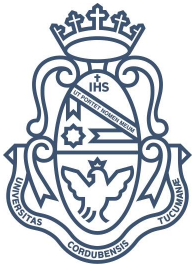
Facultad de Matemática, Astronomía,
Física y Computación

Universidad Nacional de Córdoba



Estudio de redes neuronales en escalera como método semi-supervisado para reconocimiento de entidades nombradas en textos legales

Tesis para obtener el título de

Licenciado en Ciencias de la Computación

Autor: Santiago Marro

Director: Cristian Cardellino

Córdoba, Argentina 2019

Agradecimientos

A mi familia, que me dieron su apoyo y contención incondicional.

A los profesores y profesoras que me compartieron y contagiaron su pasión por la ciencia.

A mi director de tesis por su grandiosa dedicación.

A la Universidad Nacional de Córdoba,

¡Muchas gracias a todos!

Resumen

En este trabajo se exploró el uso de un método de aprendizaje automático semi-supervisado profundo, conocido como "Redes Neuronales en Escalera". Como caso de estudio, se decidió abordar la tarea de reconocimiento y clasificación de entidades nombradas dentro del dominio legal. Como baselines se establecieron el Stanford NER-CRF y las redes recurrentes BiLSTM. El trabajo consistió en la exploración de distintos aspectos y parámetros donde se buscó evaluar el impacto de los datos no supervisados y su comparación con métodos puramente supervisados, en busca de mejorar desempeño y capacidad de generalización.

Palabras clave: Redes neuronales, Redes en escalera, Aprendizaje automático, Procesamiento del lenguaje natural, Reconocimiento de entidades nombradas.

Summary This thesis explored the use of a semi-supervised deep learning method known as "Ladder Neural Networks". As a case study, it was decided to address the task of recognizing and classifying named entities within the legal domain. The Stanford NER-CRF and the BiLSTM Recurrent Neural Networks were established as baselines. The work consisted in the exploration of different aspects and parameters where it was sought to evaluate the impact of unsupervised data and its comparison with purely supervised methods, seeking to improve performance and generalization capacity.

Keywords: Neural networks, Ladder networks, Machine learning, Natural language processing, Named entity recognition.

Índice general

Capítulo 1

Introducción y Motivación

1.1. Introducción

El campo del aprendizaje automático se ocupa de resolver el problema de cómo construir programas de computación que automáticamente mejoren con la experiencia (Mitchell, 1997). El mismo tiene como objetivo desarrollar técnicas y programas capaces de generalizar comportamientos a partir de información suministrada en forma de ejemplos.

Cuando hablamos de información en forma de ejemplos estamos hablando de datos relacionados con el problema a resolver, los cuales pueden ser etiquetados o no etiquetados.

Los datos etiquetados son aquellos donde, además del dato en sí, se proporciona información extra, generalmente proveniente de un anotador humano, llamada etiqueta, la cual indica el resultado del problema asociado con ese dato.

Por ejemplo, si nuestro problema es el de clasificar imágenes para distinguir si se encuentra un automóvil o no en la misma, nos encontramos con dos categorías distintas, llamémosles *Auto y Otro*, donde la etiqueta *Otro* se le asigna a las imágenes sin automóviles y la etiqueta *Auto* a aquellas donde sí los hay.

El **aprendizaje supervisado** es una subárea del aprendizaje au-

tomático, donde se hace uso de estos datos etiquetados para distintas tareas: clasificación de imágenes (como en el ejemplo), clasificación de documentos, clasificación de sentimiento en una oración, etc.

Dependiendo el problema se pueden obtener resultados muy buenos con estas técnicas, siempre y cuando se tenga la cantidad de datos etiquetados suficientes. Esto se da ya que estas técnicas se basan en, dados datos iniciales, aprender una función que asigne datos de entrada a datos de salida. Es decir, crear un modelo que dado datos de entrada prediga su dato de salida. Si se tienen pocos datos de entrada etiquetados, el modelo no será lo suficientemente bueno como para cubrir todos los casos reales del problema.

En contraste con los datos etiquetados, tenemos los datos *no etiquetados* los cuales no tienen etiqueta alguna o se les fueron removidas. Estos son utilizados en técnicas de **aprendizaje no supervisado** donde lo que se busca es encontrar patrones que ayuden a explicarlos.

El **aprendizaje semi-supervisado** es una técnica intermedia entre las dos anteriores, que nos permite integrar la información disponible en datos no etiquetados a aquellos modelos entrenados a partir de datos etiquetados, con el objetivo de obtener mejores modelos que se nutran de la información extra.

Particularmente, en esta tesis estudiaré y experimentaré con **redes en escalera**, un modelo de aprendizaje semi-supervisado profundo el cual utiliza una arquitectura neuronal para aprender de datos supervisados y no supervisados, presentado por Rasmus et al. (Rasmus et al., 2015).

Lo interesante de esta técnica de aprendizaje semi-supervisado en particular es que presenta resultados en su desempeño muy favorables utilizando escasos datos supervisados y a su vez es fácil de acoplar a otras técnicas comunes en el campo de estudio por lo que podría mejorar resultados en otros problemas donde haya datos supervisados limitados.

En contraste con otras técnicas de aprendizaje semi-supervisado, como el autoaprendizaje (Chapelle et al., 2010) o el aprendizaje activo (Settles, 2009), los cuales son algoritmos que utilizan un clasificador

supervisado para expandir los datos de una fuente no etiquetada, las redes en escalera combinan ambos conjuntos de datos en una sola *función objetivo* la cual es minimizada utilizando retropropagación (back-propagation) y descenso del gradiente.

La exploración de este trabajo se realizó sobre un corpus obtenido a partir de la Wikipedia, que fue específicamente procesado para representar entidades nombradas relacionadas al dominio legal en el trabajo de Cardellino et. al. (Cardellino et al., 2017).

1.2. Motivación

Para que algunos modelos, en especial aquellos basados en redes neuronales, funcionen de manera correcta y eviten sobreajustar a los datos de entrenamiento (i.e. memorizar los datos en lugar de aprender funciones que los aproximen), es necesario obtener grandes volúmenes de datos etiquetados.

Esto es algo que se torna costoso, puesto que el anotador es humano. Más aún, en distintos problemas el etiquetado puede ser dificultoso para un humano, por ejemplo si se habla de semántica, por lo cual entonces se necesitaría un equipo de personas etiquetando, o expertos de dominio, incrementando aún más su costo.

En particular, hay ciertos dominios donde los datos son aún más escasos por no poseer un uso comercial claro o bien tener una explotación comercial limitada, con una marcada carencia de los recursos para explotarlos.

Por otra parte, en la era del internet, se producen cantidades enormes de datos no estructurados. Esto es algo que permite conseguir de manera relativamente sencilla y barata datos no anotados en grandes volúmenes. El problema, es que los modelos no supervisados tienen límites muy marcados respecto a que se puede lograr con ellos.

Es aquí donde entra la importancia de explorar y estudiar técnicas de aprendizaje semi-supervisado, que aprovechen al máximo la cantidad limitada de recursos etiquetados disponibles, y los aumenten con

la inmensa cantidad de datos no anotados disponibles.

Dado que las redes en escalera son un método semi-supervisado, es interesante analizar su desempeño en una tarea cuyos datos sean en mayor o menor medida limitados.

Para ello decidí trabajar en esta tesis sobre el reconocimiento de entidades nombradas aplicado en textos legales utilizando las redes en escalera como herramienta.

La tarea del reconocimiento de entidades nombradas en texto es una tarea fundamental que da paso a la extracción de información del texto.

En particular, con el caso de los textos de dominio legal es de común conocimiento que el acceso a la información contenida en estos no es simple y requiere trabajo de gente muy capacitada (i.e. expertos de dominio como abogados, jueces, legisladores, etc.). Es por ello que se decidió explorar esta área para desarrollar una mejor plataforma utilizando el aprendizaje automático como herramienta en la construcción de soluciones útiles para problemas del dominio legal.

1.3. Estructura de la tesis

La presente tesis se estructura de la siguiente manera, en el Capítulo 2 se detallarán todos los conceptos fundamentales sobre los que se trabajó en esta tesis.

El Capítulo 3 describe la arquitectura de la red en escalera así como también sus modificaciones realizadas. Por otro lado, se detalla la metodología de experimentación con la que se trabajó a lo largo de esta tesis, describiendo los recursos trabajados, los baselines explorados y los experimentos realizados.

En el Capítulo 4 se hace un análisis detallado sobre los resultados de los experimentos detallados en el Capítulo 3 y su correspondiente comparación con los baselines.

Finalmente, el Capítulo 5 se escriben las conclusiones a las que se llegaron a través de los experimentos realizados además de los aportes

realizados por esta tesis. Por otra parte, se detallan líneas de trabajo futuro que pueden ser exploradas.

Capítulo 2

Trabajo Relacionado

En este capítulo se verá en detalle los conceptos teóricos en los cuales se apoyó esta tesis. Primero se desarrollarán aquellos conceptos relacionados con el procesamiento del lenguaje natural y luego se describirán los distintos modelos utilizados a lo largo de esta tesis.

2.1. Procesamiento de lenguaje natural

El proceso de leer y comprender el lenguaje natural es mucho más complejo de lo que parece. La interpretación puede fallar de diversas maneras, un ejemplo está dado por la siguiente oración.

Ejemplo 2.1.1. Marcela está ciega, no quiere ver la realidad.

Para un humano es fácil interpretar que "Marcela" no es una persona no vidente, sino que la oración se refiere a que no quiere aceptar una situación particular. Este conocimiento es dado por el contexto y el sentido común. Una máquina, al carecer de dichos elementos, puede interpretar la oración literalmente.

Para ayudarnos con estas tareas surge el **procesamiento de lenguaje natural**, comúnmente abreviado como PLN (o NLP en inglés), el cual es una rama de la Inteligencia Artificial cuyo objetivo final es

poder facilitar la interacción entre el humano y la máquina a través de lenguajes humanos (e.g. inglés, español, francés, etc.), es decir, la comunicación mediante lenguajes no formales (como lo son la lógica, la matemática o los lenguajes de programación).

2.1.1. Reconocimiento de entidades nombradas

El **reconocimiento de entidades nombradas**, es una súbarea del procesamiento de lenguaje natural que se encarga de reconocer, en texto libre, unidades de información como nombres de personas, organizaciones, ubicaciones, fechas, montos de dinero, etc.

El término *entidad nombrada* fue concebido en la *Sixth Message Understanding Conference (MUC-6)* (Grishman and Sundheim, 1996).

En ese momento, MUC se estaba enfocando en tareas de extracción de información estructurada a partir de textos no estructurados, como artículos de noticias. Al definir las tareas en la conferencia, la comunidad notó que es esencial reconocer las unidades de información de las entidades.

Identificando referencias a las entidades en texto fue reconocida como una de las sub-tareas mas importantes de extracción de información y fue designada como *Reconocimiento y clasificación de entidades nombradas* ("Named Entity Recognition and Classification" o "NERC" en ingles) (Nadeau and Sekine, 2007).

Un ejemplo de aplicación de un reconocimiento y clasificación de entidades nombradas sería:

Ejemplo 2.1.2. El Dr. Luis Dolina se expresó en contra del Tratado de Libre Comercio de América del Norte en 1988.

Donde se produciría un bloque de texto anotado en donde se remarcan las entidades:

Ejemplo 2.1.3. El Dr. [Luis Dolina]$_{\text{Persona}}$ se expresó en contra del [Tratado de Libre Comercio de América del Norte]$_{\text{Documento}}$ en [1988]$_{\text{Año}}$.

La expresión *entidad nombrada* especifica la tarea de clasificación a aquellas entidades de una o más cadenas de caracteres, tales como palabras o frases, de las cuales sean consistentemente reconocidas por algún referente.

Como su nombre lo indica, NERC es una tarea conceptualmente dividida en dos problemas: detección de nombres y su clasificación según el tipo de entidad a la cual se refieren.

En el ejemplo anterior, vemos que se detecta a *Luis Dolina* como un nombre de entidad, para luego clasificarlo como *Persona*.

2.1.2. Vectores densos de palabras (Embeddings)

Los modelos de aprendizaje automático requieren de algún tipo de representación matemática de sus datos de entrada sobre los cuáles aprender. Si bien hay distintas formas de representar las palabras, frases, oraciones, etc, las redes neuronales en particular han probado ser muy buenas utilizando ciertas representaciones vectoriales conocidas como *vectores densos de palabras*.

Un **vector denso de palabras** (comúnmente conocido como *word embedding*, en inglés) es una de las técnicas de representación de vocabulario más popular en el área de PLN.

Es capaz de capturar el contexto, semántica y similitud sintáctica de una palabra en un documento, en r elación con otras palabras en el mismo.

Los vectores densos de palabras son representaciones vectoriales de una palabra en particular, donde el objetivo es que las palabras relacionadas entre sí (ya sea por contexto, semántica o sintáctica) se encuentren cercanas entre sí en el espacio vectorial planteado.

Consideremos las siguientes oraciones: *Que tengas un buen día* y *Que tengas un gran día*. Semánticamente no difieren (i.e. significan lo mismo).

Si construimos un vocabulario para ellas, tendríamos:

$$V = \{Que, tengas, un, buen, gran, día\}$$

Al generar representaciones vectoriales de estas palabras, buscamos que los vectores asignados a *buen* y *gran* estén cercanos entre sí en el espacio vectorial, dado que a pesar de ser distintas, representan lo mismo *semánticamente* en las oraciones del ejemplo.

En los últimos años se ha presenciado un crecimiento importante en el uso de los embeddings y como resultado, muchas tareas en Procesamiento de Lenguaje Natural han intentado aprovechar el potencial de estos modelos. En particular, Mikolov (Mikolov et al., 2013) propuso el modelo "word2vec" que se valía de la información de coocurrencias entre palabras en un corpus de texto de gran tamaño para generar representaciones vectoriales de las palabras con resultados muy interesantes. Nos basamos en embeddings pre-entrenados generados a partir del algoritmo word2vec para representar las palabras que se les dan a los modelos de aprendizaje automático.

Representación de instancias de entrenamiento utilizando vectores densos de palabras

En el trabajo publicado por Iacobacci (Iacobacci et al., 2016) se estudia cómo utilizar embeddings de palabras para desambiguación de sentidos de palabras y se proponen distintos métodos por los cuales se pueden aprovechar los embeddings en una arquitectura supervisada.

El mismo trabajo también muestra cómo utilizando los embeddings de manera adecuada se pueden obtener mejoras significativas en el desempeño sobre las arquitecturas estándares que utilizan atributos diseñados manualmente.

A continuación describimos algunos de los modelos explorados por Iacobacci para representar las instancias de entrenamiento de nuestros modelos de aprendizaje.

Concatenación El primer método que se propone es el de la concatenación de los vectores alrededor de la palabra objetivo en un solo vector el cual tiene un tamaño de la suma de las dimensiones de cada embedding concatenado.

Sea w_{ij} el peso asociado con la i^{th} dimensión del vector j^{th} en la oración, sea D la dimensionalidad del vector, y W el tamaño de ventana el cual está definido como el número de palabras en un solo lado. Nos interesa representar el contexto de la I^{th} en la oración.

Promedio Como su nombre lo indica, este método computa el centroide de los embeddings de todas las palabras alrededor de todas las palabras. La fórmula divide cada dimensión por 2W dado que el número de palabras de contexto es el doble del tamaño de la ventana.

$$e_i = \sum_{\substack{j=I-W \\ j \neq i}}^{I+W} \frac{w_{ij}}{2W}$$

Reducción fraccionaria Otra estrategia propuesta se basa en la forma en que Word2vec combina las palabras en el contexto. Aquí la importancia de la palabra es inversamente proporcional a su distancia con respecto a la palabra objetivo. Por lo tanto, las palabras vecinas son pesadas basándose en la distancia a la palabra objetivo.

$$e_i = \sum_{\substack{j=I-W \\ j \neq i}}^{I+W} w_{ij} \frac{W - \mid I - j \mid}{W}$$

Reducción exponencial Este método funciona de manera similar al fraccionario, donde se le da más importancia al contexto mas cercano, pero en este caso el peso de las palabras es efectuado de manera exponencial.

$$e_i = \sum_{\substack{j=I-W \\ j \neq i}}^{I+W} w_{ij}(1-\alpha)^{|I-j|-1}$$

donde $\alpha = 1 - 0{,}1^{(W-1)^{-1}}$ es el parámetro de decaimiento. Se eligió tal parámetro de tal forma que las palabras inmediatamente alrededor de la objetivo contribuyan 10 veces más que las últimas palabras en ambos lados de la ventana.

2.2. Aprendizaje Automático

Como se describió en el Capítulo anterior, el aprendizaje automático es la tarea que se encarga de encontrar una función que aproxime un conjunto de datos de entrada a sus etiquetas de salida.

Hay distintos algoritmos que se encargan de encontrar estas funciones, en particular esta tesis explora sólo algunos que nos son relevantes: el "campo aleatorio condicional" y los modelos de redes neuronales.

2.2.1. Etiquetado secuencial

Si bien en general se piensa el proceso de aprendizaje automático como el aprendizaje de una función que aproxime datos discretos de entrada a datos discretos de salida (i.e. cada ejemplo es representado de manera independiente a los demás ejemplos), esto supone ciertas limitaciones sobre todo en tareas de lenguaje natural donde el contexto de la secuencia da información extra bastante importante.

Muchos modelos de aprendizaje automático (e.g. el perceptrón multicapa o las redes convolucionales) tienen esta limitación natural, mientras que otros modelos (e.g. las redes recurrentes, o el campo aleatorio condicional) pueden modelar esta información.

Este modelado del contexto en base a una secuencia de instancias (e.g. de palabras), se conoce como **etiquetado secuencial**, y es un tipo de tarea del aprendizaje automático la cual toma como dato de entrada una secuencia (e.g. una secuencia de palabras) y aprende a predecir una secuencia óptima de etiquetas (e.g. de entidades nombradas), generalmente considerando toda la información del contexto.

2.2.2. Stanford NER-CRF

Como uno de los primeros baselines explorados en este trabajo, decidimos optar por el uso de la herramienta Stanford NER-CRF (Finkel et al., 2005).

El Stanford NER-CRF es una implementación de un reconocedor y clasificador de entidades nombradas. Es una herramienta completa que incluye un extractor de atributos manuales (o "features" en inglés) para el reconocimiento de entidades nombradas y un algoritmo CRF de aprendizaje automático para entrenar el modelo supervisado de NERC. El acrónimo CRF viene del inglés *conditional random field* o "campo aleatorio condicional".

El CRF es una clase de modelo estadístico que suele aplicarse a tareas de aprendizaje automático y reconocimiento de patrones. En particular, el CRF es utilizado para el modelado estadístico de secuencias, es decir son útiles para el etiquetado secuencial que se describe en la Sección anterior.

2.2.3. Redes neuronales

Parte del trabajo de esta tesis se basó en el estudio particular de algoritmos de redes neuronales para la tarea de reconocimiento de entidades nombradas. A continuación se detallan los conceptos fundamentales de las arquitecturas de red trabajadas.

Perceptrón multicapa

El perceptrón multicapa (*multilayer perceptron*), es un tipo de red neuronal "prealimentada" (*feed-forward neural network*). Es el tipo de red neuronal más básico, y también el modelo base de aprendizaje profundo.

El objetivo de un perceptrón multicapa es el de aproximar una función f. Por ejemplo, para un clasificador donde $y = f(x)$ asigna un dato de entrada x a una categoría y. Un perceptrón multicapa define la asignación $y = f(x; \theta)$ y aprende el valor de los parámetros θ

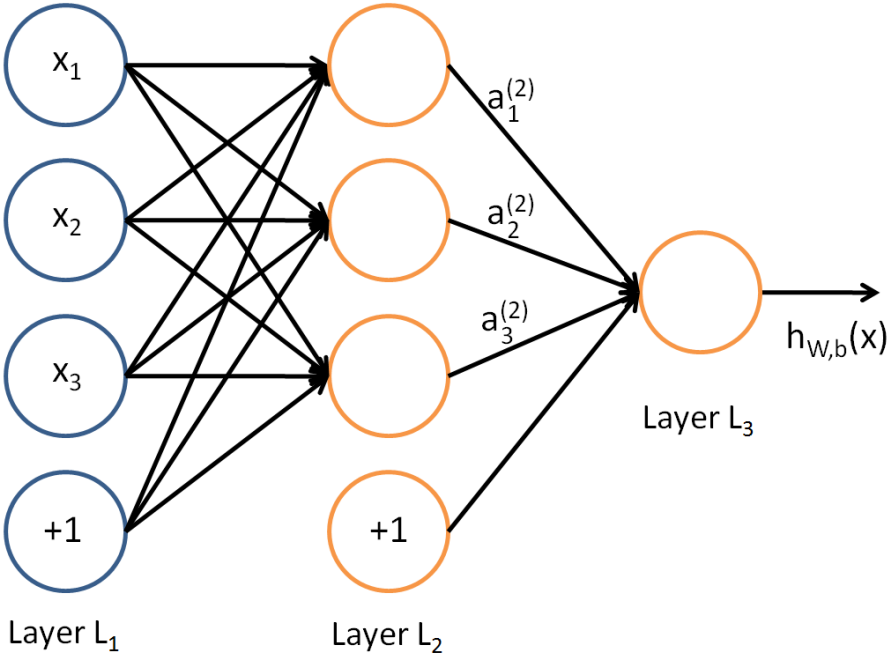


Figura 2.1: Perceptrón multicapa. Figura obtenida de (Stanford, 2013b).

que resultan en la mejor aproximación para la función f (Goodfellow et al., 2016).

En estos modelos la información fluye desde la función siendo evaluada en el valor x, por cómputos intermedios usados para definir a la misma, hasta finalmente el dato de salida y.

En la Figura 2.1 se puede observar una imagen representando al perceptrón multicapa como un grafo dirigido.

Las redes neuronales se entrenan mediante el algoritmo de retropropagación (*backpropagation*) (Rumelhart et al., 1988), que calcula el gradiente de los pesos de la red a partir del error de la misma.

Autoencoder

Los *autoencoders* (algunas veces encontrados como "autocodificadores" en la literatura en español) son un tipo de red neuronal prealimentada para aprendizaje no supervisado.

El objetivo de estas herramientas es la compresión de los datos. Para ello toman un ejemplo y tratan de reducir sus dimensiones (paso de codificación) para luego reconstruirlo a medida que aumentan las dimensiones (paso de decodificación). La red se entrena mediante el costo de reconstrucción (i.e. la diferencia entre la entrada y la salida).

Una vez entrenada la red, el codificador (o *encoder*) de la misma puede generar una representación reducida de los datos de entrada.

La Figura 2.2 muestra una imagen que representa el autoencoder como un grafo.

En términos más formales supongamos que tenemos datos no etiquetados $x(1), x(2), x(3), \ldots$ donde $x(i) \in \Re$. Un autoencoder es un algoritmo de aprendizaje no supervisado que utiliza retropropagación (back-propagation), estableciendo a los valores objetivos iguales a los datos de entrada. Es decir, $y(i) = x(i)$.

El autoencoder intenta aprender una función $h_{W,b}(x) = x$. En otras palabras, intenta aprender una aproximación a la función identidad, tal que la salida $\hat{x}$ sea similar a x.

La función identidad puede parecer una función particularmente fácil de aprender, pero estableciendo restricciones a la red, tales como limitando el número de neuronas escondidas, podemos descubrir propiedades interesantes de la estructura de los datos.

Veamos un ejemplo concreto, supongamos que los datos de entrada x son los valores de intensidad de los píxeles de una imagen de 10x10 (100 píxeles), por lo tanto tenemos $n = 100$, y hay $s2 = 50$ neuronas en la capa oculta $L2$.

Dado que hay solo 50 neuronas en la capa oculta, la red se ve forzada a aprender una representación çomprimida"del dato de entrada. Si los datos de entrada fueran completamente aleatorios esta tarea de comprensión sería muy dificultosa. Pero si se encuentra algu-

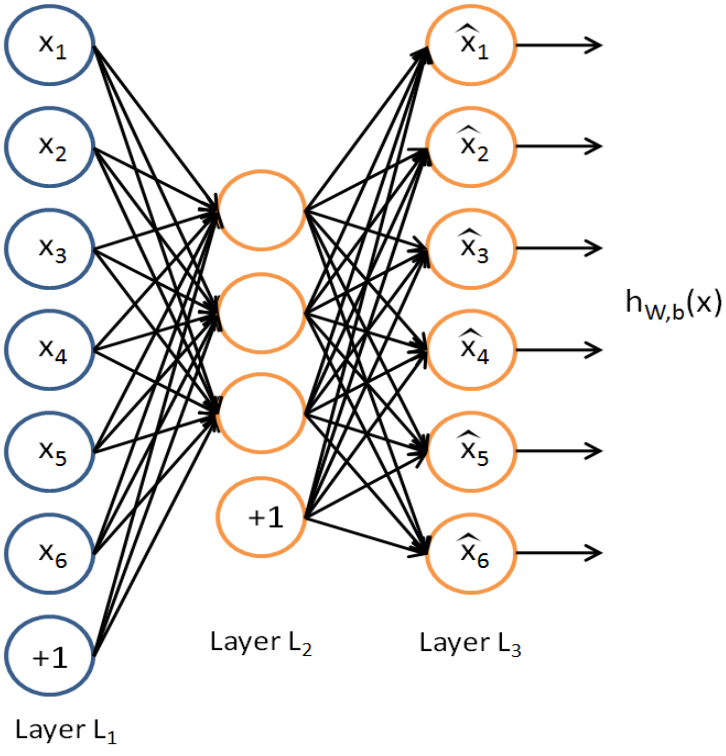


Figura 2.2: Ejemplo de un autoencoder. Figura obtenida de (Stanford, 2013a)

na estructura dentro de los datos, por ejemplo, si hay características correlacionadas en el dato de entrada, el algoritmo debería ser capaz de descubrir algunas de estas correlaciones. De hecho, un autoencoder simple muchas veces termina aprendiendo una representación con menos dimensiones adecuadas.

Red recurrente

Una limitación clara de las redes neuronales pre-alimentadas, como el perceptrón multicapa o las redes convolucionales son sus restriccio-

nes con respecto al tamaño fijo de los datos de entrada y salida. No solo esto, sino que también para producir los datos de salida tienen una cantidad fija de pasos computacionales.

Las redes recurrentes nos liberan de esta limitación permitiéndonos trabajar con secuencias como datos de entrada, donde cada secuencia esta conformada por una cantidad no fija de vectores. Más aún, los datos de salida también pueden estar conformados por secuencias.

Para lograr esto, las redes recurrentes son redes con bucles en ellas, permitiendo que "recuerden" lo aprendido por los datos de entrada anteriores en una misma secuencia mientras generan los datos de salida.

En detalle, una red neuronal recurrente toma un dato de entrada x y retorna un dato de salida y. Sin embargo, y no esta solamente influenciado por el dato de entrada x, sino que también por toda la historia de datos de entrada alimentados en el pasado.

Por la naturaleza de las redes recurrentes, estas pueden utilizarse para el modelado de secuencias, de manera similar a los CRF, al utilizar la información de cada instante de tiempo para representar una salida y entrenar la red en base a ejemplos que tengan varias salidas en simultáneo, representando una secuencia.

Un ejemplo del uso de la red recurrente para el etiquetado de secuencias puede observarse en la Figura 2.3, donde los rectángulos rojos representan los datos de entradas, los azules las salidas (etiquetas) y los verdes son los estados internos de la red. Revisar que los rectángulos verdes tienen salidas hacia los azules (i.e. las etiquetas) y hacia la representación interna del valor siguiente en la secuencia.

En esta tesis se experimenta a modo de baseline con un modelo neuronal recurrente, basado en lo que llamamos LSTM ("Long short-term memory" en ingles) (Hochreiter and Schmidhuber, 1997). La red LSTM es un tipo particular de red recurrente que modela mejor la representación de datos recurrentes mediante formas de "olvidar" información innecesaria. Más aún, se toma un modelo "bidireccional" donde las conexiones recurrentes van de izquierda a derecha y de derecha a izquierda, utilizando el contexto hacia ambos lados.

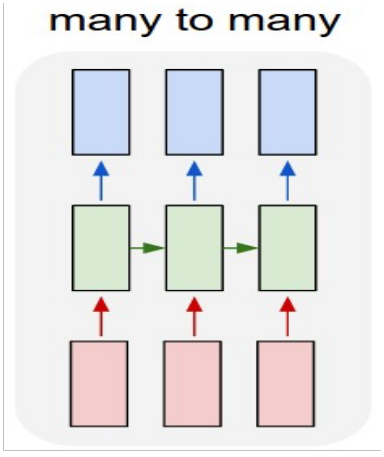


Figura 2.3: Ejemplo de una red neuronal recurrente. Obtenida de (Karpathy, 2015)

Redes en escalera

Las redes neuronales en escalera (Rasmus et al., 2015) son un modelo de aprendizaje profundo semi-supervisado, el cual utiliza una arquitectura neuronal entrenada a partir de datos anotados y no anotados.

Las redes combinan los datos tratando de optimizar una *función objetivo* en común, que busca reducir un error de regresión o clasificación (para los datos anotados) y un error de reconstrucción (para los datos no anotados).

Como en cualquier red neuronal, la optimización se lleva a cabo mediante el uso de retropropagación y algún algoritmo de optimización por gradientes. En el trabajo original, el modelo fue probado en una tarea de visión por computadoras, pero la arquitectura era lo suficientemente general como para ser aplicada en otras áreas, en particular en tareas de procesamiento de lenguaje natural como es el caso de esta tesis.

El concepto clave detrás de la construcción de una red en escale-

ra es tomar una red neuronal pre-alimentada, conocida generalmente en la literatura en inglés como, *feed-forward neural network* (como un perceptrón multicapa o una red convolucional), y tratarla como la parte de codificación de un autoencoder. Luego agregar el decodificador y usar el error de reconstrucción calculado capa por capa.

Los datos etiquetados son usados para minimizar el error dado por el encoder y una función de costo etiquetada. Los datos no etiquetados atraviesan completamente el autoencoder y el error de reconstrucción es minimizado. La función de costo de la red neuronal es la suma de las funciones de costo etiquetada y no etiquetada.

Autoencoders apilados (Vincent et al., 2010) fueron una idea clave para ayudar a entrenar las redes neuronales profundas. Usarlas para un pre-entrenamiento no supervisado ayudó a las arquitecturas neuronales profundas a converger a una solución de manera más rápida, evitando el problema del gradiente desvaneciente (o *vanishing gradient* en inglés) (Bengio et al., 1994).

Las redes en escalera se inspiran en esa idea, pero en vez de hacer el pre-entrenamiento de las capas en la red en un paso previo, se hace durante el entrenamiento de la red agregando el valor de la función de costo no supervisada (del autoencoder) a la función de costo de la red neuronal.

Capítulo 3

Metodología

Este capítulo se estructura de la siguiente manera: en la Sección 3.1 se detalla la arquitectura trabajada de la red en escalera y su implementación para el problema de reconocimiento y clasificación de entidades nombradas. La Sección 3.2 hace una descripción detallada del recurso principal utilizado para los experimentos realizados en esta tesis. Finalmente, la Sección 3.3 establece la lista de experimentos explorados y diversas hipótesis que buscaron evaluarse con dichos experimentos.

3.1. Red en escalera

En esta sección se explicará detalladamente el modelo de la red neuronal en escalera. Esto es tan solo una breve introducción del trabajo presentado por Rasmus et al. (Rasmus et al., 2015). Se recomienda al lector revisar su trabajo, el cual presenta una versión más detallada de lo que se describe aquí.

3.1.1. Arquitectura

La arquitectura de una red en escalera está basada en un *autoencoder* donde su parte *encoder* también trabaja como un clasificador

supervisado y su parte *decoder* trabaja como un aprendedor no supervisado reconstruyendo los datos de entrada.

La estructura de la red en escalera sigue los siguientes pasos:

1. Establecer un encoder el cual trabaja como un clasificador supervisado utilizando un modelo neuronal pre-alimentado (feed-forward). La red tiene dos canales del encoder –puro y corrupto. La única diferencia entre ellos es que el canal corrupto agrega ruido Gausiano a todas las capas. Agregar este ruido sirve para evitar el sobre ajuste del modelo resultante.

2. Establecer un decoder, el cual trabaja como un aprendedor no supervisado invirtiendo la dirección de cada capa del encoder. El decoder utiliza una función de eliminación de ruido para reconstruir las activaciones de cada capa dada la versión corrupta. El objetivo de cada capa es la versión pura de la activación y la diferencia entre la reconstrucción y la versión pura sirve como costo de eliminación de ruido de esa capa.

3. El costo supervisado, es decir, el error entre la etiqueta predicha y la etiqueta verdadera, es calculado desde la salida del encoder corrupto y la etiqueta objetivo. Por otro lado, el costo no supervisado es la suma de los costos de eliminación de ruido de cada capa escaladas por un hiper parámetro que denota la importancia de cada capa. Por ejemplo, las primeras capas son más importantes que la ultima para reconstruir el dato de entrada. El costo final es la suma del costo supervisado y no supervisado.

La red completa es entrenada utilizando técnicas estándares para tareas supervisadas o semi-supervisadas (tal como descenso del gradiente estocástico) para minimizar estos costos. Hay que tener en cuenta que la red en escalera puede funcionar incluso sin datos no etiquetados auxiliares, pero la motivación original era tomar clasificadores pre-alimentados (feed-forward) e incrementarlos con un encoder auxiliar.

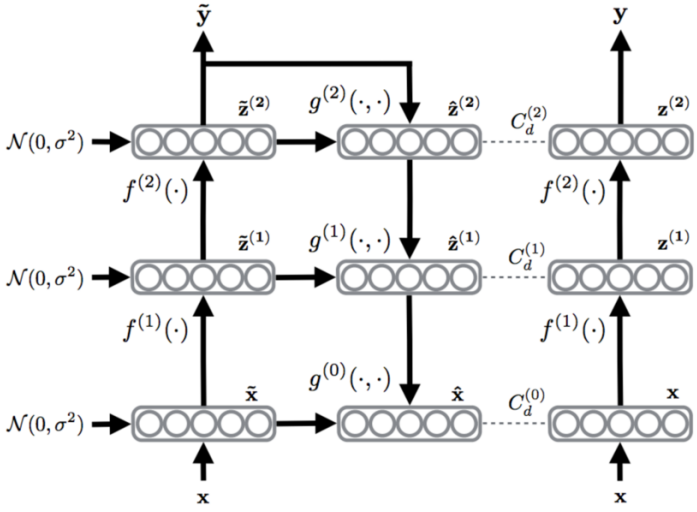


Figura 3.1: Una ilustración conceptual de una red en escalera con dos capas internas ($L = 2$). La componente supervisada ($\mathbf{x} \rightarrow \mathbf{z}^{(1)} \rightarrow \mathbf{z}^{(2)} \rightarrow \mathbf{y}$) comparte los pesos $f^{(l)}$ con la componente corrupta, i.e. el encoder ($\mathbf{x} \rightarrow \tilde{\mathbf{z}}^{(1)} \rightarrow \tilde{\mathbf{z}}^{(2)} \rightarrow \tilde{\mathbf{y}}$). El decoder ($\hat{\mathbf{z}}^{(2)} \rightarrow \hat{\mathbf{z}}^{(1)} \rightarrow \hat{\mathbf{x}}$) consiste de las funciones de eliminación de ruido $g^{(l)}$ y las funciones de costo $C_d^{(l)}$, en cada capa, orientadas a minimizar la diferencia entre $\hat{\mathbf{z}}^{(l)}$ y $\mathbf{z}^{(l)}$. La salida $\tilde{\mathbf{y}}$ del encoder puede ser también entrenada para que coincida con etiquetas disponibles $t(n)$. Figura original extraída del trabajo de Rasmus et al. (Rasmus et al., 2015).

La Figura 3.1 muestra la arquitectura de la red en escalera. El canal corrupto a la izquierda de la figura, agrega ruido Gaussiano $\mathcal{N}(0, \sigma^2)$ a cada capa del encoder.

Cada capa contribuye a la función de costo, el término $C^{(l)} = ||\mathbf{z}^{(l)} - \hat{\mathbf{z}}^{(l)}||^2$ el cual entrena las capas de arriba (del encoder y decoder) para aprender la función de eliminación de ruido $\hat{\mathbf{z}}^{(l)} = g^{(l)}(\tilde{\mathbf{z}}^{(l)}, \hat{\mathbf{z}}^{(\mathbf{l+1})})$ la cual dirige el canal corrupto $\tilde{\mathbf{z}}^{(l)}$ al resultado estimado.

Como el estimado $\hat{\mathbf{z}}^{(l)}$ incorpora todo el conocimiento previo sobre $\mathbf{z}$, el mismo término de la función de costo también entrena las capas

de abajo del encoder para encontrar atributos mejores que igualen la expectativa anterior con mayor certeza.

Dado que la función de costo necesita del término puro $\mathbf{z}^{(l)}$ y el corrupto $\tilde{\mathbf{z}}^{(l)}$, durante el entrenamiento, el encoder es ejecutado dos veces: una corrida limpia para $\mathbf{z}^{(l)}$ y una corrida corrupta para $\tilde{\mathbf{z}}^{(l)}$.

En los DAE (*denoising autoencoders* o "autoencoders que quitan ruido") (Vincent et al., 2010), un autoencoder es entrenado para reconstruir la observación original $\mathbf{x}$ dado una versión corrupta $\tilde{\mathbf{x}}$. El aprendizaje esta basado en la minimización de la norma de la diferencia entre el $\mathbf{x}$ original y $\tilde{\mathbf{x}}$ corrupto, esto significa que el costo es $||\hat{\mathbf{x}} - \mathbf{x}||^2$.

La principal diferencia con las redes en escalera es que el costo de reconstrucción es calculado capa por capa agregando las funciones de eliminación de ruido $\hat{\mathbf{z}} = g(\mathbf{z})$.

Considerando los DAE, podemos ver a la arquitectura de la Red en Escalera como una colección de DAEs anidados, los cuales comparten entre si partes de de la eliminación de ruido a través de la función de costo C_d. Si esta función no fuera utilizada, desde el punto de vista del autoencoder en la capa l, las representaciones de las capas superiores serían opacas, tratándose así como neuronas escondidas. En otras palabras, no habría ninguna razón en particular en por qué la representación intermedia $\hat{\mathbf{z}}^{(l+i)}$ producida por el decoder debería ser similar a las representaciones correspondientes $\mathbf{z}^{(l+i)}$ producidas por el encoder. La función de costo $C_d^{(l+i)}$ es lo que une esto y además fuerza la inferencia a proceder en orden inverso en el decoder. Esto ayuda a que un denoising autoenconder aprenda el proceso de eliminación de ruido mientras separa la tarea en producir representaciones intermedias.

Se aplica normalización por grupos (*batch normalization*) (Ioffe and Szegedy, 2015) a cada pre-activación, incluyendo la capa superior para mejorar la convergencia y prevenir que el costo de la eliminación de ruido fomente la solución trivial (el encoder genere valores constantes dado que son los mas fáciles de eliminar el ruido). Se utilizan conexiones directas entre una capa y su reconstrucción dada por el de-

coder. La red se llama "Red en Escalera" debido a que la arquitectura resultante es similar a una escalera, esto ya que la función de costo entre las capas espejadas se podrían ver como sogas en los escalones de una escalera.

3.1.2. Etiquetado de secuencias para redes en escalera

Como parte de este trabajo, se decidió explorar en la experimentación una modificación de la red en escalera, aplicando un modelo de etiquetado de secuencias.

Para ello se exploró la implementación de un modelo de un perceptrón multicapa que, de acuerdo a las dimensiones, representara una secuencia de datos de entrada y una secuencia de datos de salida.

Dado que el perceptrón multicapa acepta datos de entrada con tamaños fijos y las secuencias claramente no lo son, se definió como tamaño de los datos de entrada a uno fijo, que representa el máximo de todas las secuencias. Luego para cada secuencia si posee un tamaño menor al establecido, se realiza un rellenado de la misma con valores previamente definidos como no relevantes para red.

Recordemos que la salida de un modelo de perceptrón multicapa estándar suele ser una predicción de una categoría, es por ello que se debe modificar la salida del mismo para que prediga instancia por instancia para cada categoría. Se decidió tomar como salida entonces a la cantidad de categorías repetidas la cantidad de instancias en la secuencia. Es decir, si tenemos una secuencia de 50 instancias y 6 categorías, la salida sera de 300 neuronas, donde las primeras 50 neuronas representan la predicción de la primera clase para cada instancia de la secuencia y así sucesivamente.

Luego, a partir de este modelo del perceptrón multicapa "secuencial" se modificó la red en escalera de manera que los datos de entrada pudieran representar una secuencia y los datos de salida igual.

3.2. Corpus

Para los distintos experimentos que se realizaron a lo largo de este trabajo se utilizó el mismo conjunto de datos, con distintos procesamientos de acuerdo al experimento dado. El corpus utilizado fue el procesado en (Cardellino et al., 2017), el cuál fue obtenido de la Wikipedia.

Wikipedia ha sido utilizado como corpus para el reconocimiento y clasificación de entidades nombradas ya que provee una cantidad considerable de texto donde hay ocurrencias naturales de distintas entidades, las cuales están manualmente etiquetadas o enlazadas a una ontología, por ejemplo, la ontología de DBpedia (Hahm et al., 2014).

De acuerdo a la tarea (etiquetado por palabras o etiquetado secuencial), las instancias de entrenamiento pueden bien ser palabras o secuencias (i.e. oraciones).

Del corpus original, se extrajeron todas las oraciones que contenían al menos una mención de alguna entidad. Esto dio un total de, aproximadamente, 100 millones de palabras que podían o no ser una entidad nombrada. Todas estas palabras formaban un aproximado de 10 millones de oraciones.

Se particionó la totalidad de este corpus en 3 partes, donde 6 millones de oraciones con un aproximado de 60 millones de palabras conforman el corpus de entrenamiento, y 4 millones de oraciones de aproximadamente 40 millones de palabras conforman el corpus de evaluación y ajuste (2 millones de oraciones o 20 millones de palabras para cada uno).

Las categorías utilizadas para las entidades nombradas fueron 6: Person, Organization, Document, Abstraction, Act, Other.

El corpus fue anotado utilizando el formato IOB, el cual determina que a cada entidad multipalabra se le asigna su categoría correspondiente a todas las palabras, indicando cual es el comienzo de la misma (con la letra B) y su continuación (con la letra I). Esto es para separar correctamente dos entidades distintas que tengan la misma categoría.

Hastert	NNP	O
was	VBD	O
a	DT	O
supporter	NN	O
of	IN	O
the	DT	O
North	JJ	B-document
American	JJ	I-document
Free	NNP	I-document
Trade	NNP	I-document
Agreement	NNP	I-document

Cuadro 3.1: Ejemplo de oración en el corpus

El Cuadro 3.1 muestra un extracto del corpus donde se ejemplifica cómo es representado utilizando el formato IOB. La segunda columna representa la "categoría gramatical" ("part-of-speech tag" en inglés) de la palabra en cuestión.

Esta información de la categoría gramatical fue relevante para el entrenamiento del modelo Stanford NER-CRF, el cual es uno de los baselines descriptos en la Sección 3.3.1.

Se limitaron las oraciones a un tamaño máximo de 50 palabras, de lo contrario muchas se volvían imposibles de trabajar con las limitaciones de los clasificadores utilizados.

3.3. Experimentación

Esta tesis explora el uso de redes en escalera en la tarea de reconocimiento y clasificación de entidades nombradas en textos legales (NERC). Nuestra principal hipótesis, es que la red en escalera, y su naturaleza semi-supervisada ayuda en el proceso de NERC. Para ello, se realizaron varios experimentos donde se puso a prueba dicha hipótesis. A medida que los experimentos fueron desarrollándose, nos encontra-

mos con ciertas faltas sobre la hipótesis original, que nos obligaron a revisar el trabajo en busca de nuevas hipótesis.

Antes de tratar con los experimentos en sí, cabe destacar que se debieron hacer ciertas modificaciones y adaptaciones a la implementación de la red en escalera dada por (Rasmus et al., 2015).

Esto se debe a que la red fue propuesta originalmente para trabajar con imágenes. Sin embargo, como la arquitectura era suficientemente genérica e independiente del dominio, se la adaptó para el trabajo con texto.

Para el comienzo de la experimentación decidimos utilizar como parte etiquetada a una fracción de las entidades totales. Esto se debe a que dado los resultados vistos en (Rasmus et al., 2015) se obtienen buenos resultados con pocos datos etiquetados.

3.3.1. Baselines

De los baselines propuestos, dos son parte del estado del arte (*state-of-the-art*) que hoy en día existe para la tarea de NERC, el tercero fue un baseline más básico que nos sirvió como una comparación directa con los resultados obtenidos mediante las redes en escalera.

Si bien la tarea de superar el estado del arte es una tarea difícil y que nosotros no esperábamos lograr, al comparar ciertas métricas (además del desempeño final) pudimos observar que las redes en escalera tienen una tendencia interesante en lo que se conoce como la producción de modelos más generales.

Sin embargo, es necesario tener una referencia con modelos del estado del arte, para ver efectivamente que tan lejos están nuestros resultados de dichos modelos.

Los modelos explorados son: el clasificador de Stanford NER-CRF (Finkel et al., 2005), una red neuronal recurrente BiLSTM (Ma and Hovy, 2016), y un perceptrón multicapa puramente supervisado.

Experimento 1.1: Stanford NER-CRF

El primer conjunto de experimentos fue realizado sobre el modelo de Stanford NER-CRF, que aplica un modelo secuencial utilizando los CRFs sobre atributos manuales.

Este modelo se intentó entrenar utilizando originalmente los hiperparámetros por defecto propuestos en (Finkel et al., 2005). Sin embargo, por la naturaleza dispersa de los atributos (en contraste con representaciones mediante embeddings), este modelo era muy costoso de entrenar, dado que exigía de mucha memoria.

Como primera aproximación, de los hiperparámetros originales del modelo, se decidió, siguiendo indicaciones de los mismos desarrolladores de la librería, modificar dos hiperparámetros: la cantidad máxima de n-gramas de caractéres (que se redujo de 6 a 5) y el tamaño máximo de iteraciones para optimización (que se redujo de 25 a 5).

No obstante, aún con estos cambios, el modelo seguía siendo demasiado exigente como para ser entrenado con la totalidad del corpus, por lo que se decidió hacer 3 iteraciones utilizando partes del corpus: 10 %, 20 % y 50 % del total de oraciones del corpus.

Los resultados de este experimento, junto a su análisis, se pueden observar en la sección **??**.

Experimento 1.2: Perceptrón multicapa supervisado

El segundo modelo utilizado para comparar con la red en escalera fue un perceptrón multicapa supervisado, el cual como fue detallado en la sección 2.2.3, es un modelo neuronal, por lo que sus datos de entrenamiento son los mismos que para la red en escalera. Cada instancia de estos datos de entrenamiento representa a una palabra junto a su ventana de palabras.

Esto se logró mediante una representación en embeddings utilizando la técnica de reducción exponencial, la cual se detalla en la sección 1.4.

Este método no es un estado del arte, sin embargo es el método

más directo que tenemos para observar el impacto de la componente no supervisada en el resultado final.

Experimento 1.3: Red recurrente LSTM bidireccional

Para el ultimo conjunto de experimentos, que evalúaban la adaptación de la red en escalera utilizando etiquetado de secuencia, se entrenó una red recurrente LSTM bidireccional (Ma and Hovy, 2016), que al igual que el modelo de 1.1, aplica un modelo secuencial.

Esto implica que, como se detalla en la sección 2.2.1, cada instancia de entrenamiento para el modelo representa una oración del conjunto de datos entrenamiento.

A manera de simplificar la experimentación y la complejidad del modelo, el mismo se baso en el propuesto por (Ma and Hovy, 2016), utilizando únicamente las componentes de BiLSTM.

Con respecto a su estructura interna, i.e. cantidad de capas y neuronas, se utilizó la misma configuración que para la red en escalera secuencial, la cual es detallada en la sección 4.

Los resultados de este experimento, junto a su análisis, se pueden observar en la sección 4.4.

Experimento 1.4: Representación vectorial de las entidades

Para poder entrenar los modelos basados en redes neuronales utilizando este corpus (particularmente para el caso del perceptrón multicapa), necesitamos primero tener una representación vectorial del mismo que pueda ser utilizada por el modelo.

Para ello decidimos utilizar vectores densos de palabras y la representación dada por (Iacobacci et al., 2016) la cual fue descripta previamente en el Capítulo 2.

Trabajamos con 3 distintas representaciones propuestas por (Iacobacci et al., 2016): promedio, reducción fraccionaria y reducción exponencial.

Para simplificar el proceso y no tener que derivar en demasiados experimentos que le quitarían el foco al estudio de las redes en escalera, se decidió experimentar de manera rápida y superficial cual era el método de representación mas efectivo con nuestros datos.

De esto surge que el método de la reducción exponencial nos provee mejores resultados, lo que es esperable puesto que las entidades nombradas dependen mucho más de las palabras ubicadas en la ventana cercana, que de aquellas que están más alejadas.

El resto de la experimentación se trabajó sobre la base de estos resultados y sólo se exploró el uso de la representación con reducción exponencial.

3.3.2. Experimento 2: Red en escalera

Este es el primer experimento sobre el cuál se trabajó con las redes en escalera. Como primera aproximación a la experimentación con los datos y el modelo, se utilizaron directamente los hiperparámetros propuestos por (Rasmus et al., 2015).

El modelo se entrenó con una fracción aleatoria de 1 millón de instancias anotadas y 3 millones de instancias no anotadas, las cuáles no incluían el millón de instancias anotadas. Se considera en este punto a una instancia como una palabra, su ventana (representada mediante una reducción exponencial) y la etiqueta de la entidad correspondiente.

Los resultados pueden observarse en **??**. A partir de estos resultados se decidió avanzar con el experimento 2.1 para lograr mejorar el desempeño obtenido en esta instancia de experimentación.

Experimento 2.1: Búsqueda de hiperparámetros

Dados los resultados del experimento anterior, que mostraban muy bajo desempeño del modelo y una no convergencia de las funciones de costo a lo largo de las iteraciones (tanto el costo supervisado como el no supervisado), se decidió tomar un dataset aleatorio pequeño de

2000 instancias, donde se tomaron 100 instancias como el fragmento supervisado de entrenamiento. Esta reducción en el tamaño del corpus fue para hacer más rápida la exploración de hiperparámetros, ya que se volvía muy pesado hacerlo sobre la totalidad de instancias original del experimento anterior.

A partir de este conjunto de datos de menor tamaño, se prosiguió a hacer una búsqueda de hiperparámetros aleatoria, para ver si de esa manera alguna configuración de hiperparámetros obtenía buenos resultados.

Con esta experimentación concluimos que los mejores hiper-parámetros seguían siendo los propuestos por Rasmus et al. (Rasmus et al., 2015) los cuales serán los utilizados en los próximos experimentos.

Experimento 2.2: Verificación del modelo

En este punto, y teniendo en cuenta que aún con exploración de hiperparámetros no se pudieron obtener mejores resultados, se estableció la hipótesis de que la implementación del modelo en escalera estuviese sufriendo algún tipo de falla que no le permitiera aprender de los datos o bien que el modelo no fuera lo suficientemente complejo como para representar los datos de entrada.

Para descartar esta posibilidad, se estableció primero un conjunto de experimentos utilizando sólo la parte supervisada del modelo para intentar sobreajustarla, con el objetivo de comprobar que efectivamente el modelo de red tuviese la capacidad de representar este problema.

Recordemos que la red en escalera está compuesta por dos encoders y un decoder, donde un encoder entrena una función de coste supervisada. En este experimento modificamos la función de coste total de la red para que solo sea esta función de coste supervisada la que se busca optimizar. En otras palabras, se busca que solo se entrene la parte supervisada de la red como si fuera un perceptrón multicapa simple.

Los resultados de este experimento pueden observarse en la Sección

4.2.1 del Capítulo siguiente. Con estos resultados se pudo verificar que la componente supervisada de la red efectivamente puede sobreajustar los datos de la manera esperada, por lo que se descarta la hipótesis de que la implementación del modelo tenga alguna falla en en este aspecto o bien sea muy poco expresiva como para no poder representar el modelado de los datos anotados.

Experimento 2.3: Visualización de las entidades

Llegados a este punto, y a partir de los experimentos anteriores, surge la hipótesis de que los datos no anotados están agregando demasiado ruido al modelo.

Decidimos avanzar en una exploración visual de los datos con el objetivo de poder observar algún tipo de patrón que permitiera explicar la causa de la no convergencia del modelo semi-supervisado.

Para dicha exploración, se utilizo el algoritmo de t-SNE (Maaten and Hinton, 2008) para representar, gráficamente, una porción aleatoria de 20000 instancias balanceadas entre las 6 clases (i.e. incluyendo la clase "O").

t-Distributed Stochastic Neighbor Embedding (t-SNE) es una técnica de reducción de dimensionalidad particularmente adecuada para la visualización de conjuntos de datos de alta dimensión ya que le asigna a cada instancia del conjunto de datos una posición en un espacio de 3 dimensiones.

Como vimos en los resultados anteriores, el principal problema que tenemos es que la función de costo no supervisado no esta minimizando como quisiéramos. El objetivo de realizar estas gráficas es observar la distribución de las entidades con respecto a cada categoría y así ver si la estructura de las mismas ayuda al algoritmo con la parte no supervisada. Esto puede observarse en la Sección 4.2.2 del Capítulo siguiente.

Experimento 2.4: Balanceo de clases

En el corpus original, un problema muy pesado es el gran desbalanceo de clases que sufre. Esto es así porque la clase mayoritaria "O" (i.e. las palabras que no son una entidad) representan más del 90 % de las instancias del corpus.

Con el objetivo de mejorar los resultados de los experimentos anteriores, se hizo un muestreo aleatorio de instancias etiquetadas como "O", reduciéndolas a la misma cantidad de instancias de las categorías restantes, de manera tal que las instancias de clase "O" ocuparan el 50 % de las instancias totales.

A su vez se tomó el corpus con el que se venía trabajando y se le supramuestrearon aquellas clases con menor cantidad de instancias duplicándolas aleatoriamente, con el objetivo de incrementar la cantidad de datos etiquetados para clases minoritarias, llevándolos a un total de aproximadamente 2000 instancias anotadas para clases no mayoritarias, que se le dan a la parte supervisada del modelo de red en escalera.

Este incremento se hizo sólo sobre los datos etiquetados, dejando así la cantidad de datos no anotados igual que en los experimentos anteriores.

Los resultados de este experimento pueden observarse en la Sección 4.2.3 del Capítulo siguiente donde se observa una mejora sustancial del modelo original.

3.3.3. Experimento 3: Red en escalera para mejor generalización

Dado que la red en escalera basada en el perceptrón multicapa no parecía presentar resultados favorables (sobre todo en comparación al "estado del arte") en la tarea del reconocimiento de entidades nombradas, a pesar de las distintas configuraciones probadas, se decidió explorar si la componente no supervisada de la misma aportaba una mejor generalización en comparación con un modelo puramente

supervisado.

Esto es, en términos más específicos, observar si las distintas métricas a medir como la exactitud (accuracy), precisión, exhaustividad (recall) y el puntaje F1 tienen un desempeño similar o mejor a la hora probar el modelo sobre los datos de evaluación en comparación con los resultados que se obtienen de aplicar el modelo a datos del conjunto de entrenamiento.

Esto nos indicaría que la red está logrando una generalización mejor de los datos ya que obtiene un rendimiento similar ante entidades desconocidas para la misma.

Para llevar a cabo este experimento se procedió a generar un nuevo conjunto de datos de entrenamiento balanceando todas las categorías. El mismo consiste en un dataset pequeño, de 120.000 instancias de entrenamiento, distribuido en 20.000 por clase.

Dado que la categoría *act* no presentaba suficientes instancias (20.000), se decidió realizar un supramuestreo aleatorio de instancias de entidades de esta categoría para así completar la cantidad requerida.

Con este nuevo dataset se reentrenó la red en escalera, manteniendo los mismos hiper parámetros de los experimentos anteriores.

Para comprobar si generaliza mejor, se decidió entrenar la red en escalera incrementando la cantidad de datos no anotados como datos de entrada. Con esto se espera ver que en los casos donde haya mayor cantidad de datos no anotados haya una mejor generalización, y a su vez, comprobar el punto en donde suficientes datos no anotados empieza a influir en la no convergencia del modelo.

Para ello se entrenó primero con 20.000 instancias no anotadas, luego con 80.000 y finalmente con 120.000 instancias.

Como datos etiquetados siempre se utilizó la misma cantidad, 120.000 instancias, utilizando el dataset mencionado previamente. Este mismo es el que se utilizó para el entrenamiento del perceptrón multicapa.

3.3.4. Experimento 4: Red en escalera para datos secuenciales

Finalmente, en estos experimentos se trabajó con redes en escalera adaptadas a un modelo de etiquetado secuencial.

Esperamos ver si con el cambio del modelo de experimentación obtenemos mejores resultados para la red en escalera, así como también comprobar que la generalización mejora, al igual que en el experimento 3, comparando con la red recurrente BiLSTM, que es más estándar.

Para estos modelos, una instancia es representada por una secuencia de palabras, en lugar de la representación vectorial de una sola palabra.

Dada la naturaleza de los datos secuenciales, para esta serie de experimentos no se podía hacer un submuestreo de la clase mayoritaria o un supramuestreo de las clases minoritarias, por lo que los modelos tuvieron que ser explorados con los datos originales. Esto impide realizar un submuestreo o supramuestreo sobre alguna clase en particular sin modificar todas las otras instancias (de otras clases) que aparecen en la misma secuencia. Como consecuencia de esto, tendremos implícitamente modelos entrenados con un sesgo importante hacia la clase mayoritaria. No obstante, era interesante aún así explorar estos experimentos con el simple objetivo de ver si tenían sentido. Esto, sin embargo, marcará un desempeño muy bajo para el caso particular de la clase *act* que de por si posee pocos ejemplos anotados en el corpus original.

Experimento 4.1: Exploración de costos de reducción de ruido

Previo a la experimentación directa sobre la red en escalera secuencial (y su correspondiente comparación con la red recurrente BiLSTM), se realizaron algunos experimentos de ajustes de hiperparámetros. En particular, se prosiguió a experimentar más de cerca sobre distintos valores de importancia para las funciones de coste de las capas no

supervisadas del modelo (que en el trabajo original de Rasmus et. al. se denotan como "denoising cost").

Este hiperparámetro denota la importancia de la reconstrucción de cada capa en el proceso de decodificación de la componente no supervisada. En el trabajo original se utilizan valores altos para las capas más cercanas a la entrada/salida (recordemos que el modelo del autoencoder es simétrico) y valores más bajos en capas más "internas" al modelo (que son representaciones más abstractas y "elevadas" de los datos).

La sección 4.4.1 muestra los resultados de esta exploración de hiperparámetros y los valores finales utilizados para los otros subexperimentos del Experimento 4.

Experimento 4.2: Parámetro de costo no supervisado

Luego de seleccionar los mejores parámetros vistos anteriormente y analizar los resultados dados se decidió agregar un nuevo parámetro μ, el cual es utilizado como peso para el costo no supervisado en la red en escalera.

Esto quiere decir que el parámetro especifica la importancia de la componente no supervisada en el esquema completo del modelo.

La idea del experimento es verificar si cambiando el peso de este costo se obtienen mejores resultados y encontrar el mejor valor del mismo.

La componente no supervisada de la red en escalera pasaría a ser un término de regularización, similar a la regularización L1 o L2, pero con información inherente a la representación de los datos no supervisados del modelo.

Para evaluar esto se realizaron entrenamientos de la red alterando el valor de μ donde los valores estaban entre 0 y 1.

0 implica que la componente no supervisada no influye en el aprendizaje de la red mientras que 1 equivale a la función de costo propuesta originalmente por Rasmus.

Si bien se trabajaron con varios valores, no hubo realmente una

diferencia particularmente importante entre los distintos valores para el hiperparámetro μ. El valor final elegido para μ fue de 0,4 que obtenía ligeramente mejores resultados que los otros valores observados.

Experimento 4.3: Red en escalera para etiquetado secuencial

Este último es el subexperimento principal del Experimento 4, ya que evalúa específicamente la red en escalera con su modelos secuencial y lo compara en particular contra la red recurrente BiLSTM.

El experimento se realiza en pos de los resultados de los experimentos anteriores, habiendo definido los mejores hiperparámetros con ayuda de estos últimos.

Se evalúan distintas métricas y se establece su comparación contra los datos del modelo supervisado en la Sección 4.4.2.

Capítulo 4

Análisis de Resultados

A lo largo de este capitulo se detallaran los resultados obtenidos en los distintos experimentos descriptos en el Capítulo 3.

Junto a dichos resultados se hará un análisis y se extraerán conclusiones al respecto.

4.1. Resultados Experimento 1.1: Stanford NER-CRF

En esta sección se describirán los resultados obtenidos al entrenar el modelo Stanford NER-CRF que se describe en el Experimento 1.1 en la Sección 1.1.

Como fue descrito anteriormente, debido a la alta exigencia de recursos para el entrenamiento del modelo, el experimento se realizo en 3 partes, utilizando porciones pequeñas del conjunto de datos e incrementándolas iterativamente. La idea era ver hasta que punto se podía incrementar el modelo para acercarse lo más posible a los datos utilizados para los modelos trabajados posteriormente.

Para fines prácticos, si bien hubo diferencias, aunque escasas, entre los modelos entrenados con distinto tamaño, en esta sección sólo se muestran los resultados para aquél modelo entrenado con 50 % de los

Entidad	Precisión	Exhaustividad	Puntaje F1
organization	0.9050	0.9062	0.9056
person	0.9125	0.8920	0.9021
document	0.8998	0.8756	0.8875
abstraction	0.8799	0.8569	0.8682
act	0.7224	0.7477	0.7348
Totals	0.9047	0.9023	0.9035

Cuadro 4.1: Resultados obtenidos entrenando con un 50 % del conjunto de datos.

datos anotados. Más datos no pudieron utilizarse porque el clasificador no lo soportaba.

Los resultados, para tener de referencia, sobre el conjunto de datos de evaluación, pueden observarse en el Cuadro 4.1.

El modelo baseline que establece el Stanford, claramente nos deja una vara muy alta sobre la cuál buscamos mejorar. Los resultados del modelo, incluso utilizando 50 % de los datos anotados, son altos. Sin embargo, la principal limitación del Stanford, es que estos resultados también arrojan un techo sobre lo que se puede mejorar, puesto que la naturaleza dispersa del clasificador hace imposible utilizar toda la información disponible.

4.2. Resultados Experimento 2

Como ya se detalló en el Experimento 2, en la Sección 3.3.2 se entrenó la red en escalera propuesta por (Rasmus et al., 2015) sin ninguna modificación más que la adaptación a los datos de entrada y la capa de salida. Los resultados que se obtuvieron se pueden observar en el Cuadro 4.2.

Los resultados que observamos son en base al conjunto de datos de evaluación. Analizando estos datos vemos que obtenemos una pre-

Entidad	Precisión	Exhaustividad	Puntaje F1
other	0.71	0.43	0.53
organization	0.65	0.59	0.62
person	0.33	0.92	0.48
document	0.41	0.70	0.52
abstraction	0.15	0.79	0.26
act	0.13	0.19	0.15

Cuadro 4.2: Resultados obtenidos entrenando la red en escalera con todo el conjunto de datos. Se utilizaron 1 millón de instancias anotadas y 3 millones de instancias no anotadas.

cisión del 71 % para la categoría *O*, la cual a su vez es la categoría mayoritaria en el conjunto de datos. Recordemos que la precisión es una métrica que nos dice que proporción de identificaciones de entidades en esa categoría es correcta.

Esto nos indica que la red podría estar sesgada donde se inclina a clasificar como *O* a la mayoría de los datos de entrada. Continuando con el análisis de la precisión y teniendo en cuenta este argumento propuesto vemos que para la siguiente categoría *person* se obtiene un 65 %. Esta categoría de entidad es la segunda clase mayoritaria, por lo que vemos que se mantiene el mismo sesgo. Esto se da con todas las categorías a clasificar, donde la precisión disminuye a medida que la categoría tiene menos representación en la totalidad del conjunto de datos.

En comparación con los resultados obtenidos con el modelo Stanford NER-CRF en la Sección **??**, observamos que la red en escalera es mucho peor en su desempeño general que lo que muestra nuestro baseline. Esto nos obliga a formular preguntas para ver si el modelo es realmente malo o hay algún error o falla que no estamos viendo.

Para entender un poco más que es lo que sucede con la red se procedió a graficar los valores de las funciones de costo, supervisado y no supervisado por separado, a lo largo de las épocas de entrenamiento.

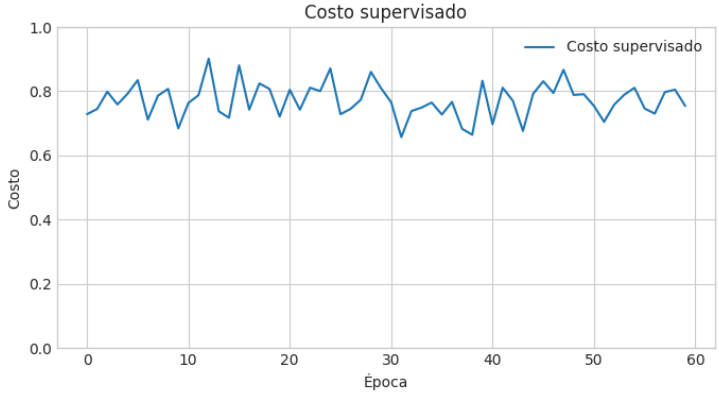


Figura 4.1: Costo supervisado en red en escalera entrenada con todo el conjunto de datos. Experimento 2

Figura 4.2: Costo no supervisado en red en escalera entrenada con todo el conjunto de datos.

Hubo resultados interesantes que se pueden observar en las Figuras 4.1 y 4.2. En estas, la línea representa el costo de la función (eje y) a lo largo de las épocas de entrenamiento (eje x).

En particular, podemos observar que para ambas funciones (su-

pervisada y no supervisada) hay un comportamiento muy errático del costo. Este no se reduce con las iteraciones, sino que simplemente salta de valor en valor de manera semi-aleatoria. Esto da lugar a pensar de que hay algo extraño ocurriendo en la red que hace que no pueda minimizarse. Se decidieron explorar distintos experimentos con el objetivo de ver que es lo que sucede.

4.2.1. Resultados Experimento 2.2

Dado que los resultados del experimento anterior no eran favorables y la exploración de hiperparámetros del Experimento 2.1 que se detalla en la Sección 2.1, no aportó nuevas mejoras, el Experimento 2.2, que se detalla en la Sección 2.2, busca verificar si la componente supervisada del modelo convergía o no (i.e. minimizaba), en pos de ver si el modelo efectivamente era muy "pequeño" para representar los datos o bien había alguna falla en la implementación del modelo.

Recordemos que la función de costo de la red en escalera es la suma del costo supervisado y el no supervisado, por lo que para entrenar únicamente la componente supervisada se modificó esta función para que sea igual al costo supervisado (e.g. multiplicando el costo no supervisado por 0).

Como vemos en la Figura 4.3, al minimizar sólo la función de costo supervisado efectivamente converge, y hasta logra sobreajustar el modelo. Esto nos sugiere que el modelo no tiene algún error en cuanto a su implementación, deduciendo así que la componente no supervisada estaría introduciendo demasiado ruido a la red, lo que estaría haciendo que esta no pueda adaptarse a los datos.

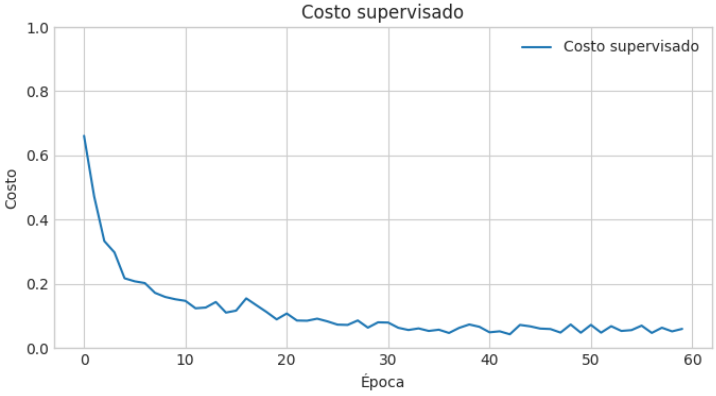


Figura 4.3: Costo de red en escalera entrenada optimizando sólo la componente supervisada a lo largo de las épocas.

4.2.2. Resultados experimento 2.3

Analizando los resultados obtenidos en el experimento anterior se procedió a explorar la naturaleza de los datos con los que se estaba trabajando.

Dado que la idea de la red en escalera es aprovechar la información proporcionada por los datos no etiquetados, se buscó comprobar si los datos muestran una distribución determinada en donde se pueda apreciar algún patrón de las entidades en el espacio.

En términos más concretos se busca comprobar si los datos divididos por las categorías definen algún esquema que pueda explotarse con la componente no supervisada del modelo de redes en escalera.

Para llevar a cabo este experimento se utilizó la técnica t-SNE (t-Distributed Stochastic Neighbor Embedding) (Maaten and Hinton, 2008) la cual fue descrita anteriormente en la descripción del Experimento 2.3 en el capítulo anterior.

Como vemos en el gráfico tenemos distintas categorías como *abstraction*, *document* o *person*, las cuales tienden a agruparse de forma más prominente, luego aquellas como *organization* que tiende a for-

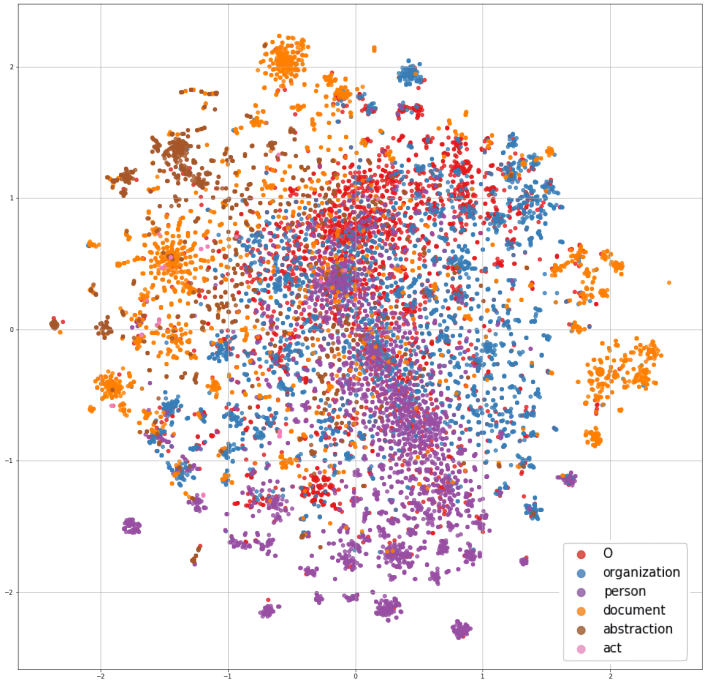


Figura 4.4: Distribución de instancias de entidades realizado con la técnica t-SNE.

mar grupos más chicos pero distribuidos a lo largo de todo el espacio y luego categorías como *act* y *O* las cuales se encuentran distribuidas a lo largo de todo el espacio sin distinción alguna.

El hecho de que algunas se agrupen, mientras que otras no, se debe principalmente al contexto de cada categoría. Esto puede explicarse en el caso de *document*, por ejemplo, que los documentos siempre ocurran bajo contextos muy similares, que los diferencia bastante de otras categorías.

Sin embargo, hay que recordar que t-SNE sólo muestra una fracción del espacio vectorial, lo que limita un poco toda la información que podemos utilizar. Sin embargo, es claro que hay ciertas clases que es fácil que se pierdan entre las demás. En particular, esto es verdadero

y algo complejo para el caso de la clase mayoritaria "O".

Esto mostraría que para algunas clases la parte no supervisada de la red puede ayudar bastante mientras que con otras no tanto. La mayoría de ellas no se encuentra distribuida de manera que pueda ayudar por lo que se puede suponer que en parte los malos resultados de los experimentos anteriores, donde se tenia una proporción de datos no supervisados mucho mayor a la de datos etiquetados, se deben a esto.

4.2.3. Resultados Experimento 2.4

Para el Experimento 2.4 se entrenó la red en escalera con un nuevo conjunto de datos, que se detallan en la Sección 2.4, con el objetivo de observar si el incremento de instancias anotadas y su mejor balanceo mejoraba el modelo.

Como vemos en el Cuadro 4.2.3 se obtiene una mejora en el modelo, aunque se mantiene el sesgo hacia las clases mayoritarias. Esto parecería indicar nuevamente que la componente supervisada de la red tiene mayor influencia en el modelo, donde si se incrementa la cantidad de instancias anotadas, se obtienen mejores resultados.

También el hecho de que las clases minoritarias tengan tan malos resultados indica que el corpus debe ser trabajado más aún, con el fin de lograr un balanceo equitativo.

4.3. Resultados de experimento 3

Una vez diagnosticado el problema que tenían las primeras experimentaciones sobre la red en escalera, se decidió, a través del Experimento 3, detallado en la Sección 3.3.3, hacer un análisis un poco más detallado de cómo las redes en escalera podían ayudar a encontrar modelos más generales.

Para analizar los resultados de este experimento se utilizaron distintas métricas como la *precisión*, la *exhaustividad* (o *recall*) y el *pun-*

Entidad	Precisión	Exhaustividad	Puntaje F1
other	0.90	0.76	0.82
organization	0.23	0.55	0.32
person	0.42	0.28	0.34
document	0.10	0.13	0.11
abstraction	0.0	0.0	0.0
act	0.0	0.0	0.0

Cuadro 4.3: Resultados entrenando la red en escalera estándar utilizando un conjunto de datos reducido y con 2000 instancias anotadas.

taje F1 (o *F1-score*) para cada categoría evaluados sobre el modelo final entrenado. Esto fue, como se detalla en la Sección 3.3.3, trabajado con distintas cantidades de datos no anotados: 20 mil, 80 mil y 120 mil. Se comprobó que el caso de 80 mil datos no anotados arrojaba los mejores resultados y, para facilitar la visualización sólo se graficarán los resultados del modelo de red en escalera utilizando 80 mil instancias no anotadas (las instancias anotadas quedan fijas).

Se compara el desempeño de los datos para el corpus de entrenamiento, validación y evaluación utilizando el *F1-Score* entre el modelo de red en escalera y sus pares supervisados (en este caso el Stanford NER-CRF y el perceptrón multicapa supervisado). Se busca observar la diferencia de desempeño entre los datos de entrenamiento y los datos de validación/evaluación, donde una diferencia menor implica la mejor generalización por parte del modelo.

La Figura 4.5 muestra los resultados del *F1-Score* por cada clase para los distintos conjuntos de datos. Es un gráfico de barras agrupadas. El eje de las y representa el valor de *F1-Score* (a mayor valor mejor desempeño), mientras que el eje de las x agrupa las distintas clases posibles. Por cada clase hay 3 barras de distintos colores, la barra azul representa el valor sobre el conjunto de datos de entrenamiento, la barra anaranjada representa el valor sobre el conjunto de datos de validación y finalmente la barra verde representa los valores

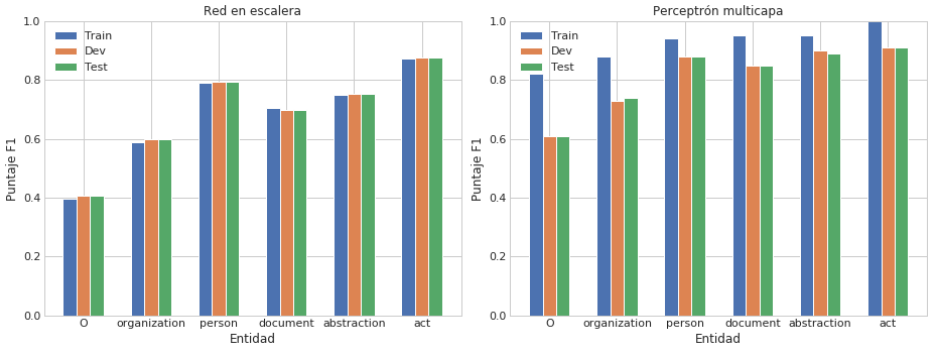


Figura 4.5: F1-Score por clase de la red en escalera con 80 mil instancias no anotadas vs. perceptrón multicapa. Comparación en distintos conjuntos de datos: entrenamiento, validación y evaluación.

sobre el conjunto de datos de evaluación. El gráfico de la izquierda representa los resultados para la red en escalera mientras que el gráfico de la derecha representa los resultados para el perceptrón multicapa.

Como vemos en la Figura, los valores de F1-Score varían según la entidad a reconocer, con valores que varían en el rango $(0{,}4, 0{,}85)$. En si mismo el objetivo de este experimento era observar si obtenemos una mejor generalización sobre los conjuntos de datos de validación y evaluación, lo cual vemos que efectivamente se obtienen valores muy similares entre si. Mas aún, comparando con lo resultados obtenidos en el perceptrón multicapa vemos que la diferencia entre el conjunto de datos entrenamiento y el de validación es mucho mayor a la de la red en escalera.

Esta comparación entre el desempeño del modelo en los distintos conjuntos de datos también puede observarse en la Figura 4.6, que análogamente al caso anterior, hace una comparación del *F1-Score* para cada una de las clases en los tres conjuntos de datos: entrenamiento, validación y evaluación. Debido a la alta exigencia de recursos en el momento de evaluar el modelo Stanford NER-CRF, solo se realizó dicha evaluación sobre el corpus de entrenamiento y test.

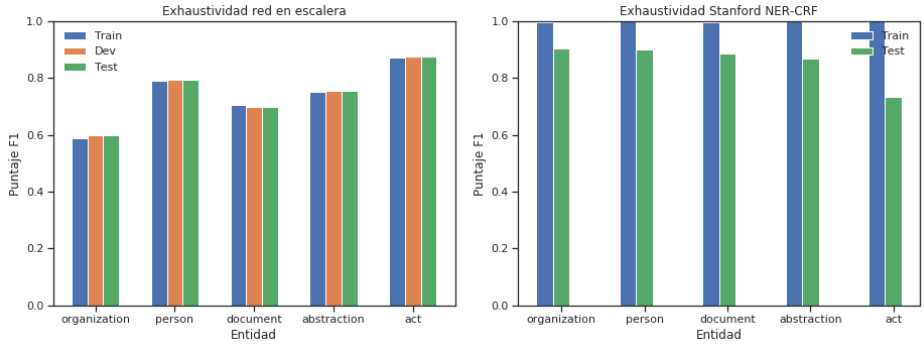


Figura 4.6: F1-Score en red en escalera con 80 mil instancias no anotadas y Stanford NER-CRF. Comparación con distintos conjuntos de datos: entrenamiento, validación y evaluación.

Otra comparación interesante para hacer, aunque en este caso sólo para el modelo de red en escalera vs. el modelo del perceptrón multicapa, es la evolución de la función de coste (en este caso, sólo la supervisada) a lo largo de las iteraciones tanto para el conjunto de datos de entrenamiento y el conjunto de datos de validación. Lo que se busca en sí es comparar las **curvas de aprendizaje** de entrenamiento y validación para ver si hay una divergencia entre los datos. Esto es una buena manera de evaluar el sobreajuste que está haciendo un modelo.

La Figura 4.7 muestra esto, donde vemos en el eje y el valor de la función de costo y en el eje x las épocas de entrenamiento. Téngase en cuenta que en este caso lo que se busca es minimizar la función de costo por lo que un valor menor en el eje y implica un mejor resultado.

Analizando la Figura 4.7 vemos que la función de costo supervisado tiende a minimizar mejor para el perceptrón multicapa aunque presentando una diferencia importante entre el corpus de entrenamiento y validación. Por otro lado en la red en escalera observamos que la función de costo para ambos conjuntos de datos se mantiene muy similar, indicándonos que el modelo presenta una mejor generalización.

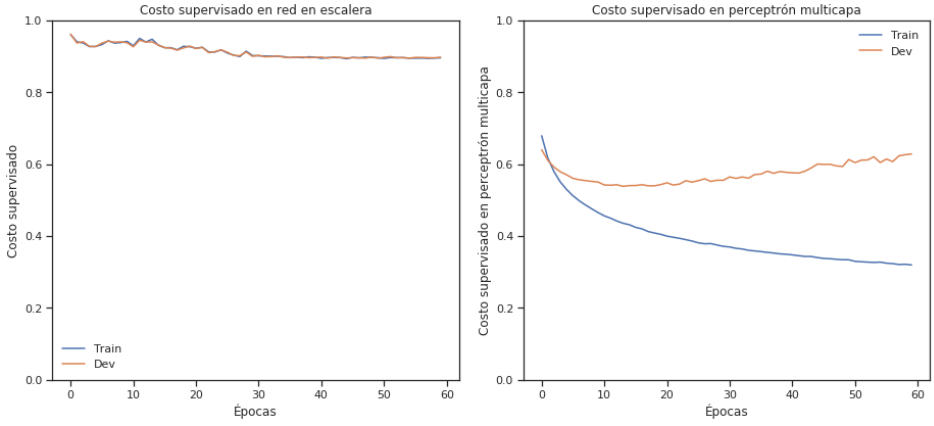


Figura 4.7: Función de coste supervisada en red en escalera con 80 mil instancias no supervisadas y perceptrón multicapa. Comparación con distintos conjuntos de datos: entrenamiento y validación.

4.4. Resultados de experimento 4

Dados los resultados del Experimento 3 que se analizaron en la Sección anterior, se decidió replicar el experimento adaptando la red en escalera (Rasmus et al., 2015) a un modelo de etiquetado secuencial, técnica mayormente usada y con mejores resultados en el ámbito del reconocimiento de entidades nombradas.

Se comparan los resultados con una red recurrente BiLSTM como se describe en la Sección 3.3.4, que es un modelo estándar a la hora de trabajar con datos secuenciales y se busca ver de esta manera como impactan los datos no anotados en el modelo final.

En particular, antes de la comparación de estos experimentos, que se da en la Sección 4.4.2, primero se trabajó con algunos

4.4.1. Resultados del experimento 4.1

Como fue detallado en la Sección 4.1, se decidió experimentar con distintos valores para los costos de reducción de ruido, con el fin de

obtener aquellos que sean más beneficiosos para la red.

Recordemos que los costos de reducción de ruido representan la importancia que se le da a cada capa del autoencoder en el momento de la reconstrucción de las activaciones, e influyen directamente sobre la función de coste no supervisada.

En contraste con los valores originales del trabajo de Rasmus, donde se tienen los siguientes costos de reducción de ruido: $1000, 10, 0{,}1, 0{,}01$, se observó que los valores altos, en las capas más cercanas a la entrada, que se tenían eran demasiados restrictivos y unos valores en ordenes de magnitud más pequeños arrojaban mejores resultados. Esto implica darle mayor importancia a la reconstrucción de capas más internas al modelo, que son las encargadas de las representaciones internas, que de manera progresivamente se vuelven más abstractas y capturan distintas relaciones entre los datos de entrada.

Los valores con los que se experimentaron fueron: $10, 1, 0{,}1, 0{,}01$ para cada capa respectivamente y $1, 0{,}1, 0{,}01, 0{,}01$. Finalmente los mejores resultados los obtuvimos con los valores de menor magnitud, $1, 0{,}1, 0{,}01, 0{,}01$, los cuales serán utilizados por el resto de los experimentos.

Entidad	F1-Score$_1$	F1-Score$_2$
other	0.77	0.77
organization	0.02	0.06
person	0.01	0.30
document	0.00	0.09
abstraction	0.0	0.24
act	0.0	0.00

Cuadro 4.4: F1-Score obtenidos para cada categoría al entrenar la red en escalera secuencial con distintos valores de reducción de ruido. F1-Score$_1$ representa los resultados obtenidos utilizando los valores de reducción de ruido del trabajo original. F1-Score$_2$ representa los mejores valores obtenidos: $1, 0{,}1, 0{,}01, 0{,}01$

En el Cuadro 4.4.1 observamos los resultados en F1-Score para los costos de reducción de ruido originales y aquellos que mejores resultados obtuvimos. Estos están denotados como F1-Score$_1$ y F1-Score$_2$ respectivamente.

Como observamos, los resultados son bastante mejores para los valores de costo de reducción de ruido explorados en este trabajo en comparación a los originales. Esto nos indica la importancia de las representaciones internas de la red a la hora de aprovechar los datos no anotados. Esto es así pues las capas más internas de la red pueden capturar relaciones más abstractas entre los datos de entrada, algo particularmente importante cuando se considera la gran cantidad de datos no anotados existentes en el modelo que pueden no necesariamente tener información relevante de manera superficial pero si en sucesivas representaciones más abstractas.

4.4.2. Resultados del experimento 4.3

Una vez obtenidos los mejores hiperparámetros se realizó el experimento entrenando la red en escalera secuencial y la red BiLSTM con el conjunto de datos previamente descrito para comparar sus resultados y el nivel de generalización.

La Figura 4.8 hace una comparación de los resultados del modelo de red secuencial en escalera y la red recurrente BiLSTM. Similar a la Sección 4.3, se muestran gráficos de barras agrupadas. El eje y representa el *F1-Score* mientras que el eje x representa cada una de las distintas clases trabajadas. Los colores de las barras representan los conjuntos de datos: azul es entrenamiento, anaranjado es validación y verde es evaluación. El gráfico de la izquierda representa el modelo de red en escalera secuencial mientras el gráfico de la derecha representa la red recurrente BiLSTM.

Observando el **??** que muestra los valores del F1-Score para la red en escalera secuencial y para BiLSTM, vemos que la red en escalera tiende a predecir bien únicamente la clase mayoritaria O, mientras que la red BiLSTM obtiene buenos resultados generales.

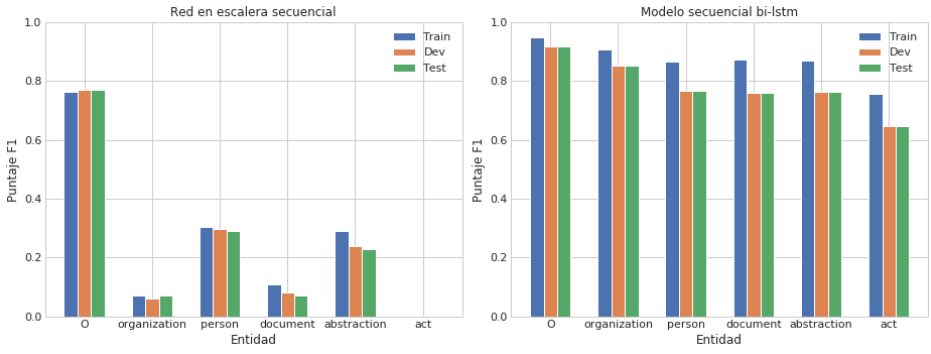


Figura 4.8: F1-Score por cada clase para el modelo de red en escalera secuencial vs. red recurrente BiLSTM. Comparación sobre distintos conjuntos de datos: entrenamiento, validación y evaluación.

El resto de las entidades obtienen valores por debajo de lo esperado, remarcando la entidad *act* la cual, como fue mencionado en la Sección 3.3.4, es la entidad minoritaria y no esperábamos resultados favorables.

Si bien el espacio de generalización en la red en escalera para cada entidad es menor al observado en el otro modelo, los resultados no proporcionan suficiente evidencia como para sostener la hipótesis a comprobar. Esto se puede deber a que la estructura propuesta para el modelo de red en escalera secuencial no sea el indicado.

Por la naturaleza del perceptrón multicapa, mucha de la información respecto a la posición se pierde, y al intentar realizar este modelo secuencial se nota que la información de contexto, al fin y al cabo, no puede representarse correctamente. El modelo claramente necesita un trabajo más profundo y algo del estilo de los embeddings posicionales que ofrecen modelos como los propuestos por la arquitectura del *Transformer* (Vaswani et al., 2017).

Capítulo 5

Conclusiones y trabajo futuro

5.1. Aportes

En esta tesis se exploró mediante la experimentación la técnica semi supervisada llamada red en escalera (Rasmus et al., 2015), aplicada al reconocimiento de entidades nombradas en textos legales.

En un comienzo el objetivo principal fue el de investigar si las redes en escalera obtienen mejores resultados con respecto a distintas técnicas del estado del arte para esta tarea clásica del procesamiento del lenguaje natural. Recordemos que la técnica de red en escalera fue concebida para tareas relacionadas con la clasificación de imágenes.

Luego con la experimentación se fue observando que, si bien los resultados no son particularmente mejores que otras técnicas, si se obtiene una mejor generalización con datos no conocidos. Es decir, que las redes en escalera tienden a generalizar mejor que las técnicas supervisadas con las cuales se experimentaron.

En un primer paso, y como para tener una base sobre la cual trabajar, se decidió utilizar el Stanford NER-CRF. Sus resultados fueron muy buenos, como se observa en la Sección **??**.

Previo a la experimentación con las redes, se exploraron distintas representaciones vectoriales para las entidades, según lo propuesto en (Iacobacci et al., 2016), como se detalla en la Sección 1.4: promedio, reducción fraccionaria y reducción exponencial, de las cuales se concluyó a partir de lo observado que la reducción exponencial provee mejores resultados.

En una primera instancia, nos encontramos con el problema de que el modelo funcionaba muy mal, esto nos obligó a explorar distintas maneras de atacar el problema en pos de obtener mejores resultados. El procedimiento se describe en la Sección 3.3.3, donde vemos como se fueron probando distintas hipótesis para encontrar el motivo por el que la red no funcionaba bien.

En particular, se exploró la naturaleza de los datos en la Sección 2.4 mediante técnicas de reducción de dimensionalidad, para comprobar si los mismos ayudan a la componente no supervisada de la red en escalera, concluyendo que los datos tienden a agruparse para ciertas entidades mientras que para otras no. Esto indicaría que la componente no supervisada de la red en escalera se va a ver beneficiada para ciertas categorías. También se comprobó un desbalance importante entre las categorías, por lo que se llevo a crear conjuntos de datos mas pequeños en donde las categorías estén lo mejor balanceadas posible.

Una vez encontradas algunas de las causas latentes de por qué el modelo estaba funcionando mal, pero a su vez entendiendo que el modelo no era la mejor solución posible para la tarea de NERC, decidimos explorar la utilidad de la componente no supervisada en el trabajo de regularización del modelo, en pos de obtener modelos que pudiesen generalizar mejor.

Este conjunto de experimentaciones puede observarse en la Sección 4.3, donde los resultados logrados no son comparables con el estado del arte, pero son prometedores ya que, si bien la componente no supervisada no provee una mejora al momento de clasificar, se observa que sí ayuda a la generalización ante datos desconocidos en comparación con las otras técnicas (pues la diferencia de desempeño entre los distintos conjuntos de datos no es tan marcada).

Durante el Experimento 4, cuyos resultados se muestran en la Sección 3.3.4, se buscó reforzar esta hipótesis de generalización, adaptando la red en escalera al método de clasificación secuencial. En este caso los resultados no nos aportan la información suficiente como para comprobar o negar la hipótesis, ya que la estructura del conjunto de datos y su desbalance no permitió generar un conjunto de datos acorde y balanceado como en el caso del Experimento 3.

5.2. Trabajo futuro

Una de las áreas que sería interesante explorar es una mejor representación de los datos que se brindan a la red en escalera. Se vio en la Sección 2.4 que la separación de los datos no es muy clara, y esto está fuertemente ligado a la representación vectorial utilizando reducción exponencial. Valdría la pena explorar otras soluciones que mejoren la separación de, al menos, los datos anotados, pues su representación interna es directamente responsable de como el autoencoder los representará en las capas más internas y por lo tanto que tan sencillo será diferenciarlos.

Por otra parte, las redes mostraron resultados prometedores en todo lo que se refiere a generalización. Quizás la componente no supervisada del modelo se pueda pensar como una componente de regularización y se podrían medir distintas formas en que esta puede ayudar al modelo a generalizar mejor. Eventualmente, la red en escalera puede brindar formas de representación interna que puedan ser transferidas a otros modelos en su generalización.

Por último, algo que queda con muchas posibilidades de ser trabajado a futuro es el estudio de las redes en escalera con la componente secuencial. Probablemente esto no pueda ser trabajado desde la red en escalera clásica, sino que necesitaría de algún modelo un poco más complejo como pueden ser las redes en escalera convolucionales o bien las redes en escalera recurrentes.

Bibliografía

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Cardellino, C., Teruel, M., Alemany, L. A., and Villata, S. (2017). A low-cost, high-coverage legal named entity recognizer, classifier and linker. In *Proceedings of the 16th Edition of the International Conference on Articial Intelligence and Law*, ICAIL '17, pages 9–18, New York, NY, USA. ACM.

Chapelle, O., Schlkopf, B., and Zien, A. (2010). *Semi-Supervised Learning*. The MIT Press, 1st edition.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Grishman, R. and Sundheim, B. (1996). Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Hahm, Y., Park, J., Lim, K., Kim, Y., Hwang, D., and Choi, K.-S. (2014). Named entity corpus construction using wikipedia and dbpedia ontology. In Chair), N. C. C., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Iacobacci, I., Pilehvar, M. T., and Navigli, R. (2016). Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–907.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

Jiang, Y., Krishnan, D., Mobahi, H., and Bengio, S. (2018). Predicting the generalization gap in deep networks with margin distributions. *arXiv preprint arXiv:1810.00113*.

Karpathy (2015). The unreasonable effectiveness of recurrent neural networks. "`http://karpathy.github.io/2015/05/21/rnn-effectiveness/`". [Online; accessed 21-September-2019].

Ma, X. and Hovy, E. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space.

Mitchell, T. M. (1997). Machine learning.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 3546–3554. Curran Associates, Inc.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA.

Settles, B. (2009). Active Learning Literature Survey. Computer sciences technical report, University of Wisconsin–Madison.

Stanford (2013a). Autoencoders. "`http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders`". [Online; accessed 21-September-2019].

Stanford (2013b). Multi-layer neural network. "`http://deeplearning.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/`". [Online; accessed 21-September-2019].

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408.