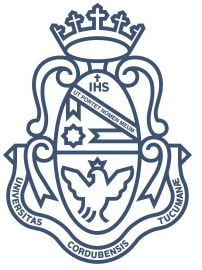
Facultad de Matemática, Astronomía,
Física y Computación

Universidad Nacional de Córdoba



Reconocimiento semi-supervisado de entidades nombradas mediante redes convolucionales en escalera

Tesis para obtener el título de

Licenciado en Ciencias de la Computación

Autor: Emiliano Kokic

Director: Cristian Cardellino

Córdoba, Argentina 2019

Resumen

El presente trabajo de tesis consiste en la exploración de un método de aprendizaje automático semi-supervisado llamado Redes Convolucionales en Escalera. La problemática que se decide abordar para la evaluación de dicho modelo es el Reconocimiento de Entidades Nombradas, una tarea muy relevante dentro del área de Procesamiento del Lenguaje Natural. Para realizar el estudio fue indispensable contar con WiNER, un corpus anotado de Wikipedia de gran calidad y fácil acceso. A su vez se estudian alternativas de representación de las palabras de acuerdo a su contexto. Se utiliza el bien conocido modelo Word2Vec para la generación de *embeddings* de palabras junto con la aplicación de estrategias que los combinan. En particular, resulta que el uso de capas convolucionales es una gran herramienta para la extracción de atributos del contexto. Se implementaron distintas arquitecturas de modelos, cada una de ellas con su versión supervisada (a modo de baseline) y semi-supervisada (al agregar las redes en escalera). Cada arquitectura tiene distintos tipos de instancias de entrenamiento, en algunos casos utilizando el etiquetado de palabras así como también el etiquetado de secuencias. Finalmente, luego de definir las métricas de evaluación se realizaron los experimentos pertinentes encontrando el modelo de Redes Convolucionales en Amplitud en Escalera como el más prometedor. Si bien los resultados obtenidos no son del estado del arte en cuanto a la tarea de reconocimiento de entidades nombradas, se visualiza que los modelos semi-supervisados de redes neuronales en escalera generalizan mejor y su performance

no disminuye en gran medida al de los supervisados gracias al uso complementario de datos no anotados.

The present work consists in the exploration of a semi-supervised machine learning method called Convolutional Ladder Networks. A very relevant task within the Natural Language Processing area is the Named-entity recognition, this is the problem that is decided to deal for the model evaluation. In order to carry out this study, it was essential to have WiNER, an annotated corpus of Wikipedia of great quality and easy access. At the same time, alternative strategies for the word representations according to their context are studied. The well-known Word2Vec model is used to generate word embeddings along with the application of strategies that combine them. In particular, it turns out that the use of convolutional layers is a great tool for context features extraction. Different model architectures were implemented, each of them with their supervised version (as a baseline) and the semi-supervised one (when adding ladder networks). Each architecture has different types of training instances, in some cases using word tagging as well as sequence tagging. Finally, once the experiments were executed, we found the Wide Convolutional Ladder Networks model as the most promising. Although the results obtained are not from the state of the art in terms of the task of recognizing named entities, it is found that semi-supervised models of ladder neural networks generalize better and their performance does not decrease greatly to that of supervised ones thanks to the complementary use of unlabeled data.

Índice general

Capítulo 1

Introducción y motivación

1.1. Introducción

El **Aprendizaje Automático** (del inglés, *Machine Learning*) es una rama de la inteligencia artificial que ha adoptado diferentes definiciones a lo largo de la historia. En particular, Tom M. Mitchell proporcionó una definición formal de los algoritmos estudiados en el campo del aprendizaje automático: "Se dice que un programa de computadora aprende de la experiencia E con respecto a alguna clase de tareas T y rendimiento P si su desempeño en las tareas T, medido por P, mejora con experiencia E." (Mitchell, 1997).

El aprendizaje automático tiene una amplia gama de aplicaciones. Un problema típico que se puede abordar es la predicción de precios de inmuebles [1]. A partir del uso de distintas características de inmuebles tales como sus metros cuadrados, cantidad de habitaciones, antigüedad, etc., y los precios asociados se genera un modelo, el cual a partir de nuevos datos de entrada ("características" o "atributos" de un inmueble) proporciona como salida una estimación del precio del mismo.

[1] `https://www.kaggle.com/c/house-prices-advanced-regression-techniques`

Los tipos de algoritmos de aprendizaje automático difieren de acuerdo a su enfoque, el tipo de datos que ingresan, o el que generan, y el tipo de tarea o problema que pretenden resolver.

Una de las tareas más comunes es el **aprendizaje supervisado**, el cual produce una función que establece una correspondencia entre los datos de entrada y los datos de salida (conocidos como "etiquetas"). Dentro del aprendizaje supervisado nos encontramos con problemas de regresión (como el anterior ejemplo de predicción de precios de inmuebles) que tratan de etiquetar a los datos en un rango numérico (i.e. la etiqueta es continua), o con problemas de clasificación que trata de etiquetar a los datos en un conjunto finito de etiquetas discretas (o clases). Un ejemplo de clasificación es el reconocimiento facial de Facebook [2] que utiliza para el auto-etiquetado de imágenes. La tarea concreta que el algoritmo debe realizar es, a partir de una imagen, detectar los rostros de personas que se encuentran en la misma y asociar a cada uno de ellos con el nombre de la persona correspondiente.

El **aprendizaje no supervisado** es otra tarea común en aprendizaje automático, donde todo el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formado tan sólo por datos que no tienen ninguna información sobre valor o categoría asociada (i.e. no tienen "etiquetas"). El objetivo de una tarea de aprendizaje no supervisado generalmente está en el descubrimiento de patrones o señales que puedan explicar los datos. Un problema interesante que el aprendizaje no supervisado puede atacar es el modelado de temas en noticias, donde el objetivo es poder agrupar palabras o incluso oraciones que pertenezcan a una misma categoría. Este es un ejemplo de lo que en aprendizaje no supervisado se conoce como *clustering* (a veces conocido en la literatura del español como "agrupamiento") donde la dificultad muchas veces se encuentra en saber elegir la cantidad de agrupaciones (*clusters*) para posteriormente hacer un análisis de las mismas.

Finalmente, existe un híbrido de los métodos nombrados anterior-

[2] `https://www.facebook.com/facialrecognitionapp/`

mente llamado **aprendizaje semi-supervisado**. El análisis del habla es un ejemplo clásico del valor que pueden aportar este tipo de modelos[3]. Abordar esta tarea en particular, con un modelo netamente supervisado, requeriría de muchos recursos humanos que establezca manualmente cuáles son las salidas que debe tener el modelo para ciertos datos de entrada (proceso que se conoce como etiquetado). Esto se debe a que etiquetar audio no es trivial y es, precisamente, donde un modelo de aprendizaje semi-supervisado puede ser de gran ayuda al requerir pocos datos anotados y muchos no anotados. En este trabajo de tesis se estudia en detalle un modelo de aprendizaje semi-supervisado.

1.2. Motivación

El **procesamiento de lenguaje natural**, abreviado PLN o NLP (del inglés *Natural Language Processing*), es un campo de las ciencias de la computación, la inteligencia artificial y la lingüística que estudia las interacciones con computadores mediante lenguaje humano. Se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas y máquinas por medio de lenguajes naturales (e.g. español, inglés, etc). PLN es en sí mismo una de las tareas más complejas que estudia la inteligencia artificial, conociéndose como un problema AI-completo. PLN está, a su vez, dividido en diversas subáreas que buscan resolver distintos problemas de índole más específico, algunos estando más cerca de ser considerados "resueltos".

La extracción de información es una de las tareas de PLN que todavía requiere trabajo. Es además, una tarea fundamental en el análisis de texto libre para poder estructurar los datos que pueden obtenerse del texto de manera automática y poder así democratizar mejor el acceso a dicha información.

[3] `https://medium.com/@jrodthoughts/understanding-semi-supervised-learning-a6437c070c87`

Una de las tareas más importantes que se engloban dentro del área de extracción de información es el **reconocimiento de entidades nombradas** [4]. Esta tarea busca ubicar y clasificar ciertas entidades de un texto no estructurado en categorías predefinidas, como nombres de personas y organizaciones, ubicaciones, expresiones temporales, cantidades, valores monetarios, entre otras. Supongamos que tenemos la siguiente oración:

Jim bought 300 shares of Acme Corp. in 2006.

Potenciales entidades a reconocer serían las siguientes:

$[\text{Jim}]_{Person}$ bought 300 shares of $[\text{Acme Corp.}]_{Organization}$ in $[2006]_{Time}$

Como tantas otras áreas de trabajo en procesamiento de lenguaje natural, existen buenos recursos para realizar dicha tarea de manera automática en inglés. Sin embargo, muchas veces dependiendo del dominio y/o el idioma, la calidad de estos datos puede disminuir considerablemente. Es entonces que entra en escena las ventajas de trabajar con métodos semi-supervisados que aprovechen la enorme cantidad de datos no anotados que circulan libremente por Internet. El problema con los métodos semi-supervisados es que a veces se hace difícil evaluarlos y compararlos directamente sobre tareas que los aprovechen, puesto que dichas tareas no tienen punto de comparación. Se hace necesario entonces saber si dichos métodos son útiles en tareas ya establecidas.

El objetivo de esta tesis es la exploración de un método semi-supervisado conocido como "redes neuronales convolucionales en escalera" (Rasmus et al., 2015). Este método es utilizado originalmente para procesamiento de imágenes, en esta tesis se adaptó a la tarea de reconocimiento de entidades nombradas. Como punto de comparación en dicha tarea, el trabajo se basa en el corpus WiNER (Ghaddar and Langlais, 2017).

[4] `https://en.wikipedia.org/wiki/Named-entity_recognition`

1.3. Estructura de la tesis

En el Capítulo 2 comenzamos analizando el trabajo de (Ghaddar and Langlais, 2017) sobre el corpus WiNER, generado a partir de la Wikipedia y seleccionado para nuestra investigación. Posteriormente se describe la problemática de representación de palabras de acuerdo al contexto en el que ocurren y estrategias que buscan resolver dicho problema a partir del trabajo de (Iacobacci et al., 2016). Finalmente hacemos una mención detallada de las redes neuronales en escalera propuestas en el trabajo de (Rasmus et al., 2015), describiendo su arquitectura y particularidades.

En el Capítulo 3 se realiza un análisis del corpus WiNER mencionando como fue procesado en base a distintas decisiones. Se introducen los distintos tipos de instancias de entrenamiento utilizadas en los modelos junto con una descripción de las métricas de evaluación seleccionadas. Por último, se detallan las arquitecturas de los modelos utilizados.

En el Capítulo 4 se identifican los distintos experimentos realizados junto con sus resultados e interpretaciones. En el Capítulo 5 se proporcionan las conclusiones de esta tesis junto con el trabajo futuro.

Capítulo 2

Trabajo relacionado

2.1. WiNER: Un corpus anotado de Wikipedia para Reconocimiento de Entidades Nombradas

Como se mencionó previamente, en esta tesis se estudiará el impacto de una técnica de aprendizaje semi-supervisado, redes convolucionales en escalera, en una tarea de reconocimiento de entidades nombradas. Para eso se buscó un recurso de referencia que pudiera servirnos a la hora de establecer una base de experimentos en donde evaluar las redes en escalera.

Si bien hay varias opciones disponibles para estudiar, muchas de ellas eran pagas o bien requerían un proceso de registro que llevaba mucho tiempo. Es por eso que se decidió trabajar con la Wikipedia como base. Ahora bien, una opción sería hacer todo el trabajo de preproceso y diseñar el corpus de base para los experimentos. Sin embargo, eso también consume mucho tiempo.

Convertir Wikipedia en un corpus de entidades nombradas anotadas con tipos es una tarea que recibió cierta atención a lo largo de los últimos años. En particular, en el trabajo de (Ghaddar and Langlais, 2017) se propuso una nueva metodología aplicada a la Wikipedia en

inglés para construir WiNER, un gran corpus anotado de alta calidad.

Para realizar esto aplicaron un pipeline de anotación, el cual consta de tres etapas. En primera instancia se consideran los enlaces directos de cada artículo objetivo, buscando en su texto los títulos de los artículos que se acceden. A su vez, cuentan con un recurso previo[1] que enumera todas las menciones del texto que se refieren al concepto principal de un artículo. Esto posibilita realizar una relación entre las correferencias del recurso y los títulos accedidos. A cada una de estas coincidencias se le asigna una etiqueta con la entidad asociada proveniente de una tabla de correferencias.

Con el fin de obtener una mayor cobertura, en una segunda etapa ya se consideran enlaces anidados, buscando en el artículo objetivo los títulos de los artículos alcanzados y en caso de haber coincidencias se les asignan las entidades nombradas correspondientes. Finalmente, estos títulos emparejados en la segunda etapa también se comparan con sus menciones en la tabla de correferencias.

En cuanto a la evaluación del corpus generado, por un lado se realiza una evaluación manual a partir de un subconjunto de 1000 menciones. En esta etapa se mide la exactitud (o *accuracy* en inglés) obtenida luego de aplicar cada etapa del pipeline.

Posteriormente, se utilizan otros corpus de entidades anotadas, entre ellos uno clásico de referencia, el CONLL-2003 obtenido en el trabajo de (Tjong Kim Sang and De Meulder, 2003). Se compara el desempeño de distintos modelos con la métrica F1-score sobre las clases para nombres de personas (PER), organizaciones (ORG) y lugares (LOC) donde lo que varía fijado un modelo es la proporción del corpus WiNER que se utiliza y el corpus de evaluación, siendo el modelo LSTM-CRF (Huang et al., 2015) el más prometedor.

[1] `http://rali.iro.umontreal.ca/rali/en/`

2.2. Embeddings para desambiguación del sentido de palabras

Uno de los problemas fundamentales a la hora de lidiar con algoritmos de aprendizaje automático es la manera de representar los datos. Esto se debe a que los algoritmos esperan como entrada un vector de números de tamaño fijo. E.g., las imágenes pueden representarse mediante la intensidad de cada uno de sus píxeles.

En el caso del trabajo con texto, la representación de los datos es algo particularmente complejo. Principalmente se debe a la naturaleza discreta de las palabras, donde las representaciones clásicas suelen ser ralas (i.e. vectores de mucha dimensionalidad, pero con muchos ceros).

Ejemplos de este tipo de representaciones son los vectores *one-hot encodings*. Para estos casos cada vector tiene la dimensionalidad del total de palabras del vocabulario, donde cada palabra se corresponde con una componente del vector. Entonces cada palabra es representada por un vector de todos 0's salvo un 1 en la componente que tenga asociada. Dado que el tamaño del vocabulario en general es considerablemente grande provoca que estos vectores sean de gran dimensionalidad. Además, hay que notar que los vectores no preservan información sobre el contexto en el que sucedieron las palabras en el texto, sino que son representaciones discretas de las mismas.

Hace unos años comenzaron a utilizarse más las representaciones distribuidas de las palabras, donde cada palabra es representada por las palabras de su vecindad. En particular, aparecieron los *word embeddings*, vectores densos y continuos que codifican la probabilidad de una palabra dada su vecindad.

Existen diferentes métodos para obtener word embeddings, pero todos tienen el objetivo de generar representaciones de palabras a partir de un corpus sin etiquetas.

En particular, el algoritmo Word2Vec (Mikolov et al., 2013), explora dos arquitecturas de modelos basados en una red neuronal que buscan codificar la probabilidad de ocurrencia de una palabra dado su

contexto. Esta red es entrenada con un gran corpus y las proyecciones son los word embeddings que el modelo Word2Vec produce. Aún así, los embeddings de palabras sólo representan palabras. Todavía queda el problema de cómo representar oraciones de palabras.

En el trabajo de (Iacobacci et al., 2016) se estudia cómo los embeddings de palabras pueden ser utilizados para la tarea de Desambiguación del Sentido de las Palabras (del inglés *Word Sense Disambiguation*). En particular, este estudio considera cuatro estrategias diferentes para integrar un modelo de word embeddings pre-entrenado en un sistema supervisado de desambiguación de sentidos: concatenación, promedio, decaimiento fraccional y decaimiento exponencial. Con estas estrategias se busca obtener una representación más informativa de cada palabra de acuerdo al contexto en la que ocurre.

El framework que utilizaron para la desambiguación de sentidos es *It Makes Sence (IMS)* (Zhong and Ng, 2010), el cual provee una plataforma extensible y flexible para la desambiguación de sentidos supervisada que permite la verificación de diferentes características (entre ellas *POS tagging*[2]) y técnicas de clasificación. IMS utiliza como clasificador un modelo de *Support Vector Machine*[3] con kernel lineal.

Se realiza una evaluación de los embeddings propuestos en base a dos tareas estándar de desambiguación de sentidos: *lexical sample* y *all-words*. El objetivo de la primera tarea es que el sistema IMS analice los contextos de los sentidos individuales de un pequeño conjunto de palabras anotadas y capture pistas que puedan usarse para distinguir diferentes sentidos de una palabra en la fase de test. El objetivo de la segunda tarea es desambiguar todas las palabras en un texto dado.

Estudiar este trabajo fue de gran interés sobretodo al comienzo de esta tesis ya que originalmente se había decidido encarar el problema de reconocimiento de entidades fijando que la entrada del modelo sea una palabra. Frente a esto, enriquecer la representación de cada palabra provista por el modelo Word2Vec con las palabras vecinas a

[2] `https://en.wikipedia.org/wiki/Part-of-speech_tagging`
[3] `https://en.wikipedia.org/wiki/Support-vector_machine`

fin de que cada instancia de entrenamiento cuente con información contextual tuvo mucho sentido. En particular, la estrategia de decaimiento exponencial resulta ser la más idónea correspondiendose con la intuición de que el impacto que tienen las palabras más cercanas a la palabra objetivo siempre es mayor y va disminuyendo a medida que se encuentran más distantes.

2.3. Aprendizaje semi-supervisado con Redes en Escalera

Utilizar aprendizaje no supervisado con el fin de complementar al supervisado no es algo nuevo. Combinar una tarea auxiliar que ayude a entrenar una red neuronal fue propuesto por (Suddarth and Kergosien, 1990). Al compartir las representaciones ocultas entre más de una tarea, la red generaliza mejor. Las redes neuronales en escalera son una combinación entre una red *feedforward* y un *autoencoder*.

Una red neuronal feedforward tiene la particularidad de que las conexiones entre los nodos no forman un ciclo. Ejemplos de este tipo de redes neuronales son el perceptrón multicapa (ver Sección 3.4.1) y las redes convolucionales (ver Secciones 3.4.2 y 3.4.3).

Por otra parte, un autoencoder es una red neuronal no supervisada, que tiene como objetivo reducir la dimensionalidad de una entrada mediante la reconstrucción de la misma. Este tipo de red está compuesta por un codificador (*encoder*) encargado de comprimir la entrada en una representación espacial latente y un decodificador (*decoder*), cuya tarea luego es reconstruir la entrada a partir de esta representación intermedia. La idea es que, el proceso de reducción de dimensiones obtendrá una representación que encuentre las propiedades más relevantes de los datos de entrenamiento.

En el modelo de red en escalera (Valpola, 2014), la tarea auxiliar es reducir el ruido de las representaciones en cada nivel del modelo. La estructura es un autoencoder con conexiones directas entre cada capa del codificador (*encoder*) hacia el decodificador (*decoder*) y la tarea

de aprendizaje es similar a la de la reducción de ruido del autoencoder pero aplicados a cada capa, no solo a las entradas.

Las conexiones entre las capas del codificador y el decodificador alivian la presión de representar detalles en las capas más internas en el modelo porque, a través de las mismas, el decodificador puede recuperar cualquier detalle descartado por el codificador. Esto es posible porque el decodificador utiliza la información sobre los costos en cada capa del pasaje limpio.

Anteriormente, la red en escalera solo había sido aplicada en aprendizaje no supervisado (Valpola, 2014), en el trabajo de (Rasmus et al., 2015) el enfoque es combinar esto con aprendizaje supervisado. Los aspectos claves que plantea este nuevo enfoque son los siguientes:

Compatibilidad con métodos supervisados La parte no supervisada se centra en los detalles relevantes encontrados por el aprendizaje supervisado. Además, se puede agregar a redes neuronales feedforward existentes, como por ejemplo el perceptrón multicapa o las redes neuronales convolucionales.

Escalabilidad resultante del aprendizaje local Además de un objetivo de aprendizaje supervisado en la capa superior, el modelo tiene objetivos de aprendizaje no supervisado locales en cada capa, por lo que es adecuado para redes neuronales muy profundas.

Eficiencia computacional La parte del codificador del modelo corresponde al aprendizaje supervisado normal. Al agregar un decodificador, se triplica aproximadamente el cálculo durante el entrenamiento, pero no necesariamente el tiempo de entrenamiento, ya que el mismo resultado se puede lograr más rápido a través de una mejor utilización de la información disponible. En general, el cálculo por actualización escala de manera similar a cualquier enfoque de aprendizaje supervisado, con un pequeño factor multiplicativo.

En la red en escalera el proceso de inferencia puede ser aprendido usando el principio de reducción de ruido, que ha sido utilizado, por ejemplo, en autoencoders que reducen ruido (en siglas: *dAE*, del inglés *denoising autoencoders*), y en separación de fuente con reducción de ruido (en siglas: *DSS*, del inglés *denoising source separation*).

En dAE, un autoencoder es entrenado para reconstruir la observación original $\mathbf{x}$ a partir de una versión corrupta $\tilde{\mathbf{x}}$. El aprendizaje se basa simplemente en minimizar la norma de la diferencia de la $\mathbf{x}$ original y su reconstrucción $\hat{\mathbf{x}}$ de $\tilde{\mathbf{x}}$; el costo a minimizar es, entonces $\|\hat{\mathbf{x}} - \mathbf{x}\|^2$.

Mientras que las dAE normalmente sólo están capacitadas para reducir el ruido de las observaciones, las DSS se basan en la idea de utilizar funciones de reducción de ruido $\hat{\mathbf{z}} = g(\mathbf{z})$ de las variables latentes $\mathbf{z}$ para entrenar una función $f(\mathbf{x}) = \mathbf{z}$ que modela la probabilidad de las variables latentes en función de las observaciones. La función de costo es idéntica a la usada en un dAE salvo el hecho de que las variables latentes $\mathbf{z}$ reemplazan a las observaciones $\mathbf{x}$; esto es, $\|\hat{\mathbf{z}} - \mathbf{z}\|^2$. Lo único que hay que tener en cuenta es que $\mathbf{z}$ debe normalizarse de alguna manera, ya que de lo contrario el modelo tiene una solución trivial $\mathbf{z} = \hat{\mathbf{z}} = \textit{constante}$. En un dAE esto no puede suceder ya que el modelo no puede cambiar la entrada $\mathbf{x}$.

La Figura 2.1 muestra la estructura de una red en escalera. Cada capa (l) contribuye, a la función de costo C_d, un término de la forma $C_d^{(l)} = \|\mathbf{z}^{(l)} - \hat{\mathbf{z}}^{(l)}\|^2$ que entrena las capas superiores (tanto codificador como decodificador), para aprender la función de reducción de ruido $\hat{\mathbf{z}}^{(l)} = g^{(l)}(\tilde{\mathbf{z}}^{(l)}, \hat{\mathbf{z}}^{(l+1)})$ que asigna el valor corrupto $\tilde{\mathbf{z}}^{(l)}$ al estimador de reducción de ruido $\hat{\mathbf{z}}^{(l)}$. Como la estimación $\hat{\mathbf{z}}^{(l)}$ incorpora todo el conocimiento previo sobre $\mathbf{z}$, el mismo término de función de costo también entrena las capas del codificador, con el objetivo de encontrar características más limpias que coincidan mejor con el valor esperado previo.

Dado que la función de costo necesita tanto lo limpio $\mathbf{z}^{(l)}$ como lo corrupto $\tilde{\mathbf{z}}^{(l)}$, durante el entrenamiento el codificador se ejecuta dos veces: un pasaje limpio por $\mathbf{z}^{(l)}$ y un pasaje corrupto por $\tilde{\mathbf{z}}^{(l)}$.

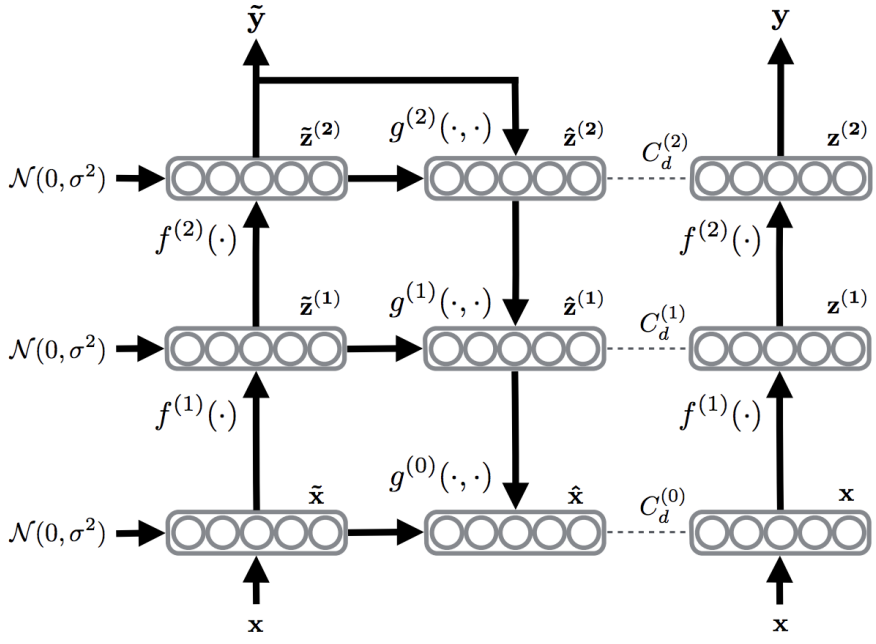


Figura 2.1: Ilustración conceptual de una red en escalera con L = 2 obtenida del trabajo de (Rasmus et al., 2015). El camino feedforward ($\mathbf{x} \rightarrow \mathbf{z}^{(1)} \rightarrow \mathbf{z}^{(2)} \rightarrow \mathbf{y}$) comparte las asignaciones $f^{(l)}$ con el camino feedforward corrupto, o codificador ($\mathbf{x} \rightarrow \tilde{\mathbf{z}}^{(1)} \rightarrow \tilde{\mathbf{z}}^{(2)} \rightarrow \tilde{\mathbf{y}}$). El decodificador ($\tilde{\mathbf{z}}^{(l)} \rightarrow \hat{\mathbf{z}}^{(l)} \rightarrow \hat{\mathbf{x}}$) consiste en las funciones de reducción de ruido $g^{(l)}$ y tiene funciones de costo $C_d^{(l)}$ en cada capa, tratando de minimizar la diferencia entre $\hat{\mathbf{z}}^{(l)}$ y $\mathbf{z}^{(l)}$. La salida $\tilde{\mathbf{y}}$ del codificador también se puede entrenar para que coincida con las etiquetas disponibles $t(n)$.

Una forma de visualizar la red en escalera es considerarla como una colección de autoencoders anidados que comparten partes de la maquinaria de reducción de ruido entre sí. Desde el punto de vista del

autoencoder en la capa l, las representaciones de las capas superiores se pueden tratar como neuronas ocultas. En otras palabras, no hay una razón particular por la cual $\hat{\mathbf{z}}^{(l+i)}$ producido por el decodificador, deba parecerse a la correspondiente representación $\mathbf{z}^{(l+i)}$ producida por el codificador. Solo la función de costo $C_d^{(l+i)}$ los une y obliga a la inferencia a proceder en orden inverso en el decodificador. Este intercambio ayuda a un autoencoder de reducción de ruido profundo a aprender el proceso de reducción de ruido a medida que divide la tarea en subtareas significativas de reducción de ruido en representaciones intermedias.

Capítulo 3

Entorno de experimentación

3.1. Análisis y procesamiento del corpus

Siendo el objetivo de este trabajo la evaluación de un método semi-supervisado aplicado al reconocimiento de entidades nombradas se hizo necesario contar con un corpus para realizar dicha tarea. Por ello, se decidió utilizar a WiNER , un corpus anotado a partir de artículos de la Wikipedia en inglés construido en el trabajo de (Ghaddar and Langlais, 2017).

Dicho corpus consta de 3.2M de artículos de Wikipedia, que comprende más de 54M de oraciones, de las cuales 41M contienen al menos una anotación de entidad nombrada, generando un total de 106M de anotaciones (un promedio de 2 entidades por oración).

Los tipos de entidades anotadas en este conjunto de datos son PER, LOC y ORG que denotan nombres de personas, lugares y organizaciones respectivamente. También se cuenta con la etiqueta MISC, categoría que incluye aquellas entidades no pertenecientes a las categorías anteriormente mencionadas. *Agatha Christie* (PER), *Belgian* (LOC), *Warner Bros* (ORG) y *Black Coffee* (MISC) son algunos ejemplos de entidades nombradas de este corpus. En caso de que se trate de una instancia que no corresponde a una entidad nombrada se utiliza, por convención, la etiqueta O.

Una de las primeras decisiones tomadas durante el procesamiento del corpus fue considerar entidades cuya cardinalidad a nivel de palabra fuese uno. Esto se logró separando aquellas entidades formadas por más de una palabra en nuevas entidades donde a cada palabra le corresponde el mismo tipo de entidad de la cual fue separada. En el ejemplo anterior de *Agatha Christie*, ahora se considera *Agatha* y *Christie* como dos entidades independientes, ambas del tipo PER.

El corpus contiene un total de 3223 documentos. El Cuadro 3.1 muestra la división de datos para conformar los conjuntos de entrenamiento, evaluación y validación.

Corpus	**Documentos**
Entrenamiento	0 - 1999
Validación	2000 - 2599
Evaluación	2600 - 3223

Cuadro 3.1: División de los datos para entrenamiento, validación y evaluación.

A partir de esa división se tomaron a su vez muestras aleatorias de cada conjunto de documentos detallados en el Capítulo 4. Esto fue necesario por cuestiones de limitación de recursos computacionales y tiempo para la ejecución de los distintos experimentos. Se realizó un preprocesamiento sobre los 3223 documentos, cada uno de estos documentos tiene asociado aproximadamente 1000 artículos identificados con un número único. Para cada artículo existe entonces un archivo asociado cuya primera línea tiene su wikiID seguidos de sus oraciones una por línea. En los artículos, los tokens (palabras, símbolos, signos, etc.) están representados por un ID unívoco.

Ejemplo

63903 24730 9 7 3035 4104 7584 1 355 17 22825 9984 2

Es equivalente a:

Hercule Poirot is a fictional Belgian detective , created by Agatha Christie .

Gracias a esta metodología de almacenamiento del corpus no fue necesario aplicar ningún tokenizador sobre el texto. A su vez, se decidió no remover signos de puntuación al igual que las palabras más comunes del idioma *stopwords* con la premisa de que en el caso de removerlas se perdería información útil sobre el contexto de cada palabra para nuestra tarea particular de reconocimiento de entidades nombradas. Una vez realizada la transformación y obtenidas las oraciones, se construyeron las distintas instancias de entrenamiento que se mencionan en la sección 3.2. Finalmente, se encuentran codificadas las entidades de cada artículo. El formato es el siguiente:

ID <wikiID> sentIdx begin end entityType

Donde *sentIdx* representa el número de oración del artículo, *begin* la posición donde comienza una entidad en la oración, *end* la posición donde termina y *entityType* el tipo de entidad que se trata:

entityType[0] = PER	entityType[1] = LOC
entityType[2] = ORG	entityType[3] = MISC

Continuando el ejemplo anterior:

ID	1000		
0	0	2	0
0	5	6	1
0	10	12	0

Resulta en:

[Hercule Poirot]$_{PER}$ is a fictional [Belgian]$_{LOC}$ detective , created by [Agatha Christie]$_{PER}$.

3.2. Instancias de entrenamiento

En este trabajo se exploraron tres maneras de representar a los datos en los modelos de aprendizaje.

3.2.1. Palabra - etiqueta

En este caso se hizo necesario investigar metodologías que nos permitieran representar a una palabra teniendo el cuenta el contexto en el cual ocurre. Como primera medida, se decidió utilizar los *embeddings* pre-entrenados de Google[1] para transformar a cada palabra en un vector que la representa en un espacio de dimensionalidad igual a 300.

Siguiendo el trabajo de (Iacobacci et al., 2016) se exploraron cuatro estrategias distintas para obtener una representación más informativa de cada palabra.

Concatenación

Este método consiste en concatenar los vectores de palabras que rodean una palabra objetivo en un vector más grande, que tiene un tamaño igual a las dimensiones sumadas de todas las proyecciones (*embeddings*) individuales.

- w_{ij} = peso asociado con la i-ésima dimensión del vector de la j-ésima palabra en la oración. Con los vectores de palabras de una oración se forma una matriz w^{DxL} donde L es la cantidad de palabras de esa oración.
- D = dimensionalidad de los word vectors originales. Por ejemplo, al usar el modelo Word2Vec de Google se tiene $D = 300$.
- W = tamaño de ventana que se define como el número de palabras en un solo lado.

[1] `https://code.google.com/archive/p/word2vec/`

Nos interesa representar el contexto de la I-ésima palabra de la oración. La i-ésima dimensión del vector de concatenación, que tiene un tamaño de $2WD$, se calcula de la siguiente manera:

$$e_i = \begin{cases} w_{i \bmod D,\ I - W + \lfloor \frac{i}{D} \rfloor} & \lfloor \frac{i}{D} \rfloor < W \\ w_{i \bmod D,\ I - W + 1 + \lfloor \frac{i}{D} \rfloor} & c.c. \end{cases}$$

Al tomar una ventana simétrica, se realiza un relleno (*padding*) con ceros a izquierda y/o derecha según corresponda para mantener la misma dimensionalidad en cada nuevo vector generado.

Promedio

Como su nombre indica, se calcula el centroide de los embeddings de todas las palabras circundantes. La fórmula divide cada dimensión en $2W$ ya que el número de palabras del contexto es dos veces el tamaño de la ventana:

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

Decaimiento fraccional

Una tercera estrategia para construir un vector de características en base a los embeddings de palabras contextuales está inspirada en la forma en que Word2vec combina las palabras en el contexto. Aquí, se supone que la importancia de una palabra para nuestra representación es inversamente proporcional a su distancia respecto a la palabra objetivo.

Por lo tanto, las palabras contextuales se ponderan en función de su distancia de la palabra objetivo:

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

Decaimiento exponencial

Funciona de manera similar al decaimiento fraccional, que le da más importancia al contexto cercano, pero en este caso la ponderación se realiza exponencialmente:

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

donde $\alpha = 1 - 0{,}1^{(W-1)^{-1}}$ es el parámetro de decaimiento. Se elige el parámetro de tal manera que las palabras inmediatas que rodean a la palabra objetivo contribuyen 10 veces más que las últimas palabras en ambos lados de la ventana.

3.2.2. Fragmento de oración - etiqueta

Para la aplicación de capas convolucionales la entrada del modelo deja de ser solo una palabra. Una primera alternativa fue tomar fragmentos de oraciones donde la palabra ubicada en el centro es la palabra a clasificar. Se decidió tomar una ventana simétrica de 2 palabras a izquierda y derecha de la palabra objetivo con la premisa de que el contexto relevante para la tarea de clasificación de la palabra objetivo son aquellas palabras de ocurrencia cercana. Para mayor claridad, un ejemplo de esto puede observarse en el Cuadro 3.3.

La información sobre el contexto de la palabra objetivo será obtenida entonces a partir de la aplicación de capas convolucionales sobre el fragmento en cuestión.

Contexto	Etiqueta
['It', 'aired', '**on**', 'NBC', 'from']	O
['aired', 'on', '**NBC**', 'from', 'February']	ORG
['on', 'NBC', '**from**', 'February', '2002']	O
['NBC', 'from', '**February**', '2002', 'to']	MISC
['from', 'February', '**2002**', 'to', 'May']	MISC
['February', '2002', '**to**', 'May', '2003']	O

Cuadro 3.2: Ejemplo de fragmentos de oración y etiquetas respectivas al contexto.

3.2.3. Oración - etiquetas

En aprendizaje automático, el etiquetado de secuencias, del inglés *sequence labeling*, es un tipo de tarea que implica la asignación de una etiqueta categórica a cada miembro de una secuencia de valores observados.

El etiquetado de secuencias es una aproximación clásica cuando se trata de lidiar con reconocimiento de entidades en particular (y otras tareas de PLN en general). Esto es así porque la disposición de las palabras en el lenguaje natural es inherentemente secuencial, muchas veces una palabra hace referencia a un hecho que sucedió anteriormente en el texto y se quiere que el modelo de aprendizaje automático capture esa información siendo idóneo que la entrada del mismo sea una secuencia.

Debido a que la entrada del modelo debe que ser un vector de longitud fija se hizo necesario determinar un tamaño máximo para las oraciones. Se observó entonces la distribución de la cantidad de palabras por oración y se decidió tomar sentencias de longitud igual a 30. Como consecuencia, se truncan aquellas oraciones de longitud mayor a 30 y se completan con 0's a derecha aquellas cuya longitud es menor.

Secuencia
['It', 'aired', 'on', 'NBC', 'from', 'February', '2002', 'to', 'May', '2003', '.']
Etiquetas
[O, O, O, ORG, O, MISC, MISC, O, MISC, MISC, O]

Cuadro 3.3: Ejemplo de oración y etiquetas.

3.3. Métricas de evaluación

Una de las métricas más conocidas para la evaluación de modelos de aprendizaje automático en problemas de clasificación es la exactitud (del inglés *accuracy*). Esta métrica en particular cuenta la cantidad de aciertos que tuvo el modelo a la hora de clasificar un dato de entrada y la divide sobre el total de datos clasificados. Se entiende por acierto cada vez que el modelo predice la etiqueta correcta para un dato ingresado. Más formalmente, en un problema de clasificación binaria con dos clases Positivo y Negativo:

$$\text{Exactitud (accuracy)} = \frac{TP + TN}{TP + TN + FP + FN}$$

Donde:

$TP =$ *True Positive* o Verdaderos Positivos
$TN =$ *True Negatives* o Verdaderos Negativos
$FP =$ *False Positives* o Falsos Positivos
$FN =$ *False Negatives* o Falsos Negativos

Esta definición se extiende fácilmente a un problema de más de dos clases.

La exactitud es una métrica estándar, pero tiene algunos defectos a la hora de ser utilizada para evaluar un modelo. En particular, está muy sesgada para los casos de clases desbalanceadas.

Muchas veces puede ocurrir que la distribución de las clases en un conjunto de datos observado no es homogénea, sino por el contrario se

encuentra muy desbalanceada. Esto es especialmente común en tareas de lenguaje natural por lo que se conoce como *Ley de Zipf* (Zipf, 1949).

Tomando como ejemplo un problema de clasificación binaria, supongamos que nuestra muestra contiene 100 ejemplos de entrenamiento de los cuales 99 pertenecen a la clase negativa y sólo 1 a la clase positiva. Un modelo muy sencillo que sin importar el dato de entrada siempre lo clasifique como negativo obtendrá un valor de *accuracy* de 0.99 pero este modelo nunca va a ser capaz de clasificar correctamente un dato perteneciente a la clase positiva. Si bien existen maneras para tratar el desbalance de clases, existen a su vez otras métricas que nos dan más confiabilidad respecto al desempeño del modelo en este escenario.

Por un lado tenemos la métrica *recall* que mide la capacidad de un modelo para encontrar todos los casos relevantes dentro de un conjunto de datos. Entendamos casos relevantes como aquellos pertenecientes a una clase en particular, i.e. la clase positiva. Formalmente se define como:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Por otro lado tenemos la métrica *precision* que expresa la proporción de los datos que nuestro modelo dice que eran relevantes (positivos) realmente eran relevantes. Para esto, se tienen en cuenta los falsos positivos, es decir, la cantidad de veces que el modelo de manera incorrecta etiquetó como positivo un dato que era en realidad negativo. Formalmente se define como:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Finalmente, existe una métrica que representa la media armónica entre las anteriores:

$$\text{F1-score} = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Notar que esta métrica penaliza valores extremos. Por ejemplo, con una *precision* igual a 0 y una *recall* igual a 1 el *f1-score* resulta igual a 0 mientras que con solo tomar el promedio se obtiene un valor igual a 0.5 que no representa bien la combinación de las métricas anteriores.

Una característica fundamental que un buen modelo de aprendizaje automático debe poseer es la *generalización*. Entendemos por generalización la capacidad que tiene el modelo de obtener buenos resultados con datos nuevos, es decir, el modelo también debe ajustarse de manera correcta a aquellos datos que no se utilizaron para el entrenamiento del mismo. Para medir entonces que tan bien generaliza un modelo es una convención realizar una división de los datos en tres conjuntos: entrenamiento, validación y test. A partir de esto, se analiza que tanto difieren los resultados de performance del modelo para cada conjunto de datos. Mientras menor sea la diferencia entre los datos en entrenamiento y los datos nuevos (validación y test) podemos afirmar que el modelo posee mayor capacidad de generalizar.

Otra manera más robusta de medir la generación es utilizar lo que se conoce como *validación cruzada* (del inglés, *cross-validation*). Cuando usamos los conjuntos de entrenamiento y validación, hay una parte aleatoria que puede influir los resultados. Dependiendo cómo dividamos los datos en estos dos conjuntos tendremos una estimación diferente del error de generalización. En unos casos pensaremos que el modelo generaliza mejor y en otros peor. Para combatir este problema, la validación cruzada propone crear los conjuntos de entrenamiento y validación varias veces, cada vez con una separación diferente. De esta forma obtendremos estimaciones diferentes. La media de todos los errores de validación se considera la estimación del error de generalización. Es una estimación más robusta. El inconveniente es que es computacionalmente más costoso. La solución de compromiso es usar validación cruzada pero limitar el número de veces que la hacemos.

3.4. Arquitectura de los modelos

3.4.1. Perceptron Multicapa

El modelo Perceptrón Multicapa (MLP) es una red neuronal artificial del tipo *feed-forward* (Hornik et al., 1989). Consiste en un grafo dirigido de múltiples capas, cada capa contiene una cantidad variable de nodos los cuales están totalmente conectados con los nodos de capa anterior y/o posterior según corresponda. En la Figura 3.1 se observa un ejemplo de este tipo de red donde cada nodo adquiere el nombre de neurona que aplica una función de activación no lineal. De esta manera, su composición le brinda la capacidad de distinguir datos de naturaleza no linealmente separables, limitación que sufren modelos de aprendizaje más tradicionales.

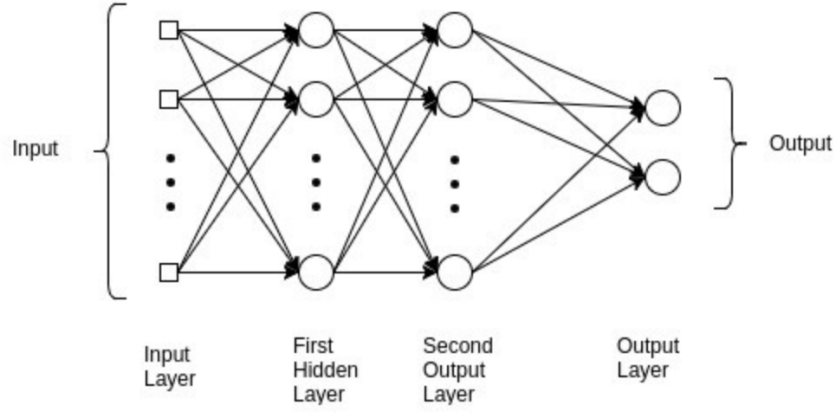


Figura 3.1: Ejemplo de una red neuronal perceptron multicapa.

Se decidió utilizar este modelo a modo de *baseline*, y para determinar cuál estrategia de representación de palabras del trabajo de (Iacobacci et al., 2016) funcionaba mejor. La capa de entrada del modelo toma una palabra representada a través de un vector, resultante

de haber aplicado alguna de las estrategias citadas en la sección 3.2.1. La capa de salida contiene cinco neuronas, ya que se cuenta con cinco clases: PER - LOC - ORG - MISC - O, posteriormente se aplica la función de activación softmax que asigna la salida no normalizada a una distribución de probabilidad sobre las clases de salida.

3.4.2. Redes convolucionales en amplitud

La entrada de la mayoría de los modelos que realizan tareas de procesamiento de lenguaje natural son oraciones o documentos representados en una matriz. Cada fila de la matriz corresponde a un *token*, típicamente una palabra, pero también podría ser un carácter, representado a través de un vector.

Al aplicar los filtros de las convoluciones, usualmente los mismos se deslizan sobre las filas completas de la matriz (palabras). Por lo tanto, el ancho de nuestros filtros suele ser el mismo que el ancho de la matriz de entrada. La altura es lo que puede variar, tomando ventanas de n palabras a la vez.

La Figura 3.2 muestra un ejemplo de esta arquitectura[2]. Se toman filtros de alturas 2, 3 y 4 y se supone un *stride* (desplazamiento del filtro en cada paso) igual a 1. Luego de la convolución se aplica una función de activación y se obtienen distintos vectores que contienen información sobre la sentencia original. Posteriormente se aplica una capa de *max pooling* cuyo objetivo es reducir la dimensionalidad pero con la promesa de mantener la información más relevante. Finalmente se construye un único vector al concatenar los vectores resultantes y se aplica una función de activación softmax a la capa densa de salida para realizar la clasificación correspondiente.

[2] `http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/`

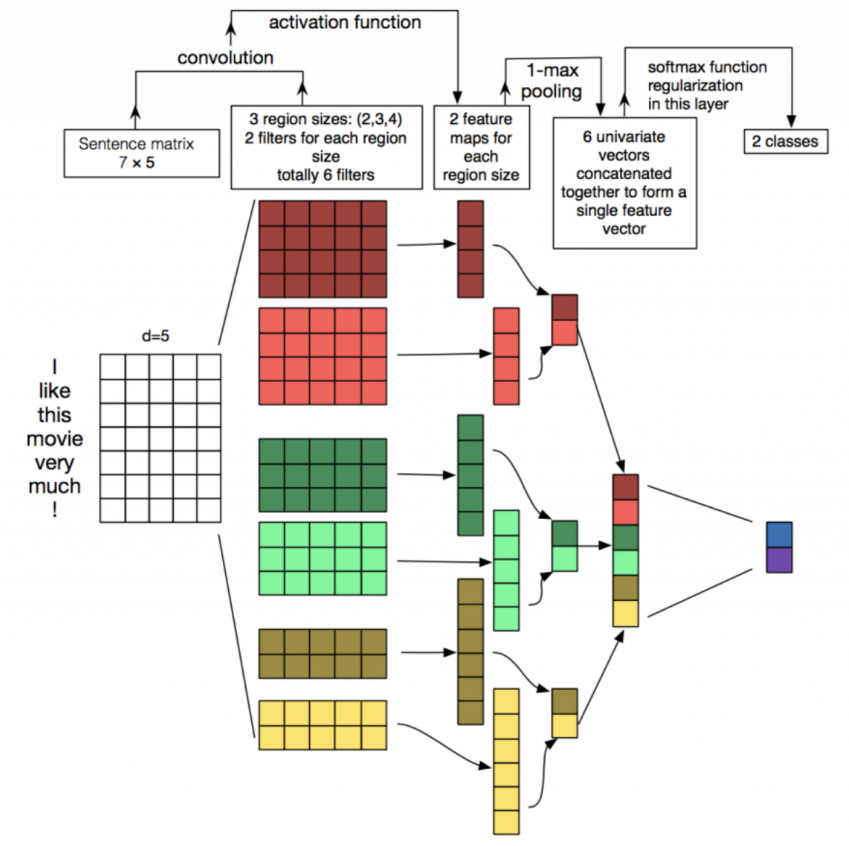


Figura 3.2: Ejemplo de una red convolucional en amplitud para análisis de sentimiento.

3.4.3. Redes convolucionales en profundidad

Una alternativa al modelo de redes convolucionales en amplitud que también le permite a las representaciones intermedias obtener información sobre el contexto de la sentencia es el modelo de redes convolucionales en profundidad.

La idea es hacer crecer el campo receptivo a medida que se apilan

más y más capas convolucionales. Esto significa que, de forma predeterminada, cada paso en la representación de la convolución ve toda la entrada en su campo receptivo.

En la Figura 3.3 se puede ver como al agregar más capas se incrementa el campo receptivo[3]. Esto significa que dada una profundidad suficiente, nuestra red podría mirar toda la capa de entrada, a través de unas pocas abstracciones.

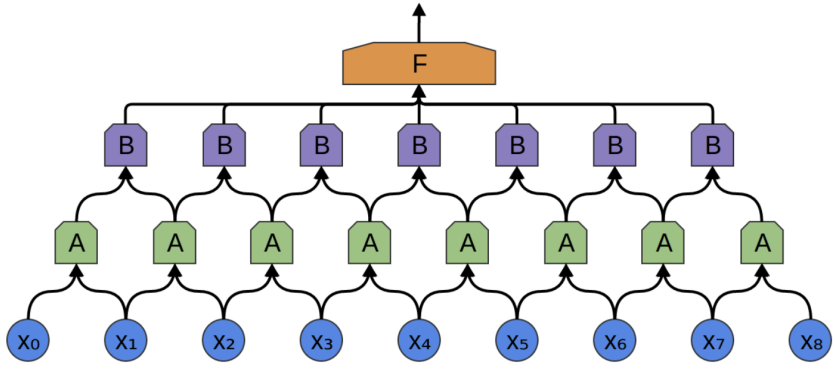


Figura 3.3: Ejemplo de una red convolucional en profundidad.

3.4.4. Redes neuronales en escalera

La idea detrás de las redes neuronales en escalera es entrenar simultáneamente una red neuronal del tipo *feed-forward* (como lo son la red perceptrón multicapa y las redes convolucionales) junto con un autoencoder con pesos compartidos. La red se entrena aprendiendo dos objetivos diferentes: uno supervisado, dado por el error de predicción de los datos etiquetados; y otro no supervisado, dado por el error de reconstrucción de los datos sin etiquetar.

[3] `https://medium.com/@TalPerry/convolutional-methods-for-text-d5260fd5675f`

En el presente trabajo, se exploró el uso de un parámetro extra sobre la función de costo de la red en escalera original. Al factor no supervisado de la función de costo se lo multiplica por un parámetro μ que viene a representar la relevancia de la tarea no supervisada en el modelo. La función de costo final es:

$$Cost = \text{supervised\_cost} + \mu \text{ unsupervised\_cost}$$

La estructura del modelo es un autoencoder con *skip connections* desde el codificador hacia el decodificador y la tarea de aprendizaje es similar a los autoencoders de eliminación de ruido popularmente conocidos como denoising autoencoders pero en este caso dicha tarea se aplica a cada capa, no solo a las entradas. Las *skip connections* liberan la presión de representar los detalles en las capas más altas del modelo porque a través de estas conexiones el decodificador puede recuperar cualquier detalle descartado por el codificador. Una descripción más detallada de este modelo se encuentra en la sección 2.3 como así también remitimos al lector a (Rasmus et al., 2015).

Finalmente, vale la pena destacar la estructura del decodificador cuando se utiliza como red feedforward las redes convolucionales en amplitud. La primera capa del decodificador es una transpuesta de la misma capa del codificador. La segunda capa, que se corresponde a la capa de max pooling del codificador, resulta ser una capa de 'upsampling' que simplemente toma la entrada y la repite para mapear las dimensiones de las capas convolucionales. La última capa es una 'convolucional transpuesta' (también conocida como capa deconvolucional), que mapea las convoluciones a la matriz de entrada original.

Capítulo 4

Experimentación y análisis de resultados

4.1. Baselines supervisados

Previo a la experimentación con los modelos de redes convolucionales en escalera, era necesario buscar un punto de comparación de estos modelos con algún baseline. Por lo cuál, se seleccionaron tres baselines basándonos en una estructura de red neuronal puramente supervisada, es decir, sin utilizar datos no anotados como es el caso de la red en escalera. Estos se detallaron en el Capítulo anterior, y a continuación se denotarán los experimentos y mostrarán los resultados de los mismos.

4.1.1. Perceptrón multicapa

El primer conjunto de experimentos fue realizado sobre el perceptrón multicapa común, que se describe con mayor detalle en la Sección 3.4.1. El conjunto de datos para entrenamiento y evaluación de este modelo se establecieron en 100000 instancias de entrenamiento y 20000 para evaluación y validación. Para conseguir esta muestra de ejemplos primero se realizó una división de los documentos del corpus

como se detalla en el Cuadro 3.1.

Luego, sobre cada subconjunto de documentos se tomo una muestra aleatoria de las instancias mencionadas anteriormente.

Notar que para estas muestras se realizó un *subsampling* de la clase O tanto para las instancias de tipo *palabra-etiqueta* como para *fragmento oración-etiqueta*. Esta decisión fue tomada con el objetivo de balancear las clases.

Para el caso del perceptrón multicapa se muestran tres experimentos, uno para cada estrategia de representación, con los hiperparámetros que mejores resultados obtuvieron seleccionados de manera aleatoria. Estos pueden verse detallados en el Cuadro 4.1.

Experimento	**Hiperparámetro**	**Valor**
MLP.1	Representación	Promedio
	Ventana	5
	Capas	[300, 256, 128, 5]
	Tasa de aprendizaje	0.3
MLP.2	Representación	Decaimiento fraccional
	Ventana	5
	Capas	[300, 256, 128, 5]
	Tasa de aprendizaje	0.3
MLP.3	Representación	Decaimiento exponencial
	Ventana	5
	Capas	[300, 256, 128, 5]
	Tasa de aprendizaje	0.3

Cuadro 4.1: Hiperparámetros del conjunto de experimentos con perceptrón multicapa supervisado.

Resultados

La Figura 4.1 muestra los resultados de aplicar los tres experimentos detallados en el Cuadro 4.1. Es un gráfico de barras con agrupa-

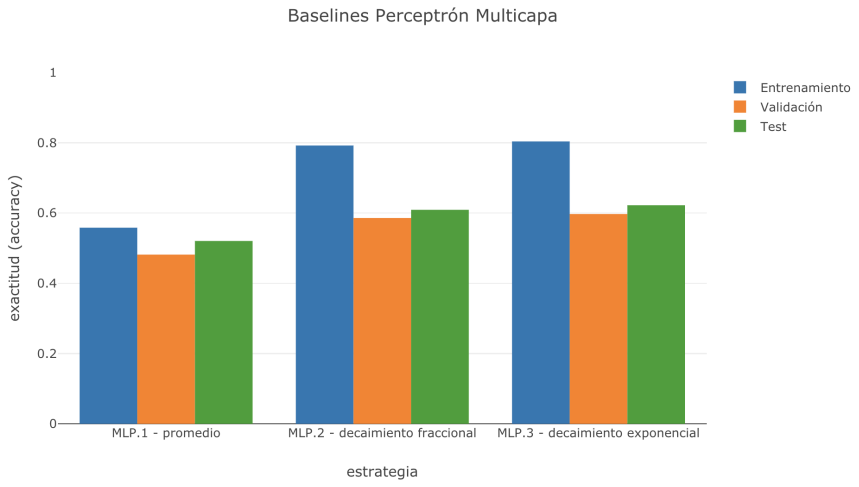


Figura 4.1: Resultados de los experimentos MLP. Comparación utilizando distintas representaciones de datos de entrada.

ción. El eje y del gráfico muestra la exactitud (*accuracy*) del experimento, donde a mayor valor, mejor el resultado.

Por otro lado, el eje x representa la estrategia aplicada (i.e. el experimento de acuerdo al Cuadro 4.1), agrupando tres barras de acuerdo al conjunto de datos sobre el que se muestra el resultado. De esta manera, la barra azul muestra el resultado sobre el conjunto de entrenamiento, la barra naranja sobre el conjunto de validación y la barra verde sobre el conjunto de evaluación (*test*).

Como se puede observar en la Figura 4.1, utilizar el decaimiento exponencial y fraccional como estrategias de representación de una palabra son una buena alternativa.

Esto posiblemente se deba a que la información de contexto que le aporta una palabra inmediatamente vecina a la palabra objetivo es mucho más relevante que una más distante.

Por otro lado, simplemente tomar el promedio de las palabras circundantes no tiene buenos resultados y justamente no tiene en cuenta

lo mencionado anteriormente. De estos experimentos el experimento MLP.3 es el que mejor resultados obtiene y es el seleccionado para su comparación en los experimentos mostrados en las siguientes secciones.

4.1.2. Redes convolucionales en amplitud

Para estos experimentos un ejemplo de entrenamiento es un fragmento de oración como se detalla en la Sección 3.2.2.

El conjunto de datos para el entrenamiento y evaluación de este modelo se establecieron 100000 instancias de entrenamiento y 20000 para validación y evaluación de manera análoga a los experimentos anteriores.

El Cuadro 4.2 establece la lista de experimentos e hiperparámetros utilizados. El objetivo de los experimentos es analizar si el modelo sufre variaciones al utilizar diferentes proporciones de instancias de entrenamiento anotadas.

Experimento	**Hiperparámetro**	**Valor**
Hiper. generales	Ventana simétrica W	2
	Número de filtros	100
	Tamaño de max pooling	1
	Tamaño de kernels	[2,3,4]
	Dropout	0.5
	Capas densas	[128, 5]
	Regularizador L2	1
CNN-Wide-Sup.1	% ejemplos entrenamiento	25 %
CNN-Wide-Sup.2	% ejemplos entrenamiento	50 %
CNN-Wide-Sup.3	% ejemplos entrenamiento	75 %
CNN-Wide-Sup.4	% ejemplos entrenamiento	100 %

Cuadro 4.2: Hiperparámetros para CNN Supervisado en Amplitud.

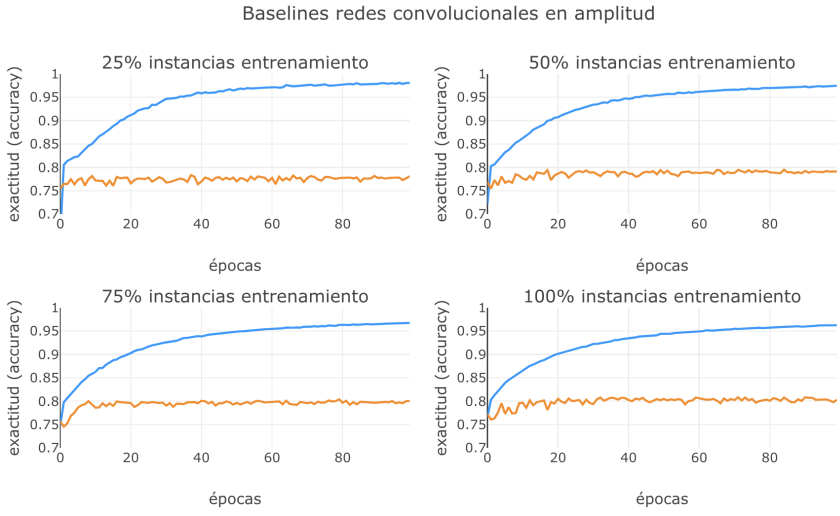


Figura 4.2: Resultados redes convolucionales en amplitud. Comparación utilizando distintas porciones de corpus como datos de entrenamiento para el modelo.

Resultados

La Figura 4.2 muestra los resultados de aplicar los cuatro experimentos detallados en el Cuadro 4.2. Se observan las curvas correspondientes al valor de exactitud que logra cada modelo a lo largo de las épocas de entrenamiento. Se utiliza el color azul para denotar la exactitud en los datos de entrenamiento mientras que el anaranjado para los datos de validación. No se observan diferencias significativas al utilizar una menor cantidad de datos anotados para el entrenamiento del modelo. Más adelante se tendrá en consideración el experimento CNN-Wide-Sup.4 que utiliza el 100 % de las instancias de entrenamiento para su comparación con el equivalente en la versión en escalera.

4.1.3. Redes convolucionales en profundidad

Para estos experimentos la entrada del modelo es una oración y la salida una secuencia de etiquetas que se corresponde una a una con cada *token* de la oración ingresada tal como se detalla en la Sección 3.2.3.

El conjunto de datos para el entrenamiento y evaluación de este modelo se establecieron 100000 instancias de entrenamiento y 20000 para validación y evaluación. Para este caso en el que las instancias son oraciones y no palabras no se realiza un *subsampling* de la clase mayoritaria O, por lo que el conjunto de datos se presenta inherentemente desbalanceado.

Para el caso del modelo de redes convolucionales en profundidad supervisado se muestran cuatro experimentos, con los hiperparámetros que mejores resultados obtuvieron seleccionados de manera aleatoria. En particular, se busca ver si el uso de varias capas convolucionales mejora el desempeño del modelo. Estos pueden verse detallados en el Cuadro 4.3.

Experimento	**Hiperparámetro**	**Valor**
Hiper. generales	Tamaño de kernels	2
	Dropout	0
	Regularizador L2	0
CNN-Depth-Sup.1	Capas convolucionales	3
	Número de filtros por capa	50
CNN-Depth-Sup.2	Capas convolucionales	10
	Número de filtros por capa	50
CNN-Depth-Sup.3	Capas convolucionales	3
	Número de filtros por capa	100
CNN-Depth-Sup.4	Capas convolucionales	5
	Número de filtros por capa	100

Cuadro 4.3: Hiperparámetros del conjunto de experimentos con redes convolucionales en profundidad supervisado.

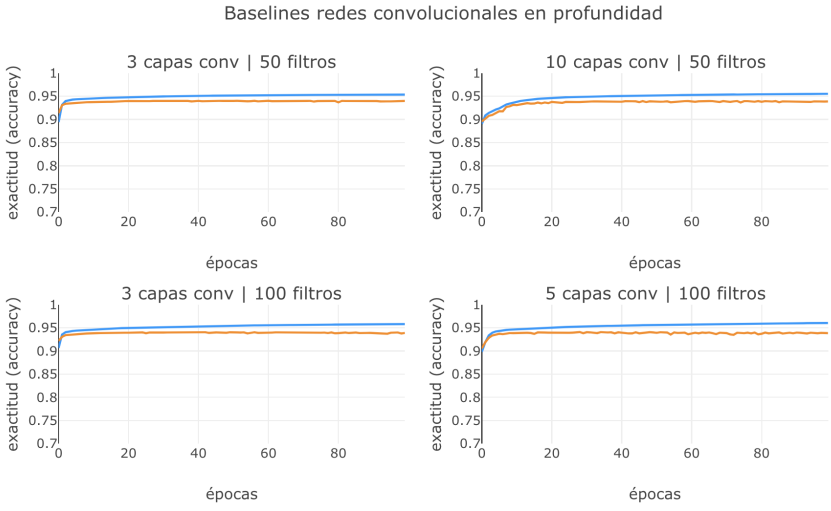


Figura 4.3: Resultado redes convolucionales en profundidad supervisado. Comparación utilizando distinta cantidad de capas convolucionales y cantidad de filtros por capa.

Resultados

En la Figura 4.3 se observan cuatro experimentos, los cuales representan los mejores resultados de los distintos experimentos que se ejecutaron sobre este modelo supervisado.

Nuevamente el color azul denota el valor de exactitud en los datos de entrenamiento y el anaranjado para el caso de los datos de validación. Un aspecto a tener en cuenta es que utilizar como métrica la exactitud para su comparación con los modelos anteriores es algo que hay que trabajar con cuidado, ya que al manejar instancias a nivel de oración las clases se encuentran inherentemente desbalanceadas, por lo que la exactitud puede ser alta simplemente marcando todo como la clase mayoritaria.

Sin embargo, por cuestiones comparativas, sobre todo con los modelos anteriores, se utiliza el experimento CNN-Depth-Sup.2 para di-

cha tarea y con su variante implementación en escalera (en la Figura 4.3, el de arriba a la derecha, con 10 capas convolucionales y 50 filtros por capa).

4.1.4. Comparación de baselines

Al obtener los mejores hiperparámetros para los modelos baseline, se decidió compararlos para tener una idea clara de lo que estamos tratando de lograr superar con la ayuda de las redes en escalera. Para esto, el método de comparación es con el conjunto de datos de evaluación, en lugar del conjunto de datos de validación.

La Figura 4.4 compara los resultados de los mejores modelos de cada variante supervisada. Es un gráfico de barras con agrupación. El eje y del gráfico muestra la exactitud (*accuracy*) del experimento, donde a mayor valor, mejor el resultado. Por otro lado, el eje x representa cada uno de los mejores experimentos, agrupando tres barras de acuerdo al conjunto de datos sobre el que se muestra el resultado. De esta manera, la barra azul muestra el resultado sobre el conjunto de entrenamiento y la barra verde sobre el conjunto de evaluación (*test*).

En la Figura 4.4 se puede observar la notable mejora que aportan los modelos basados en redes convolucionales. Se valida el hecho de que aplicar capas convolucionales provoca que se capture información del contexto de cada palabra generando buenos atributos (*features*) intermedios que el modelo utiliza para la tarea de clasificación de entidades nombradas.

En otras palabras, el uso de capas convolucionales con la finalidad de extraer características del entorno de una palabra parece ser una muy buena alternativa.

Por otro lado, se observa que las redes convolucionales en profundidad arrojan los mejores resultados. De todas maneras, cabe recordar que el uso de exactitud puede esconder el desempeño del modelo para clases minoritarias, por lo que hay que tomarlo con cuidado.

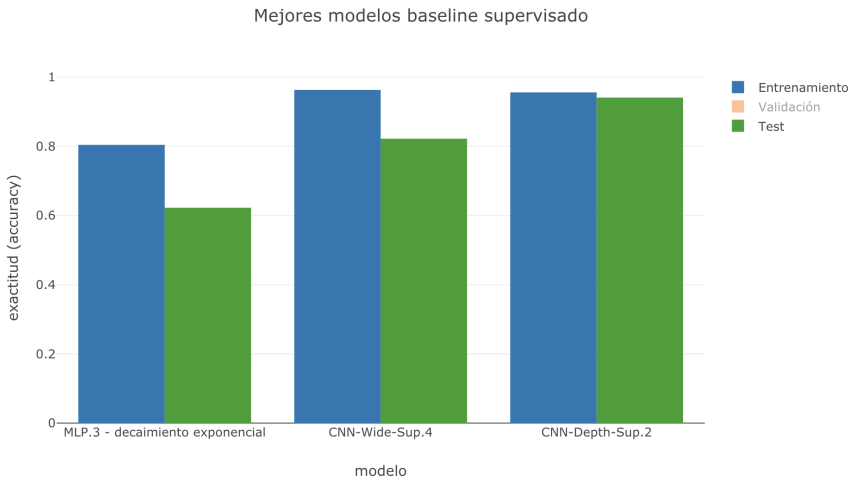


Figura 4.4: Mejores modelos baseline supervisados. Comparación de resultados entre datos de entrenamiento y datos de evaluación

4.2. Redes neuronales en escalera

Una vez realizados los experimentos netamente supervisados a modo de baseline se prosiguió con la incorporación de la tarea no supervisada al utilizar las redes en escalera. Para cada variante se utilizó la misma división y conjunto de datos que en los experimentos anteriores, de manera tal que el impacto de los datos no supervisados pudiese ser medido acorde al experimento realizado.

4.2.1. Impacto de los datos no anotados: Perceptrón multicapa en escalera

En primer lugar, se desea medir el impacto de la tarea no supervisada que brinda la arquitectura en escalera. Al incorporar al entrenamiento la utilización de datos no anotados queremos observar si efectivamente derivan en una mejora de rendimiento de la red en

Experimento	Hiperparámetro	Valor
MLP-Ladder.1	Representación	Decaimiento exp.
	Ventana	5
	Capas	[300, 256, 128, 5]
	Factor de ruido por capa	0.3
	Costo de reducción de ruido	[100, 0.1, 0.1, 0.01]
	Tasa de aprendizaje	0.02

Cuadro 4.4: Hiperparámetros del conjunto de experimentos con perceptrón multicapa en escalera.

escalera más básica: el perceptrón multicapa.

Se seleccionó el experimento con mejores resultados del modelo de perceptrón multicapa en escalera para su inmediata comparación con su versión supervisada. El Cuadro 4.4 muestra los hiperparámetros de dicho experimento.

Evolución de datos de entrenamiento vs. validación a través de épocas

En la Figura 4.5 se observan dos gráficos, las trazas azules describen el valor de exactitud del modelo a lo largo de las épocas en los datos de entrenamiento y las trazas anaranjadas de manera análoga en los datos de validación. El gráfico a izquierda corresponde al experimento MLP.3 que utiliza la estrategia de decaimiento exponencial para representar cada palabra y que consiguió el mejor desempeño en la Sección 4.1.1.

Por otro lado, el gráfico a derecha es un experimento equivalente (i.e. con los mismos hiperparámetros y datos de entrenamiento y validación) en su versión en escalera.

Se puede apreciar como el modelo en escalera permite una mejor generalización que el netamente supervisado al observar que la curva correspondiente a los datos de validación acompaña a la de entrenamiento (i.e. el 'gap' entre entrenamiento y validación no es tan alto).

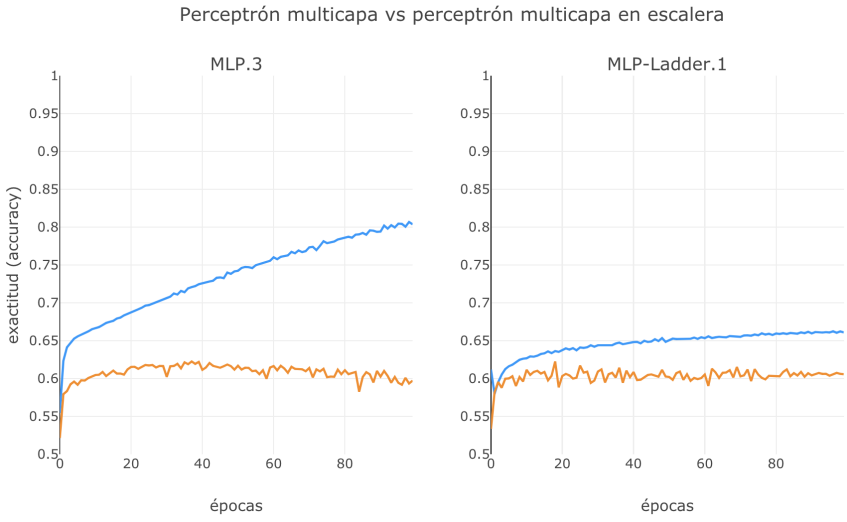


Figura 4.5: Perceptrón multicapa supervisado vs semisupervisado (escalera). Comparación de evolución de datos de entrenamiento y datos de validación

Más aún, se observa como para el caso del experimento puramente supervisado, después de ciertas iteraciones el desempeño sobre validación comienza a caer. En el caso del experimento semi-supervisado, por el contrario, el desempeño entre entrenamiento y validación se mantiene bastante constante a lo largo del tiempo, lo que es un primer indicio de que el modelo semi-supervisado sirve para regularizar el modelo.

Comparación de valores finales de desempeño sobre datos de entrenamiento vs. evaluación

La figura 4.6 muestra los valores de exactitud para los datos de entrenamiento y evaluación de los modelos del perceptrón multicapa supervisado y su variante en escalera.

Es un gráfico de barras con agrupación. El eje y del gráfico muestra

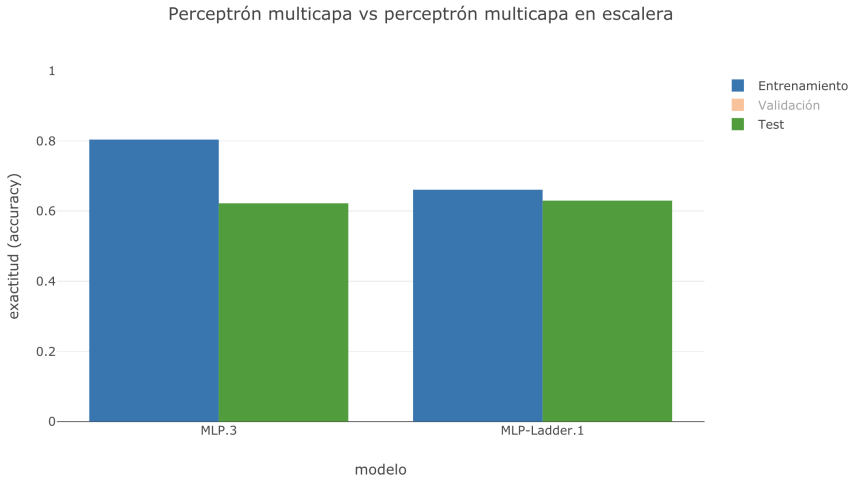


Figura 4.6: Perceptrón multicapa supervisado vs semisupervisado (escalera). Comparación de resultados finales sobre datos de entrenamiento y datos de validación

la exactitud (*accuracy*) del experimento, donde a mayor valor, mejor el resultado. Por otro lado, el eje x representa el modelo utilizado, agrupando dos barras de acuerdo al conjunto de datos sobre el que se muestra el resultado.

De esta manera, la barra azul muestra el resultado sobre el conjunto de entrenamiento y la barra verde sobre el conjunto de evaluación (*test*). Nuevamente se observa una mayor generalización por parte del modelo perceptrón multicapa en escalera (*MLP-Ladder.1*) ya que la diferencia en cuanto al desempeño en los datos de evaluación y entrenamiento es notablemente menor, mientras que el desempeño en el conjunto de datos de evaluación para el método semi-supervisado es igual (sino mayor) al mostrado para el caso puramente supervisado. Junto con los resultados de la Figura 4.5, podemos concluir que el uso de datos no anotados en el modelo arroja resultados interesantes a la hora de regularizar dicho modelo.

4.2.2. Impacto de selección de datos anotados y no anotados para el entrenamiento: Redes convolucionales en escalera en amplitud

Este conjunto de experimentos busca medir el impacto de utilizar distintos tipos de datos de entrenamiento para los modelos semi-supervisados de redes en escalera. En particular, nos interesa observar como se comporta un modelo de red en escalera convolucional en amplitud a partir de la que se define en la Sección 3.4.2.

Se observan distintas combinaciones de datos anotados y no anotados como entrenamiento para los modelos de la red convolucional en escalera. No obstante, se mantienen las mismas 20000 instancias de validación y 20000 de evaluación utilizadas en la versión supervisada para poder comparar con el baseline supervisado de la Sección 4.1.2.

Experimento	**Hiperparámetro**	**Valor**
Hiper. generales	Ventana simétrica W	2
	Número de filtros	100
	Factor de ruido por capa	0.3
	Costo de reducción de ruido	[100, 0.1, 0.01]
	Factor costo no supervisado	0.3
	Tasa de aprendizaje	0.02
	Tamaño de max pooling	1
	Tamaño de kernels	[2,3,4]
CNN-Wide-Ladder.1	Datos anotados	100k
	Datos no anotados	mismos 100k
CNN-Wide-Ladder.2	Datos anotados	100k
	Datos no anotados	100k disjuntos
CNN-Wide-Ladder.3	Datos anotados	100k
	Datos no anotados	200k

Cuadro 4.5: Hiperparámetros del conjunto de experimentos con redes convolucionales en amplitud en escalera.

Notar que en estos casos, a diferencia del perceptrón multicapa de la sección anterior, un ejemplo de entrenamiento es un fragmento de oración como se detalla en la Sección 3.2.2.

El Cuadro 4.5 detalla los experimentos realizados con las redes convolucionales en escalera en amplitud, basadas en lo detallado en la Sección 3.4.2.

Para el caso del modelo de redes convolucionales en amplitud en escalera se muestran tres experimentos, con los hiperparámetros generales que mejores resultados obtuvieron seleccionados de manera aleatoria.

Lo que varían en estos experimentos son las instancias utilizadas para el entrenamiento del modelo, la razón de esto es medir el impacto que generan los datos no anotados dependiendo si coinciden o no con los anotados y que sucede si variamos la proporción utilizada.

Comparación de distintos tipos de datos de entrada: iguales, disjuntos y contenidos

En la figura 4.7 se observan tres gráficos correspondientes a los experimentos realizados sobre el modelo de redes convolucionales en amplitud en escalera.

A izquierda se ubica el gráfico correspondiente a utilizar las mismas 100k instancias de entrenamiento como datos anotados y no anotados.

El gráfico central es el resultante de haber utilizado 100k instancias anotadas y 100k instancias disjuntas no anotadas.

Por último, el gráfico de la derecha utiliza las 100k instancias originales como datos anotados y no anotados y, a su vez, suma 100k nuevos no anotados.

El experimento CNN-Wide-Ladder.2 nos muestra que agregar nuevos datos no anotados parece ser clave para mejorar el rendimiento del modelo. Sin embargo agregar demasiados datos no anotados puede afectar al modelo de manera negativa como se observa en el experimento CNN-Wide-Ladder.3. Esto último puede deberse al hecho de que la red no es lo suficientemente capaz de manejar la reconstrucción

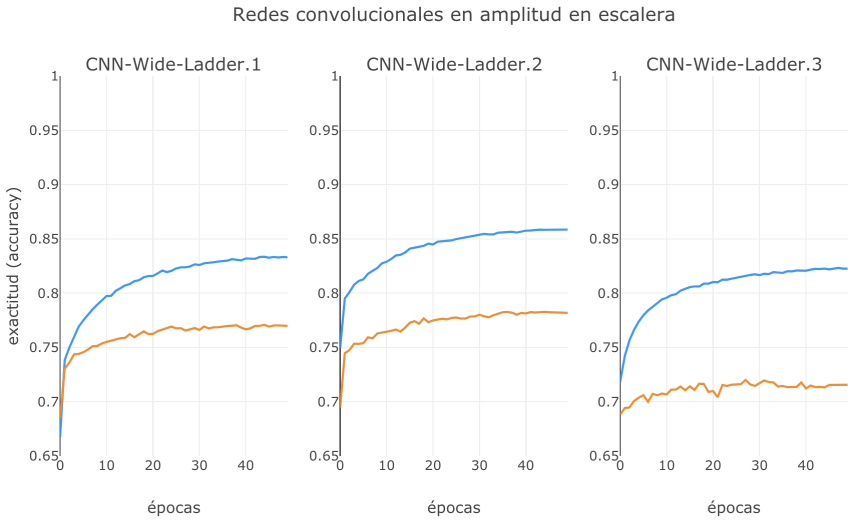


Figura 4.7: Redes convolucionales en amplitud en escalera. Comparación de resultados utilizando distintos conjuntos de entrenamiento anotado y no anotado.

de tantos datos no anotados, quizás se requeriría una red de mayor tamaño para dicha tarea.

Comparación de resultados supervisados y semi-supervisados en la evolución de entrenamiento vs. validación por épocas

La Figura 4.8 hace una comparación del mejor modelo obtenido en la Sección 4.1.2 y el modelo obtenido con CNN-Wide-Ladder.2.

Como se puede observar, la implementación del modelo de redes convolucionales en amplitud en escalera, la curva anaranjada (validación) acompaña a una distancia más cercana a la curva azul (entrenamiento) que en el caso del modelo supervisado.

Más aún, si bien en este caso el desempeño del modelo semisupervisado sobre el conjunto de datos de validación es ligeramente menor

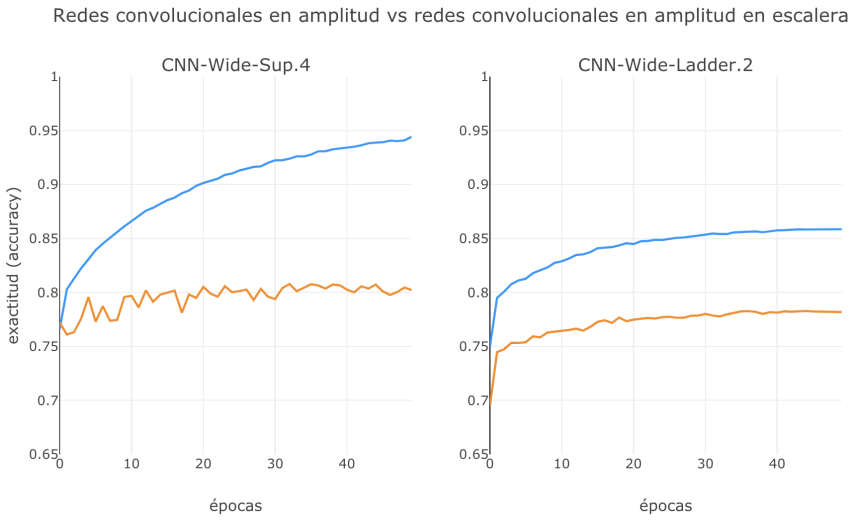


Figura 4.8: Modelo de redes convolucionales en amplitud supervisado vs semi-supervisado (escalera). Comparación de evolución en épocas del desempeño en corpus de entrenamiento y corpus de validación.

que para el modelo puramente supervisado (medido mediante la exactitud), se muestra menor ruido en el desempeño.

De manera análoga al modelo perceptrón multicapa en escalera, se comprueba que la tarea no supervisada colabora en la generalización del modelo.

Comparación de precisión y exhaustividad por clase supervisados y semi-supervisados con resultados sobre datos de evaluación

Finalmente, para una comparación más detallada, con métricas de precisión y exhaustividad (*precisión & recall*), la Figura 4.9 presenta una comparación clase por clase entre los dos modelos para el conjunto de datos de evaluación. En el gráfico, cada cuarto representa la clase a comparar (PER, LOC, ORG, y MISC, la clase O no se com-

para). Es un gráfico de barras agrupadas de acuerdo al modelo (redes convolucionales en amplitud supervisadas y no supervisadas).

La barra de color rojo representa el valor de exhaustividad que obtuvo cada modelo para cada clase en particular sobre el conjunto de datos de evaluación. La barra de color color ocre representa los valores de precisión obtenidos. No se observan grandes diferencias entre los modelos, con el modelo supervisado superando en algunos casos al semi-supervisado y viceversa. Posteriormente se utilizarán estos resultados para su comparación con el modelo de redes convolucionales en profundidad.

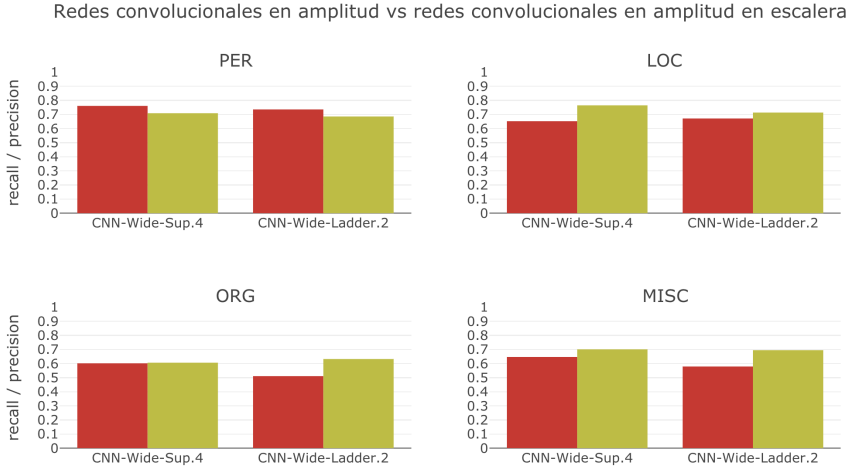


Figura 4.9: Modelo de redes convolucionales en amplitud supervisado vs semi-supervisado (escalera). Comparación de desempeño mediante precisión y exhaustividad por clase sobre conjunto de datos de evaluación.

4.2.3. Impacto de datos anotados para entrenamiento: Redes convolucionales en escalera en profundidad

En esta serie de experimentos buscamos medir el impacto del desempeño del modelo modificando la cantidad inicial de datos anotados sobre los cuales entrenar la parte supervisada del modelo.

Como caso de estudio, se utilizaron las redes convolucionales en escalera en profundidad definidas en la Sección 3.4.3, ya que en estas el desempeño de la parte supervisada del modelo era esencial puesto que no se pueden, por la naturaleza de los datos secuenciales, subsamplear la clase mayoritarias "O". Por lo que queremos ver efectivamente que tan importante es el uso de datos anotados para el desempeño general del modelo semi-supervisado.

Experimento	**Hiperparámetro**	**Valor**
Hiper. generales	Número de filtros	100
	Capas convolucionales	10
	Factor de ruido por capa	0.3
	Costo ruido capa de entrada	100
	Costo ruido capa convolucional	0.1
	Costo ruido capa de salida	0.01
	Factor costo no supervisado	0.1
	Tasa de aprendizaje	0.02
	Tamaño de kernels	2
	Datos entrenamiento no anotados	100k
CNN-Depth-Ladder-a	Datos entrenamiento anotados	25k
CNN-Depth-Ladder-b	Datos entrenamiento anotados	50k
CNN-Depth-Ladder-c	Datos entrenamiento anotados	75k
CNN-Depth-Ladder-d	Datos entrenamiento anotados	100k

Cuadro 4.6: Hiperparámetros del conjunto de experimentos con redes convolucionales en profundidad en escalera.

En los experimentos se varía la proporción de datos anotados en la etapa de entrenamiento con la promesa de que el modelo semi-supervisado puede suplir la falta de datos anotados gracias al uso de datos no anotados. En estos experimentos, al utilizarse un modelo de etiquetado de secuencias, se evalúan datos del tipo "Oración - Etiquetas" como se definen en la Sección 3.2.3. Notar que este experimento también se realizó en otras arquitecturas en escalera con resultados similares pero con fines de no ser reiterativos se decidió mostrar los resultados para esta arquitectura en particular.

En el Cuadro 4.6, se detallan los cuatro experimentos realizados, con los hiperparámetros generales que mejores resultados obtuvieron seleccionados de manera aleatoria. Lo que varía son las instancias anotadas utilizadas para el entrenamiento del modelo. No obstante, se decidió mostrar únicamente la comparación entre los modelos CNN-Depth-Ladder-a y CNN-Depth-Ladder-d que mostraban los resultados mas relevantes en cuanto a la validación de la hipótesis propuesta.

Comparación de resultados semi-supervisados al variar la proporción de datos anotados para el entrenamiento.

En la Figura 4.10 se observan los valores de *recall* (en color rojo) y de *precision* (en color verde claro) para cada una de las clases que representan una entidad nombrada sobre el conjunto de evaluación.

Se compara el modelo de redes convolucionales en profundidad en escalera utilizando 25000 instancias etiquetadas (CNN-Depth-Ladder-a) contra utilizar 100000 instancias etiquetadas (CNN-Depth-Ladder-d) para el entrenamiento del modelo.

En general, se obtiene entre 1 y 2 puntos de mejora al utilizar la totalidad de ejemplos etiquetados salvo para los casos de *precision* en la clase PER y de *recall* en la clase MISC donde incluso tomar una proporción menor (25 %) del total de instancias de entrenamiento generó mejores resultados.

Vemos entonces la ventaja de trabajar con este modelo semi-supervisado donde tomar una proporción significativamente menor de datos para el

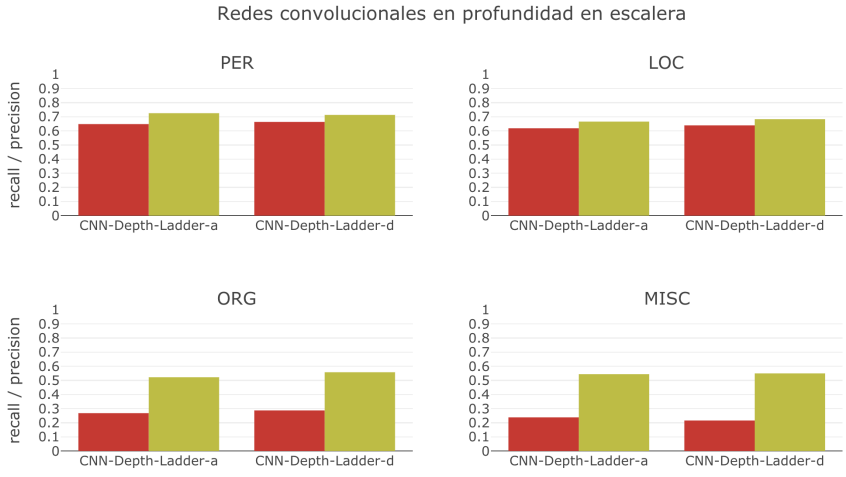


Figura 4.10: Redes convolucionales en profundidad en escalera.

entrenamiento no se corresponde con una gran pérdida de desempeño.

Comparación de mejores resultados de los modelos de redes convolucionales en amplitud y profundidad en escalera.

En la Figura 4.11 se comparan los mejores resultados de los modelos de redes convolucionales en amplitud y en profundidad ambos en su variante en escalera.

El color rojo se utiliza para los valores de *recall* y el verde claro para los valores de *precision* alcanzados por cada uno de los modelos en sus correspondientes conjuntos de evaluación.

El modelo de redes convolucionales posee un desempeño superior visible en todas las clases y de manera más notable en las categorías ORG y MISC, sólo en el caso de *precision* para la categoria PER el modelo de redes convolucionales en profundidad es ligeramente superior.

Si bien esto es algo que hay que analizar con cuidado puesto que

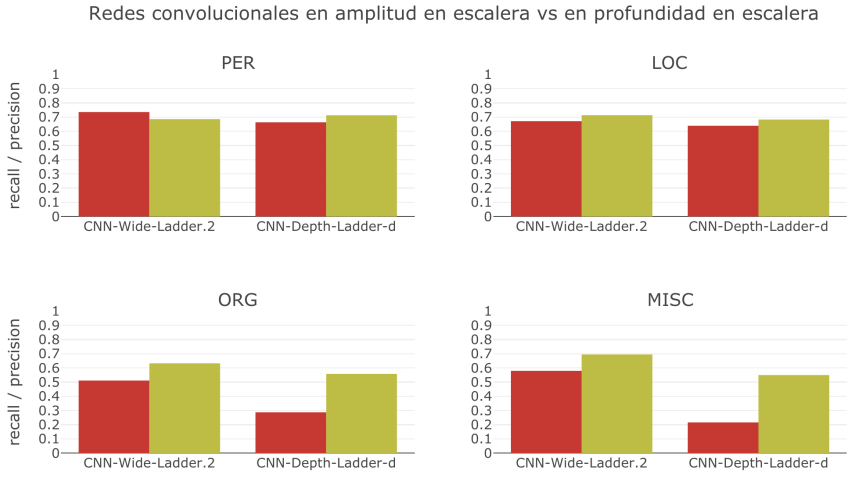


Figura 4.11: Redes convolucionales en amplitud en escalera vs en profundidad en escalera.

los datos no son de la misma naturaleza, una conclusión que puede obtenerse de este resultado está en el ámbito de la restricción de no poder utilizar ninguna técnica de submuestreo en el caso de las redes convolucionales en profundidad porque ello ocasionaba problemas con la estructura de la clasificación de secuencias.

4.2.4. Impacto del costo no supervisado en el aprendizaje del modelo: Redes convolucionales en escalera en profundidad

Una hipótesis que surgió a lo largo de este trabajo fue si brindar demasiada importancia a la tarea no supervisada perjudica el entrenamiento del modelo ya que el mismo se concentraría en la reconstrucción de la entrada en lugar de la tarea prioritaria de predicción.

Para verificar esta hipótesis, se agregó un factor μ, detallado en 3.4.4, que nos permitiera ajustar el peso del costo no supervisado y

Experimento	**Hiperparámetro**	**Valor**
Hiper. generales	Número de filtros	100
	Capas convolucionales	10
	Factor de ruido por capa	0.3
	Costo ruido capa de entrada	100
	Costo ruido capa convolucional	0.1
	Costo ruido capa de salida	0.01
	Factor costo no supervisado	0.1
	Tasa de aprendizaje	0.02
	Tamaño de kernels	2
	Datos entrenamiento no anotados	100k
u_cost 0	Valor de μ	0
u_cost 0.1	Valor de μ	0.1
u_cost 0.3	Valor de μ	0.3
u_cost 0.5	Valor de μ	0.5

Cuadro 4.7: Hiperparámetros del conjunto de experimentos con redes convolucionales en profundidad en escalera.

de esa manera poder analizar que tanta importancia tiene este peso en el desarrollo final de la red.

Comparación de las curvas de loss durante el entrenamiento del modelo.

En la Figura 4.12 se observan los resultados de cuatro experimentos realizados sobre el modelo de redes convolucionales en profundidad en escalera. Dichos experimentos se encuentran detallados en el Cuadro 4.7. La variación en estos experimentos es sobre el valor de u_cost, que es un factor multiplicativo aplicado a la función de costo no supervisada del modelo (denotado como μ en la sección 3.4.4).

A mayor u_cost, mayor es la ponderación que se le da a la función de costo no supervisada que el modelo debe minimizar conjuntamente con la supervisada.

En el gráfico, el color azul representa el valor de pérdida (*loss*) supervisado del modelo a lo largo de las épocas sobre el conjunto de entrenamiento mientras que el anaranjado sobre el conjunto de validación.

Lo que se puede observar es como luego de cierto valor sobre el parámetro μ, en este caso de 0.5, esto afecta directamente el desempeño de la optimización sobre la función de costo supervisada. Luego, la hipótesis de que el modelo se comienza a centrar más en la tarea de reconstrucción y dando menos importancia a la tarea supervisada de clasificación, es efectivamente cierta.

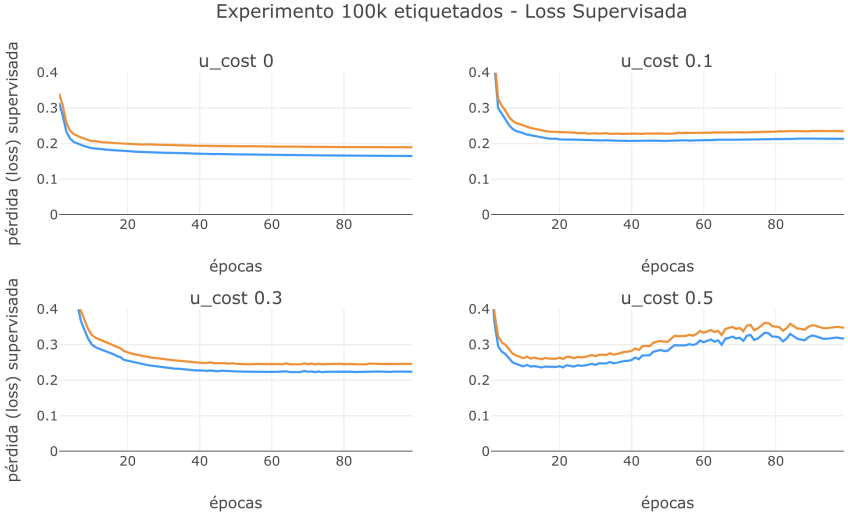


Figura 4.12: Redes convolucionales en escalera experimento CNN-Depth-Ladder-d con variando el factor de la función de costo no supervisado.

4.2.5. Resultados finales

En los Cuadros 4.8 y 4.9 se comparan los mejores resultados utilizando la métrica F1-score de los modelos convolucionales. Se observa como los modelos supervisados superan a sus variantes en escalera y que la arquitectura del modelo de redes convolucionales en amplitud performa mejor al modelo de redes convolucionales en profundidad.

	PER	LOC	ORG	MISC	O
CNN wide supervised	**0.9849**	**0.9751**	**0.9775**	**0.9753**	**0.9909**
CNN wide ladder	0.8152	0.8137	0.7453	0.7194	0.9269
CNN depth supervised	0.7923	0.7847	0.6953	0.6173	0.9837
CNN depth ladder	0.6994	0.6628	0.4025	0.3261	0.9732

Cuadro 4.8: F1-score de los mejores modelos por clase sobre los conjuntos de **entrenamiento** de cada uno de ellos.

	PER	LOC	ORG	MISC	O
CNN wide supervised	**0.7341**	0.7042	**0.6039**	**0.6726**	0.9325
CNN wide ladder	0.71	0.6921	0.565	0.6317	0.906
CNN depth supervised	0.7326	**0.7089**	0.5828	0.4897	**0.9789**
CNN depth ladder	0.6876	0.6605	0.3791	0.3099	0.9696

Cuadro 4.9: F1-score de los mejores modelos por clase sobre los conjuntos de **evaluación** de cada uno de ellos.

Las Figuras 4.13 y 4.14 muestran los valores de F1-score para los datos de entrenamiento (en color azul) y evaluación (en color verde) comparando los modelos supervisados y en escalera de los modelos que utilizan las capas convolucionales en amplitud y en profundidad. En ambos casos nuevamente se observa la capacidad de generalización que aporta la tarea no supervisada de los modelos en escalera donde los valores obtenidos sobre el conjunto de evaluación son cercanos al conjunto de entrenamiento.

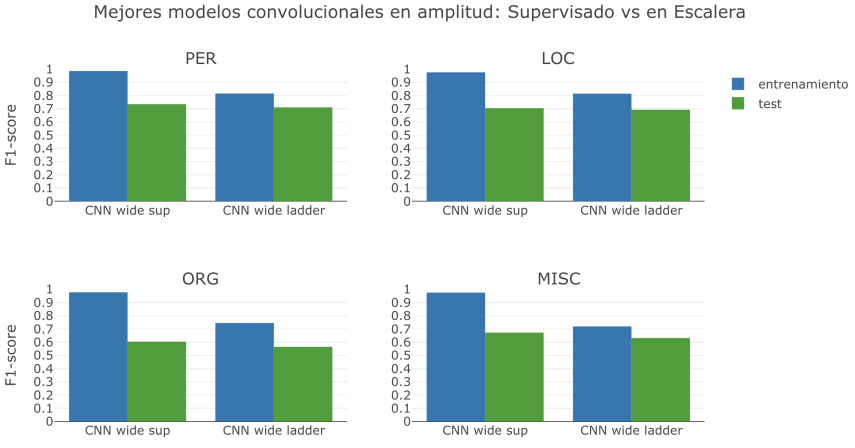


Figura 4.13:

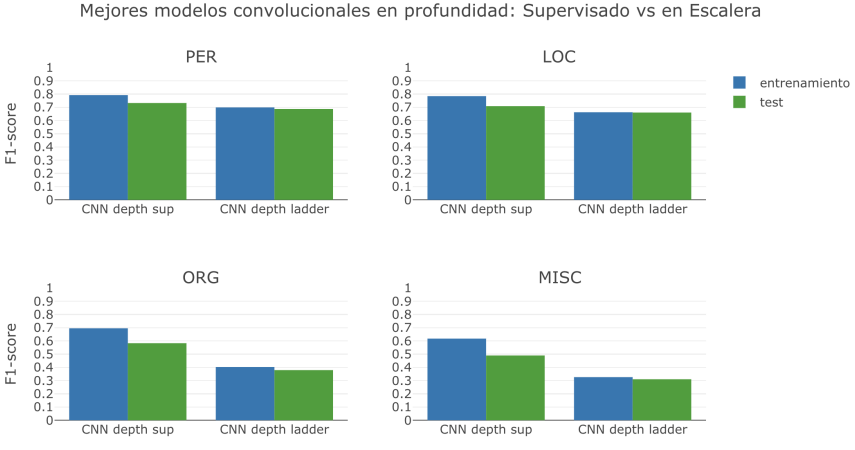


Figura 4.14:

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

Una de las primeras problemáticas estudiadas fue la representación vectorial de una palabra dada su semántica en el contexto en el cual ocurre. La estrategia de decaimiento exponencial es la que mejor performa por sobre el resto de estrategias propuestas en el trabajo de (Iacobacci et al., 2016). Ponderar en mayor medida a las palabras inmediatamente cercanas a la palabra objetivo tiene sentido, ya que en general aporta mayor información que una palabra distante.

A lo largo del trabajo también se experimentaron modelos basados en redes convolucionales, con el afán de capturar información del contexto de cada palabra al aplicar capas convolucionales. Los resultados determinaron que aplicar convoluciones para extraer características del entorno de una palabra es una alternativa exitosa que supera a la estrategia de decaimiento exponencial.

Al incorporar la tarea no supervisada que proponen las redes neuronales en escalera se observó en distintos experimentos una mayor generalización de estos modelos respecto a los netamente supervisa-

dos. Esta mayor generalización puede ser consecuencia de haber compartido las representaciones ocultas de la red entre más de una tarea.

Agregar una cierta cantidad de datos no anotados al entrenamiento del modelo de redes convolucionales en amplitud en escalera parece ser clave para mejorar el rendimiento del mismo. Sin embargo, se debe tener cuidado de no excederse ya que la red puede no ser lo suficientemente capaz de realizar la tarea no supervisada de reconstrucción de tantos datos no anotados y afectar negativamente al modelo en cuestión. A su vez, es importante regular el peso que se le asigna a la función de costo no supervisada. Si se excede, el modelo diverge ya que se centra demasiado en la tarea de reconstrucción dejando de lado la tarea supervisada de clasificación.

Sobre el modelo de redes convolucionales en profundidad se observó que entrenando el modelo con la totalidad de instancias de entrenamiento y utilizando solo un 25 % se obtienen resultados similares. Vemos entonces la ventaja de trabajar con este modelo semi-supervisado donde tomar una proporción significativamente menor de datos no se corresponde con una gran pérdida de desempeño.

Finalmente, comparando los mejores resultados de cada modelo concluimos los modelos supervisados superan a sus variantes en escalera y que la arquitectura del modelo de redes convolucionales en amplitud tiene mejor desempeño que el modelo de redes convolucionales en profundidad.

5.2. Trabajo futuro

Una metodología interesante para la representación de palabras son los modelos de embeddings de caracteres, en donde el vector representante de una palabra se construye a partir de los *n-grams* de caracteres que la componen. Dado que los caracteres son compartidos entre palabras, estos modelos de *character embeddings* funcionan mejor para representar palabras que están fuera del vocabulario en cuestión. En contraste, modelos de *word embeddings*, como por ejem-

plo el modelo Word2Vec utilizado en este trabajo de tesis no pueden representar una palabra fuera del vocabulario ya que tratan a cada palabra de forma atómica. Otro aspecto a tener en cuenta es que los modelos de *character embeddings* tienden a generar mejores representaciones para aquellas palabras que aparecen con poca frecuencia, ya que los *n-grams* de caracteres que se comparten entre las palabras aún pueden aprender buenos embeddings. Por otro lado, los modelos de *word embeddings* sufren la falta de oportunidad de entrenamientos suficientes para palabras poco frecuentes.

En el trabajo de (Ghaddar and Langlais, 2017) se evalúa el desempeño de distintos modelos sobre el conjunto de datos WiNER. En particular, el modelo LSTM-CRF del trabajo de (Huang et al., 2015) obtiene los mejores resultados. Este modelo combina el modelo LSTM (Long Short-Term Memory), una red neuronal del tipo recurrente y el método de modelado estadístico CRF (Conditional Random Field). Ambos modelos pertenecen a la familia de modelado de secuencias ya que por su composición les permite lidiar con problemas de series temporales. Surge entonces la idea de combinar las redes convolucionales en escalera estudiadas en este trabajo de tesis con el modelo LSTM-CRF donde el primero se utilizaría para la extracción de features y el segundo encargado de la tarea de clasificación.

Finalmente, como alternativa y en pos de conseguir mejores representaciones de palabras nos gustaría explorar modelos del estado del arte como ELMo (*Embeddings from Language Models*) del trabajo de (Peters et al., 2018) y BERT (*Bidirectional Encoder Representations from Transformers*) del trabajo de (Devlin et al., 2018).

Bibliografía

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Ghaddar, A. and Langlais, P. (2017). Winer: A wikipedia annotated corpus for named entity recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

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, Berlin, Germany. Association for Computational Linguistics.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *CoRR*, abs/1802.05365.

Rasmus, A., Valpola, H., Honkala, M., Berglund, M., and Raiko, T. (2015). Semi-supervised learning with ladder network. *CoRR*, abs/1507.02672.

Suddarth, S. C. and Kergosien, Y. L. (1990). Rule-injection hints as a means of improving network performance and learning time. In *Neural Networks, EURASIP Workshop 1990, Sesimbra, Portugal, February 15-17, 1990, Proceedings*, pages 120–129.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Valpola, H. (2014). From neural PCA to deep unsupervised learning. *CoRR*, abs/1411.7783.

Zhong, Z. and Ng, H. T. (2010). It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, ACLDemos '10, pages 78–83, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zipf, G. (1949). *Human Behavior and the Principle of Least Effort*. Addison–Wesley, Cambridge, MA.