Modelos predictivos basados en redes neuronales recurrentes de tiempo discreto

Juan Antonio Pérez Ortiz

Tesis doctoral dirigida por
Mikel L. Forcada y Jorge Calera Rubio

Esbozo

- Redes neuronales recurrentes
- Problemas resolubles mediante predicción de secuencias:
  - Compresión de secuencias simbólicas
  - Aprendizaje de lenguajes con fuertes dependencias a largo plazo
  - Desambiguación categorial
  - Predicción de señales de voz

Índice

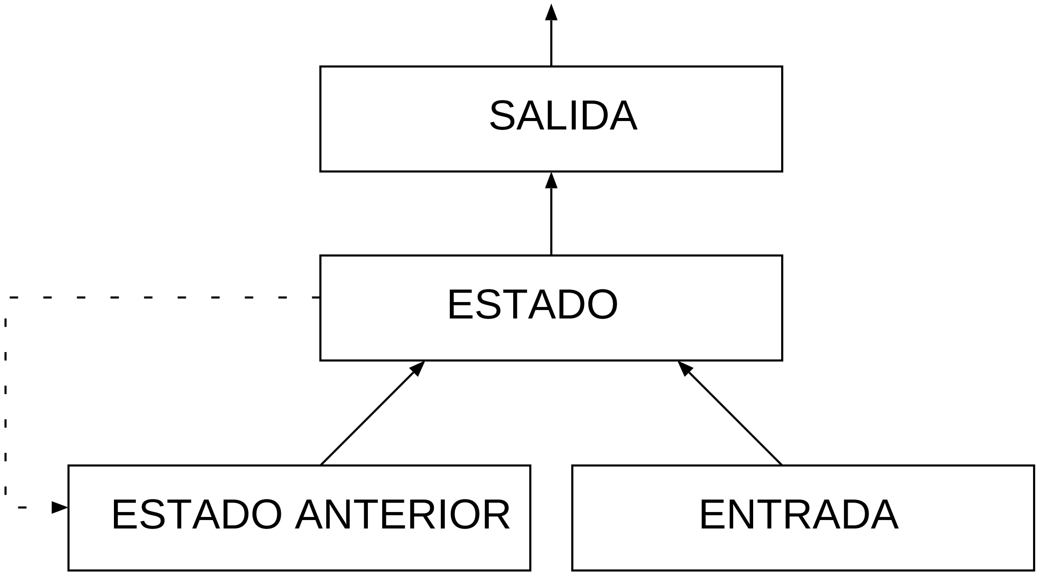
- Redes neuronales recurrentes de tiempo discreto (RNR): concepto, modelos, entrenamiento, predicción
- Problemas estudiados en la tesis:
  - introducción, trabajos previos, método, resultados, discusión y próximos trabajos
- Conclusiones

Redes neuronales recurrentes de tiempo discreto

- Adecuadas para el procesamiento de secuencias temporales, al contrario que las redes hacia adelante
- Representación en forma de estados de información sobre la historia de la secuencia
- Esta tesis estudia problemas abordables a través de la predicción de secuencias

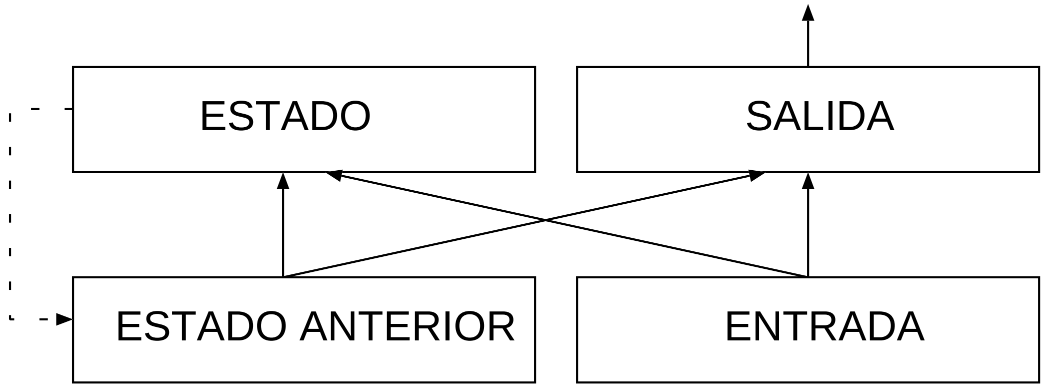
Red recurrente simple

- RRS (Elman, 1990)



Red parcialmente recurrente

- RPR (Robinson y Fallside, 1991)



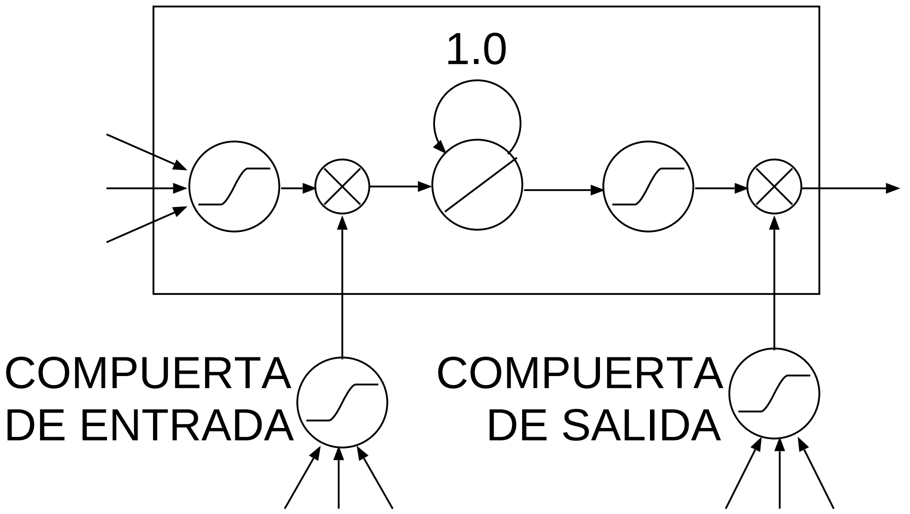
Memoria a corto y largo plazo

- LSTM (Hochreiter y Schmidhuber, 1997)
- Respuesta al problema del gradiente evanescente: la influencia de la señal de error se debilita rápidamente con el tiempo
- LSTM resuelve muchos problemas no resolubles con redes recurrentes tradicionales
- La topología y la forma de calcular las derivadas simplifican el manejo de las dependencias a largo plazo

Celdas de memoria de LSTM

- La celda de memoria es el componente básico de una red LSTM: una unidad lineal autorrecurrente (carrusel) rodeada de una serie de compuertas

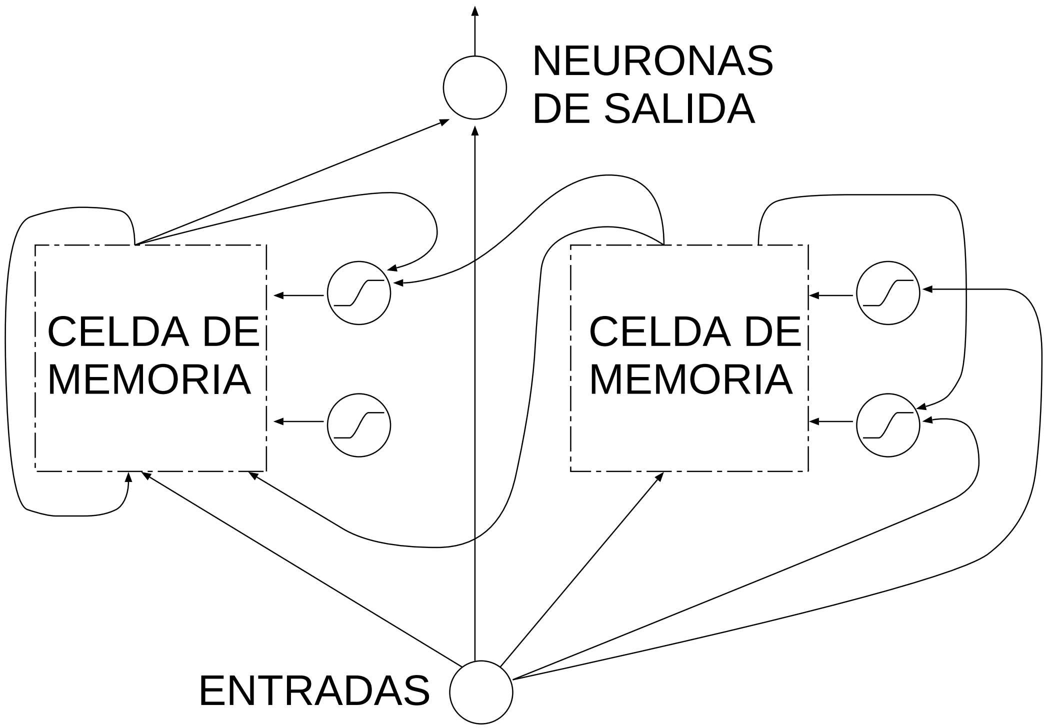
- La unidad lineal puede almacenar información durante largos periodos de tiempo: el problema del gradiente evanescente se desvanece



Red LSTM

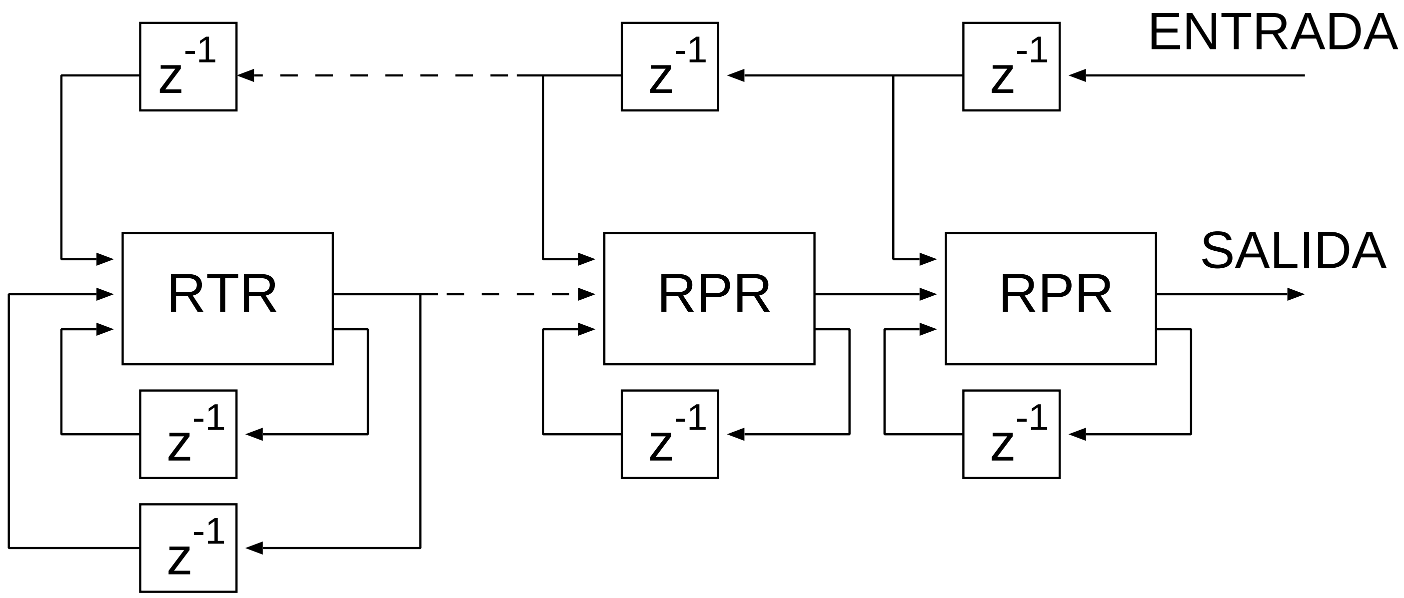
- Las celdas de memoria forman la capa oculta de una red LSTM; las neuronas de salida son convencionales

- El estado de una red LSTM no está acotado



Red neuronal recurrente en cascada

- RNRC (Haykin y Li, 1995)
- Modelo especializado usado en pocas aplicaciones: diseño ad hoc
- Los pesos son compartidos por todos los módulos



Métodos basados en el gradiente

- Derivada de la función de error con respecto a los parámetros ajustables del modelo (entrenamiento supervisado):

$$\frac{\partial E[t]}{\partial \square}$$

- Metodos para calcular el gradiente:
  - Aprendizaje recurrente en tiempo real (RTRL)
  - Retropropagación a través del tiempo (BPTT)
  - Métodos mixtos (como el usado en la red LSTM)
  - ...

Algoritmos de entrenamiento basados en el gradiente

- La información proporcionada por las derivadas puede usarse de distintas formas:
  - Descenso por el gradiente (DG)
  - Filtro de Kalman extendido desacoplado (FKED)
  - ...
- Según la frecuencia de actualización de los pesos:
  - En línea
  - Fuera de línea

Filtro de Kalman extendido desacoplado

- El descenso por el gradiente (DG) es normalmente lento porque se basa en el último valor del gradiente: no se tiene en cuenta la historia del entrenamiento
- Los algoritmos basados en el filtro de Kalman consideran recursiva y eficientemente toda la información disponible hasta el momento actual
- El FKED (Puskorius y Feldkamp, 1994) se basa en el filtro extendido de Kalman (no lineal)

Predicción con RNR

- Predicción del siguiente componente de la secuencia
- Dos posibilidades:
  - Secuencias numéricas
  - Secuencias simbólicas

Predicción de secuencias numéricas

- La muestra $u[t]$ se introduce en la red
- La salida deseada es $u[t+1]$
- Si la red aprende correctamente, la salida de la red será una estimación del valor de la siguiente muestra de la señal

Predicción de secuencias simbólicas

- Similar al caso de secuencias numéricas
- Los símbolos del alfabeto se suelen codificar mediante codificación exclusiva (salidas deseadas y entradas)
- Bajo determinadas circunstancias, la activación de una neurona de salida será una estimación de la probabilidad de que el siguiente símbolo de la secuencia sea el asociado a dicha neurona

Problemas estudiados en la tesis

① Compresión de secuencias simbólicas

② Aprendizaje de lenguajes con dependencias a largo plazo

③ Desambiguación categorial

④ Predicción de señales de voz

① Compresión de secuencias simbólicas

- Uso de RNR como modelo de probabilidades en línea para un compresor aritmético

- La predicción en línea es una tarea más difícil que la inferencia gramatical clásica con RNR (fuera de línea con más de una secuencia)

- Objetivo: secuencias de texto en inglés

- Aperitivo: secuencias de estados finitos, secuencias caóticas

Conjetura de Elman

- En el caso del aprendizaje fuera de línea puede demostrarse la convergencia de las probabilidades emitidas por la red a las frecuencias de aparición observadas (demostración sencilla)

- Conjetura de Elman (1990): bajo el aprendizaje en línea la salida de la red puede seguir considerándose como una aproximación a las probabilidades reales (empíricamente comprobable, pero no demostrado)

Trabajos previos

- Todos fuera de línea
- Secuencias de estados finitos: un clásico (Cleeremans et al., 1989). Además, las RNR pueden emular máquinas de estados finitos (Carrasco et al., 2000)
- Secuencias caóticas: Tiňo y Köteles (1999)
- Secuencias de texto: Schmidhuber y Heil (1996)

Compresión aritmética

- La compresión aritmética es una forma natural de evaluar la corrección del predictor

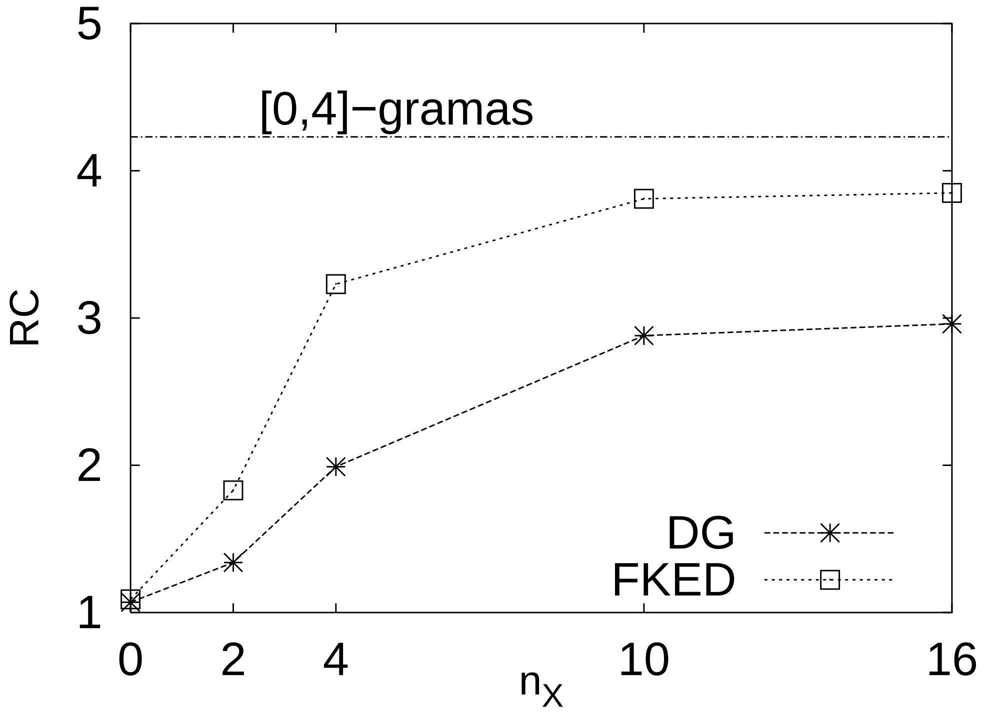
- A mejor modelo de probabilidad, mayor razón de compresión (RC)

Modelo alternativo

- Como modelo comparativo, se considera un modelo de probabilidad basado en n-gramas
- En un modelo de n-gramas la probabilidad del siguiente símbolo de la secuencia depende de los $n-1$ símbolos que le preceden
- En los experimentos se utiliza un modelo combinado de n-gramas con $n \in [0, 4]$

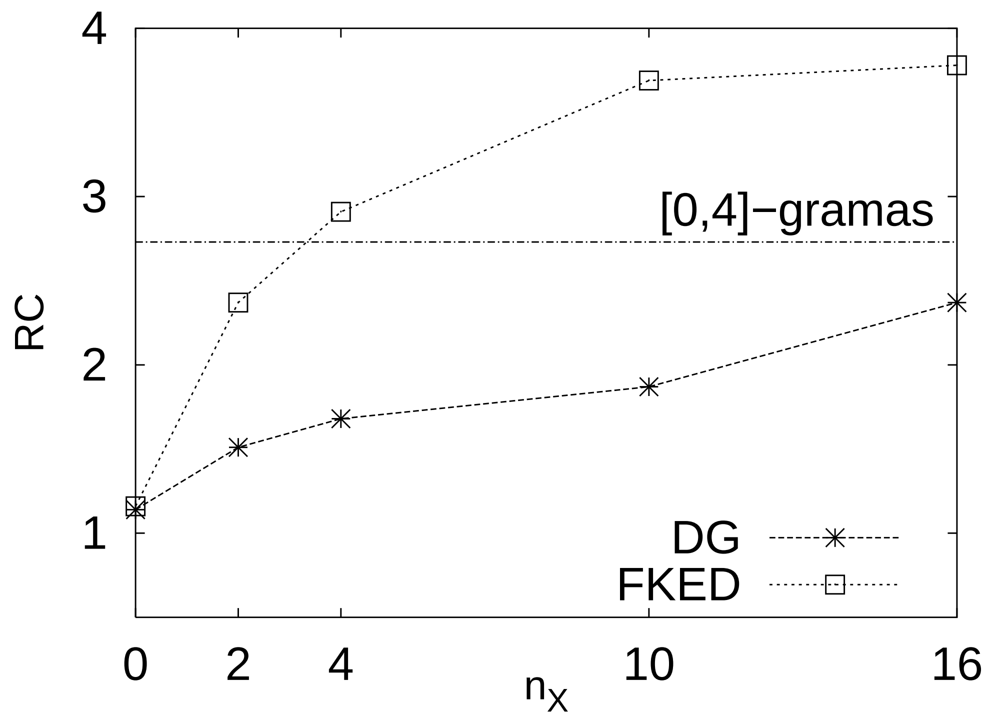
Secuencias de estados finitos

- Autómata simétrico de Reber continuo (Smith y Zipser, 1989)
- Longitud: 20 000



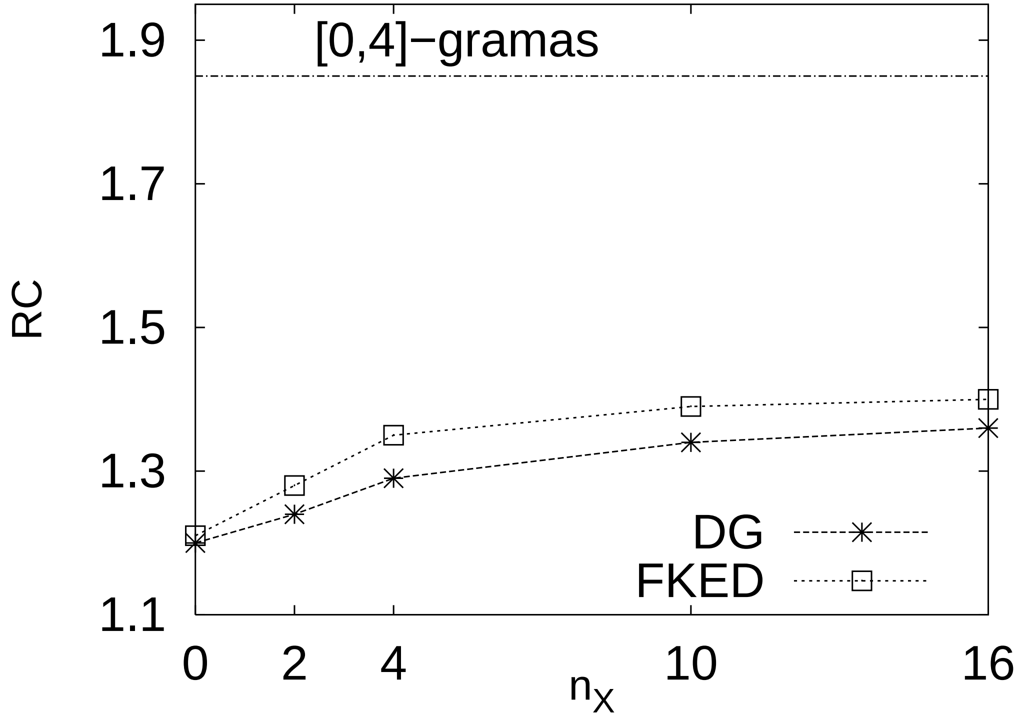
Secuencias caóticas

- Codificación simbólica de las activaciones de un láser en regimen caótico (Tiňo y Köteles, 1999)
- Longitud: 10 000



Secuencias de texto

- Ensayo en inglés sobre la obra del director Krzysztof Kieslowski
- Longitud: 62 648



Discusión

- El FKED supera siempre al DG, independientemente del tipo de secuencia involucrada en la tarea de predicción
- El rendimiento de las RNR en línea es aceptable con secuencias de estados finitos y caóticas: similar a $[0, 4]$-gramas
- Sin embargo, la dinámica de los textos en lenguaje humano es más difícil de aprender en línea

Próximos trabajos

- Estudio del modelo de probabilidad desarrollado cuando una RNR trabaja en línea y extracción del autómata correspondiente (si es posible)

- Aplicación de la red LSTM a la tarea de compresión de secuencias simbólicas y estudio de su rendimiento

② Inferencia de lenguajes con dependencias a largo plazo

- Aprendizaje simbólico en línea
- Aprendizaje de $a^n b^n c^n$

Aprendizaje en línea

- La adición de la compuerta de olvido al modelo LSTM debería posibilitar su uso en línea
- Sin embargo, ningún trabajo previo ha estudiado este hecho
- Aquí se aplica la red LSTM a la predicción en línea sobre una secuencia simbólica con dependencias a largo plazo

Introducción

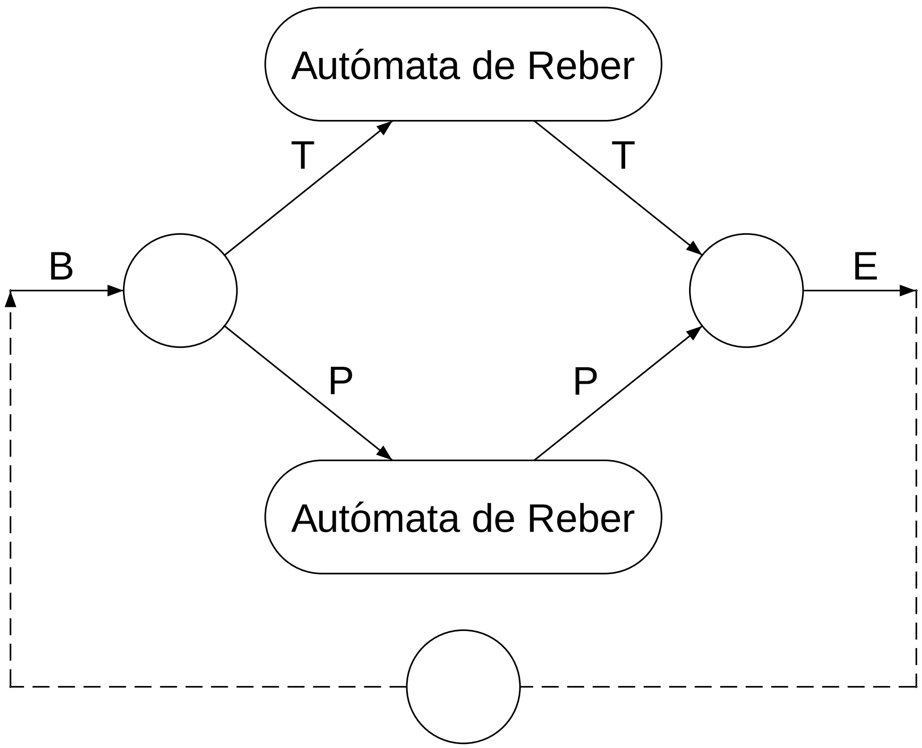
- Predicción en línea sobre una secuencia con dependencias a largo plazo
- Tarea similar a la desarrollada en los experimentos de compresión de secuencias simbólicas

Trabajos previos

- Gers et al. (2000) estudiaron un problema similar pero desde un enfoque a medio camino entre el aprendizaje en línea y fuera de línea
- La red LSTM se reiniciaba cada vez que cometía un error y las secuencias eran cortas
- ¿Cómo se comportará la red sin el apoyo de esta reiniciación al procesar secuencias de longitud arbitraria?

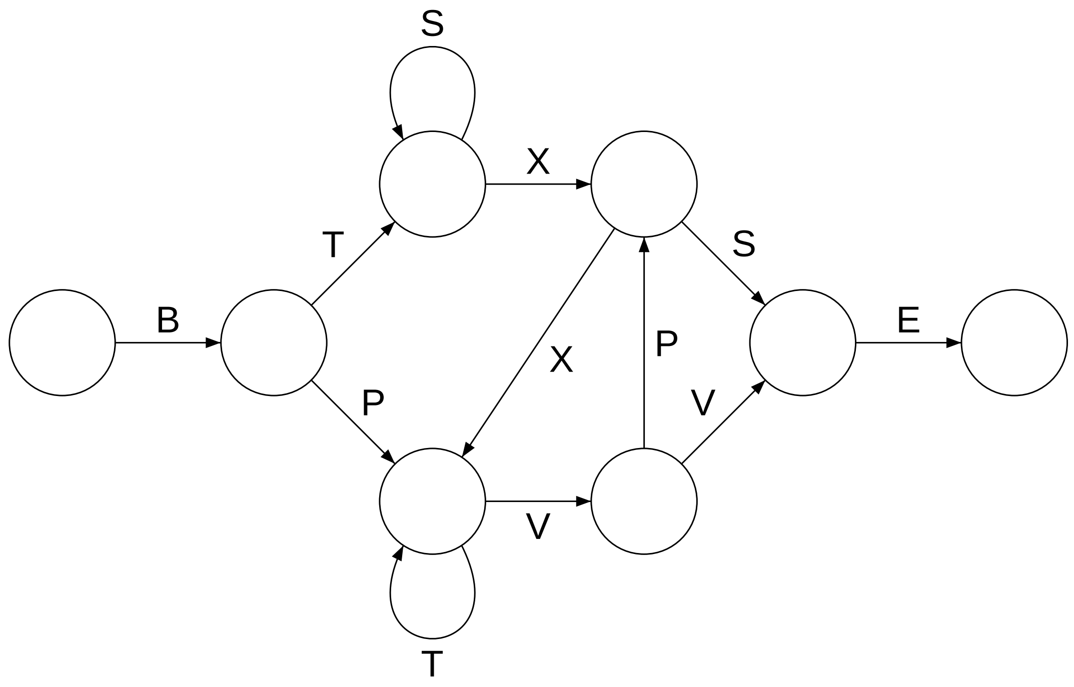
Método

- Secuencias extraídas del autómata de Reber simétrico continuo
- Los símbolos P y T introducen dependencias a largo plazo



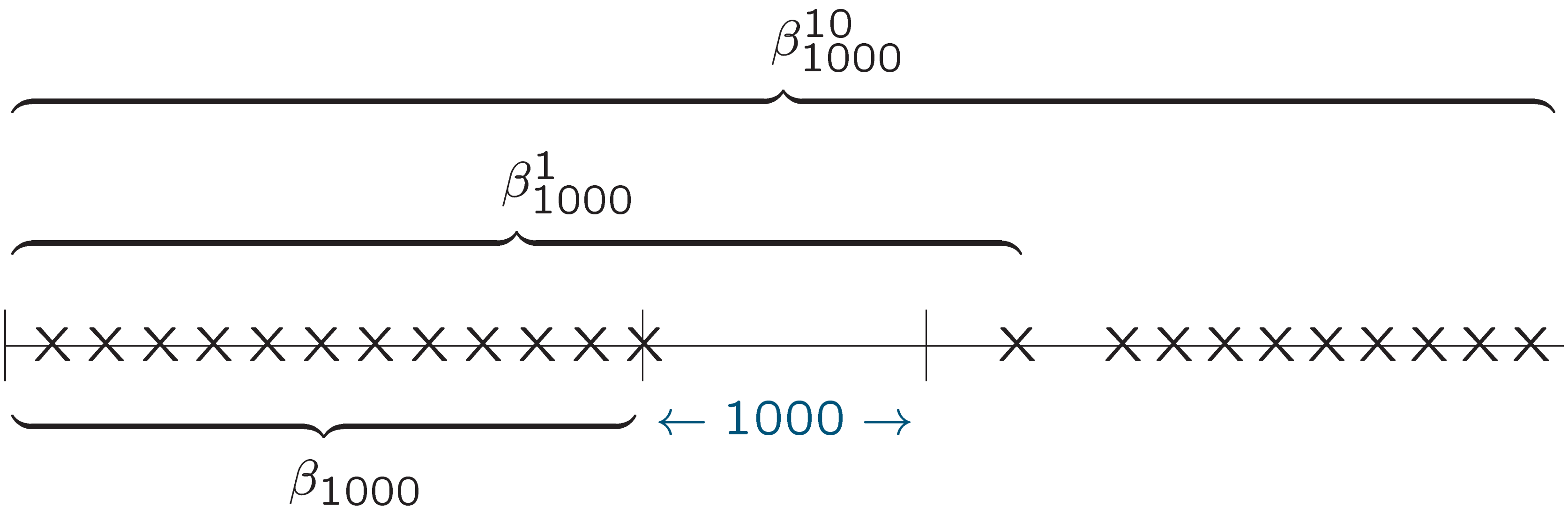
Autómata de Reber

- Solo presenta dependencias a corto plazo (trigramas)



Predicción sostenible

- β_{1000}: número de símbolos necesarios para no obtener errores durante al menos 1 000 predicciones (predicción sostenible)
- β^{1}_{1000}: símbolo en el que se produce el primer error tras una predicción sostenible
- β^{10}_{1000}: símbolo en el que se produce el décimo error tras una predicción sostenible



Mejores resultados de LSTM

- Descenso por el gradiente:

$$
\begin{aligned}
\beta_{1000} &= 39\,229 \\
\beta_{1000}^{1} &= 143\,563 \\
\beta_{1000}^{10} &= 178\,229
\end{aligned}
$$

- FKED:

$$
\begin{aligned}
\beta_{1000} &= 16\,667 \\
\beta_{1000}^{1} &= 29\,826 \\
\beta_{1000}^{10} &= 1\,000\,000^{+}
\end{aligned}
$$

Mejores resultados de la RRS

- Descenso por el gradiente: β_{1000} imposible de obtener y β_{100} muy de tarde en tarde

- FKED:

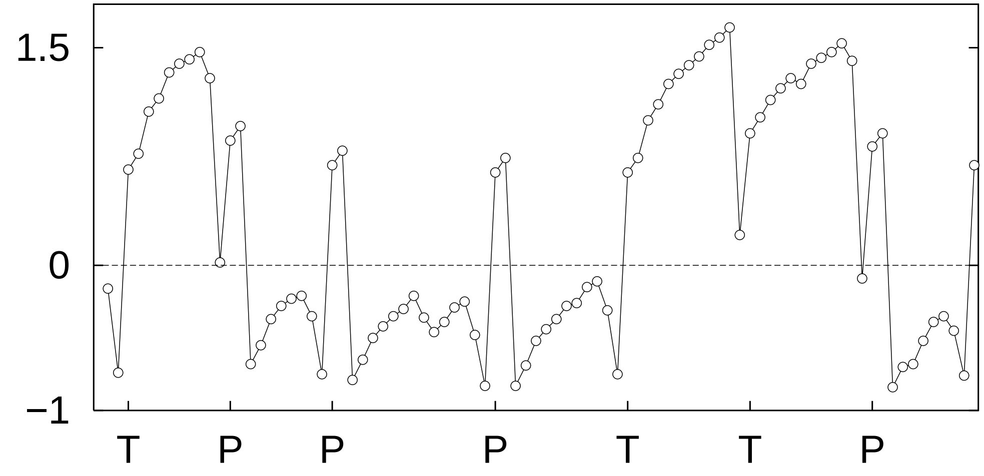
$$\begin{aligned} \beta_{1000} &= 58\,077 \\ \beta_{1000}^{1} &= 59\,222 \\ \beta_{1000}^{10} &= 65\,463 \end{aligned}$$

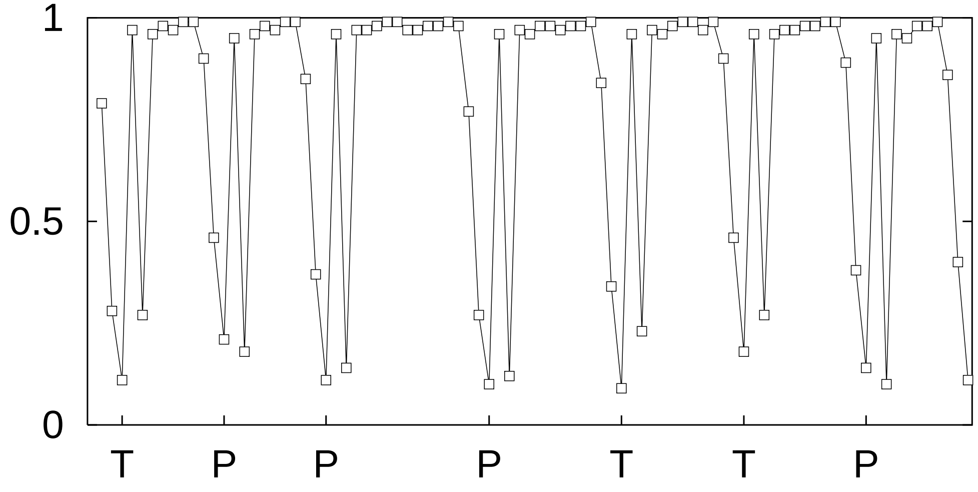
Soluciones al problema

- Las predicciones a corto plazo dejan de ser un problema para ambos modelos neuronales tras los primeros 4 000 símbolos aproximadamente
- Las probabilidades estimadas por la RRS para P y T están siempre en torno a $1/2$
- LSTM, por contra, sabe a ciencia cierta si el siguiente símbolo es P o T

Solución de LSTM

- Similar a la de anteriores estudios que no eran en línea





Discusión

- El FKED produce valores inferiores para β_{1000} que el descenso por el gradiente (DG), aunque parece olvidar más rápido (no en los mejores resultados)

- Las redes tradicionales no logran aprender la tarea con el DG; con el FKED lo consiguen parcialmente

- LSTM es aplicable a tareas de procesamiento en línea: la compuerta de olvido aprende a reiniciar la red cuando la memoria histórica ya no es necesaria y delimita, por tanto, secuencias independientes

Aprendizaje de $a^n b^n c^n$

- El problema: aprender el lenguaje $a^n b^n c^n$ con una RNR
- Los trabajos anteriores usaron:
  - Algunas RNR tradicionales con resultados pobres
  - LSTM con resultados mucho mejores
- El algoritmo de entrenamiento era descenso por el gradiente (DG) en todos los casos
- ¿Cómo funciona el FKED en esa tarea?

Introducción

- $a^n b^n c^n$: lenguaje sensible al contexto (lejano a los lenguajes regulares)
- Principal dificultad: dependencias a largo plazo
- Nos interesa estudiar la capacidad de la red para aprender el conjunto de entrenamiento y también para generalizar

Trabajos previos

- Redes recurrentes tradicionales:
  - Red secuencial en cascada de segundo orden (Bodén y Wiles, 2000) y RRS (Chalup y Blair, 1999)
  - Entrenamiento difícil, generalización ínfima
- LSTM (Gers y Schmidhuber, 2001):
  - Aprendizaje sencillo, buena generalización

Resultados para $n \in [1, 10]$

- Todos los modelos tienen aproximadamente el mismo número de pesos (≈ 70)
- Conjunto de entrenamiento: $a^n b^n c^n$ con $n \in [1, 10]$

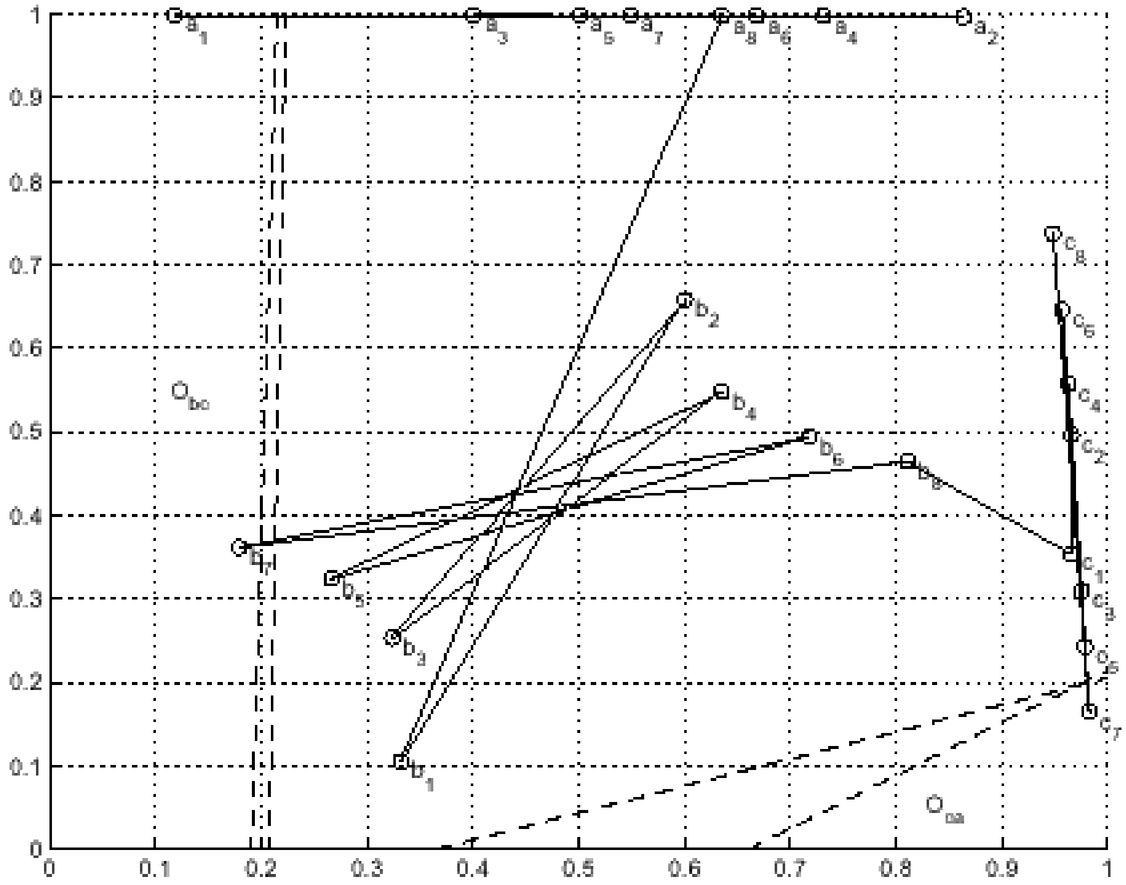
Modelo	**Sec. de entren.**	**% aprendido**	**Generaliz. media**	**Mejor generaliz.**
Tradicional + DG	20 000	8	[1,12]	[1,18]
LSTM + DG	45 000	90	[1,28]	[1,52]
LSTM + FKED	2 000	100	[1,434]	[1,2743]

El mejor resultado para $n \in [1, 10]$

- El mejor resultado sobre todas las redes LSTM entrenadas con $a^n b^n c^n$ con $n \in [1, 10]$
- Conjunto de entrenamiento aprendido completamente tras 2 000 secuencias
- Generalización: $n \in [1, 22\,463\,683]$

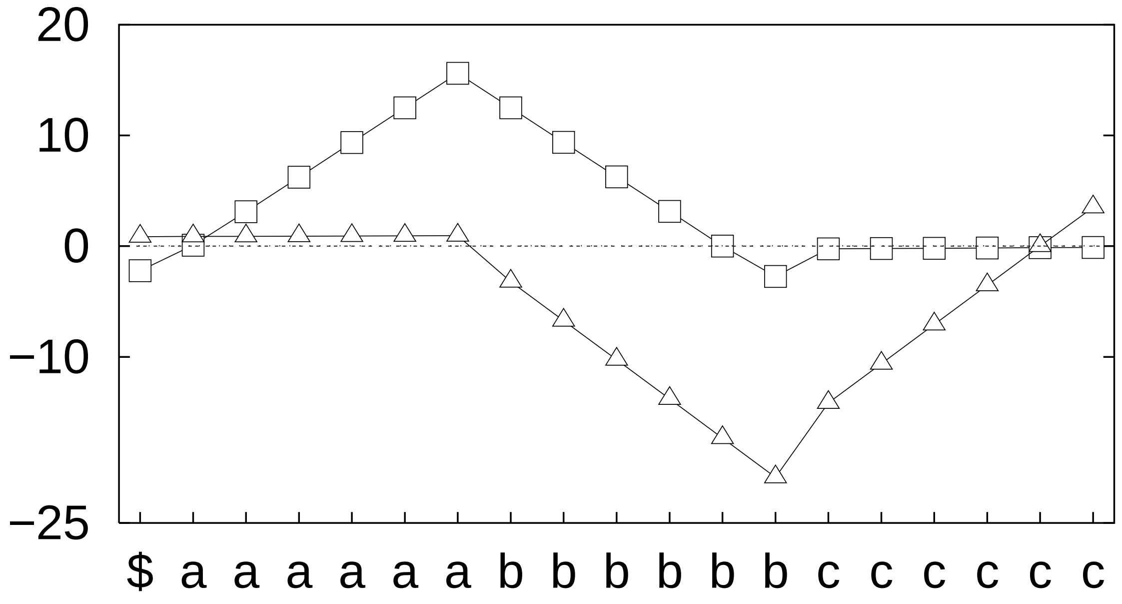
Solución encontrada por las RNR tradicionales

- Oscilaciones en torno a puntos fijos del espacio de estados acotado



Solución encontrada por LSTM

- Contadores monótonos en el espacio de estados no acotado
- Una celda de memoria resuelve $a^n b^n$ y la otra resuelve $b^n c^n$
- La misma solución tanto con DG como con el FKED



Discusión

- La red LSTM con el FKED necesita muchas menos secuencias de entrenamiento, aprende con mayor frecuencia y generaliza mucho mejor que la red LSTM tradicional (también con $n \in [20, 21]$)

- La complejidad temporal adicional del FKED se ve compensada con una velocidad de aprendizaje mucho mayor

- El FKED vuelve a superar al DG, esta vez con LSTM

Próximos trabajos

- Estudio de la red LSTM que generaliza hasta $n = 22\,463\,683$
- Uso de muestras negativas y de otros lenguajes sensibles al contexto
- Caracterización de la clase de lenguajes aprendibles con LSTM

③ Desambiguación categorial

- Desambiguación de las categorías léxicas de las palabras de una oración con ayuda de una RNR

- Un desambiguador de este tipo es un módulo principal de muchas aplicaciones relacionadas con el tratamiento del lenguaje humano

Categoría léxica

- Categoría léxica o parte de la oración: la categoría léxica de "trampolín" es nombre

- Etiquetador léxico: un programa que asigna una categoría léxica a cada palabra de una oración

Ambigüedad

- El problema: la ambigüedad
- Palabra no ambigua: el adjetivo “evanescente”
- Palabras ambiguas: “vino” y “refresco”, que pueden ser tanto nombre como verbo
- Clase de ambigüedad: “vino” y “refresco” pertenecen a la misma clase de ambigüedad {nombre,verbo}

Textos etiquetados

- Texto parcialmente etiquetado (no desambiguado):

  El/{artículo} vino/{nombre,verbo} pone/{verbo} alegre/{adjetivo,verbo}

- Texto completamente etiquetado (desambiguado y más difícil de obtener):

  El/artículo vino/nombre pone/verbo alegre/adjetivo

Trabajos previos

- Suposición: el contexto puede ayudar a determinar la categoría léxica correcta
- Enfoques:
  - Basados en reglas (Brill, 1992)
  - Estadísticos, como los modelos ocultos de Markov (Cutting et al., 1992)
  - Neuronales (Schmid, 1994)
  - Mixtos. . .

Trabajos neuronales

- Todos los trabajos neuronales se basan en el de Schmid (1994)
- Se entrena un perceptrón para producir la etiqueta adecuada a partir del contexto
- Se necesita extraer información estadística de un texto completamente etiquetado
- Los resultados son similares a los de un modelo oculto de Markov (MOM) en el que los observables son las clases de ambigüedad y los estados ocultos las categorías léxicas

Método

- Se basa en la información almacenada en el estado de una red recurrente simple (RRS)
- Originalidad:
  - Solo se necesita un texto parcialmente etiquetado
  - La tarea se aborda desde un enfoque predictivo
- El entrenamiento consta de dos fases

Primera fase

- Las entradas y las salidas deseadas son clases de ambigüedad
- Se entrena una RRS para predecir la clase de ambigüedad de la siguiente palabra
- Es de esperar que la red desarrolle una representación de estados de la información secuencial de las clases de ambigüedad

Segunda fase

- Los pesos de la RRS se congelan en esta fase
- Se introduce cada clase de ambigüedad en la RRS entrenada y se obtiene el correspondiente vector de estado
- Se entrena un perceptrón para obtener la etiqueta correcta a partir de este vector
- Salida deseada para las palabras ambiguas: 1 para las neuronas asociadas a las categorías de la clase de ambigüedad y 0 para el resto

Resultados

- Corpus del Penn Tree Bank
- Entrenamiento mediante descenso por el gradiente y bien RTRL, bien retropropagación
- Comparación con un modelo oculto de Markov (MOM) con un número similar de parámetros y con un etiquetador aleatorio
- Los resultados son el porcentaje de etiquetas incorrectas asignadas a palabras ambiguas

Mejores tasas de error

- Etiquetador aleatorio: 62 %
- MOM entrenado mediante el algoritmo de Baum y Welch: 45.3 %
- **Método basado en la RRS:** 44.2 %
- Tasa global de etiquetados correctos ≈ 92 %
- Sin embargo, la complejidad temporal de los enfoques neuronales es muy superior

Discusión

- El método basado en una RNR y el basado en un MOM tienen una capacidad desambiguadora similar
- Este método es el primer trabajo basado en una red neuronal que no necesita un texto completamente etiquetado y que resuelve la tarea con un enfoque predictivo
- A diferencia de los MOM, la desambiguación se realiza usando solo el contexto previo

Próximos trabajos

- Extracción de un modelo de estados finitos (si lo hay) a partir del estado de la RRS para formular el conjunto de reglas aprendidas por el sistema

- Evaluación de la influencia del tamaño del corpus de entrenamiento

- Aplicación de la red LSTM a esta tarea

④ Predicción de señales de voz

- Predicción de señales de voz
- Los únicos trabajos con RNR se basan en la red neuronal recurrente en cascada (RNRC)
- ¿Cuál es el rendimiento de las redes recurrentes tradicionales?

Introducción

- La predicción se realiza directamente sobre la señal muestreada $u[t]$, es decir, no hay preprocesamiento

- Si el predictor es eficiente, la codificación predictiva permite comprimir la señal

- Diversos estándares como ADPCM se basan en esta idea

- Suposición: $\hat{u}[t]$ puede obtenerse a partir de $u[t-1], u[t-2], \ldots$

Trabajos previos

- Multitud de enfoques descritos en la bibliografía
- Nos centraremos en:
  - Predictores lineales
  - Predictores basados en RNR: deberían tener en cuenta los aspectos no lineales de la voz
  - La única RNR considerada para esta tarea era la RNRC

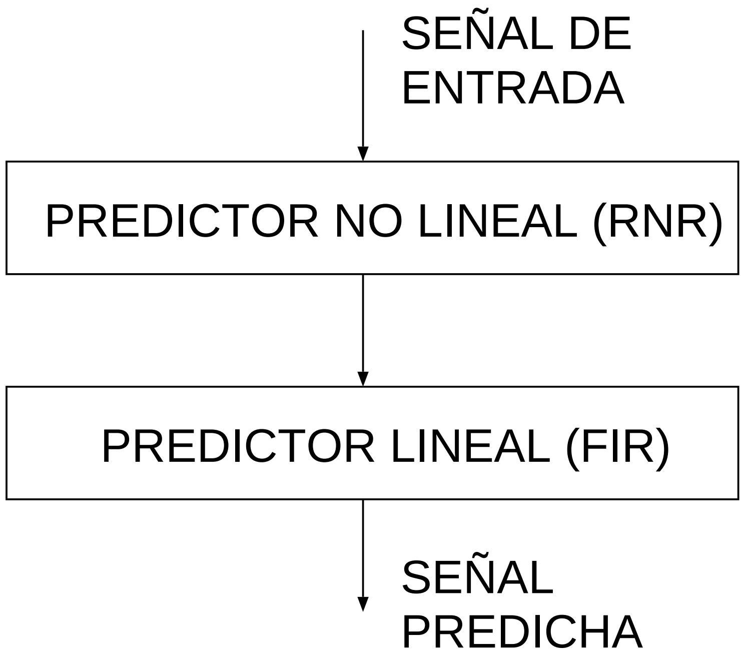
Filtros lineales

- Predictor simple tomado como rendimiento base: $\widehat{u}[t] = u[t-1]$
- Filtro de respuesta finita al impulso (FIR) en el que la predicción es una combinación lineal de las muestras previas (q es el orden del filtro):

$$\widehat{u}[t] = \sum_{i=1}^{q} w_i \, u[t-i]$$

Configuración del predictor

- Configuración propuesta por Haykin y Li (1995)
- En este trabajo sustituimos la RNRC por redes tradicionales
- Es de esperar que el bloque no lineal "linealice" la señal y que el filtro FIR pueda sacar partido de ello



Entrenamiento

- Algoritmos de entrenamiento del filtro lineal: mínimos cuadrados (LMS) y mínimos cuadrados recursivo (RLS)
- Algoritmos de entrenamiento de las RNR: descenso por el gradiente (DG) y el FKED
- Los resultados con RNR tradicionales se mostrarán para una RPR

Parámetros

- Los resultados son la media de 7 inicializaciones de los pesos diferentes (excepto para la RNRC: parámetros ad hoc)
- El filtro FIR tiene orden $q = 12$
- La RNRC y la RPR tienen aproximadamente el mismo número de pesos ≈ 35
- El rendimiento se mide con la ganancia de predicción G
- A mayor G, mejor es el predictor

Peores resultados

- Modelos que producen resultados peores que los del filtro FIR entrenado con mínimos cuadrados (LMS):

Modelo	G (dB)
RPR + FIR	≤ 3.99
⋆ $\hat{u}[t+1] = u[t]$	4.61
RPR (DG)	5.80
⋆ FIR (LMS)	5.82

Mejores resultados

- Modelos que producen resultados mejores que los de un filtro FIR entrenado con LMS:

Modelo	G (dB)
RNRC (DG) + FIR (LMS)	7.30
RPR (FKED)	8.61
RNRC (DG) + FIR (RLS)	9.24
$\star$ FIR (RLS)	9.66
RNRC (FKED) + FIR (RLS)	10.90

Discusión

- Problemas de las RNR tradicionales para manejar secuencias de voz
- La adición de un filtro lineal solo parece útil en el caso de la RNRC
- El FKED supera parcialmente algunas de estas limitaciones
- Solo la RNRC seguida por un filtro FIR entrenado con mínimos cuadrados recursivo (RLS) supera los resultados individuales del filtro

Próximos trabajos

- Aplicación de LSTM a esta tarea, aunque experimentos preliminares arrojaron resultados similares
- Evaluación exhaustiva de la RNRC

Contribuciones de la tesis

- Se confirma la idea de que el FKED debería considerarse como algoritmo de entrenamiento en cualquier experimento con RNR

- Primer uso de la red LSTM en tareas de predicción completamente en línea y primer uso del FKED sobre LSTM

- La red LSTM entrenada con el FKED obtiene una generalización sin precedentes sobre el lenguaje $a^n b^n c^n$

- Primer desambiguador categorial neuronal predictivo que no necesita un corpus completamente etiquetado (comparable a los MOM)

- Las RNR tradicionales muestran claras dificultades a la hora de predecir en línea señales de voz digitalizada