la combinación log-lineal, y finalmente se han elegido los mejores pesos con respecto al criterio $100 \times BLEU + 4 \times NIST$.

3.2.5. Resultados

La tabla 3.2 muestra resultados de perplejidad con la partición de validación. Se compara entre la perplejidad de modelos entrenados con el SRILM [Sto02] y la perplejidad de los MLCs, utilizando las mismas particiones para entrenar los modelos y evaluarlos. En la tabla 3.3 se pueden encontrar los resultados en BLEU, NIST y METEOR[1], tanto del sistema baseline de traducción, como del sistema que incluye los MLCs entrenados para la tarea.

Se puede observar que para la mayoría de las configuraciones, los MLCs obtienen un BLEU mayor que para el baseline. El valor que debe superar el BLEU para que el resultado sea considerado estadísticamente significativo[2] es 27.40 para la partición de validación y 22.05 para la de test. Incorporar los MLCs en el proceso de traducción ha conseguido una mejora de 0.5 puntos de BLEU, tanto para la partición de validación como la de test. El resultado ha sido estadísticamente significativo para 2 de las 6 configuraciones utilizadas.

[1] BLEU es una medida que mide la calidad de la traducción a partir de la distancia a un conjunto de referencia usando un n-grama [PRWZ02]. NIST es una medida basada en el BLEU, pero ponderando el valor de los n-gramas [Dod02]. METEOR es una medida que está infravalorada, y se calcula como la media ponderada de la precisión y la cobretura (recall), considerando la correspondencia entre lexemas y sinónimos [BL05].

[2] Se ha utilizado el método de *bootstrap resampling* tal y como se describe en [Koe04] para una confianza del 95 % y un valor de 1 000 para el muestreo.

	Validación			*Test*		
	BLEU	NIST	METEOR	BLEU	NIST	METEOR
baseline	27.00	7.29	53.83	21.77	6.65	49.89
			k=10			
3-grama	27.43	7.27	53.82	21.91	6.61	49.62
4-grama	**27.54**	7.36	54.21	**22.26**	6.72	50.18
Mixutra 3- 4-gramas	27.34	7.14	53.26	21.54	6.49	49.24
			k=12			
3-grama	27.37	7.32	54.10	21.73	6.63	49.68
4-grama	27.47	7.31	53.98	21.91	6.65	49.82
Mixutra 3- 4-gramas	27.47	7.35	54.33	22.07	6.71	50.20

Tabla 3.3: *Resultados comparativos para la tarea de traducción Árabe-Inglés*

3.3. Tareas de etiquetado PoS

Esta línea de investigación es muy interesante debido a lo poco explorada que está mediante técnicas conexionistas. Sin embargo los resultados logran ser competitivos, y en algunos casos superan el estado del arte. Aquí abordaremos el problema con los métodos explicados en el capítulo anterior. Para ello necesitamos extraer del corpus una lista que nos permita saber, a cada palabra, qué etiquetas le son asignables. Las palabras con una única etiqueta serán consideradas como no ambiguas. Las que tengan más de una etiqueta serán las palabras ambiguas. Esta lista es la que se utilzará con el algoritmo que genera el GAD de cara a la evaluación del sistema.

3.3.1. Corpus

El corpus elegido es la parte del Wall Street Journal [PB92] (WSJ) que fue procesada en el Penn TreeBank Project. Este corpus consiste en un conjunto de textos en inglés extraídos del Wall Street Journal, y distribuidos en 25 directorios con una gran cantidad de frases. El número total de palabras ronda el millón, siendo el vocabulario de una talla de 39 000 palabras. El corpus entero fue etiquetado con etiquetas PoS y también analizado sintácticamente. El total de etiquetas PoS es de 45, a las que añadimos una nueva etiqueta que servirá de context cue inicial (el context cue final es el ".", que ya está como etiqueta y sólo se utiliza como punto final en *todas* las frases del corpus). Las principales características de las particiones elegidas como entrenamiento, validación y test están en la tabla 3.4

Considerando únicamente las palabras ambiguas del conjunto de entrenamiento, obtenemos un vocabulario de más de seis mil palabras. Vamos a reducir ese vocabulario cogiendo únicamente aquellas palabras con una frecuencia de aparición mayor o igual a 10 en la partición de entrenamiento. Si una palabra está por debajo de este umbral, es considerada una palabra desconocida. También se ha reducido el vocabulario de palabras ambiguas realizando un proceso de limpieza de las etiquetas asociadas a cada palabra. Si una etiqueta posible para una palabra tiene una frecuencia de aparición de menos del 90 % que la etiqueta más frecuente de esa misma palabra, y además en valor absoluto aparece menos de 3 veces, esa etiqueta es eliminada de la lista de esa palabra. Después de aplicar este proceso, el número de palabras ambiguas se ha reducido a 2 564, incluyendo una palabra especial para la desconocida. El número de palabras que se han convertido en palabras desconocidas es de 2 826 al aplicar el corte de las palabras que aparecen menos de 10 veces en el corpus de entrenamiento.

Partición	**Dir.**	**No. of frases**	**No. de tokens**	$\|\Omega\|$
Entrenamiento	00-18	38 219	912 344	34 064
Validación	19-21	5 527	131 768	12 389
Test	22-24	5 462	129 654	11 548
Total	00-24	49 208	1 173 766	38 452

Tabla 3.4: *Particiones del Penn TreeBank utilizadas para entrenamiento, validación y test. El número de tokens es el número total de palabras en cada partición.*

	$\|f\|$					
	0		1		2	
$\|p\|$	*Unknown*	*KnownAmb*	*Unknown*	*KnownAmb*	*Unknown*	*KnownAmb*
0	74.86 %	15.93 %	61.14 %	10.03 %	69.14 %	13.91 %
1	49.71 %	9.41 %	37.88 %	6.00 %	48.78 %	8.17 %
2	49.49 %	9.04 %	**36.89** %	**5.74** %	44.64 %	7.68 %
3	51.57 %	9.32 %	37.19 %	5.85 %	45.89 %	8.01 %
4	50.23 %	9.45 %	38.29 %	6.14 %	46.55 %	8.35 %

Tabla 3.5: *Resultados de clasificación con la partición de valicación para el barrido de contextos. Estos resultados acotan de inferiormente el error que podemos llegar a conseguir con el método de evaluación basado en GADs.* Unknown *es el error de etiquetar las palabras desconocidas, y* KnownAmb *es el error de etiquetado en las palabras ambiguas conocidas.*

3.3.2. Entrenamiento de los modelos conexionistas de etiquetado

El primer paso para montar el sistema de etiquetado PoS con modelos conexionistas, es entrenar las redes neuronales necesarias para etiquetar las palabras. Se utilizan los modelos explicados en la sección 2.2.2. Para entrenarlos se asume que el modelo es un clasificador, de manera que no se pretende entrenar modelos capaces de etiquetar una frase, sino entrenar modelos que, dada una palabra y las etiquetas de su contexto, nos clasifique la palabra en alguna de las t_i del vocabulario de etiquetas $\mathcal{T}$. La evaluación del modelo se lleva a cabo siguiendo el mismo razonamiento, así pues el resultado de evaluación es más bien un resultado de aciertos/errores de clasificación, con lo que este resultado no es una buena medida para comparar esta técnica con otras. Por ello se ha desarrollado el algoritmo basado en GADs de la sección 2.2.3 que sí está preparado para etiquetar las frases de forma correcta, utilizando para ello estos clasificadores y un modelo de lenguaje.

Se ha realizado un entrenamiento con validación para evitar el sobreentrenamiento. El modelo etiquetador inicial tiene una topología con 80 unidades de capa oculta y 64 de proyección para la palabra de entrada. Se ha hecho un barrido de contexto pasado y futuro para buscar el mejor etiquetador. La entrada de la red dependerá del contexto, con lo que será $46|p| + |\Omega| + 46|f|$, siendo $|\Omega| = 2\,564$, que es el número de palabras ambiguas. La salida de la red tiene una unidad por cada etiqueta, con lo que hay 46 unidades con función de activación *softmax*. Para las unidades del resto de capas gastaremos la función de activación *tangente hiperbólica*. El factor de aprendizaje ha sido fijado a 0.005 y el *momentum* a 0.001. Se ha fijado un factor de caída para estos parámetros de manera que en cada iteración se multiplica su valor por 0.99. También se ha usado el parámetro *weight decay* con un valor de 10^{-7}, y un factor de caída de 0.99. En la tabla 3.5 se puede ver el resultado de clasificación con la partición de validación para el barrido de contextos. El mejor modelo se ha encontrado para $|p| = 2$ y $|f| = 1$, con lo que a partir de ahora éste será el contexto que usaremos en todos los experimentos. A este modelo lo denominaremos RN_1.

Tras el barrido de contextos se ha decidido entrenar otro modelo etiquetador con una

n-grama	Perplejidad
2	10.42
3	9.26
4	8.00

Tabla 3.6: *Perplejidad de los MLCs de etiquetas para la tarea de etiquetad PoS.*

topología un poco diferente para la configuración de $|p| = 2$ y $|f| = 1$. El modelo tiene 100 unidades en la capa oculta y 128 unidades en la capa de proyección de la palabra a etiquetar. El resto de parámetros se mantiene igual que antes. Este modelo obtiene un error de clasificación con $Unknown = 38.05\,\%$ y con $KnownAmb = 5.73\,\%$. A este modelo lo denominaremos RN_2.

3.3.3. Entrenamiento de los MLCs

No sería necesario entrenar un modelo de lenguaje para lanzar el etiquetado una vez se ha generado el grafo de etiquetas con el modelo etiquetador de la sección anterior. Bastaría con buscar con Viterbi el mejor camino en el grafo. Pero, como ocurre en la gran mayoría de tareas donde se trabaja con secuencias, el modelo de lenguaje puede ayudar mucho al proceso de búsqueda para que encuentre una solución mejor. Así que, para ello, entrenamos MLCs de etiquetas a partir del etiquetado de las frases de entrenamiento.

El esquema de entrenamiento sigue siendo con validación para evitar sobreentrenar, y las redes tienen la siguiente topología, $(n-1)46$ unidades de entrada, donde n es el valor del n-grama, 80 unidades en la capa oculta de tipo *tangente hiperbólica* y 46 unidades en la capa de salida de tipo *softmax*. En estos MLCs no necesitamos capa de proyección ya que el tamaño del vocabulario es muy pequeño. El factor de aprendizaje ha sido fijado a 0.001 y el *momentum* a 0.002. Se ha fijado un factor de caída para estos parámetros de manera que en cada iteración se multiplica su valor por 0.999. También se ha usado el parámetro *weight decay* con un valor de 10^{-7}, y un factor de caída de 0.99. Las redes convergen en muy pocas épocas. Se han entrenado MLCs para 2-gramas, 3-gramas y 4-gramas. En la tabla 3.6 se puede ver el resultado de perplejidad de estos modelos con la partición de validación.

3.3.4. Evaluación de los modelos

Para evaluar los modelos se ha utilizado el algoritmo basado en GADs que se explicó en la sección 2.2.3. El tamaño de los contextos probados para la evaluación ha sido de $|p| = 2$ y $|f| = 1$, por ser los que mejor resultado de clasificación obtenían, y se ha hecho un barrido con cada uno de los n-gramas entrenados. Se han probado las dos topologías de modelos etiquetadores entrenadas, RN_1 y RN_2, y también una mixtura de RN_1 y RN_2, en donde el coeficiente de mezcla de ambos modelos es de 0.5 para cada uno. A esta mixtura la denominaremos MIX_{RN_1,RN_2}. En la tabla 3.7 se pueden ver los resultados de este barrido con la partición de validación. Cabe destacar que por separado, tanto RN_1 como RN_2 obtienen los mismos resultados, sin embargo al mezclarlos se consigue reducir un poco el error, sobre todo con palabras ambiguas desconocidas.

3.3.5. Resultados

El mejor resultado con validación ha sido obtenido por el modelo MIX_{RN_1,RN_2} usando como modelo de lenguaje el MLC de 4-gramas. Por lo tanto, para comparar esta aproximación con el estado del arte, se ha lanzado el algoritmo de evaluación con la partición de test, obteniendo un resultado de 7.04 % de errores en el etiquetado, como se puede ver en la tabla 3.8. Destacaremos, que aunque los resultados presentados todavía están lejos de llegar al estado del

Modelo	*KnownAmb*	*Unknown*	*Total*
RN_1 + 2-gramas	10.26 %	42.60 %	7.76 %
RN_1 + 3-gramas	9.85 %	39.06 %	7.39 %
RN_1 + 4-gramas	9.79 %	38.02 %	7.32 %
RN_2 + 2-gramas	10.26 %	42.60 %	7.76 %
RN_2 + 3-gramas	9.85 %	39.06 %	7.39 %
RN_2 + 4-gramas	9.79 %	38.02 %	7.32 %
MIX_{RN_1,RN_2} + 4-gramas	9.73 %	37.56 %	**7.27** %

Tabla 3.7: *Resultados de etiquetado PoS con la partición de validación.* KnownAmb *es el error de desambiguación para las palabras ambiguas conocidas.* Unknown *al error de desambiguación de las palabras ambiguas desconocidas.* Total *el error total de desambiguación sobre todo el conjunto de palabras de la partición (ambiguas y no ambiguas).*

Model	*KnownAmb*	*Unknown*	*Total*
SVMs [GM04]	6.09 %	10.99 %	2.84 %
MT [GS07]	-	23.46 %	3.46 %
TnT [Bra00]	7.84 %	14.14 %	3.54 %
NetTagger [Sch94]	-	-	3.78 %
RANN [POF01]	-	-	8.00 %
MIX_{RN_1,RN_2} + 4-gramas	9.71 %	38.06 %	7.04 %

Tabla 3.8: *Resultados con la partición de test. Otras aproximacions: SVMs (Support Vector Machines), MT (Machine Translation techniques), TnT (Statistic Models), RANN (Recurrent Neural Networks).*

arte, queda mucho trabajo por hacer en cuanto al etiquetado de las palabras desconocidas, como por ejemplo agrupar las palabras desconocidas siguiendo reglas morfológicas, algo que otros autores sí realizan. Los resultados presentados, tan sólo superan el resultado del sistema que utiliza redes neuronales recurrentes, que obtiene un 8.00 % de error en el etiquetado.

3.4. Tareas de reconocimiento automático del habla

En esta línea se han realizado diversos experimentos con el corpus Wall Street Journal (WSJ) [PB92]. El problema ha sido abordado desde dos perspectivas. MOMs clásicos entrenados con la herramienta HTK [WLO+95, WY93], y MOMs hibridados con Redes Neuronales Artificiales (RNA) entrenados con la herramienta `April` [EZCG07]. Los modelos han sido entrenados siguiendo la receta de la publicación [Ver06], entrenando únicamente con el corpus acústico WSJ0, y usando los modelos de 8 gaussianas por estado. En ambos casos la evaluación del reconocimiento con los MLCs está basado en el parsing de grafos de palabras, obtenidos a partir de un proceso de reconocimiento inicial. En el caso de los MLCs se utiliza el algoritmo de parsing conexionista de la sección 2.1.4. Para los modelos estadísticos existe una implementación parecida a este algoritmo que también existe como utilidad de la herramienta `April`.

3.4.1. Corpus

De entre las tareas suministradas con el corpus WSJ se ha escogido la de diccionario cerrado de 5 K sin signos de puntuación.

Partición	**No. de frases**	**No. de tokens**	$\|\Omega\|$
Modelos de lenguaje			
Entrenamiento	1 125 442	25 783 882	145 118
Validación	250 000	5 734 939	81 493
Test	250 000	5 720 955	81 543
Acústica			
Entrenamiento	13 268	10 115 429	8 908
Validación	1 000	757 771	3 907
Nov92	330	–	1 271
Test	400	–	357

En reconocimiento: diccionario cerrado de 4 989 palabras (longitud de *wlist5c.nvp* +3)

Tabla 3.9: *Estadísticas de la parte del WSJ usada para modelos de lenguaje y de la parte usada en acústica. En el caso de modelos de lenguaje el número de tokens hace referencia al número de palabras, y en el caso de acústica hace referencia al número de tramas acústicas (vectores de parámetros) de cada partición.*

Corpus para los modelos de lenguaje

Las particiones para el entrenamiento de modelos de lenguaje se han obtenido a partir de los datos suministrados por el propio corpus para entrenar estos modelos. De entre los diccionarios suministrados se ha escogido el `wlist5c.nvp` que es un diccionario cerrado de 4 986 palabras. El corpus para estos modelos está en el directorio `lm_train/np_data`, y ha sido divido en tres particiones aleatorias con las características que aparecen en la tabla 3.9.

Corpus para los modelos acústicos

Se han utlizado las particiones `si_tr_s` y `Nov92` de la receta [Ver06]. La primera de ellas se ha divido en dos porciones aleatorias, y se han generado las particiones de entrenamiento y validación de la parte acústica. La partición `Nov92` ha sido utilizada como conjunto de validación para ajustar los parámetros del reconocedor: Grammar Scale Factor, Beam Search, Histogram Pruning, etc. La partición de test ha sido extraída de un conjunto de 400 frases del directorio `si_dt_05` del WSJ. En concreto se han extraído todas las muestras etiquedas con este formato $XXXaXXXX$, donde X es un número hexadecimal y a es la letra a. Esta partición es completamente independiente y sólo será utilizada para evaluar en última instancia todos los sistemas presentados. En la tabla 3.9 se muestra información sobre estas particiones. Del vocabulario de 357 palabras del corpus de test, 105 están fuera del vocabulario utilizado para el léxico del reconocedor.

Se ha utilizado HTK para parametrizar las grabaciones del WSJ de cada partición. El parametrizador extrae 12 cepstrales más la energía para cada vector de parámetros. También se han añadido las derivadas y las aceleraciones de estos 13 parámetros, haciendo un total de 39 por cada trama. Esta parametrización ha sido utilizada en todos los experimentos.

3.4.2. Entrenamiento de MLCs

Se han entrenado MLCs mediante un esquema de entrenamiento basado en la evaluación de una partición de validación para evitar el sobreentrenamiento. Se han entrenado MLCs para 3-gramas, de manera que cada palabra se ha codificado con 4 989 unidades. Una por cada palabra del vocabulario más otras 3 palabras que se han añadido al vocabulario para poder representar los context cues `<s>` y `</s>`, y también la palabra desconocida `<unk>`. Por lo tanto el modelo tiene $4989 \times 2 = 9\,978$ unidades de entrada. La capa de proyección ha sido de

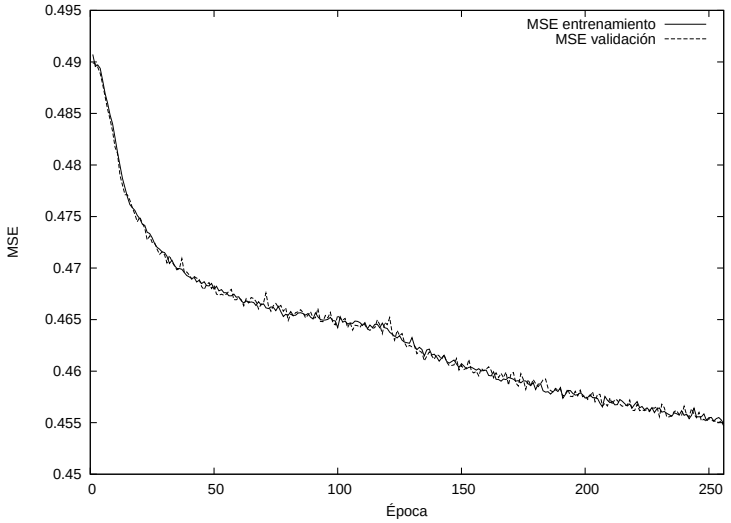

Figura 3.1: *Evolución del MSE durante el entrenamiento del Modelo de Lenguaje Conexionista de 3-gramas con el corpus WSJ.*

$P = 160$ unidades, con lo que cada palabra ha sido proyectada en 80 unidades para obtener su codificación distribuida. La capa oculta elegida ha sido de $H = 128$ unidades. Y la capa de salida de $O = 4\,989 = |\Omega|$ unidades. La función de activación de todas las capas ha sido la *tangente hiperbólica* (tanh), salvo en la última capa donde se ha elegido una activación tipo *softmax*. El factor de aprendizaje ha sido fijado a 0.001 y el *momentum* a 0.002. Se ha fijado un factor de caída para estos parámetros de manera que en cada iteración se multiplica su valor por 0.999. También se ha usado el parámetro *weight decay* con un valor de 10^{-7}, y un factor de caída de 0.99. Dada la gran cantidad de patrones del corpus, el entrenamiento de los modelos se ha realizado con reemplazamiento. Esto quiere decir que en cada época se le ha suministrado a la red neuronal un subconjunto aleatorio de cada partición. En concreto se han usado 1 000 000 de patrones para entrenar cada época, y 300 000 patrones para la validación de cada época.

La mejor red ha sido encontrado en la iteración 254, y el entrenamiento ha tenido el aspecto de la figura 3.1.

Para poder comparar los resultados, hemos tomado el modelo de lenguaje de 2-gramas estadísticos que viene ya entrenado con el corpus WSJ, y además se ha entrenado otro modelo de 3-gramas usando para entrenar el modelo la suma de los corpus de entrenamiento y de validación. Los 3-gramas han sido entrenados usando el SRILM [Sto02] y el suavizado *Good Turing*. En la tabla 3.10 se pueden ver resultados de perplejidad con el corpus de test para comparar todos los modelos de lenguaje utilizados. Se puede observar que los MLCs, a partir de la gráfica y de sus resultados de perplejidad, no están todavía lo suficientemente entrenados, todavía no han dejado de converger. Por lo tanto se espera obtener mejores resultados cuando los modelos sean definitivos.

3.4.3. Experimentos con MOMs

Los MOMs han sido entrenados siguiendo la receta [Ver06], en la cual se definen modelos de Markov de 3 estados para cada fonema, haciendo un total de 41 fonemas. 8 mixturas de

	2-gramas	3-gramas
SRILM	91.07	80.18
MLC	–	107.60

Tabla 3.10: *Perplejidad de los modelos de lenguaje utilizados en los experimentos con la partición de Test.*

Configuración	WER	GSF
Baseline - HTK + 2-grama	22.53 %	15
3-gramas SRI + LM *score* HTK	20.81 %	13
3-gramas MLC + LM *score* HTK	22.92 %	6
3-gramas SRI	19.35 %	31
3-gramas MLC	25.33 %	26

Tabla 3.11: *Mejores resultados de los barridos de GSF con la partición Nov92 con los MOMs clásicos. Se indica el WER obtenido y con que valor de GSF.*

gaussianas por cada estado. En un momento dado del entrenamiento, se ha pasado de modelos incontextuales a modelos contextuales de trifonemas, de manera que cuando termina el entrenamiento hay un total de 19 054 MOMs entrenados para esta tarea, de un total de 65 561 trifonemas (de los cuales muchos están ligados). Se ha utilizado la herramienta HTK, entrenando los modelos con las particiones de entrenamiento y validación. Para ajustar los parámetros del reconocedor hemos escogido la partición `Nov92`.

Una vez han sido entrenados los modelos, se ha usado la herramienta `HVite` para generar grafos de palabras en el formato de HTK. Para ello se ha utilizado el 2-grama estándar que viene junto con el corpus WSJ y que ha sido especificado antes. Este 2-grama ha servido para guiar el reconocedor, y el WER obtenido con la 1-best de este proceso de reconocimiento ha sido escogido como el baseline del resto de la experimentación. Después se ha utilizado la herramienta `April` para hacer una reestimación de las probabilidades del grafo de palabras de HTK con cada uno de los modelos de lenguaje que se quería probar. Destacaremos que, aprovechando que HTK genera grafos y ofrece por separado tanto el *score* acústico como el *score* del modelo de lenguaje utilizado, se han realizado experimentos combinando el *score* del modelo de lenguaje de HTK con los modelos lanzados en `April`, y experimentos sin utilizar este *score*.

Para ajustar el reconocedor se han realizado diversos barridos de Grammar Scale Factor (GSF), con 3-gramas estadísticos, con MLCs, combinando los *scores* de HTK con los de `April` y sin hacer esta combinación. En total se han realizado 4 barridos con la partición `Nov92`, como se ve en las figuras 3.2 y 3.3. Con los mejores resultados de cada barrido se ha generado la tabla 3.11.

En la tabla 3.11 se puede ver una tabla comparando los diferentes resultados obtenidos para cada modelo de lenguaje. Se puede observar que el MLC no ha sido todo lo bueno que se esperaba, aunque tiene cierto sentido, ya que la perplejidad que se ha obtenido no ha sido satisfactoria.

3.4.4. Experimentos con MOMs Híbridos

Se han entrenado MOMs híbridados con redes neuronales hacia-delante utilizando las mismas particiones que para el entrenamiento de MOMs siguiendo la receta [Ver06]. La topología de los modelos ha sido exactamente la misma, y únicamente se han entrenado modelos incontextuales (monofonemas), con lo que en total han sido entrenados 41 modelos, junto con una red neuronal para el cáculo de las probabilidades de emisión de los estados de cada MOM. Para entrenar este sistema se ha utilizado la herramienta `April`. La red neuronal tiene una topología

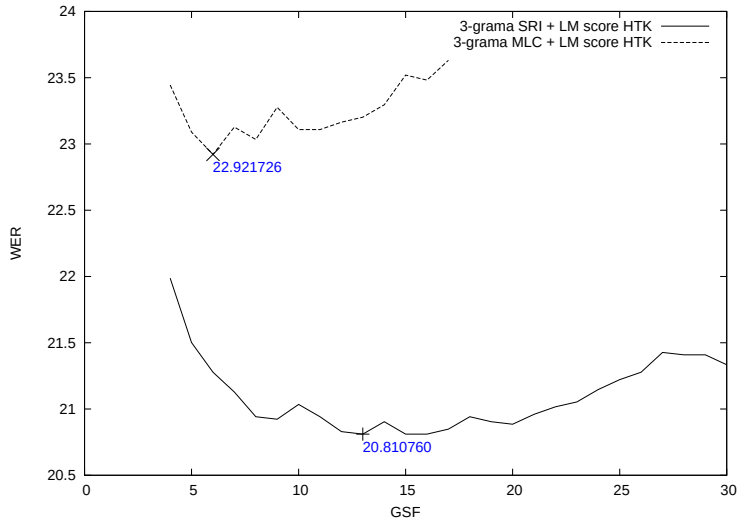

Figura 3.2: *Gráficas del barrido de GSF con la partición Nov92 para ajustar el reconocimiento, combinando el score del modelo de lenguaje usado en HTK con el de* `April`.

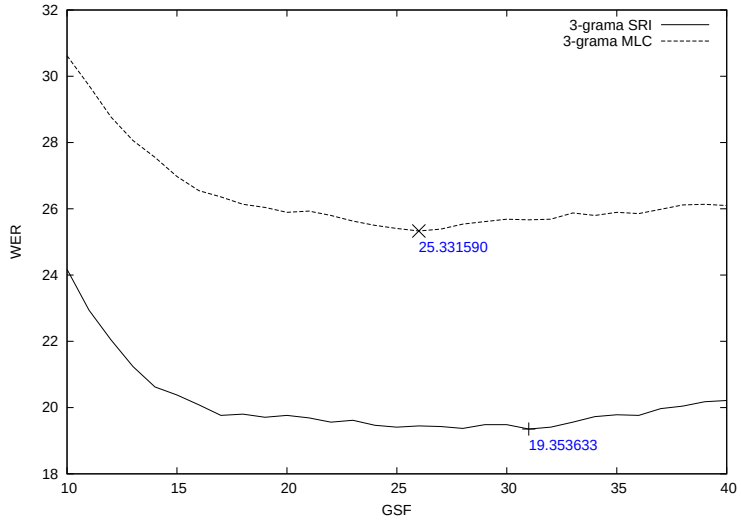

Figura 3.3: *Gráficas del barrido de GSF con la partición Nov92 para ajustar el reconocimiento, utilizando de los grafos de palabras de HTK únicamente el score acústico.*

GSF	4	5	6	7	8
WER	15.75 %	13.99 %	13.66 %	**13.28** %	13.34 %
GSF	9	10	11	12	
WER	14.50 %	15.39 %	16.57 %	17.92 %	

Tabla 3.12: *Barrido GSF de MOMs Híbridos con la partición Nov92.*

	WER		
	2-gramas WSJ	3-gramas SRI	3-gramas MLC + LM *score* HTK
MOMs	22.53 %	19.35 %	22.92 %
MOMs Híbridos	13.28 %	**9.55** %	–

Tabla 3.13: *Resultados con la partición Nov92 de la mejor configuración de cada uno de los modelos.*

capa a capa con 351 entradas, 128 unidades en una primera capa oculta con función de activación *sigmoide*, 128 unidades en una segunda capa oculta con la misma función de activación y 120 unidades de salida con función de activación *softmax*. Cada unidad de salida es una emisión de uno de los MOMs. Las emisiones pueden estar ligadas y, de hecho, los modelos de silencio tienen sus emisiones ligadas.

Se han entrenado los modelos, y después se ha ajustado el GSF mediante un barrido con la paritición `Nov92`, como se puede ver en la tabla 3.12, siendo el mejor GSF= 7. Tras este barrido se han extraído resultados de reconocimiento con 3-gramas de SRI con el corpus `Nov92` para comparar los resultados con los MOMs clásicos.

Con estos modelos no se han podido utilizar MLCs porque `April` sólo admite sistemas de reconocimiento con los MLCs acoplados, lo cual es inviable a día de hoy por el coste computacional de evaluar las redes. Está pendiente como trabajo futuro implementar la posibilidad de generar grafos de palabras como salida del reconocedor, y realizar toda esta experimentación igual para el caso de MOMs clásicos, haciendo reestimación de grafos de palabras.

3.4.5. Resultados

Para finalizar, se ha lanzado el reconocimiento de ambos sistemas con el corpus de test de la tabla 3.9. Ha sido probado el 2-grama estándar que viene junto con el WSJ, además del 3-grama calculado con SRI y el MLC para 3-gramas. Los resultados se han presentado en la tabla 3.14.

Se puede observar que el corpus de test tiene una complejidad mayor, por lo que el WER sube de forma drástica con todos los sistemas. El mejor sistema para todos los experimentos, tanto con la partición `Nov92` como con la partición de test, ha sido el de MOMs híbridados con RNAs, usando como modelo de lenguaje el 3-grama calculado con SRI (una mejora del 50 % con la partición `Nov92`, y del 3 % con la partición de test). En cuanto al resto de configuraciones se produce una discrepancia entre los resultados de test y de la partición `Nov92`. En el caso de

	WER		
	2-gramas WSJ	3-gramas SRI	3-gramas MLC + LM *score* HTK
MOMs	52.52 %	51.36 %	51.74 % %
MOMs Híbridos	52.18 %	**49.77** %	–

Tabla 3.14: *Resultados con la partición de test de la mejor configuración de cada uno de los modelos.*

MOMs clásicos, el paso de 2-gramas a 3-gramas siempre produce una mejora. Pero el paso al MLC de 3-gramas, con la partición `Nov92` no obtiene ninguna mejora, pero con la partición de `test` el MLC consigue superar al baseline de 2-gramas del WSJ. Esto puede ser debido a una mayor robustez de los MLCs, aunque la mejora que se obtiene es muy pequeña. A pesar de todo, el MLC todavía necesita una mayor experimentación dado que los resultados perplejidad del modelo no son buenos. En todos los casos el MLC se ha comportado mejor al combinarlo con el 2-grama del WSJ. Esto dejaría abierta la posibilidad de probar a combinarlo también con el 3-grama de SRI, esperando en este caso obtener un mejor resultado.

Capítulo 4

Conclusiones

En este trabajo ha sido presentada la aplicación del PLN mediante técnicas conexionistas a 3 tareas diferentes. Se han presentado dos modelos conexionistas, ambos basados en las ideas de [BDV00, BB00], y también dos algoritmos, uno de parsing de grafos con Viterbi y modelos conexionistas, y otro de generación de grafos de etiquetas para tareas de etiquetado PoS. Para poder evaluar la calidad de estos modelos se ha experimentado en 3 líneas: traducción automática estadística, etiquetado PoS y reconocimiento automático del habla.

De la tarea de TAE han surgido dos publicaciones [KFZ+08a, KFZ+08b], una de las cuales ha sido presentada en este texto. De ambas publicaciones se puede observar que los MLCs presentados ayudan en una cierta medida a obtener mejores traducciones (alrededor de 0.5 puntos de BLEU).

De la tarea de etiquetado PoS todavía no se ha podido publicar ningún resultado. Sería interesante profundizar la investigación explorando técnicas para reducir el error de etiquetar palabras ambiguas desconocidas. Tras esto el siguiente paso sería publicar los resultados, presentando la idea de generar los grafos de etiquetas utilizando clasificadores. Los resultados que se han obtenido son prometedores, pero no son definitivos.

La tarea de RAH tampoco ha dado lugar a publicaciones todavía. Falta ampliar la experimentación con MLCs, y poder lanzar experimentos de MOMs híbridos utilizando como modelo de lenguaje los MLCs. Tras esta experimentación se pasaría a publicar los resultados. Caben esperanzas de conseguir mejoras con los MLCs, aunque hasta el momento los resultados no son positivos, quizá debido a un insuficiente entrenamiento de los modelos, como se puede ver en la gráfica 3.1.

En conclusión, ésta es una línea de trabajo muy abierta en la que todavía quedan muchas cosas por hacer. Los resultados hasta el momento son esperanzadores. A pesar de los problemas de convergencia de los modelos, parece que se están comportando bien para tamaños grandes de vocabulario. Los resultados obtenidos en TAE prometen conseguir mejoras en las otras tareas dedicando un poco más de esfuerzo a la experimentación.

Capítulo 5

Trabajos futuros

Como trabajo futuro de forma general, queda acelerar el proceso de reconocimiento con MLCs para permitir hacer un sistema acoplado con ellos. En esta línea hay varias ideas que sería interesante implementar. Analizando dónde reside el coste computacional de los MLCs, es fácil darse cuenta de que el cuello de botella es la capa de salida, donde fácilmente se pueden tener varios miles de unidades con función de activación *softmax*. Esto obliga a tener que calcular la activación de *todas* las unidades de salida cada vez que se quiere consultar la probabilidad de unos pocos n-gramas. Esto es así por culpa de la normalización necesaria para la función de activación *softmax*. Se está trabajando en conseguir entrenar redes con una capa de salida con activación *sigmoide*, para la cual no tendríamos este problema, y permitiría acelerar mucho el proceso de reconocimiento acoplado.

5.1. Tarea de etiquetado PoS

En esta tarea los resultados parecen prometedores. A pesar del mal funcionamiento de los modelos con las palabras desconocidas, el sistema está unos pocos puntos por encima del estado del arte. Como trabajo futuro se plantea probar más configuraciones, y más mixturas de diferentes modelos. También se plantea hacer un preproceso mejor del corpus, en el sentido de no tener un único tipo de palabra desconocida. Se podría diferenciar entre diversos tipos de palabras desconocidas a partir de información morfológica de las palabras. Esta aproximación forma parte de los sistemas que son estado del arte. También se podrían probar más topologías para los modelos de etiquetado conexionistas, y combinarlos todos entre sí, como ya se ha hecho con RN_1 y RN_2. De nuevo en la línea de preproceso, se podría categorizar el corpus y añadir, además de información de las etiquetas PoS, información de las categorias de las palabras para entrenar los modelos.

5.2. Tarea de traducción automática estadística

En esta línea de investigación nos queda abierto un nuevo frente que todavía nadie ha atacado con modelos conexionistas, el reordenamiento de las palabras de la frase de entrada. Es sabido por diversos trabajos [SC06, ZNWS04, KVM+05] que reordenar las palabras de la frase origen, para más tarde hacer una traducción monótona, puede ayudar a obtener mejores resultados en el sistema de traducción completo. Habitualmente se utilizan modelos de reordenamiento basados en distancias entre las posiciones originales de las palabras y la posición reordenada. Se nos plantea la posibilidad de generar un grafo con los posibles reordenamientos de las palabras de una frase origen, y buscar mediante un modelo de lenguaje el reordenamiento

de mayor probabilidad. Está búsqueda estaría acoplada al proceso de generación del grafo, permitiendo aplicar técnicas como ramificación y poda, o incluso programación dinámica usando poda basada en histograma para reducir la complejidad del problema. Quedaría como trabajo futuro profundizar en esta línea.

5.3. Tarea de reconocimiento automático del habla

En esta tarea nos quedan diversas líneas abiertas:

- Obtener resultados de MOMs Híbridos utilizando MLCs.
- Entrenar mejor los MLCs, realizar una experimentación más exhaustiva para mejorar el resultado de perplejidad de los modelos, con la intención de mejorar también el WER.
- Entrenar modelos de 4-gramas y 5-gramas, aprovechando la fácil escalabilidad de los MLCs.
- Combinar modelos, tanto MLCs con otros MLCs de topologías diferentes, como estos con modelos de lenguaje estadísticos calculados con SRI.

Bibliografía

[ABCK02] Ahmed, S. Bapi, Pammi Chandrasekhar, and M. Krishna. Application of multilayer perceptron network for tagging parts-of-speech. In *Language Engineering Conference*, page 57, 2002.

[BB00] S. Bengio and Y. Bengio. Taking on the Curse of Dimensionality in Joint Distributions Using Neural Networks. *IEEE Transaction on Neural Networks special issue on data mining and knowledge discover*, pages 550–557, 2000.

[BCP+90] P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, J.D. Lafferty, R. Mercer, and P.S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

[BDV00] Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. A neural probabilistic language model. In *NIPS*, pages 932–938, 2000.

[Ben] Yoshua Bengio. Journal of machine learning research 3 (2003) 1137–1155 submitted 4/02; published 2/03 a neural probabilistic language model.

[Bis95] C. M. Bishop. *Neural networks for pattern recognition.* Oxford University Press, 1995.

[BL05] S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, 2005.

[BMM07] Salvador España Boquera, Jorge Gorbe Moya, and Francisco Zamora Martínez. Semiring Lattice Parsing Applied to CYK. In Joan Martí, José Miguel Benedí, Ana Maria Me ndonça, and Joan Serrat, editors, *Pattern Recognition and Image Analysis*, volume 4477 of *LNCS*, pages 603–610. Springer, June 2007.

[Bra00] T. Brants. TnT: a statistical part-of-speech tagger. In *Proceedings of the 6th conference on Applied Natural Language Processing*, pages 224–231, 2000.

[BW90] H. Bourlard and C.J. Wellekens. Links Between Markov Models and Multilayer Perceptrons. *IEEE Trans on Pattern Analisys and Machine Intelligence*, 12(12):1167–1178, 1990.

[Cas97] M.J. Castro. Reconocimiento del habla mediante redes neuronales. *Inteligencia Artificial, Monografía "Redes Neuronales Artificiales"*, 1:16–21, 1997.

[CG96] 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.

[CMdG04] J. M. Crego, J. Mariño, and A. de Gispert. Finite-state-based and Phrase-based Statistical Machine Translation. In *Proceedings of the Int. Conf. on Spoken Language Processing*, pages 37–40, 2004.

[CMdG05] J. M. Crego, J. Mariño, and A. de Gispert. An Ngram-based Statistical Machine Translation Decoder. In *Proceedings of INTERSPEECH05*, pages 3185–3188, 2005.

[CS70] J. Cocke and J.T. Schwartz. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University, 1970.

[Dod02] G. Doddington. Automatic evaluation of machine translation quality using n-grams co-occurrence statistics. In *In HLT 2002 (Second Conference on Human Language Technology)*, pages 128–132, 2002.

[ECZG07] Salvador España, María José Castro, Francisco Zamora, and Jorge Gorge. Efficient Viterbi algorithms for lexical tree based models. In *An ISCA Tutorial and Research Workshop on Non Lonear Speech Processing*, May 2007.

[EZCG07] Salvador España, Francisco Zamora, María José Castro, and Jorge Gorbe. Efficient BP Algorithms for General Feedforward Neural Networks. In *Bio-inspired Modeling of Cognitive Tasks*, volume 4527, pages 327–336, June 2007.

[GEZC08] J. Gorbe, S. España, F. Zamora, and M. J. Castro. Handwritten Text Normalization by using Local Extrema Classification. In Alfons Juan-Císcar and Gemma Sánchez-Albaladejo, editors, *Pattern Recognition in Information Systems: Proceedings of the 8th International Workshop on Pattern Recognition in Information Systems (PRIS 2008)*, pages 164–172, Barcelona, Spain, June 2008. INSTICC PRESS. ISBN: 978-989-8111-42-5.

[GM03] J. Giménez and L. Márquez. Fast and accurate Part-of-Speech tagging: the SVM approach revisited. In *Proceedings of the 4th RANLP*, 2003.

[GM04] D. Gimenez and L. Marquez. Svmtool: A general pos tagger generator based on support vector machines. In *Proceedings of the Fourth Conference on Language Resources and Evaluation.*, 2004.

[GMA] Roberto Gemello, Franco Mana, and Dario Albesano. Hybrid HMM/Neural Network based Speech Recognition in Loquendo ASR.

[GS07] Guillem Gascó and Joan Andreu Sánchez. Part-of-speech tagging based on machine translation techniques. In *Pattern Recognition and Image Analysis*, volume 4477, pages 257–264, 2007.

[Jel97] F. Jelinek. Statistical methods for speech recognition. *MIT PRess, Cambridge, MA, USA*, 1997.

[Kas65] T. Kasami. An efficient recognition and syntax analysis algorithm for context-free languages. Technical report, Air Force Cambridge Research Laoratory, 1965.

[KFZ+08a] Maxim Khalilov, José A. R. Fonollos, F. Zamora, María J. Castro, and S. España. Neural Network Language Models for Translation with Limited Data. In *Proceedings of the ICTAI*, 2008.

[KFZ+08b] Maxim Khalilov, José A. R. Fonollosa, F. Zamora, María J. Castro, and S. España. Arabic-English translation improvement by target-side neural network language modeling. In *Proceedings of the Workshop HLT & NLP within the Arabic world. (LREC 2008)*, May 2008. ISBN: 2-9517408-4-0.

[Koe04] P. Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP) 2004*, pages 388–395, 2004.

[KVM+05] Stephan Kanthak, David Vilar, Evgeny Matusov, Richard Zens, and Hermann Ney. Novel reordering approaches in phrase-based statistical machine translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, pages 167–174, 2005.

[MB95] N. Morgan and H. Bourlard. Continuous speech recognition: An introduction to the hybrid HMM/connecionist approach. *IEEE Signal Processing Magazine*, 12(3):25–42, 1995.

[MVUG02] M. T. Martín Valdivia, L. A. Ureña, and M. García. Resolución de la ambigüedad mediantes redes neuronales. *Procesamiento del Lenguaje Natural*, 29:39–45, 2002.

[NS89] M.Ñakamura and K. Shikano. A study of English word category prediction based on neural networks. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'89)*, pages 731–734, May 1989.

[PB92] D. Paul and J. Baker. The Design for the Wall Street Journal-based CSR Corpus. In *Proceedings of the DARPA Speech and Natural Language Workshop*, 1992.

[PM03] F. Pla and A. Molina. Improving Part-of-Speech Tagging using Lexicalized HMMs. *Natural Language Engineering*, 2003.

[POF01] J. A. Pérez-Ortiz and M. L. Forcada. Part-of-speech tagging with recurrent neural networks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2001.

[PRWZ02] K. Papineni, S. Roukos, T. Ward, and W. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the ACL 2002*, pages 311–318, 2002.

[Rat96] A. Ratnaparkhi. A Maximum Entropy Part-Of-Speech Tagger. In *1st Conference on Empirical Methods in Natural Language Processing*, 1996.

[RSC07] Brian Roark, Murat Saraclar, and Michael Collins. Discriminative n-gram language modeling. *Comput. Speech Lang.*, 21(2):373–392, 2007.

[SC06] Germán Sanchis and Francisco Casacuberta. N-best reordering in statistical machine translation. In *IV Jornadas en Tecnologia del Habla*, pages 99–104, Zaragoza, Spain, November 2006.

[Sch94] H. Schmid. Part-of-Speech tagging with neural networks. In *Proceedings of COLING-94*, pages 172–176, 1994.

[Sch07] Holger Schwenk. Continuous space language models. *Comput. Speech Lang.*, 21(3):492–518, 2007.

[SCjF06] H. Schwenk, M. R. Costa-jussà, and J. A. R. Fonollosa. Continuous space language models for the IWSLT 2006 task. In *Proceedings of IWSLT 2006*, pages 166–173, 2006.

[SDG06] H. Schwenk, D. Déchelotte, and J. L. Gauvain. Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 723–730, 2006.

[SG05] Holger Schwenk and Jean-Luc Gauvain. Training neural network language models on very large corpora. In *HLT/EMNLP*, 2005.

[SIMY99] M. Shajith Ikbal, Hemant Misra, and B. Yegnanarayana. Analysis of autoassociative mapping neural networks. In *Int. Joint Conf. on Neural Networks*, 1999.

[Sto02] A. Stolcke. SRILM: an extensible language modeling toolkit. In *Proceedings of the Int. Conf. on Spoken Language Processing*, pages 901–904, 2002.

[TC06] Salvador Tortajada and María José Castro. Diferentes aproximaciones a la codificación del vocabulario en modelado de lenguaje conexionista. In *IV Jornadas en Tecnologia del Habla*, pages 59–63, November 2006.

[TCP05] Salvador Tortajada, María-José Castro, and Ferran Pla. Part-of-speech tagging based on artificial neural networks. In *2nd Language & Technology Conference Proceedings*, 2005.

[Ver06] K. Vertanen. HTK Wall Street Journal Training Recipe, 2006. http://www.inference.phy.cam.ac.uk/kv227/htk/.

[VI90] J. Veronis and N. M. Ide. Word Sense Disambiguation with Very Large Neural Networks Extracted from Machine Readable Dictionaries. In *COLING-90*, pages 389–394, 1990.

[WLO+95] P.C. Woodland, C.J. Leggetter, J.J. Odell, V. Valtchev, and S.J. Young. The 1994 HTK large vocabulary speech recognition system. In *Proceedings of the 1995 International Conference onAcoustics, Speech, and Signal Processing*, volume 1, pages 73–76, May 1995.

[WY93] P. Woodland and S. Young. The htk tied-state continuous speech recogniser, 1993.

[XR00] Wei Xu and Alex Rudnicky. Can Artificial Neural Networks Learn Language Models? In *Proceedings of the Int. Conf. on Spoken Language Processing*, 2000.

[You67] D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208, 1967.

[ZEC07] Francisco Zamora, Salvador España, and M.J. Castro. Behaviour-based Clustering of Neural Networks applied to Document Enhancement. In *Computational and Ambient Intelligence*, volume 4507 of *LNCS*, pages 144–151. Springer, 2007. 9th International Work-Conference on Artificial Neural Networks, IWANN 2007, San Sebastián, Spain, June 20-22, 2007. Proceedings.

[ZNWS04] Richard Zens, Hermann Ney, Taro Watanabe, and Eiichiro Sumita. Reordering constraints for phrase-based statistical machine translation. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 205, Morristown, NJ, USA, 2004. Association for Computational Linguistics.