Aprendizaje profundo para el procesamiento del lenguaje natural

José Ángel González Barba
Universidad Politécnica de Valencia

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

IARFID

Valencia 2017

Dirigido por:
Lluís Felip Hurtado Oliver
Ferran Pla Santamaría

"Daría todo lo que sé por la mitad de lo que ignoro."
René Descartes

Abstract

The work carried out is part of natural language processing area and it is focused specifically on text classification problems like *sentiment analysis* and *topic detection* about content published on *Twitter*, using deep learning models such as convolutional and recurrent neural networks.

In order to estimate the parameters of these models in problems where the input is a sequence of linguistic units, a representation that retains the most discriminative information for a given task (contextual information, semantic, syntactic, etc.) is required.

For this reason, in the present work, suitable representations for the addressed tasks are proposed and comparisons are made among the different representations, as well as among the different models of neural networks that use these representations.

All the tasks addressed have been proposed in workshops of national and international congresses such as SemEval and IberEval. Furthermore, in these tasks we have obtained competitive results, reaching a 1st place in the tasks COSET and *Gender* (*Ibereval 2017@SEPLN*), a 4th place in *Stance* (*Ibereval 2017@SEPLN*) and a 3rd place in *Sentiment Analysis in Twitter* (*SemEval 2017@ACL*).

Keywords: Deep learning, Text classification, IberEval, SemEval.

Resumen

El trabajo realizado se enmarca en el área del procesamiento del lenguaje natural y se centra, concretamente, en problemas de clasificación de texto como *sentiment analysis* y *topic detection* sobre contenido publicado en *Twitter*, haciendo uso de modelos basados en aprendizaje profundo como redes convolucionales y redes recurrentes.

Para poder estimar los parámetros de estos modelos, en problemas donde la entrada es una secuencia de unidades lingüísticas, se requiere una representación que retenga la información más discriminativa para una tarea determinada (información contextual, semántica, sintáctica, entre otras).

Por este motivo, en el presente trabajo se proponen representaciones adecuadas para las tareas tratadas y se realizan comparaciones entre las diferentes representaciones, así como entre los diferentes modelos de redes neuronales que hacen uso de dichas representaciones.

Todas las tareas abordadas han sido propuestas en talleres de congresos nacionales e internacionales como SemEval e IberEval. Además, en dichas tareas se han obtenido resultados competitivos, llegando a alcanzar un 1º puesto en las tareas COSET y *Gender* (*Ibereval 2017@SEPLN*), un 4º puesto en *Stance* (*Ibereval 2017@SEPLN*) y un 3º puesto en *Sentiment Analysis in Twitter* (*SemEval 2017@ACL*).

Palabras clave: Aprendizaje profundo, Clasificación de texto, IberEval, SemEval.

Resum

El treball realitzat s'emmarca en l'àrea del processament del llenguatge natural i es centra, concretament, en problemes de classificació de text com *sentiment analysis* i *topic detection* sobre contingut publicat en *Twitter*, fent ús de models basats en aprenentatge profund com xarxes convolucionals i xarxes recurrents.

Per poder estimar els paràmetres d'aquests models, en problemes on l'entrada és una seqüència d'unitats lingüístiques, es requereix una representació que retinga la informació més discriminativa per a una determinada tasca (informació contextual, semàntica, sintàctica, entre altres).

Per aquest motiu, en el present treball es proposen representacions adequades per las tasques tractades i es realitzen comparacions entre les diferents representacions, així com entre els diferents models de xarxes neuronals que fan ús d'aquestes representacions.

Totes les tasques abordades han sigut propostes en tallers de congressos nacionals i internacionals com SemEval i IberEval. A més, en aquestes tasques s'han obtingut resultats competitius, aconseguint un 1r lloc en les tasques COSET i *Gender* (*Ibereval 2017@SEPLN*), un 4t lloc en *Stance* (*Ibereval 2017@SEPLN*) i un 3r lloc en *Sentiment Analysis in Twitter* (*SemEval 2017@ACL*).

Paraules clau: Aprenentatge profund, Classificació de text, IberEval, SemEval.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

> La ciencia se trata de saber, la ingeniería de hacer.
>
> Henry Petroski

1.1. Proyecto

El trabajo fin de máster realizado se enmarca dentro del proyecto *Audio Speech and Language Processing for MULtimedia ANalytics* (ASLP-MULAN) [84] y tiene como objetivo la formación del alumno y el desarrollo e implementación de técnicas para problemas relacionados con el área de *Procesamiento del Lenguaje Natural* (PLN), de forma que éstas se puedan generalizar y aplicar al proyecto ASLP-MULAN.

Concretamente, el proyecto ASLP-MULAN tiene como intención generar la combinación adecuada de tecnologías de audio, habla y lenguaje para tratar con grandes cantidades de datos. Entre algunas de las tecnologías que se emplean en dicho proyecto destacan el procesamiento del lenguaje natural, la transcripción automática del habla o la descripción de contenidos multimedia no estructurados [84].

Así, en el proyecto fin de máster se trata con profundidad el primero de los casos, PLN. Para esto, se ha participado en talleres propuestos en diferentes congresos, que tratan problemas de clasificación de texto (*text classification*) como análisis de sentimientos (*sentiment analysis*) o detección de temas (*topic detection*), mediante otro de los puntos clave del presente proyecto, modelos basados en redes neuronales, concretamente en aprendizaje profundo (*deep learning*).

Sin embargo, a pesar de que no se discute en la memoria, también se han desarrollado e implementado aplicaciones para tratar algunos problemas adicionales de utilidad en el proyecto ASLP-MULAN. Algunas de las aplicaciones son: bots conversacionales y descripción de imágenes mediante modelos *sequence-to-sequence* [73], generación de imágenes mediante *Generative Adversarial Networks* [28] o modelos de lenguaje recurrentes para generación de texto [2].

1.2. Clasificación de texto

La tarea de clasificación de texto, del inglés *text classification*, consiste en asignar un conjunto de clases a un determinado documento. Formalmente, dado un documento d y un conjunto fijo de clases $C = \{c_1, c_2, ..., c_n\}$ un clasificador f debe asignar al documento su clase (o clases) correcta, $f(d) = c_d^*$. Previamente a dicha clasificación, se suelen emplear enfoques supervisados en los que dado un conjunto de entrenamiento de m documentos etiquetados, se aprende un clasificador f.

Así, los documentos a clasificar pueden ser de tipos muy diversos como texto, audio, imágenes, vídeos, etc., sin embargo, todos los problemas abordados en el presente proyecto consideran documentos de texto y, más concretamente, *tweets*.

Debido a la utilización de textos de Twitter, la tarea se hace más compleja ya que no siempre se trata con textos gramaticales tanto a nivel léxico como sintáctico, es decir, hay ciertas características peculiares de Twitter como palabras alargadas, eliminaciones de caracteres, emoticonos, urls, hashtags, etc., que dificultan el tratamiento lingüístico de textos.

Con respecto a la aplicabilidad de los problemas de *text classification*, éstos tienen gran interés en la actualidad en el análisis, principalmente, de contenido alojado en redes sociales, medios de comunicación, buscadores, etc. Algunos de los problemas más significativos son *topic detection* [22] [25], *sentiment analysis* [65] [27], *stance detection* [75] [26] (minería de opiniones en general) o incluso detección de tendencias políticas [60] y seguimiento *online* de radicalización [10].

Por último, el gran interés que suscitan las tareas de *text classification* se ve reflejado en diferentes tareas propuestas en talleres de congresos que versan sobre *text*

classification como SemEval@ACL, IberEval@SEPLN y TASS@SEPLN, cuyas ediciones de 2017 han sido abordadas en el presente proyecto.

1.3. Aprendizaje profundo

El aprendizaje profundo, del inglés *deep learning*, constituye una de las áreas de investigación del aprendizaje automático más interesantes en la actualidad y que mejores resultados está obteniendo. Generalmente, no existe una definición clara sobre lo que dicho término significa, sin embargo, es comúnmente aceptado que se trata de modelos de aprendizaje automático basados en redes neuronales con una gran cantidad de capas no lineales, cada una de las cuales aprende un nivel de abstracción más alto que sus anteriores, pudiendo llegar a abstraer estructuras complejas como formas, semántica, etc.

El área de las redes neuronales estuvo en boga desde el 1943, cuando W. McCulloch y W. Pitts propusieron empíricamente el primer modelo sobre el funcionamiento de una neurona natural [50], hasta mediados de la decada de los 90, unos años después de que D. Rumelhart y G. Hinton redescubriesen el algoritmo de aprendizaje *Backpropagation*. Sin embargo, debido a la falta de potencia de cómputo y a la imposibilidad de entrenar modelos muy profundos, éstas se abandonaron hasta el año 2006, cuando G. Hinton, entre otros, publicaron un método efectivo para entrenar redes neuronales profundas [34].

Desde entonces, las técnicas utilizadas para entrenar modelos profundos han cambiado bastante, tal como se verá en el segundo capítulo de este documento, aún así, la filosofía y el interés en las redes neuronales profundas se mantiene, llegando a alcanzar abstracciones cada vez de más alto nivel para modelar estructuras y comportamientos complejos en los datos.

Así, en la actualidad, debido a los nuevos enfoques para entrenar redes neuronales profundas, a la capacidad de cómputo y a la cantidad de datos existentes, este tipo de modelos de *deep learning* están teniendo mucha repercusión actualmente para proponer soluciones a problemas complejos como la segmentación de objetos en imágenes o vídeos [61], la descripción de escenas [79] o la extracción de la semántica/contexto

de palabras y frases [51], [53], [45], [40].

Por este motivo, durante el proyecto realizado se han empleado profusamente este tipo de modelos, obteniendo resultados muy competitivos en las tareas de PLN abordadas como *sentiment analysis*, *topic detection* o análisis de sentimiento a nivel de aspectos (*stance detection*), tal como se muestra en los capítulos cuatro y cinco de la presente memoria.

1.4. Metodología

En el presente proyecto se abordan problemas de *text classification*, por lo que es necesario emplear representaciones de texto y modelos que manejen dichas representaciones para llevar a cabo la clasificación. Los modelos utilizados están basados en *deep learning* ya que obtienen resultados competitivos en problemas de clasificación [65].

Entre los modelos de *deep learning* utilizados destacan el perceptrón multicapa, las redes convolucionales y las redes recurrentes. Además, se ha experimentado con una gran variedad de representaciones como *bag of words* (a nivel de n-gramas de palabras y caracteres), vectores *one-hot*, *embeddings* y *specific word embeddings*. Todas estas técnicas se discuten con detalle en los capítulos dos y tres del presente documento y se han empleado *frameworks* como Keras [14] y Scikit-learn [59], que implementan o permiten implementar algunas de las técnicas mencionadas.

Por otro lado, el método seguido para llevar a cabo las experimentaciones consiste, en primer lugar, en analizar ciertos aspectos del corpus como el número de muestras, la talla del vocabulario o la distribución de las muestras en cada clase, para, en función de éstos, tomar una serie de decisiones iniciales como la topología de la red, el número de capas y de neuronas por capa, entre otros.

En base a dichas decisiones, se emplean representaciones y modelos simples y diferentes con el objetivo de determinar un subconjunto de los parámetros que mayor influencia tienen en la discriminación (e.g. el tipo de representación y algunos hiperparámetros de los modelos como el algoritmo de aprendizaje).

Una vez se determinan estos parámetros se realizan más pruebas, generalmente variando hiperparámetros de los modelos (ajuste), haciendo uso de una partición de validación extraída del conjunto de entrenamiento de cada tarea (*hold-out*), con el tipo de representación y algunos parámetros ya fijados.

Por último, se realiza un estudio de los errores cometidos por los clasificadores, ajustados tras la fase de ajuste junto con la mejor representación escogida, en casos peculiares que dificultan la clasificación como por ejemplo la ironía, el sarcasmo o la negación en problemas de *sentiment analysis*.

1.5. Estructura de la memoria

La memoria realizada sobre el trabajo fin de máster está dividida en siete capítulos.

- **Capítulo 1**. En el capítulo se expone una introducción al proyecto realizado y algunos aspectos de este, a destacar, los problemas de clasificación de texto tratados, la metodología empleada para llevar a cabo el trabajo experimental y el enfoque principal para abordar las tareas (aprendizaje profundo).

- **Capítulo 2**. Trata, con cierta profundidad, el área de las redes neuronales y el aprendizaje profundo, concretamente, se enuncia la teoría subyacente de los modelos basados en redes neuronales que han sido utilizados en los talleres abordados, así como algunos problemas que se derivan de la utilización de éstos y las técnicas utilizadas para solucionarlos.

- **Capítulo 3**. Este capítulo estudia diferentes representaciones para palabras y frases. En concreto, *embeddings*, *specific word embeddings*, secuencias de vectores *one-hot* y estrategias clásicas tales como *bag-of-words*.

- **Capítulo 4**. En el cuarto capítulo se describe la participación del equipo ELiRF-UPV en la tarea 4 de SemEval2017@ACL [27] en la que se emplea un enfoque basado en el uso de tres *stacks* de redes convolucionales y recurrentes, así como en la combinación de *embeddings* específicos (extraídos de la tarea, *in-domain*)

y generales (extraídos de conjuntos externos, *out-domain*) con lexicones de polaridad.

- **Capítulo 5**. En este capítulo se discute la participación del equipo ELiRF-UPV en las tareas *Classification Of Spanish Election Tweets* (COSET) [25] y *Stance and Gender Detection in Tweets on Catalan Independence* de IberEval2017@SEPLN [26]. En este caso, los enfoques han variado en función de la tarea abordada, estando basados, generalmente, en modelos de redes neuronales que han obtenido resultados muy competitivos.

- **Conclusiones**. Trata algunas conclusiones obtenidas tras haber realizado completamente el trabajo fin de máster y otras tareas contempladas por el proyecto ASLP-MULAN.

- **Trabajo futuro**: En el último capítulo se proponen un conjunto de propuestas y posibles mejoras a aplicar, sobre los métodos desarrollados en este proyecto, para llevar a cabo trabajos futuros que estudien y planteen soluciones a problemas similares a los tratados en el presente documento.

Capítulo 2

Redes neuronales y aprendizaje profundo

> Siempre he creído en los números. En las ecuaciones y la lógica que llevan a la razón. Pero, después de una vida de búsqueda me digo, ¿qué es la lógica? ¿quién decide la razón?
>
> John Nash

2.1. Introducción

Las redes neuronales son modelos no paramétricos que forman parte de los planteamientos conexionistas basados en funciones discriminantes. Este tipo de modelos están inspirados en cerebros biológicos. Sus componentes simulan las interacciones entre los axones, que emiten impulsos nerviosos, y las dendritas, que reciben los impulsos nerviosos. Cuando se supera un cierto umbral de actividad, alimentan el soma de la neurona para que su axón emita un nuevo impulso nervioso.

Este tipo de modelos ha demostrado buen comportamiento en problemas muy complejos. Principalmente, en problemas relacionados con actividades sensitivas de los seres humanos, como por ejemplo, la segmentación de objetos en imágenes o videos [61], la descripción de escenas [79] o la extracción de la semántica/contexto de palabras y frases [51] [53], [45], [40].

Destacar que el uso de modelos basados en redes neuronales y en *deep-learning* constituye una de las principales aportaciones de este proyecto, debido a que se han

utilizado profusamente en todas las tareas de *text classification* abordadas, obteniendo en algunas de ellas resultados muy competitivos. Sin embargo, es cierto que en determinadas tareas, típicamente caracterizadas por un conjunto reducido de muestras, otros modelos como *Support Vector Machine* (SVM) [16] han alcanzado resultados ligeramente superiores.

En las siguientes secciones se realiza una introducción a algunos de los modelos y técnicas, basadas en redes neuronales, utilizadas en el trabajo y que han dado lugar a diversas participaciones en *workshops* de *text classification* y a la realización de algunas tareas del proyecto ASLP-MULAN.

2.2. Perceptrón

Se conoce como perceptrón a cualquier clasificador lineal $c(x) = \underset{c}{\operatorname{argmax}}\, g_c(x)$ donde $g_c(x)$ es función discriminante lineal (e.g. $g_c(x) = w_c^t x + w_{c0}$) asociada a la clase c, aprendida mediante el algoritmo Perceptron [64].

Así, podemos ver un perceptrón como una red neuronal de cero capas ocultas con función de activación lineal ($w_c^t x + w_{c0} \equiv f(w_c^t x + w_{c0})$ si f es una función lineal). Esto se puede observar más claramente en la Figura 2.1.

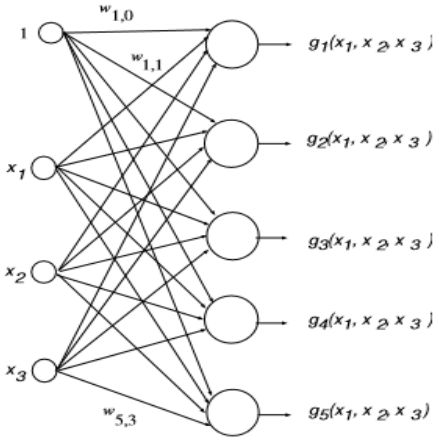

Figura 2.1: Arquitectura de un perceptrón [8].

Este modelo se puede extender a funciones discriminantes no lineales si combinamos modelos perceptrón y empleamos funciones de activación no lineales al resultado

de la preactivación $w_c^t x + w_{c0}$. Algunas de las funciones de activación más utizadas son *sigmoid*, *hyperbolic tangent* y *Rectified Linear Unit* (ReLU). En la siguiente sección se presenta el perceptrón multicapa, resultado de combinar varios modelos perceptrón.

2.3. Perceptrón multicapa

La potencia del modelo perceptrón se puede extender si se añaden más capas a la arquitectura original (considerando siempre conexiones hacia delante, redes *feed-forward*). En la Figura 2.2 se muestra un ejemplo de un perceptrón multicapa (MLP) de una capa oculta.

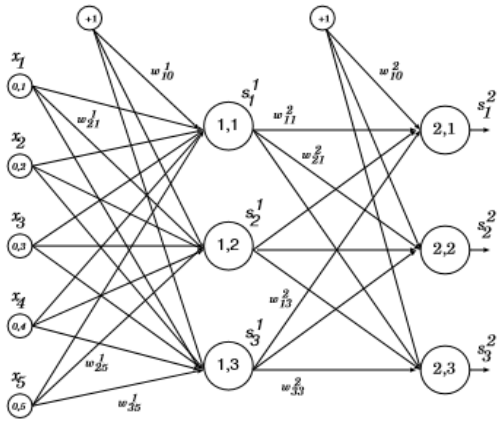

Figura 2.2: Arquitectura de un perceptrón multicapa de una capa oculta [8].

En este caso, la función discriminante que aprende el MLP de una capa oculta para cada clase c es $g_c(x;\theta) \equiv s_c^2(x) = f(\sum_{j=0}^{M_1} w_{kj}^2 s_j^1(x))$, donde x es la entrada, s_j^i es la salida de la neurona j de la capa i (calculado como en el perceptrón simple, aplicando a la salida una función de activación, f, potencialmente no lineal) y w_{jk}^i es el peso que conecta la neurona k de la capa i con la neurona j de la capa $i-1$ ($\theta \equiv \mathbf{w}$). Por tanto, desarrollando $s_j^1(x)$, la expresión anterior queda como sigue, $g_c(x;\theta) \equiv s_c^2(x) = f(\sum_{j=0}^{M_1} w_{kj}^2 f(\sum_{j'=0}^{M_0} w_{jj'}^1 x_{j'}))$.

Como se puede observar, este modelo es fácilmente generalizable a N capas ocultas y el problema del entrenamiento se plantea de la misma manera. Dado un conjunto de entrenamiento $A = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ con $x_i \in \mathbb{R}^{M_0}$, $y_i \in \mathbb{R}^{M_N}$, se deben

encontrar los pesos $\mathbf{w} : s^N(x_j) = y_j, 1 \leq j \leq |A|$.

Para abordar el problema del entrenamiento, se hace uso del algoritmo *Backpropagation* [66], que permite calcular el gradiente de la función de pérdida o *loss* (métrica empleada para medir la bondad de las predicciones) con respecto a los pesos de la red y, con ello, optimizar dichos pesos en función del gradiente. Este algoritmo se emplea intensivamente en el entrenamiento de este tipo de modelos, existiendo diferentes variantes del algoritmo para otros tipos de redes como *Backpropagation Through Time* (BPTT) para redes neuronales recurrentes, que se discutirán en apartados posteriores.

2.4. Procesamiento de secuencias

Con lo visto hasta ahora únicamente podemos procesar como entrada, vectores de una dimensionalidad fija i.e. $\mathbf{x} \in \mathbb{R}^d$, pero qué ocurre cuando los datos de entrada son secuencias de elementos (generalmente vectores), i.e. $\mathbf{x} = \{x_1, x_2, ..., x_n\}, x_i \in \mathbb{R}^d, 1 \leq i \leq n$, como es el caso de problemas de PLN.

Una solución trivial para plantear el problema consiste en transformar cada secuencia de elementos de talla n y dimensionalidad d en un único vector $x \in \mathbb{R}^{n \cdot d}$ mediante algún tipo de transformación, e.g. suma de los elementos de una secuencia dada (es importante notar que, en este caso, la solución planteada conlleva una pérdida de información temporal).

Con ello, para abordar el problema, tal como se comentará detalladamente en el próximo capítulo, podemos plantear dos enfoques: representar los documentos con un único vector que contenga información sobre las palabras de dicho documento (representación vectorial) o representarlos mediante una secuencia de elementos que identifiquen las palabras que lo componen (representación secuencial).

En el uso de la representación secuencial de nuestros documentos, podemos plantear dos enfoques diferentes para procesar dicho tipo de representación. Por un lado, utilizar, de alguna manera diferente, los modelos ya conocidos (perceptrón y MLP) o por otro lado, plantear nuevos modelos que permitan manejar de forma natural una

representación secuencial.

Con respecto a los tipos de problemas secuenciales a resolver y siguiendo a [2], podemos encontrar tres tipos diferentes en función de la cardinalidad de la salida esperada. Éstos se representan gráficamente en la Figura 2.3:

1. $N \rightarrow 1$: en el presente proyecto, surge en problemas de *text classification* donde se clasifican secuencias en una clase determinada.

2. $N \rightarrow N$: en este tipo de problemas se debe generar una secuencia de salida con la misma longitud que la secuencia de entrada, es típico en problemas semejantes a *Part Of Speech* (POS) *tagging* (en el marco del proyecto ASLP-MULAN se ha desarrollado un POS *tagger* basado en redes recurrentes bidireccionales, aunque se ha excluido de la presente memoria).

3. $N \rightarrow M$: a diferencia del tipo anterior, la salida no necesariamente debe tener la misma longitud que la entrada. Algunos problemas de este tipo son la descripción de imágenes o vídeos y el modelado conversacional (de nuevo, en el marco de ASLP-MULAN, se ha desarrollado un bot conversacional usando técnicas basadas en *sequence-to-sequence* [78] que se ha excluido de este documento).

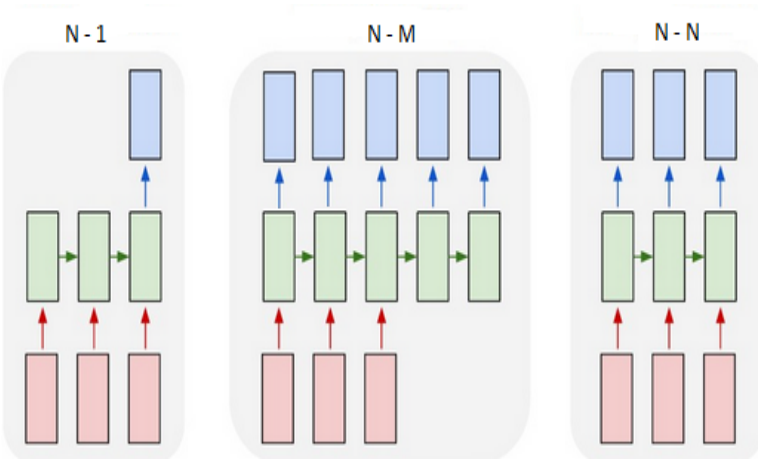

Figura 2.3: Tipos de problemas secuenciales [2].

Estos problemas se pueden resolver con algunas técnicas que se discuten con detalle en las siguientes subsecciones, desde redes dinámicas que emplean MLP sobre la secuencia de entrada, redes recurrentes y algunas variantes de estas, capaces de resolver algunos problemas relacionados con el *vanishing gradient*, hasta redes convolucionales que nos permiten extraer características de alto nivel de las secuencias de

entrada para realizar una posterior clasificación.

2.4.1. Redes dinámicas

Se conoce como redes dinámicas al método empleado para aprender, mediante MLP, de un comportamiento dependiente del tiempo a partir de un conjunto de datos compuesto por un conjunto de secuencias $\mathbf{x} = \{x_1, x_2, ..., x_n\}, x_i \in \mathbb{R}^d, 1 \leq i \leq n$ [9].

En este proyecto consideraremos como redes dinámicas todas aquellas redes neuronales que procesen secuencias y no consideren para cada elemento x_i, la salida obtenida con el elemento x_{i-j} o x_{i+j} con $j \geq 1$. De forma general, las redes dinámicas carecen de lo que se conoce como *estado interno*, por este motivo se distingue entre redes dinámicas y redes recurrentes.

El comportamiento dinámico de la red se puede conseguir con los modelos ya vistos, si se procesa cada elemento de la secuencia independientemente de elementos anteriores o posteriores, i.e. cada vector de la secuencia pasa a considerarse como una muestra más, independiente del resto. Un ejemplo detallado de este proceso se muestra en la Figura 2.4.

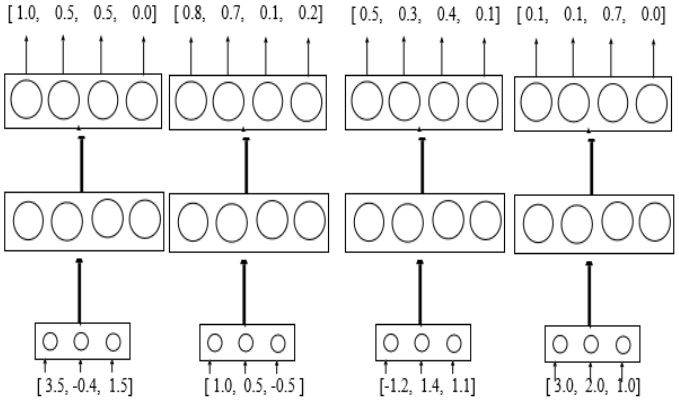

Figura 2.4: Ejemplo de redes dinámicas [9].

Es conveniente notar que este tipo de redes son útiles en problemas $N \rightarrow N$, siendo incapaces de resolver, de forma natural, los problemas $N \rightarrow M$ y $N \rightarrow 1$.

Se han propuesto algunas generalizaciones de este modelo para, principalmente, considerar más elementos de la secuencia a la hora de obtener una determinada salida. Una de estas técnicas se discute en la siguiente subsección (aunque existen otras, e.g. *time-delayed networks*).

2.4.1.1. Net-Talk

En primer lugar, una generalización del modelo anterior consiste en emplear varios elementos de la secuencia de entrada para generar una salida, similar a n-gramas de elementos de la secuencia. De esta forma, los n-gramas de elementos se concatenan y se procesan juntos (nótese que no se considera la salida de elementos anteriores y/o posteriores).

Este modelo se conoce como Net-Talk y fue propuesto en los 80 por T. Sejnowski y C. Rosenberg [68]. Además, puede ser visto como un *precursor* de las redes convolucionales ya que el modelo que se aplica en cada n-grama de la secuencia puede considerarse una especie de filtro o *kernel*. Un ejemplo de esta red se muestra en la Figura 2.5.

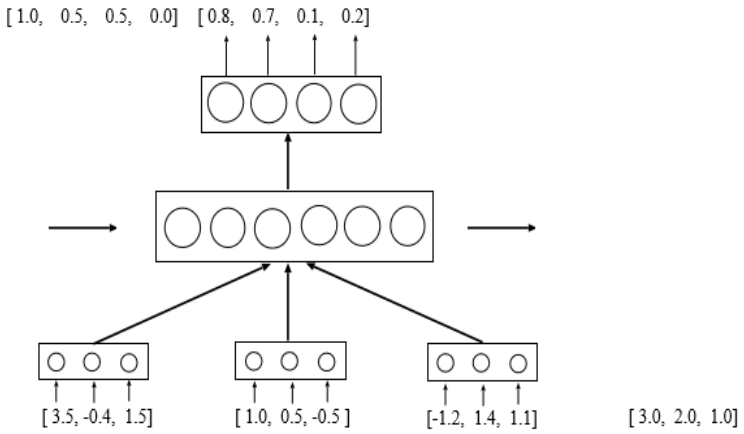

Figura 2.5: Net-Talk tras procesar el primer *3-grama* $\{\mathbf{0}x_1x_2\}$ de la secuencia [9].

Como se puede observar en la Figura 2.5, el modelo resuelve problemas $N \rightarrow M$, $M \leq N$ ($M = N$ en el caso en que se emplee un desplazamiento unitario). También es importante notar la necesidad de añadir elementos ficticios (*padding*) al principio y final de la secuencia, debido a que los primeros y últimos elementos no

tienen un contexto por la izquierda ni por la derecha respectivamente.

2.4.2. Redes recurrentes

Distinguimos entre redes dinámicas y redes recurrentes debido a la capacidad que tienen estas últimas de mantener un estado interno o memoria en función del cual, son capaces de modelar un comportamiento temporal dada una secuencia de entrada $\mathbf{x}$, donde $\mathbf{x} = \{x_1, x_2, ..., x_n\}, x_i \in \mathbb{R}^d, 1 \leq i \leq n$.

Dentro de este tipo de redes con conexiones recurrentes, podemos encontrar varios subtipos, a destacar: redes recurrentes simples [9], aumentadas [17] y bidireccionales [67]. Así, en las siguientes subsecciones se discuten estos tipos con más detalle, limitándonos únicamente a aquellos con los que se ha experimentado en este proyecto.

2.4.2.1. Redes recurrentes simples

Siguiendo a [9], la entrada del modelo es una secuencia de n componentes, $X = \{x_1, x_2, ..., x_n\} : x_i \in \mathbb{R}^{d_X}$, donde d_X es la dimensionalidad de cada componente de la secuencia; mientras que la salida del modelo es una secuencia de la misma longitud que X cuyos componentes son de dimensionalidad d_Y, i.e. $Y = \{y_1, y_2, ..., y_n\} : y_i \in \mathbb{R}^{d_Y}$.

De esta forma, a cada componente x_i de una secuencia de entrada (siguiendo un orden de izquierda a derecha de la secuencia), se calcula la componente de salida y_i de la forma $f(W^y y_{i-1} + W^x x_{i-1})$, donde W^x es la matriz de pesos asociada a las conexiones entre las neuronas de la capa recurrente y la componente x_i de entrada, W^y es la matriz de pesos asociada a las conexiones entre las neuronas de la capa recurrente y la componente y_{i-1} (vector de salida anterior) y f es una función de activación. La topología de una red recurrente simple se muestra en la Figura 2.6.

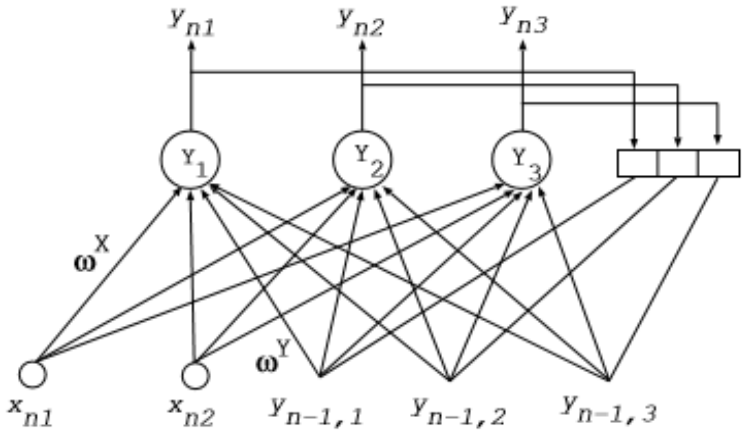

Figura 2.6: Red recurrente simple para $X = \{x_1, x_2, ..., x_m\} : x_n \in \mathbb{R}^2$ [9].

Como se puede observar más claramente en la Figura 2.6, para calcular la salida del elemento x_i de la secuencia (y_i) se emplea la salida y_{i-1} del modelo, por lo que se puede considerar que los modelos hacen uso de cierta memoria para aprender una secuencia temporal. Sin embargo, dicha memoria se diluye conforme se incrementa la longitud de la secuencia de entrada, i.e., no es capaz de dar cuenta de dependencias temporales muy amplias entre los elementos de una secuencia de entrada. Además, aparecen otros problemas asociados como *exploding* y *vanishing gradient*.

Así, por un lado, para ampliar la capacidad del modelo se han planteado otras redes recurrentes simples como las redes recurrentes de segundo orden (son Turing completas) [69] o *Nonlinear AutoRegressive models with eXogenous inputs* (NARX) [47]. Mientras que, por otro lado, para resolver los problemas de dependencias temporales y de gradientes, se plantearon soluciones como *Long Short Term Memory* (LSTM) [35] o *Gated Recurrent Units* (GRU) [15], la primera de las cuales se trata en el siguiente apartado.

2.4.2.2. Long-short Term Memory

Como se ha comentado en el apartado anterior, las redes recurrentes simples no son capaces de resolver problemas donde la salida dependa de información muy antigua, ni tampoco de ajustar o determinar el contexto necesario de la secuencia para calcular una determinada salida. Además, problemas como *exploding* y *vanishing gradient* también afectan durante el entrenamiento de este tipo de redes mediante BPTT.

Para solucionar estos problemas, se han propuesto arquitecturas como LSTM [35]. La topología de una red LSTM se muestra en la Figura 2.7.

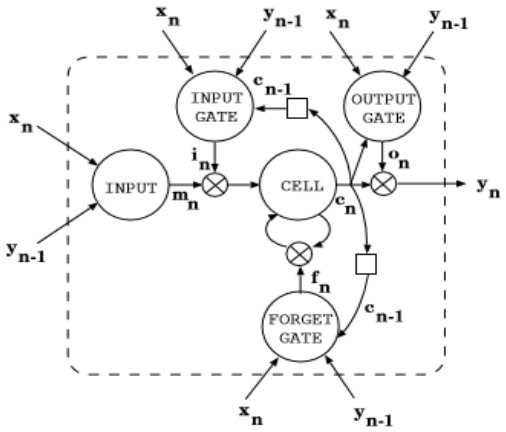

Figura 2.7: Topología de una red LSTM [9].

En la Figura 2.7 se pueden observar las diversas celdas de las que consta una red LSTM, y es necesario destacar como, haciendo uso únicamente de la componente actual x_n de la secuencia y la salida anterior y_{n-1}, es capaz de aprender todo lo necesario para resolver los problemas anteriormente mencionados, mediante el *pipeline* de celdas que la componen.

Más detalladamente, el funcionamiento es el siguiente. En la primera celda (*input*), se computa m_n de forma análoga a como se calcula la salida en una red recurrente simple, i.e. $m_n = f(W^X x_n + W^Y y_{n-1})$ [1]. Posteriormente, se calcula la salida de input gate, $i_n = f(W^Y y_{n-1} + W^X x_n + W^C c_{n-1})$ y se pondera m_n con i_n ($m_n * i_n$). Esto se puede ver como una forma de resaltar o reducir determinada información de la salida de una red recurrente simple.

A continuación, se calcula $c_n = f_n \times c_{n-1} + i_n \times m_n$, haciendo uso de la salida de la celda *forget gate*, $f_n = f(W^Y y_{n-1} + W^X x_n + W^C c_{n-1})$. Esto es similar a lo realizado al calcular $m_n * i_n$ y pretende determinar qué cantidad de información de $m_n * i_n$ debe olvidar en función de un cálculo intermedio, c_{n-1}, de la componente anterior de la secuencia (nótese como tanto en este caso, como en el anterior, c_{n-1} es la salida sin combinar con la salida de *output gate*).

[1] W^Y, W^X y W^C son diferentes para cada celda, no se trata de pesos compartidos.

Como último paso, se obtiene la salida de la red, $y_n = o_n \times f(c_n)$, donde o_n es la salida de *output gate*, $o_n = f(W^Y y_{n-1} + W^X x_n + W^C c_n)$ y f es una función de activación (generalmente *hyperbolic tangent*).

Con respecto al tipo de problemas que pueden resolver este tipo de redes, éstas pueden tratar también, de manera natural, problemas $N \to M$. Sin embargo, debido a su capacidad de seleccionar el contexto necesario y de mantener una amplia memoria, también es posible resolver de forma eficaz problemas $N \to 1$, tomando únicamente la salida tras procesar el último elemento de la secuencia.

2.4.2.3. Redes recurrentes aumentadas

Las redes recurrentes aumentadas están compuestas por una red recurrente simple y una red *feed-forward* [9]. Se pueden ver como una generalización de las redes recurrentes simples. Una de las redes más conocidas de este tipo es la red de Elman [21], compuesta por una red *feed-forward* de una única capa. Esta red se muestra en la Figura 2.8.

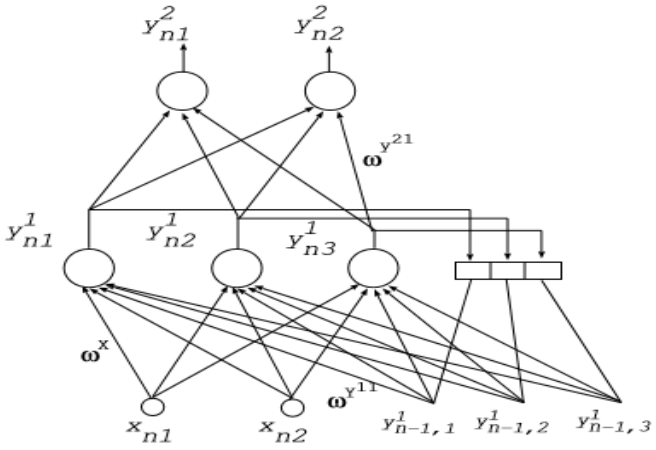

Figura 2.8: Red de Elman con entrada $x : x_i \in \mathbb{R}^2$ y salida $y : y_i \in \mathbb{R}^2$ [9].

Es importante notar cómo, en este caso, la conexión recurrente únicamente se realiza sobre la primera capa, por lo que la salida de la red *feed-forward*, y^2_n, no se empleará nunca como historia para computar la salida del siguiente elemento de la secuencia. Además, es posible generalizar este tipo de redes mediante la inclusión de nuevas capas en la red *feed-forward* o de nuevas capas en la red recurrente (red de

Jordan [39], véase la Figura 2.9).

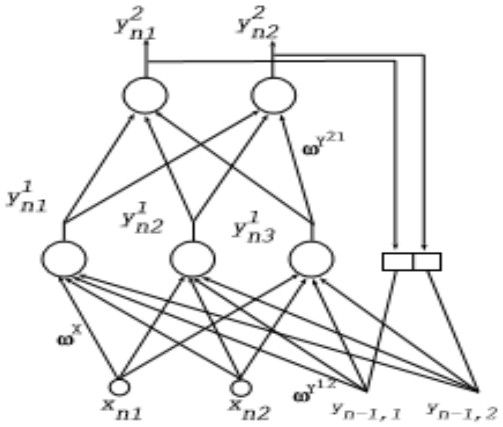

Figura 2.9: Red de Jordan con entrada $x : x_i \in \mathbb{R}^2$ y salida $y : y_i \in \mathbb{R}^2$ [9].

Por último, mencionar que la generalización de las redes de Elman mediante la inclusión de nuevas capas a la red *feed-forward*, también se ha empleado en gran parte de las competiciones abordadas, concretamente, en las redes utilizadas se emplea como red recurrente una LSTM y posteriormente, una red *feed-forward* con varias capas ocultas (previamente a estas también se han empleado, generalmente, redes convolucionales).

2.4.2.4. Redes recurrentes bidireccionales

Hasta ahora, hemos supuesto que el orden de procesamiento de las secuencias es un orden natural de izquierda a derecha, por lo que las redes comentadas solo son capaces de tener en cuenta dependencias con eventos pasados. Sin embargo, también pueden existir dependencias con los siguientes elementos de una secuencia.

Un ejemplo de ésto, puede darse en las tareas de *sentiment analysis* abordadas, donde la polaridad hasta un punto de una secuencia, no depende únicamente de los elementos ya procesados, si no que puede haber algún elemento posterior que influya en dicha polaridad e.g. «Todo perfecto, excepto que el objetivo que traía no tiene estabilizador».

En dicho ejemplo, la polaridad en el elemento $x_2 = perfecto$ no es positiva debido a que la palabra $x_3 = excepto$ modifica la polaridad de su contexto. Para abordar

este tipo de problemas, se propusieron las redes recurrentes bidireccionales [67].

Estas redes combinan los resultados de procesar bidireccionalmente (de izquierda a derecha y viceversa) las secuencias, tal como se muestra en la Figura 2.10 y se han empleado también (LSTM bidireccionales) en los problemas tratados.

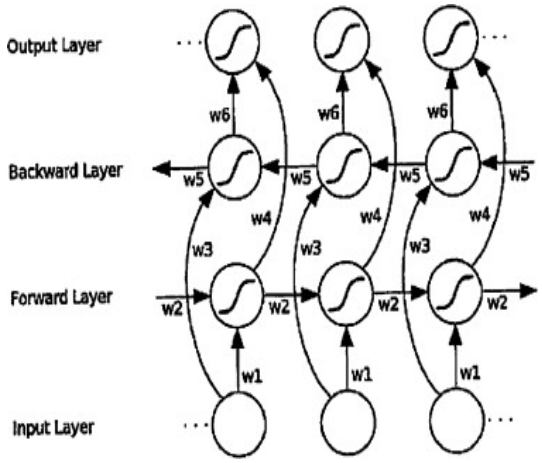

Figura 2.10: Ejemplo de red recurrente bidireccional.

2.4.3. Redes convolucionales

Las redes convolucionales están inspiradas en el funcionamiento de la corteza visual de cerebros biológicos, formada por un conjunto de neuronas sensibles a la presencia de aristas con ciertas orientaciones, regiones y formas específicas [19].

Por este motivo, la aplicación natural de éstas redes son entradas 2D que representan imágenes. Sin embargo, su aplicación se puede generalizar a secuencias de vectores 1D, como es el caso de los problemas tratados en este proyecto si se emplea una representación secuencial.

Con respecto a su capacidad, son capaces de aprender, de forma automática mediante *back-propagation*, una jerarquía de características que abstraen información espacial y pueden ser usadas para problemas de clasificación [72], es decir, permiten aprender filtros con altas respuestas a determinados patrones que, posteriormente, pueden ser discriminativos para realizar una clasificación.

Generalmente, se denomina red convolucional a una red compuesta por capas convolucionales, *pooling* y una capa de *reshape*, todo ello seguido de un modelo de clasificación, por ejemplo, MLP [58] (un ejemplo de ello se muestra en la Figura 2.11). Sin embargo, en este proyecto se han utilizado, generalmente, *stacks* de capas convolucionales y recurrentes con MLP, por ello, se ha prescindido de la capa *reshape*, obteniendo un único vector mediante la última salida de una red LSTM.

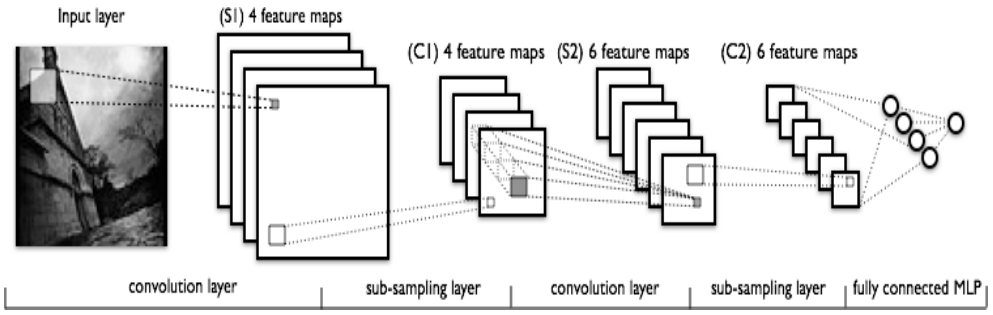

Figura 2.11: Ejemplo de red convolucional para clasificación de imágenes.

A continuación se presenta con más detalle los componentes mencionados que forman parte de redes convolucionales.

2.4.3.1. Convolución

Una convolución es una operación matemática entre dos funciones f y h, que produce una tercera función típicamente vista como una versión modificada de f en función de h [29]. En nuestro caso, f es una secuencia de vectores 1D (e.g. *embeddings*) y h es un filtro lineal que consigue que cada nuevo elemento de la salida sea una suma ponderada de los elementos en el contexto de cada elemento procesado de la secuencia. Un ejemplo de este tipo de filtros para un problema de detección de caras, se puede observar en la Figura 2.12.

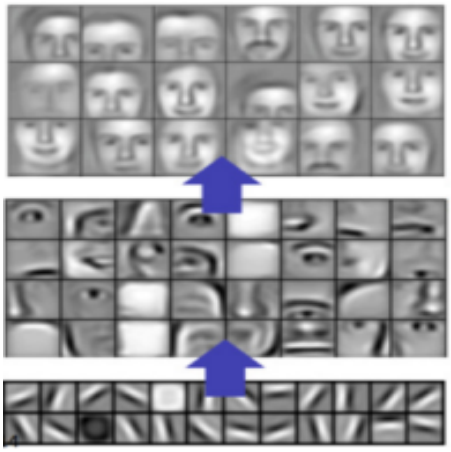

Figura 2.12: Ejemplos de filtros posibles para un problema de detección de caras.

Así, sobre la secuencia de entrada (o sobre salidas de convoluciones previas) se aplica un conjunto de filtros lineales, siempre con los mismos pesos para toda la secuencia, en el caso de redes convolucionales, aprendidos mediante *back-propagation*. Esto consigue ciertas propiedades en la secuencia de salida, siendo una de las más importantes la invarianza a la traslación.

Dicha invarianza es importante en determinados problemas de *text classification*, debido a que un contexto específico es igual de discriminativo independientemente de la posición de la frase en la que aparezca. Hay otros problemas donde esto no se cumple, generalmente, aquellos problemas donde la propiedad más discriminativa es la presencia de componentes clave como determinadas palabras.

Matemáticamente, dada una entrada f y un filtro h, cada elemento (i, j) de la salida g de una convolución se define como $g(i, j) = \sum_{k,l} f(i - k, j - l)h(k, l)$, donde k y l son la altura y la anchura del filtro respectivamente [29]. En la Figura 2.13 se muestra un ejemplo de esta operación. Nótese la necesidad de incluir elementos adicionales para el cálculo de algunas salidas mediante técnicas como *padding*, *clamp-to-edge*, *wrap*, etc.

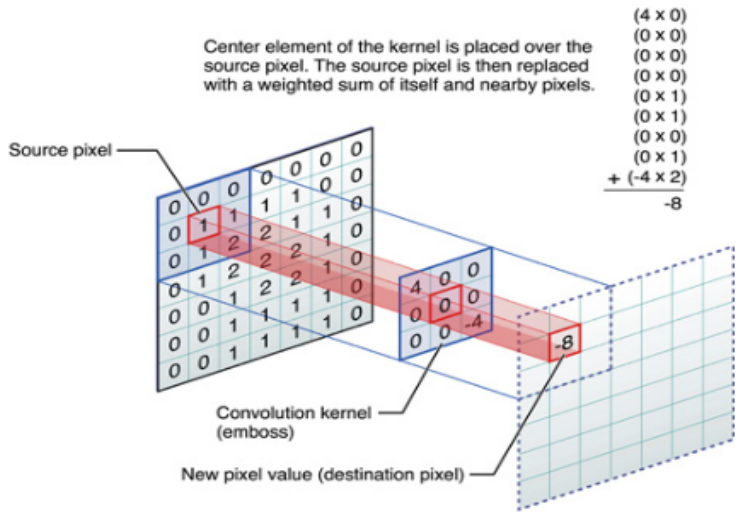

Figura 2.13: Convolución 2D con *zero-padding*, $k = 3$, $l = 3$, $f_h = 7$ y $f_w = 7$.

Es importante resaltar que para problemas de PLN, las regiones locales de los elementos de una secuencia de entrada son independientes entre sí (e.g. con $k = 2$ y $l = 2$, no existe ninguna relación espacial entre $x_{1,1}$, $x_{1,2}$, $x_{2,1}$ y $x_{2,2}$). Sin embargo, sí existe una relación espacial entre los elementos (vectores 1D completos) de la secuencia. Es por ello por lo que en este tipo de casos se emplea como anchura de filtro, l, la dimensionalidad de los vectores, d_X. Asumiendo ésto, $g(i, j)$ pasa a ser $g(i) = \sum_k f(|i - k|)h(k)$.

2.4.3.2. Pooling

El principal objetivo de las operaciones de *pooling* consiste en reducir el tamaño de los mapas de salida, imprescindible en problemas donde el tamaño de la entrada es demasiado grande, lo que provoca que tras aplicar convoluciones la salida también siga siendo de gran tamaño ($g_h = (f_h - k) + 1$ y $g_w = (f_w - l) + 1$, suponiendo un desplazamiento $stride = 1$ del filtro y sin *padding*).

Con el *pooling* también se consigue una reducción en el coste temporal del cálculo de posteriores convoluciones, debido a que se reduce el tamaño de su entrada. Aunque existen enfoques para acelerar dicho cálculo como *Lowering* [13] o *Winograd* [44], esto sigue siendo necesario en algunos casos.

Además, al ser una forma de realizar *down-sampling*, generalmente no lineal, sobre la entrada, los operadores de *pooling* permiten tratar con multi escala, es decir,

fijándonos en una componente del resultado obtenido tras aplicar esta operación sobre una entrada estaríamos considerando una región mayor de dicha entrada en la escala original. Esto es similar a las pirámides espaciales, tal como se muestra en la Figura 2.14.

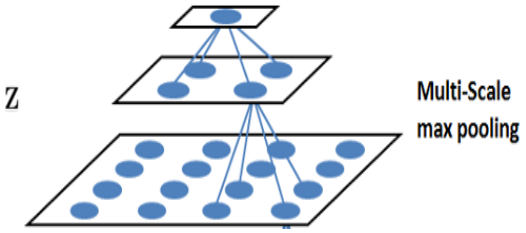

Figura 2.14: Ejemplo de multi escala con operaciones de *pooling*.

Entre las operaciones de *pooling* más comúnmente utilizadas podemos encontrar *max pooling* y *average pooling*, que sustituyen regiones de $k \cdot l$ de la entrada por un único componente calculado como el máximo o la media de la región respectivamente. Una representación gráfica de estas operaciones se muestra en la Figura 2.15.

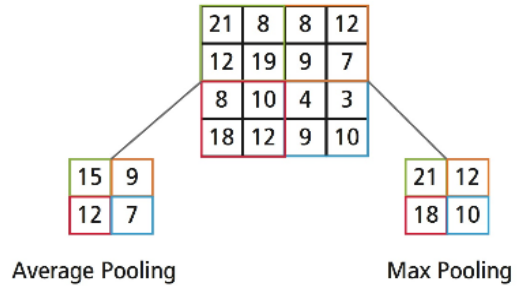

Figura 2.15: Ejemplo de *max pooling* y *average pooling*.

2.5. Deep Models

El objetivo de este tipo de modelos es dejar de lado la definición de características de forma manual, debido a que dichos modelos son capaces de extraer características de alto nivel (más complejas cuantas más capas se empleen) de forma automática. Así, con las características extraídas de forma automática, únicamente es necesario aplicar un modelo discriminativo, posiblemente basado en redes neuronales, para resolver un determinado problema de clasificación.

Algunos de los modelos vistos hasta ahora podrían considerarse, debido a la ambigüedad del término, como modelos de *deep-learning* (MLP con más de tres capas ocultas, *stack* de CNN, LSTM y MLP, etc.) ya que no se trata de modelos simples y son capaces de extraer características de alto nivel de forma automática a partir de las entradas.

Sin embargo, la utilización de modelos muy complejos compuestos por muchas capas conlleva una serie de problemas relacionados con la optimización mediante *back-propagation*, la generalización (debido a la gran cantidad de parámetros y, normalmente, la poca disponibilidad de datos) y la complejidad temporal. Todos estos problemas se discuten con más detalle en las siguientes subsecciones.

2.5.1. Optimización

Al entrenar redes neuronales mediante *back-propagation* se aplican, de forma iterativa, actualizaciones sobre los pesos w_{ij}^l en función del gradiente de la función de *loss*, E_A, con respecto a dichos pesos. Estos gradientes se computan, empezando desde la última capa, mediante la regla de la cadena aplicada sobre cada neurona de cada capa de la red.

Más concretamente, el incremento a aplicar en cada peso es $\Delta w_{ij}^l = -\rho \frac{\partial E_A}{\partial w_{ij}^l} = \rho\ \delta_i^l\ s_j^i$, donde δ_i^j es el error en la neurona i de la capa j, s_i^j es la salida de la neurona i de la capa j y ρ es el *learning rate*. Un ejemplo de esto se puede ver en la Figura 2.16.

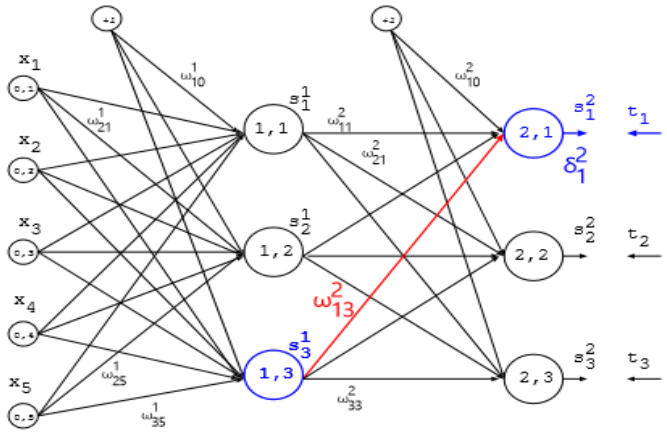

Figura 2.16: Incremento sobre el peso $w_{1,3}^2$ en un MLP de una capa oculta [8].

Si observamos δ_i^l, ésta se calcula de forma diferente si se trata del error en una neurona de la última capa o de una capa intermedia. Si se trata de la última capa (como en la figura anterior), $\delta_i^l = (t_i - s_i^l) \cdot f'(z_i^l)$, donde f' es la derivada de la función de activación, z_i^l es la preactivación de la neurona i de la capa l (salida sin aplicar una función de activación) y t_i es la componente i de la salida correcta para la muestra analizada.

Si por el contrario se trata de una neurona de una capa intermedia, el error es $\delta_i^l = (\sum_r^{\#(l+1)} \delta_r^{l+1} w_{r,i}^{l+1}) f'(z_i^l)$. En este caso, se trata de una combinación lineal entre el error en todas las neuronas r de la capa $l+1$ conectadas a la neurona i de la capa l y el peso que las une, considerando también la derivada de la función de activación sobre la preactivación de la neurona.

En resumen, como se ha enunciado a lo largo de la sección, se emplea la derivada de la función de activación en cualquier caso y el incremento de los pesos se calcula como un producto de tres elementos, por lo que, si alguno de estos tuviera un valor muy bajo, el incremento sería prácticamente cero. Este problema se conoce como parálisis de la red o *vanishing gradient* y en las siguientes subsecciones se detallan posibles soluciones a este problema.

2.5.1.1. Rectified Linear Unit

Tradicionalmente, se han empleado funciones de activación como *sigmoid* o *hyperbolic tangent* cuya derivada tiene la peculiaridad de tomar valores muy bajos en ciertas situaciones, lo que provoca el problema de parálisis de la red según lo mencionado anteriormente. Se muestra la derivada de *sigmoid* en la Figura 2.17.

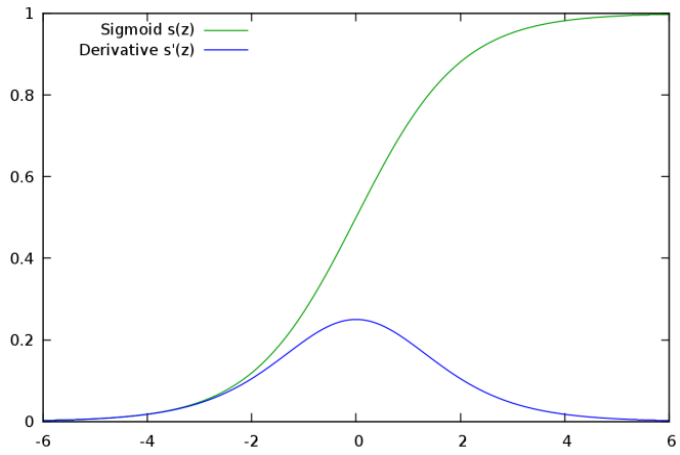

Figura 2.17: Función de activación *sigmoid* y su derivada.

Para solucionar este problema se han propuesto varias soluciones. Entre éstas destaca la utilización de la función de activación ReLU ($f(x) = max(0, x)$) , que resuelve el problema mencionado ya que su derivada es $f'(x) = 1$ si $x > 0$ y cero en caso contrario, lo que provoca que el gradiente se propague de forma completa hacia atrás en caso de que $x > 0$. Una gráfica de dicha función de activación se muestra en la Figura 2.18.

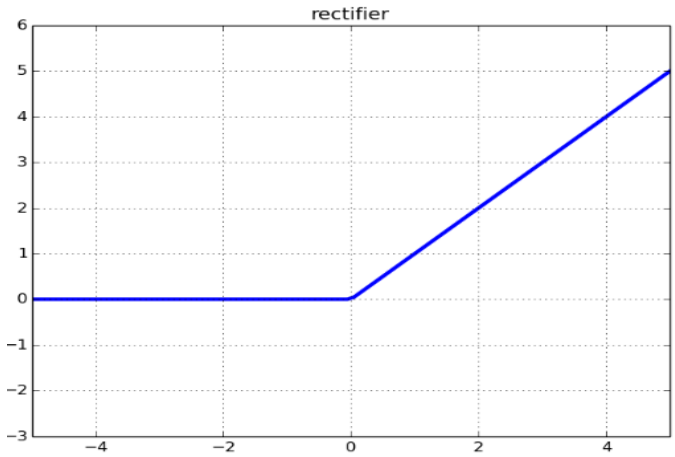

Figura 2.18: Función de activación ReLU.

Sin embargo, a pesar de esta ventaja (y otras como la invarianza a la escala y la simplicidad de cómputo), ReLU sufre de un problema conocido como *dying ReLU problem*, que ocurre cuando las preactivaciones son en su mayoría negativas. Para esto, se han propuesto funciones como *Leaky ReLU* (LReLU) [49].

2.5.1.2. Batch Normalization

La normalización de los datos también es importante para evitar problemas de *vanishing gradient* y de oscilamiento durante entrenamiento. Típicamente se realiza una normalización, únicamente de los datos de entrada, que consiste de forma general en desplazar las muestras para que que tengan media $\mu = 0$ y desviación típica $\sigma = 1$.

Sin embargo, los pesos modifican los valores que fluyen a través de la red y dicha normalización se pierde durante el flujo de datos, por lo que la distribución de las salidas es diferente en cada capa, esto se conoce como *Internal covariance shift*.

Para solucionar este problema y conseguir que todos los datos de cada *mini-batch* que fluyen por la red estén normalizados se propuso *Batch Normalization* [37]. Además, esta técnica facilita la convergencia debido a que la salida de todas las capas de la red siguen la misma distribución y permite utilizar valores de *learning rate* más altos, lo que acelera el entrenamiento. El cálculo de la normalización se realiza tal como se muestra en la Figura 2.19 [37].

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figura 2.19: *Batch Normalization* aplicado a la activación x sobre un *mini-batch* [37].

2.5.2. Generalización

Debido a la gran cantidad de parámetros que tienen los modelos de *deep-learning* y, generalmente, a la reducida cantidad de muestras que se disponen para ajustarlos,

surgen problemas de generalización. Entre los problemas de generalización podemos encontrar *underfitting* y *overfitting*, siendo este último el que ocurre con mayor frecuencia durante las experimentaciones con redes neuronales. Ambos problemas se representan gráficamente en la Figura 2.20.

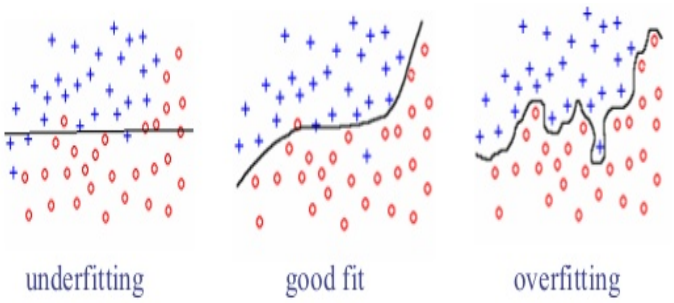

Figura 2.20: Representación gráfica de *underfitting* y *overfitting*.

Con la finalidad de mitigar este problema se han propuesto diversas soluciones, principalmente orientadas a dificultar, de forma intencionada, el aprendizaje de la red durante el entrenamiento, bien sea especializando de forma dispersa las neuronas de cada capa (*dropout* [71] o *drop-connect* [80]) o añadiendo ruido extraído de una determinada distribución [30]. Además, otro enfoque clásico para lidiar con el *overfitting* consiste en generar muestras sintéticas a partir de las muestras de entrenamiento (*data augmentation*) [11].

Por último, es necesario considerar que existen otras técnicas de generalización como emplear diversos tipos de regularización tanto en pesos como en gradientes (*weight decay* [43], *L1-normalization*, *MaxNorm* [7], etc.), sin embargo, no se han expuesto en esta memoria debido a que no se han utilizado en la experimentación realizada.

2.5.2.1. Dropout

Una técnica simple para prevenir el *overfitting* es *dropout* [71]. Inspirada en la pérdida biológica de neuronas y la necesidad de que el resto se especialice en las tareas que anteriormente realizaban las neuronas perdidas, lo que conlleva una aproximación a la combinación exponencial (*sampling*) de muchas arquitecturas de redes neuronales diferentes a partir de una única red.

El término *dropout* hace referencia a la anulación de la salida de determinadas neuronas (tanto de capas ocultas como de capas visibles) con una probabilidad dada, de forma diferente a otras técnicas como *drop-connect* que anulan las conexiones entre neuronas en lugar de sus activaciones.

Con ello, una variable aleatoria $r_j^l \sim Bernoulli(p)$, controla la anulación de la neurona j de la capa l con una probabilidad p, de tal forma que la nueva salida de la neurona $\tilde{s}_j^l$, se define como $\tilde{s}_j^l = r_j^l s_j^l$ donde s_j^l es la salida original de la neurona.

Este comportamiento de anulación aleatoria durante entrenamiento provoca que, para cada *mini-batch* en cada iteración, las neuronas que se activen sean diferentes y tengan que especializarse. De aquí proviene la idea de la combinación de modelos, tal como se muestra en la Figura 2.21.

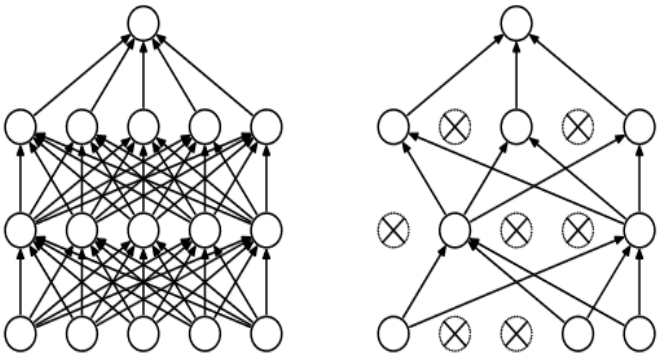

Figura 2.21: Red neuronal convencional y red neuronal tras aplicar *dropout* [71].

Por último, destacar que la anulación de neuronas solo se aplica durante el entrenamiento. En la fase de *test*, las neuronas siempre están activas y su activación se multiplica por la probabilidad de retener cada una de dichas neuronas, es decir, $\tilde{s}_j^l = s_j^l(1-p)$.

2.5.2.2. Data Augmentation

Uno de los problemas que causa *overfitting* es la disponibilidad de pocas muestras de entrenamiento cuando el número de parámetros del modelo es muy elevado en relación al número de muestras. Para solucionar este problema, un posible enfoque, dejando de lado técnicas aplicadas específicamente sobre el modelo, consiste en la

generación de muestras sintéticas a partir de datos reales.

En primer lugar, una solución trivial consiste en emplear recursos léxico-semánticos como *WordNet* [54] para sustituir las palabras por sinónimos, sentidos, etc. También es posible realizar sustituciones utilizando *Word2Vec* [51], [53], [62], reemplazando, mediante algún criterio, constituyentes (palabras, caracteres, n-gramas, etc.) por alguno de sus vecinos más cercanos.

En segundo lugar, otro método sencillo consiste en llevar a cabo un proceso de corrupción de muestras mediante la aplicación de ruido extraído de distribuciones caracterizables [30] [38], típicamente ruido gaussiano $r \sim \mathcal{N}(\mu = 0, \sigma)$. Esto puede ser visto como un proceso de *data augmentation* y, además, también es posible aplicarlo de forma general en modelos basados en redes neuronales sobre las activaciones de cada neurona, complicando aún más la tarea de la red para alcanzar el *overfitting*.

En tercer lugar, una técnica muy conocida para problemas de PLN es *Synthetic Minority Over-sampling Technique* (SMOTE) [11]. Concretamente, la técnica genera, dada una muestra x de la clase c y sus k-vecinos más cercanos a ésta de su misma clase c, una nueva muestra sintética que en el espacio de representación se encuentra en un segmento de longitud aleatoria entre la muestra original y uno de sus k-vecinos más cercanos (también seleccionado aleatoriamente).

Un ejemplo de esta técnica se muestra en la Figura 2.22. Además, la técnica es fácilmente generalizable para vectores de características discretas. Algunas variantes de esta técnica pueden ser SMOTE-D [76] o P-SMOTE [81].

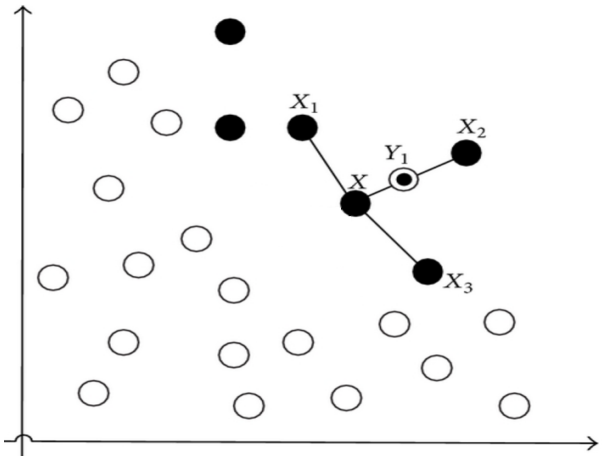

Figura 2.22: Representación gráfica de SMOTE para generar Y_1 [11].

Otros métodos para generación de muestras sintéticas están basados en modelos de *deep-learning* generativos y se han hecho muy populares en los últimos dos o tres años, a destacar *Variational Autoencoders* (VAE) [42], *Generative Adversarial Networks* (GAN) [28] y sus variantes como InfoGAN [12] o DiscoGAN [41]. En ambos casos, la aplicación para vectores de características discretas no es posible, por lo que es necesario recurrir a representaciones continuas de frases para la generación de muestras sintéticas.

Una posible forma de emplearlos en la generación de muestras sintéticas consiste en usarlos en combinación con modelos *sequence-to-sequence*. De esta manera, dado un modelo *sequence-to-sequence* ya entrenado, es posible emplear técnicas como GAN y VAE para generar un vector de características reales (muestra sintética) que represente el vector colapsado W (Figura 2.23), resultado del *encoder* del modelo *sequence-to-sequence*.

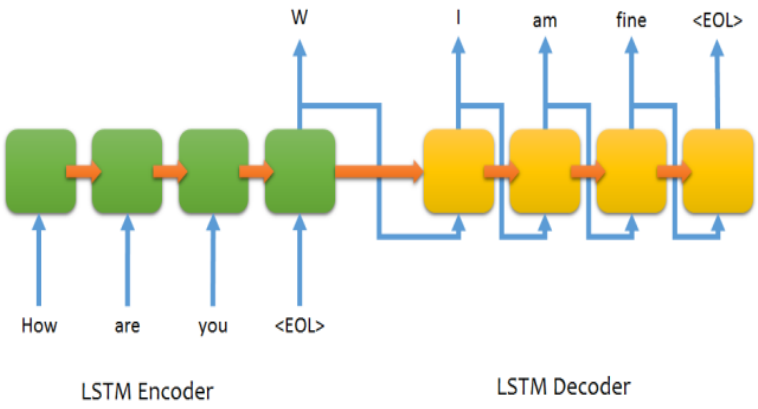

Figura 2.23: Ejemplo de modelo *sequence-to-sequence* [1].

También es necesario destacar que hasta ahora se ha supuesto únicamente la utilización de *data augmentation* sobre muestras de entrenamiento. Sin embargo, también es posible su uso en la fase de *test* para determinar, para n muestras sintéticas generadas a partir de una muestra de test, la clase predicha mediante una nueva función de las n salidas e.g. mediante votación.

Por último, destacar que se han implementado algunas de estas técnicas pero, por brevedad, no se han discutido con mucho detalle en esta sección. Concretamente, las técnicas implementadas son: sustitución por sinónimos y vectores *Word2Vec* y SMOTE.

2.5.3. Problemas adicionales

En la presente subsección se presentan algunos problemas relacionados con la utilización de las técnicas expuestas a lo largo del capítulo y que han surgido en las diversas tareas tratadas durante la realización del proyecto.

En primer lugar, se analiza con detalle el problema del desbalanceo en determinadas tareas donde la cantidad de muestras de una clase o conjunto de clases es superior al resto. En este caso, las redes neuronales han demostrado poca capacidad de generalización, dando lugar a un sesgo en las predicciones hacia las clases con mayor probabilidad *a priori* [26] [25].

En segundo lugar, se enuncia el método utilizado para entrenar modelos recurrentes, que manejan representaciones secuenciales de los datos, mediante BPTT usando *mini-batches* compuestos por muestras de la misma longitud (*bucketing*).

2.5.3.1. Desbalanceo

Como ya se ha comentado, el desbalanceo entre el número de muestras de las clases de un determinado corpus provoca que algunos modelos no sean capaces de generalizar y sus salidas están sesgadas hacia las clases con mayor probabilidad a priori. Este problema es muy común y ha ocurrido en todas las tareas abordadas en este proyecto, exceptuando la tarea de detección de género (*IberEval2017@SEPLN*).

Para solucionar este problema se han empleado diversas soluciones en las tareas tratadas [27] [26] [25]. La primera solución que se empleó consiste en reducir el número de muestras de las clases más pobladas hasta que las probabilidades a priori de todas las clases sean iguales (o, al menos próximas) [27].

Sin embargo el principal problema de esta técnica es el desaprovechamiento de muestras. Además, se requiere un buen criterio de selección de muestras para no perder aquellas más representativas de las clases reducidas. Por este motivo se buscaron soluciones alternativas.

La siguiente solución con la que se ha experimentado consiste en incrementar el número de muestras, de forma sintética, de aquellas clases con menor probabilidad a priori mediante técnicas de *data augmentation*. Concretamente se experimentó con SMOTE y sustitución por palabras semejantes mediante *Word2Vec* con el corpus de la tarea COSET (IberEval2017@SEPLN), sin embargo, los resultados obtenidos no fueron satisfactorios.

Debido a que hemos hecho uso, casi exclusivamente, de modelos basados en redes neuronales, se experimentó con un método de escalado de la función de *loss* de forma que cada clase tiene asociado un factor multiplicativo a aplicar sobre la *loss*. Así, si le asignamos a las clases minoritarias un factor mayor, al cometer errores con las muestras de dichas clases la *loss* será mucho mayor, forzando así a la red para que clasifique correctamente dichas muestras [26].

Normalmente, para definir los pesos de cada clase p_c se utiliza la clase mayoritaria como clase de referencia ($p_{mayoritaria} = 1$) y se le asigna a las demás clases un peso proporcional a la diferencia de muestras entre dicha clase y la referencia. Sin embargo, para corpus muy desbalanceados este método puede hacer que la *loss* crezca mucho y provocar inestabilidades en entrenamiento. Una solución a ésto consiste en aplicar un suavizado a los pesos, en nuestro caso $p_c = \log(\mu \frac{\#referencia}{\#clase})$, donde el parámetro μ se debe ajustar en fase de ajuste.

Por último, destacar que el método de balanceo que mejor se ha comportado, generalmente, ha sido el escalado por la función de *loss* que ha demostrado ser un factor clave en la obtención de sistemas competitivos en las tareas abordadas como COSET [25] o Stance [26].

2.5.3.2. Bucketing

Un problema que surge al entrenar redes recurrentes mediante BPTT al usar *mini-batches* (también ocurre algo similar con las redes convolucionales) es que es necesario conocer previamente la longitud de las secuencias de un *mini-batch* dado para poder expandir la red un número de niveles fijo. Sin embargo, en problemas de PLN donde se emplean representaciones secuenciales, las longitudes de las secuencias son variables.

Una solución trivial a ésto consiste en emplear *padding*, añadiendo o eliminando vectores de las secuencias de entrada hasta que todas sean de la misma longitud. El principal problema de ésto es que en los dominios tratados, *Twitter* generalmente, las diferencias entre las longitudes de las secuencias son muy pronunciadas y se añade demasiado ruido no inherente al problema.

Otra posible solución consiste en entrenar muestra a muestra, lo que conlleva un incremento considerable en el tiempo necesario para resolver las tareas abordadas. Por ello, para evitar ambos problemas, la solución empleada consiste en la utilización de *buckets*, que se pueden ver como *mini-batches* pero asociados a una determinada longitud l que están compuestos por secuencias de dicha longitud l, lo que permite llevar a cabo la expansión de la red durante el entrenamiento [20].

Capítulo 3

Representación de texto

> Existe un lenguaje que va más allá de las palabras.
>
> Paulo Coelho

3.1. Representación de constituyentes

Uno de los problemas fundamentales a abordar en las tareas de *text classification*, es la representación de texto. El objetivo consiste en representar numéricamente documentos de texto para hacerlos matemáticamente computables [85]. Sin embargo, para representar documentos es necesario representar de alguna manera los constituyentes que los componen.

Estos constituyentes pueden ser las palabras, los caracteres de un documento, n-gramas de palabras o caracteres, o incluso información morfológica como las categorías gramaticales (POS), información de polaridad, sentidos de WordNet, etc.

Cada uno de estos constituyentes conlleva ventajas y desventajas, que se utilizan para escoger el que mejor se ajuste a la tarea. Algunas ventajas e inconvenientes de los constituyentes se listan a continuación:

- **Palabras**: como ventajas, destacar la facilidad de extracción de este tipo de constituyentes y el reducido coste espacial que conllevan en corpus de reducido tamaño. Además, las relaciones existentes entre palabras están más claras que entre otros constituyentes e.g. entre caracteres. La principal desventaja radica en la dimensionalidad de las representaciones en corpus de gran tamaño (dependiente de la talla de vocabulario, que generalmente es elevada) al emplear

representaciones *one-hot*, sin embargo, el coste se reduce al emplear otras representaciones como, por ejemplo, *embeddings*.

- **Caracteres**: también destacan por la facilidad de extracción y la reducida dimensionalidad que conlleva su representación en algunos casos, debido a que el número de caracteres (talla de vocabulario de caracteres) es muy reducido en comparación al número de palabras (talla de vocabulario de palabras). Sin embargo, en el caso de las representaciones secuenciales, el coste espacial de representar un documento es mucho mayor y los algoritmos de aprendizaje automático se hacen más complejos ya que tienen que dar cuenta de las relaciones existentes entre caracteres (más complejas que a nivel de palabras).

- **Constituyentes complejos**: en este caso, la extracción de este tipo de constituyentes es más compleja, requiriendo recursos externos como lexicones de polaridad (en el caso de representaciones de polaridad) o sistemas que etiqueten de alguna manera los constituyentes (POS-Taggers o sistemas de Word Sense Desambiguation). A pesar de la dificultad de extracción, permiten obtener información que no existe *per se* en los constituyentes, por lo que se trata de representaciones más ricas y que aportan información adicional a los dos tipos anteriores.

- **n-gramas de constituyentes**: en los casos anteriores no se tiene en cuenta el contexto de los contituyentes (aunque se puede considerar en la representación secuencial a nivel de documento) y esto puede ser un inconveniente para determinadas tareas. Por este motivo, los n-gramas de constituyentes permiten dar cuenta de cierto contexto, sin embargo, dicho contexto es limitado debido al coste espacial que generalmente conlleva.

Con todo ello, para resolver problemas de *text classification* debemos representar documentos de texto en función de las representaciones de sus constituyentes. Por este motivo, en los siguientes apartados se enuncian con detalle algunas técnicas de representación de constituyentes individuales para, posteriormente, emplearlas mediante otras técnicas de representación a nivel de documento.

3.1.1. One-Hot

El concepto de representación *one-hot* proviene de la teoría de circuitos digitales, donde intuitivamente se puede ver como un conjunto de bits de tal forma que solo uno de estos bits se encuentra activo (1) y todos los demás anulados (0) [32].

Con ello, es posible generalizar el concepto para representar de forma unívoca los constituyentes de un determinado documento si se considera que cada componente x_i del vector de representación $x = \{x_1, x_2, ..., x_n\}$ está asociado a un constituyente i concreto. Así, para representar un constituyente i, la componente x_i de x se encontrará activa mientras que los demás $j \neq i$ estarán a cero, $x(i) = \{x_1, x_2, ..., x_n\} : x_i = 1, x_j = 0 \forall j \neq i$. Un ejemplo de ésto se muestra en la Figura 3.1.

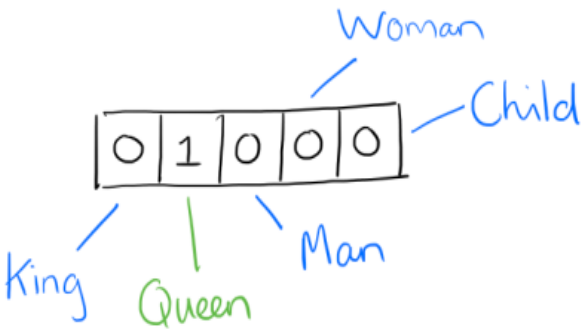

Figura 3.1: Ejemplo de representación *one-hot* del constituyente *Queen*.

Entre las ventajas de este tipo de representación destacan la facilidad de implementación y de comprobación de errores o la eliminación de asunciones erróneas por parte de los algoritmos de aprendizaje. Por ejemplo, en un problema de clasificación en tres clases y_1, y_2, y_3, podemos asignar como etiquetas $y_1 = 1$, $y_2 = 2$, $y_3 = 3$, sin embargo, se estarían haciendo asunciones como que la clase y_2 es la media de las clases y_1 e y_3 o que la clase $y_3 = y_1 * 3$, cuando esto es no es cierto en la mayoría de los casos.

Por otro lado, las desventajas más importantes son la incapacidad para dar cuenta de relaciones entre palabras y el coste espacial debido a que la dimensionalidad de un único vector de representación coincide con la talla del vocabulario. Aún así, en la práctica, es posible emplear vectores dispersos para utilizar estas representaciones en problemas con un vocabulario de gran tamaño.

3.1.2. Word Embeddings

Debido a los problemas que presenta la representación *one-hot*, se han explorado representaciones vectoriales continuas que preservan regularidades lingüísticas para diferentes problemas del área de PLN.

Así, varios métodos, principalmente basados en redes neuronales, han sido propuestos durante los últimos años para obtener este tipo de representaciones. Tambien se han explorado otros enfoques como *Latent Dirichlet Allocation* [6] o *Latent Semantic Analysis* [18].

El primer método fue propuesto por Y.Bengio en [3] y se conoce como *Feedforward Neural Network Language Model* (NNLM). Concretamente es una red neuronal de una capa oculta no lineal, una capa proyección lineal y una *lookup table* compartida. Además, una capa *softmax* en la salida computa la probabilidad sobre V elementos, esto hace que el modelo no sea útil en tareas con vocabularios de gran tamaño. Este enfoque se muestra en la Figura 3.2.

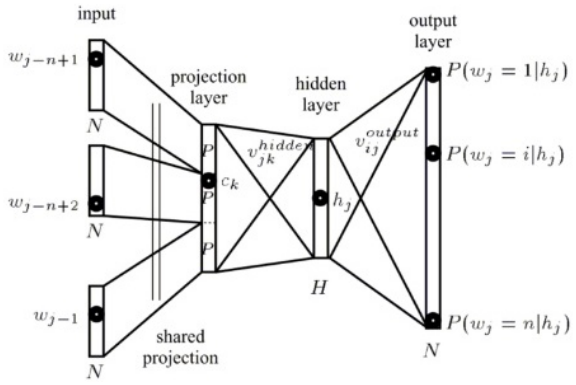

Figura 3.2: Arquitectura NNLM [3].

Otro enfoque propuesto en [52] emplea redes neuronales recurrentes para intentar capturar relaciones a largo término entre los constituyentes de entrada. Sin embargo, se trata de una red recurrente simple como las comentadas en el capítulo anterior y no es tan capaz de capturar dichas relaciones como, por ejemplo, las redes LSTM. Una representación gráfica de esta arquitectura se muestra en la Figura 3.3.

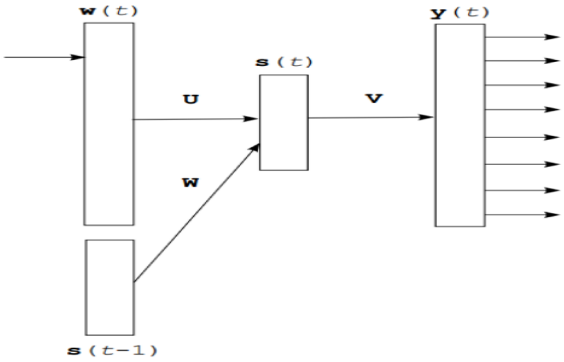

Figura 3.3: Arquitectura RNNLM [52].

Para solucionar los problemas con vocabularios de gran tamaño y con el objetivo de mejorar la eficiencia del entrenamiento de redes dedicadas a la obtención de *embeddings*, T.Mikolov propuso en [51] [53] las arquitecturas *Continuous Bag Of Words* (CBOW) y *skip-gram* (SG) combinadas con técnicas para optimizar el cómputo de la capa *softmax* como *hierarchical softmax* o *negative sampling*. Estas arquitecturas se discuten con detalle en los próximos apartados y han sido tratadas en otros documentos como [24].

Por último, es necesario destacar la capacidad que tienen estas arquitecturas para capturar gran cantidad de propiedades lingüísticas complejas como el genéro o conceptos como la hiponimia. De esta forma, se observan ciertas propiedades en el espacio de representación de los *embeddings* como por ejemplo, una artimética de *embeddings* que permite dar cuenta de relaciones como las mostradas en las Figuras 3.4 y 3.5.

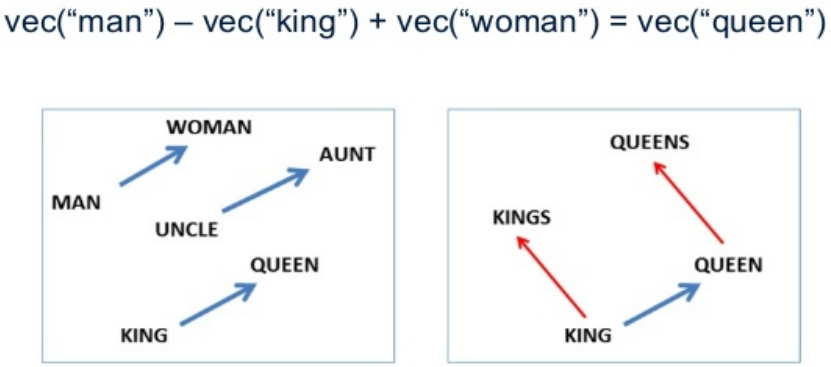

Figura 3.4: Ejemplo de aritmética de *embeddings*.

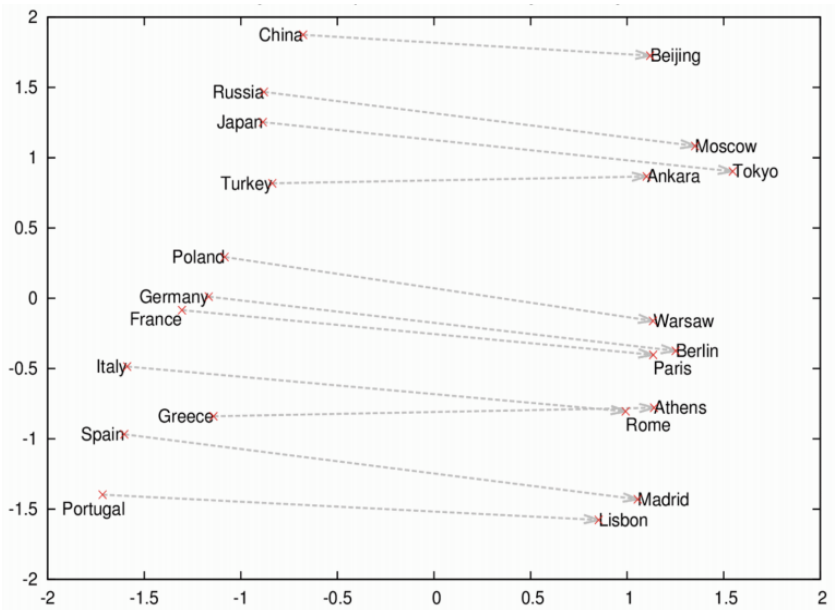

Figura 3.5: Relación «es capital de» capturada por los *embeddings*.

3.1.2.1. CBOW

La primera arquitectura propuesta en [51] [53] para reducir la complejidad computacional de los métodos de obtención de *embeddings* anteriores, es el modelo *Continuous Bag Of Words* (CBOW), similar a NNLM, tal como se puede observar en la Figura 3.6. En este caso, se elimina la capa oculta no lineal de las redes neuronales utilizadas en NNLM, manteniendo compartida la *lookup table*.

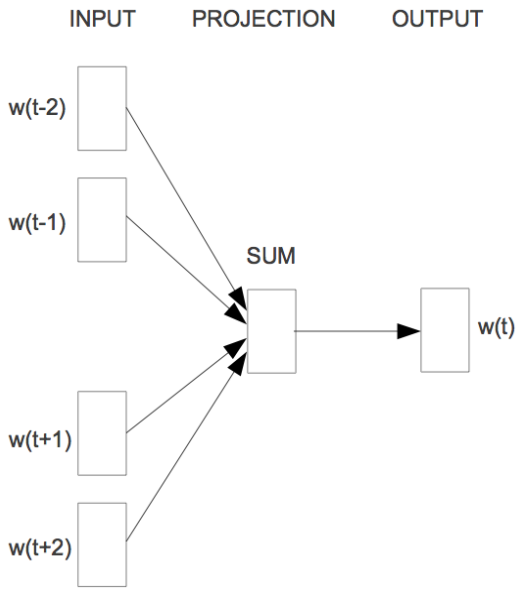

Figura 3.6: Arquitectura CBOW [51], [53].

Formalmente, considerando que w_t es el constituyente analizado de un documento, la entrada de la red neuronal es $w_{t-k}, ..., w_{t-1}, w_{t+1}, ..., w_{t+k}$, siendo k el tamaño de la ventana y la salida w_t. La tarea de la red consiste en predecir el constituyente w_t de un documento conociendo su contexto. Además, destacar que la entrada $w_{t-k}, ..., w_{t-1}, w_{t+1}, ..., w_{t+k}$ es una secuencia de vectores *one-hot* (o un vector de índices) que permite obtener un vector concreto de la *lookup table*, actualizándose dicha matriz durante el proceso de entrenamiento de la red (ésto también se aplica al modelo *skip-gram*).

Con respecto a la complejidad, si N es la dimensión de los elementos de la secuencia de entrada, D la dimensionalidad de los *embeddings* (filas o columnas de la *lookup table*) y V el tamaño del vocabulario, el coste temporal Q de entrenar el modelo CBOW está determinado por la expresión $Q = N * D + D * log_2 V$. Nótese como el coste para estimar $p(w_t|w_{t-k}, ..., w_{t-1}, w_{t+1}, ..., w_{t+k})$, en la última capa de la red, se reduce utilizando técnicas como *hierarchical softmax* o *negative sampling* de V a $log_2 V$.

Por último, comentar que esta arquitectura suaviza la distribución de probabilidad modelada [53], es por ello que, sobre corpus de entrenamiento con grandes cantidades de muestras, los vectores de representación obtenidos no capturan tantas propiedades lingüísticas como los obtenidos mediante el modelo *skip-gram*. En resumen, el

comportamiento de CBOW es mejor en problemas donde este efecto regulador es beneficioso.

3.1.2.2. Skip-gram

La segunda arquitectura tratada en [51] [53] es el modelo *skip-gram* (SG), que pretende predecir el contexto de un constituyente w_t dado, de forma inversa a CBOW cuyo objetivo es predecir w_t conociendo su contexto delimitado en ambos extremos por una ventana de tamaño k. En la Figura 3.7 se puede visualizar gráficamente la arquitectura mencionada.

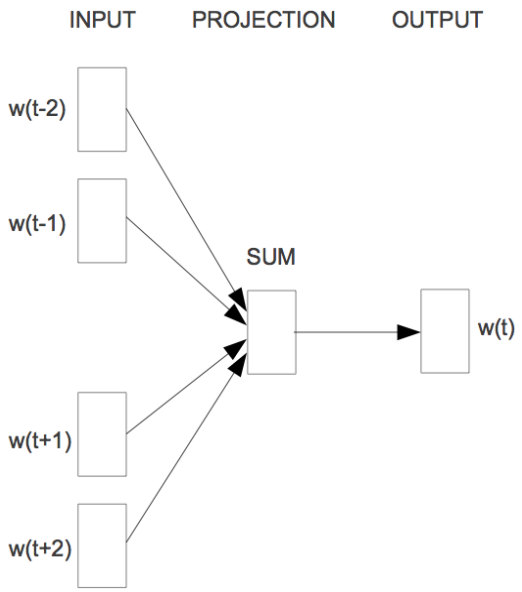

Figura 3.7: Arquitectura SG [51], [53].

Formalmente, dada una secuencia de constituyentes de entrenamiento $w_1, w_2, ..., w_T$, el objetivo es maximizar el logaritmo de la probabilidad $\frac{1}{T}\sum_{t=1}^{T}\sum_{-k\leq j\leq k, j\neq 0}\log(w_{t+j}|w_t)$, donde T es la longitud de la secuencia, y k es el tamaño del contexto.

Si el parámetro k tiene un valor elevado, los ejemplos de entrenamiento serán más completos al considerar un mayor número de constituyentes en la salida (nótese el incremento en la dificultad y la complejidad temporal de la tarea), permitiendo generar vectores de representación de mayor calidad. Ésto provoca también, un buen comportamiento del modelo ante corpus con un número elevado de muestras de aprendizaje, ya que el modelo es capaz de extraer mucha más información que CBOW en estos

casos.

Con respecto a la formulación de la probabilidad condicional de ocurrencia de un contexto $w_{t+j} - k \leq j \leq k, j \neq 0$ dada una palabra w_t, $p(w_{t-k}, ..., w_{t-1}, w_{t+1}, ..., w_{t+k}|w_t)$, se define mediante la función *softmax* de la misma manera que en el modelo CBOW, $\frac{exp((v'_{w_O})^T v_{w_I})}{\sum_{w=1}^{V} exp((v'_{w_O})^T v_{w_I})}$. Donde v_w y v'_w son los vectores de representación de entrada (extraídos de la *lookup table*) y salida respectivamente de w y V es la talla del vocabulario.

Por último, la complejidad temporal de entrenamiento de la presente arquitectura está determinada por la expresión $Q = C * (D + D * \log_2 V)$. Donde C es la máxima distancia entre palabras del corpus de entrenamiento y D es la dimensión de los *embeddings*. Nótese que el coste temporal es mayor en comparación con la arquitectura anterior y, como en este caso, es necesaria la utilización de técnicas para reducir la complejidad de cálculo en la capa *softmax* como *Hierarchical Softmax* o *Negative Sampling* [51] [53].

3.1.3. Specific Word Embeddings

Como se ha observado en apartados anteriores, los *embeddings* son capaces de capturar regularidades lingüísticas en función de los contextos de los constituyentes, haciendo que constituyentes semejantes en contexto tengan representaciones similares. Sin embargo, este comportamiento puede ser un problema en el caso de algunas tareas.

Un ejemplo de este problema se puede observar en la Figura 3.8, donde palabras con polaridad totalmente opuesta ocurren en contextos similares por lo que sus *embeddings* serán semejantes. Por ejemplo, en problemas de *sentiment analysis* esto no es deseable. Además, este mismo problema puede ocurrir en otras tareas si constituyentes que, en principio son discriminativos para la clasificación realizada, tienen *embeddings* similares.

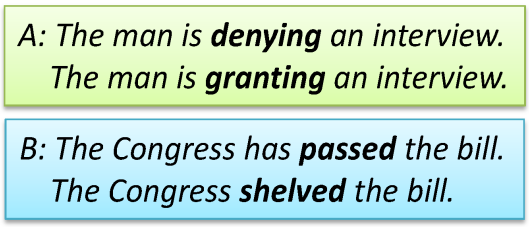

Figura 3.8: Palabras con polaridades opuestas tienen contextos similares.

Por este motivo surge la necesidad de especializar los *embeddings* no solo para capturar contextos similares, si no para lidiar también con propiedades más específicas como la polaridad (a partir de aquí consideraremos el caso de la polaridad, pero puede ser generalizado para otros conceptos distintos). Los *embeddings* especializados para una tarea determinada se denominan *Specific Word Embeddings* y requieren datos supervisados para la tarea, siendo ésto uno de los principales inconvenientes.

Así, se distinguen tres modelos diferentes para la obtención de este tipo de *embeddings*, siguiendo a [74], especializados en tareas de *sentiment analysis* y conocidos como *Sentiment Specific Word Embeddings* (SSWE).

1. El modelo SSWE-Hard usa únicamente la polaridad (con valores discretos) de cada documento para estimar los *embeddings* de los constituyentes que lo forman, de tal forma que solo dicha polaridad se emplea como salida a predecir dados los constituyentes del documento, tal como se muestra en la Figura 3.9 (nótese que se pierde la información contextual).

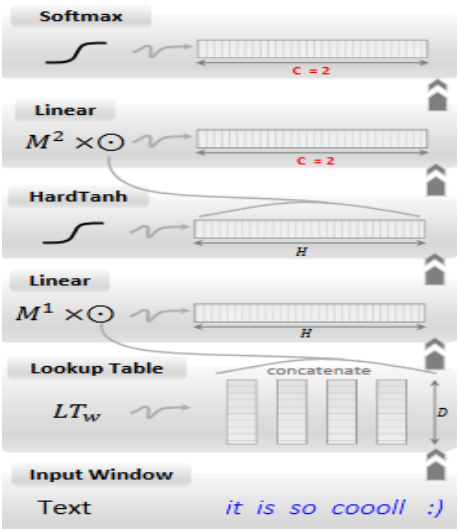

Figura 3.9: Modelo SSWE-Hard [74].

2. Por otro lado, la discretización de la polaridad en la salida puede no ser una buena idea en casos donde los constituyentes no tengan polaridades extremas e.g. totalmente positivo o totalmente negativo, si no que puedan tener un determinado grado de positivo y de negativo. El modelo que elimina dicha discretización (véase que la última capa del modelo es lineal, en lugar de utilizar una *softmax* como en el caso anterior) y emplea en su salida valores continuos para estimar los *embeddings* en la *lookup table* se denomina SSWE-Soft y se muestra en la Figura 3.10 (nótese que también se pierde la información contextual).

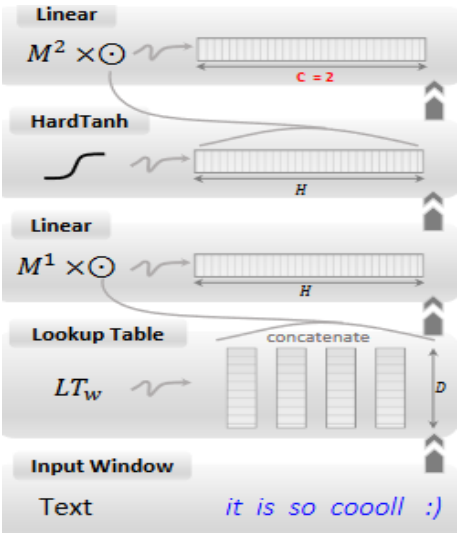

Figura 3.10: Modelo SSWE-Soft [74].

3. El último modelo propuesto en [74], SSWE-Unified pretende incorporar información contextual a los modelos anteriores, ya que en ambos casos únicamente se tiene en cuenta la polaridad para estimar los *embeddings*. Para ésto, se emplea la misma red que en el caso de SSWE-Soft pero se emplea una función de *loss* que combina linealmente una *loss* para el contexto sintáctico y otra para un concepto específico como la polaridad. Por tanto, dicha función de *loss* es de la forma $loss(y, \hat{y}) = \alpha \cdot loss_{si}(y, \hat{y}) + (1 - \alpha) \cdot loss_{sp}(y, \hat{y})$, donde $loss_si$ y $loss_sp$ son la *loss* sintáctica y específica respectivamente (se pueden encontrar más detalles en [74]).

Así, en algunas tareas abordadas en el presente proyecto se ha empleado un modelo híbrido, SSWE-Híbrido, similar a SSWE-Unified, pero optimizando en dos pasos la *loss* sintáctica y la *loss* específica (polaridad). Concretamente, en primer lugar se optimiza la *loss* sintáctica utilizando un modelo *skip-gram* para, posteriormente, refinar los *embeddings* de la *lookup table* usando *tweets* «supervisados» de forma automática en función de determinados emoticonos presentes en dichos tweets i.e. $pos = \{x_1, x_2, ..., x_n\}/$ { :), :D, ... } $\in x_i$ y $neg = \{x_1, x_2, ..., x_n\}/$ { :(, ;(, ... } $\in x_i$.

Por último, en la tabla 3.1 se puede observar cómo afecta la especialización de *embeddings*, en una tarea de *sentiment analysis*, sobre la polaridad de las palabras. Con ello, al obtener las cinco palabras más cercanas a «mejor» y a «peor», el modelo *skip-gram* considera que palabras con polaridad totalmente opuesta a la referencia son próximas a ésta, como es el caso de «peor» para la referencia «mejor» y viceversa (marcadas en negrita en la tabla). Así, el modelo SSWE-Híbrido soluciona dicha problemática.

SSWE-Híbrido		Skip-gram	
Mejor	Peor	Mejor	Peor
mejot	malo	mejorr	peorrrr
importante	triste	mejorrr	**mejor**
mejorr	pobre	mejorrrr	peorr
mejir	mala	mejir	peorrrrr
mejorrrrrrr	tampoco	**peor**	peor...

Tabla 3.1: Comparación entre SSWE-Híbrido y *Skip-gram*.

3.2. Representación de documentos

Hasta ahora se han comentado métodos para representar constituyentes de documentos, sin embargo, el principal objetivo del proyecto consiste en resolver problemas de *text classification* a nivel de documento y no de constituyentes individuales. Un punto a tener en cuenta es que la representación de documentos se realiza en base a las representaciones de sus constituyentes, por lo que si se tienen buenas representaciones de constituyentes, generalmente, se obtienen representaciones de documentos de calidad.

Formalmente, dado un conjunto de documentos $D = \{d_1, d_2, ..., d_n\}$ donde d_i es un documento determinado, el problema consiste en representar cada $d_i \in D$ como un punto s en un espacio métrico o pseudométrico S, donde se puede definir una función de distancia entre los elementos de dicho conjunto S [85].

Con respecto a los tipos de representación, podemos encontrar dos tipos generales. Por un lado, aquellas representaciones con pérdida de información temporal y de las relaciones entre los constituyentes del documento. Entre estas destacan *bag of words* (BOW) y colapsado de *embeddings* (en el caso de que se empleen *embeddings* como representación de los constituyentes). Además, modelos de redes neuronales, que traten con vectores de entrada, como MLP, pueden manejar este tipo de representación.

Por otro lado, es posible representar sin pérdida de información temporal, al mismo tiempo que se mantienen las relaciones entre los constituyentes, un determinado documento mediante una representación secuencial similar a las secuencias de cepstrales en problemas de *speech*. En este caso, ha sido necesario el uso de modelos de redes neuronales que manejen secuencias, e.g. redes neuronales recurrentes o convolucionales.

Además, las representaciones no son excluyentes entre sí y podemos emplear combinaciones de representaciones, de tal forma que cada una de dichas representaciones dé cuenta de información diferente a la que capturan el resto de representaciones utilizadas, consiguiendo así una mayor comprensión del documento que, posiblemente, permita simplificar la resolución de las tareas.

Por último, en las siguientes subsecciones se comentan con detalle todas las representaciones enunciadas, comenzando por aquellas que condensan globalmente la información del documento sin tener en cuenta relaciones temporales (BOW y colapsado de *embeddings*) y finalizando por aquellas que, potencialmente, pueden mantener dicha información temporal.

3.2.1. Bag of words

Mediante este tipo de representación, inicialmente propuesto en [33], un documento se representa como un único vector cuya dimensionalidad coincide con la talla del vocabulario, de tal forma que cada componente x_i del vector de representación $x = \{x_1, x_2, ..., x_n\}$ está asociado a un constituyente i concreto. Así, podemos considerar una representación binaria de los constituyentes o la frecuencia de éstos en el documento (TF). Por tanto, para representar un documento d, la componente x_i de su representación x contendrá la presencia o frecuencia del constituyente i. Un ejemplo de ésto se muestra en la Figura 3.11.

Figura 3.11: Ejemplo de representación *bag of words*.

Por otro lado, es importante notar cómo esta representación ignora el orden de los constituyentes en el documento y provoca una pérdida de las relaciones temporales entre estos (aunque se pueden emplear n-gramas para capturar ordenaciones locales a corto término entre constituyentes). Además, nótese la similitud con la representación *one-hot* de constituyentes, ya que la representación *bag of words* de un documento consiste en la suma de las representaciones *one-hot* de los constituyentes que lo componen.

Sin embargo, a pesar de la pérdida de información temporal, este tipo de representación ha presentado buenos resultados en las tareas abordadas donde el aspecto más discriminativo es la frecuencia de ciertas palabras clave, en lugar de sus relaciones con otros constituyentes del documento.

3.2.2. Colapsado de embeddings

Otra opción para la representación vectorial de documentos consiste en llevar a cabo una composición de *embeddings* de constituyentes, capaces de capturar gran parte de la semántica presente en dichos documentos (o secuencias de unidades lingüísticas con significado). Esta área todavía es un tema de investigación activo y abierto [5] [55] y están por determinar las propiedades de cada tipo de composición en tareas concretas.

Así, de entre las posibles composiciones que se pueden emplear para combinar *embeddings* de constituyentes, destacamos, siguiendo a [24], los siguientes métodos:

- **Suma**: el enfoque consiste, únicamente, en sumar los vectores de representación de todos los constituyentes w que conforman un documento s para obtener su representación vectorial continua, $repr(s) = \sum_{w \in s} emb(w)$. Adecuada ante casos en los que es posible afirmar que la semántica de un documento se puede expresar como la suma de las semánticas de los constituyentes que lo forman. Este enfoque se ha empleado en algunas tareas, obteniendo resultados muy competitivos sin necesidad de normalizar por la longitud del documento, debido al tipo de dominio tratado (Twitter).

- **Centroide**: idéntico a la representación anterior, pero normalizando con respecto al número de constituyentes que se encuentran en el documento y han aparecido durante el entrenamiento del modelo de *embeddings* (m), $repr(s) = \frac{1}{|\{s_1, s_2, ..., s_n\} : s_i \in m|} \sum_{w \in s} emb(w)$. De esta forma, se evita que la norma del vector $repr(s)$ se incremente cuanto mayor sea el número de palabras en s.

- **Palabra próxima al centroide**: en este caso, se considera como representación de la frase el vector de representación del constituyente, presente en el documento, que más próximo se encuentre con respecto al centroide comentado en el punto anterior, $repr(s) = emb(w_i) : w_i \in s \ \wedge \ \nexists w_j \in s : dist(w_j, centroide) < dist(w_i, centroide)$. Esta representación es útil cuando el significado de una frase o documento viene determinado por la semántica de una palabra de referencia que se encuentre en ella.

- **Ponderada *embeddings* con *tf-idf***: otro de los enfoques que surge tras definir la representación mediante la suma de palabras, consiste en realizar una combinación lineal de la forma $repr(s) = \sum_{i=1}^{n} k_i * emb(w_i)$, considerando los vectores de constituyentes y un término que permita ponderarlos de una manera determinada. Entre los tipos de ponderaciones que es posible realizar, podemos destacar la aleatoria, donde cada constituyente es ponderado por un valor aleatorio o la ponderación basada en información de POS, en la que no se ponderan de igual forma las categorías gramaticales. Otra posible ponderación consiste en hacer uso del peso *tf-idf* de cada constituyente para obtener vectores de representación de frases, $repr(s) = \sum_{w \in s} emb(w) * tfidf(w)$.

- **Producto**: el planteamiento es idéntico al de la suma de palabras, pero empleando la función producto en lugar de la suma, $repr(s) = \prod_{w \in s} emb(w)$. El principal inconveniente de esta representación es que las componentes del vector $emb(w)$ son reales pequeños cuyo producto da lugar a reales todavía menores, con lo que se puede alcanzar una pérdida de precisión considerable. Además, la intuición de aplicar la operación producto para combinar constituyentes no es tan clara como la suma de *embeddings*.

- **Derivada**: es similar a las fórmulas de diferencias finitas y consiste en acumular las derivadas de cada dimensión del vector según la expresión, $repr(s) = \sum_{i=0}^{|s|-1} emb(w_{i+1}) - emb(w_{i-1})$. Es adecuado cuando la semántica, o propiedades objetivo como la polaridad, vienen determinadas por la diferencia entre las semánticas o contextos de palabras adyacentes en la frase.

Así, de la misma forma que con la representación basada en *bag-of-words*, el principal problema de colapsar toda la información de los constituyentes es la pérdida de las relaciones temporales. Esto es debido a que el documento deja de considerarse como una secuencia de unidades para ser representado por un vector de representación global que considera la información de todos los constituyentes independientemente de sus posiciones en el documento y las relaciones entre éstos. Por este motivo, en el siguiente apartado se enuncia un tipo de representación que sí mantiene dicha información secuencial.

3.2.3. Representación secuencial

Este tipo de representación consiste en considerar un documento como una secuencia de representaciones de los contituyentes que lo forman. Es muy similar a las representaciones utilizadas en problemas de *speech* ya que un documento queda representado como una sucesión de vectores que son la representación de sus constituyentes (véase la Figura 3.12. Nótese la problemática de tratar con secuencias de longitud variable debido a que no todos los documentos constan del mismo número de constituyentes.

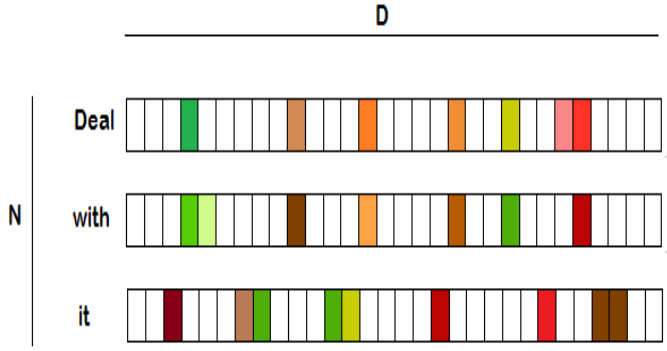

Figura 3.12: Ejemplo de representación secuencial para la frase «Deal with it».

En este caso, es necesario destacar la capacidad de esta representación para mantener las relaciones temporales entre los constituyentes de los documentos. Sin embargo, los modelos de clasificación deben dar cuenta de relaciones más complejas que en el caso de las representaciones globales como BOW, por lo que es necesario disponer de un mayor número de datos para obtener resultados competitivos.

Por último, mencionar que en todas las tareas del presente trabajo se ha expermientado con este tipo de representaciones de documentos, junto con redes neuronales recurrentes y convolucionales, considerando como representaciones de los constituyentes *one-hot* y *embeddings* extraídos de diversas fuentes en función de la tarea tratada.

3.2.4. Combinación de representaciones

Un caso interesante en la representación de documentos consiste en la combinación de representaciones [87]. Para ello, es posible realizar cualquier tipo de combinación de

representación de documentos en función de diferentes representaciones de constituyentes, típicamente, en base a constituyentes complejos extraídos de diversas fuentes (POS, synsets, lexicones de polaridad, etc.) que, idealmente, aporten información complementaria y no solapada.

Algunos casos destacados con los que se ha experimentado durante el presente proyecto y en [24] son los siguientes:

- ***Embeddings in-domain, out-domain* y lexicones de polaridad**: con el objetivo de refinar los *embeddings out-domain* (obtenidos con fuentes de texto externas al corpus de la tarea e.g. Wikipedia) y tener información específica sobre la tarea, se hace uso de *embeddings in-domain* (obtenidos a partir del corpus de la tarea), además, esto permite reducir el número de palabras fuera de vocabulario tanto en el conjunto de entrenamiento como en el de test. Por otro lado, se emplean lexicones de polaridad para representar un documento de acuerdo a las polaridades de sus constituyentes. En resumen, la representación de un documento consiste en una 3-tupla de secuencias, una de ellas compuesta por *embeddings out-domain*, otra por *embeddings in-domain* y la última por vectores *one-hot* de las polaridades. Esta representación se discute con más detalle en el siguiente capítulo y se ha empleado en una tarea de *sentiment analysis* en SemEval2017@ACL, requiriendo un modelo de clasificación específico que también se discute en dicho capítulo.

- ***Embeddings* ponderados con *tf-idf***: esta combinación ya ha sido mencionada en el apartado dedicado a colapsado de *embeddings* y no requiere un modelo específico para ser procesada ya que únicamente se lleva a cabo una ponderación de los *embeddings* de los constituyentes mediante sus pesos *tf-idf* estimados con el corpus de entrenamiento para refinar los *embeddings* de acuerdo a la tarea. Esta representación fue utilizada en un problema de *topic detection* sobre el corpus TC-STAR en [24].

Por último, a pesar de que únicamente se ha experimentado con las combinaciones mencionadas, se pueden tener en cuenta otros tipos de combinaciones para la representación de documentos con el objetivo de cubrir la mayor cantidad de información relevante para solucionar una determinada tarea de PLN.

Capítulo 4

SemEval 2017: Task 4

> Caballeros, debo recordarles que,
> mis probabilidades de éxito,
> aumentan en cada nuevo intento.
>
> John Nash

4.1. Introducción

En esta sección se describe la participación del equipo ELiRF-UPV en la tarea 4 de SemEval2017 [27]. Nuestro enfoque está basado en el uso de tres *stacks* de redes convolucionales y recurrentes, así como en la combinación de *embeddings* específicos (extraídos de la tarea, *in-domain*) y generales (extraídos de conjuntos externos, *out-domain*) con lexicones de polaridad. Además, hemos participado en todas las subtareas de la tarea propuesta, tanto en árabe como en inglés usando el mismo sistema con pequeñas variaciones en los parámetros determinados en la etapa de ajuste.

Twitter se ha convertido en una fuente que recoge gran cantidad de información de forma distribuida, por ello, proporciona un amplio abanico de posibilidades para realizar investigaciones en campos de PLN como *sentiment analysis*. *Sentiment analysis* o *opinion mining*, es un área de investigación dentro de PLN cuyo objetivo es identificar la emoción o polaridad subyacente en un determinado documento, frase o aspecto [48].

Con respecto a las aplicaciones donde surge este problema, podemos identificar, entre otros, la clasificación de opiniones [77, 57], la generación de resúmenes basados en aspectos [36] o la identificación de tendencias políticas [60].

Los organizadores de la tarea 4 de SemEval 2017 han propuesto cinco tareas diferentes. Todas estas tareas están relacionadas, en general, con *sentiment analysis* en *Twitter*, sin embargo, entre ellas hay algunas diferencias significativas. Además, en la edición de este año 2017, las cinco tareas también se han propuesto para idioma árabe. En resumen, los participantes pueden elegir entre diez subtareas a abordar.

La subtarea A consiste en predecir la polaridad global de un *tweet* como positiva, negativa o neutra. En las subtareas B y C, dado un mensaje y un tema, el sistema debe clasificarlo en una escala de polaridad de dos puntos o de cinco puntos respectivamente ($\tilde{p} \in \{0, 1\}$ o $\tilde{p} \in \{-2, -1, 0, 1, 2\}$).

Las subtareas D y E tratan el problema de cuantificación de *tweets*, esto es, dado un conjunto de *tweets* sobre un tema dado, estimar la distribución de los *tweets* de acuerdo a una escala de dos puntos o de cinco puntos respectivamente.

Con respecto a la evaluación, se emplean métricas diferentes para cada subtarea. Para las subtareas A y B se emplea *macroaveraged recall* (*recall* promediado sobre las tres clases) y en la subtarea C la métrica es *macroaveraged mean absolute error*. Por otro lado, las subtareas D y E se evalúan con la divergencia Kullback-Leibler y *Earth Mover's Distance* respectivamente.

El resto del capítulo se organiza como sigue: primero se discuten algunos detalles de los corpus empleados, posteriormente se describe el sistema desarrollado y los recursos de los que hace uso, para, finalmente, comentar los resultados obtenidos con dicho sistema en las diferentes subtareas e idiomas.

4.2. Corpus

En la tarea tratada se proporcionan seis corpus diferentes, de los cuales, la mitad son para las subtareas en inglés y el resto para las subtareas en árabe. De forma general, hay dos corpus para la subtarea A, dos para las subtareas B-D y otros dos para las subtareas C-E.

En todos los casos, es importante notar que el corpus está desbalanceado, existiendo grandes diferencias en las probabilidades *a priori* de las clases en la mayoría

de las subtareas para ambos idiomas. Este desbalanceo se ha reducido mediante la eliminación de muestras de las clases más pobladas de forma aleatoria. El desbalanceo mencionado se puede observar en las Figuras 4.1, 4.2, 4.3, 4.4, 4.5 y 4.6.

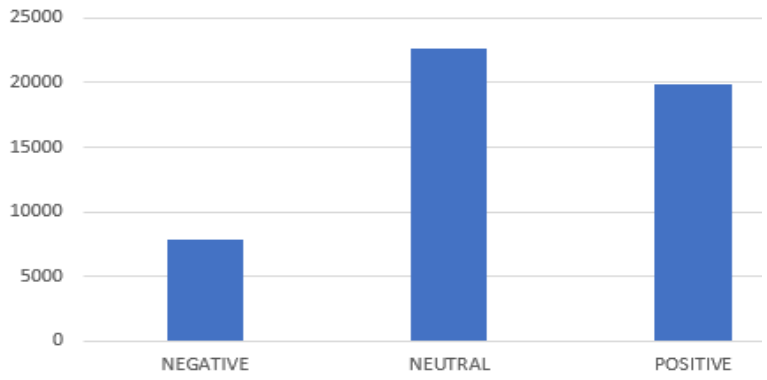

Figura 4.1: Número de muestras por clase en la subtarea A inglés.

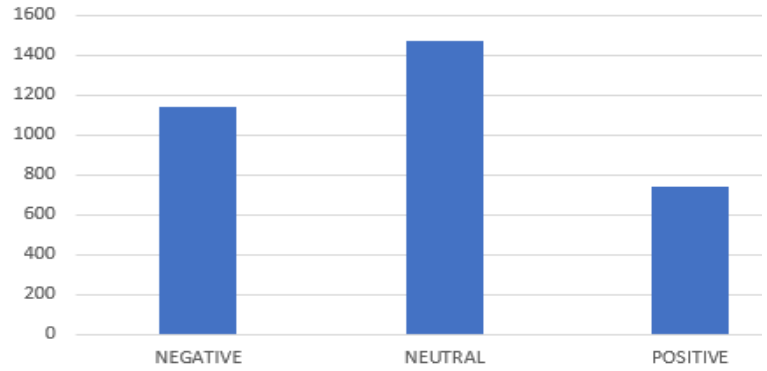

Figura 4.2: Número de muestras por clase en la subtarea A árabe.

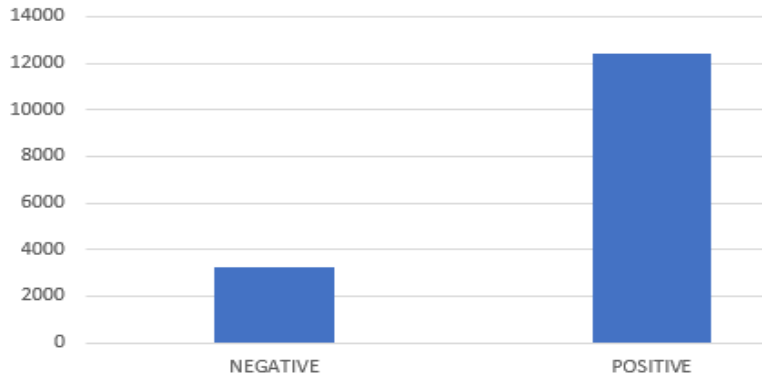

Figura 4.3: Número de muestras por clase en las subtareas B-D inglés.

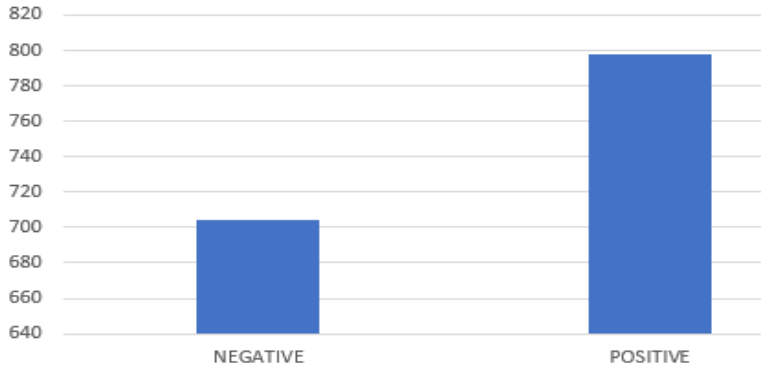

Figura 4.4: Número de muestras por clase en las subtareas B-D árabe.

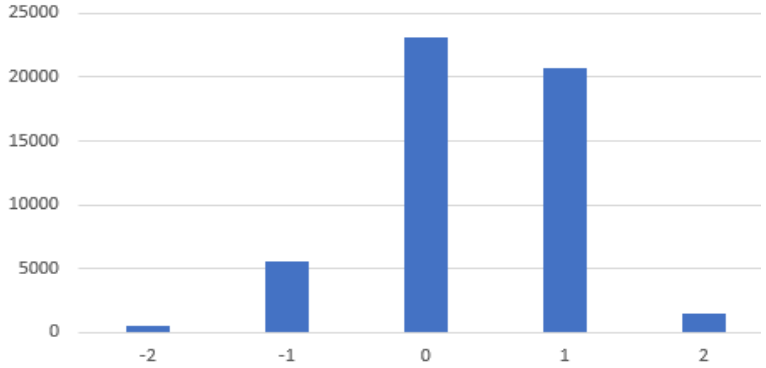

Figura 4.5: Número de muestras por clase en las subtareas C-E inglés.

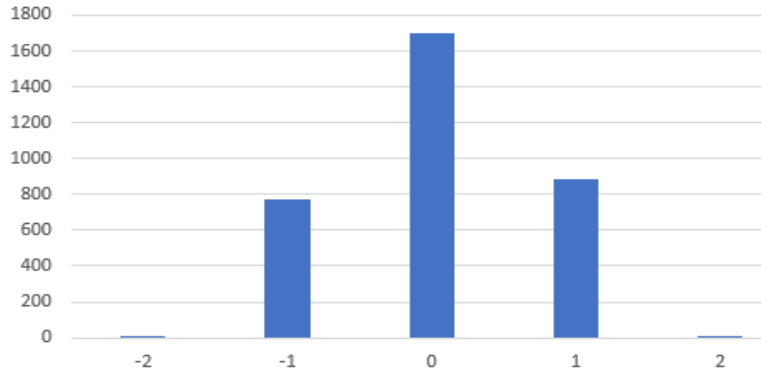

Figura 4.6: Número de muestras por clase en las subtareas C-E árabe.

Además, en todas las subtareas se han extraído, de forma aleatoria a partir de las muestras proporcionadas, conjuntos de validación formados por el 20 % del total de las muestras de cada subtarea para ajustar los parámetros de los modelos.

4.3. Descripción del sistema

En esta sección se describe la arquitectura del sistema utilizado para todas las subtareas. El sistema está basado en el uso de redes convolucionales y recurretes y la combinación de *word embeddings* específicos y generales con lexicones de polaridad.

Para la adaptación del sistema a cada subtarea e idioma, se han realizado ciertas modificaciones con el objetivo de ajustar los parámetros en función de los recursos disponibles en ésta, esto es el tamaño del corpus de dicha subtarea.

El sistema combina tres *stacks* de redes convolucionales y recurrentes para aprender características de alto nivel [46] a partir de representaciones ruidosas [38] para mejorar la generalización de los modelos. La entrada de estas tres redes son: *out-domain embeddings*, *in-domain embeddings* y secuencias de vectores *one-hot* que representan la polaridad de las palabras de la frase. La salida de estas tres subredes se combina y se utiliza como entrada para un modelo discriminativo implementado en términos de un MLP. La Figura 4.7 representa gráficamente el enfoque propuesto.

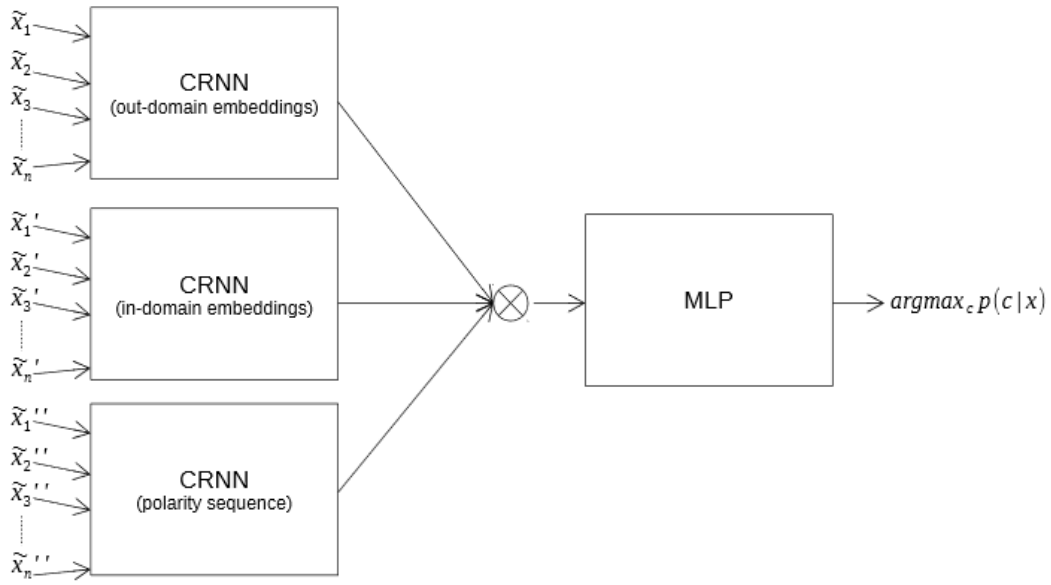

Figura 4.7: Arquitectura general del sistema para SemEval 2017 Task 4.

Los *stacks* de redes convolucionales y recurrentes utilizan como primera capa una capa convolucional unidimensional, que permite abstraer relaciones espaciales entre los vectores que representan las palabras (tanto *embeddings* como polaridades) de la

frase o *tweet*.

El número de capas convolucionales varía entre uno y dos, el número de filtros o *kernels* entre 8 y 96 y la altura de los filtros entre 2 y 4 dependiendo de la tarea. Además, se utiliza Batch Normalization y funciones de activación ReLU tras cada capa convolucional. También, en algunas subtareas se realiza un proceso de *down-sampling* mediante capas *max pooling*.

La salida de la última capa *max pooling* se utiliza como entrada para una LSTM. Además, debido a que la polaridad de una subsecuencia de la sentencia no depende únicamente de palabras pasadas, sino que también depende de palabras futuras, hemos utilizado las LSTM de forma bidireccional [35, 67]. En la mayoría de las subtareas, solo se ha empleado una capa LSTM bidireccional y la dimensionalidad de salida de dicha capa se ha variado entre 32 y 256.

La Figura 4.8 muestra una representación gráfica de los *stacks*, donde $\tilde{x}_i$ es una versión corrupta de la entrada mediante ruido gaussiano $r \sim \mathcal{N}(\mu = 0, \sigma = 0{,}3)$, c_i son los kernels de la capa convolucional, p_i representa la operación de *max pooling* y y_s es la salida del *stack* s.

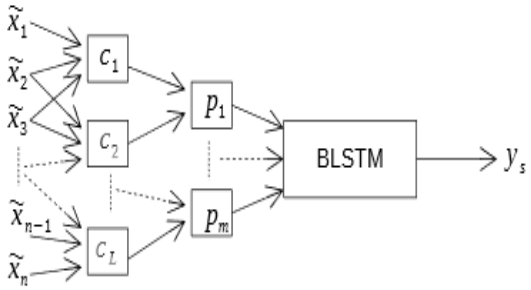

Figura 4.8: Arquitectura de un *stack* de redes convolucionales y LSTM.

La última red utilizada en nuestro sistema es un MLP para llevar a cabo la clasificación a partir de las características extraídas por los *stacks*. Dependiendo de la subtarea se han utilizado entre 1 y 3 capas ocultas con funciones de activación ReLU. El número de neuronas se ha variado en función de la subtarea entre 64 y 1024. También, se ha empleado una función de activación en la capa de salida para estimar $p(c|x)$ y el número de neuronas en esta capa depende del número de clases en la

subtarea tratada.

En la Figura 4.9 puede verse una representación gráfica del MLP utilizado, donde y_i es la salida del *stack* i, que son concatenadas para utilizarlas como entrada del MLP. Además, nótese que en este caso, no se ha aplicado ruido a las entradas debido a que de esta manera se han obtenido mejores resultados en la fase de ajuste.

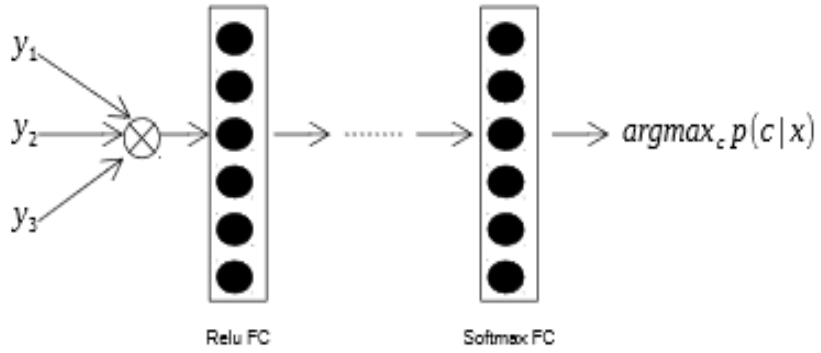

Figura 4.9: Representación del MLP utilizado para SemEval 2017 Task 4.

4.4. Recursos

Como se ha mencionado anteriormente, se han utilizado dos tipos de *embeddings* (*in-domain* y *out-domain*) como entrada del sistema para todas las subtareas, tanto en árabe como en inglés.

Se han utilizado ambos modelos de *embeddings* con el objetivo de reducir el número de palabras no vistas. De esta forma, nosotros combinamos representaciones específicas de la tarea que solo consideran palabras vistas en el conjunto de entrenamiento (*in-domain embeddings*) con representaciones más generales entrenadas con grandes conjuntos de datos.

Así, para las subtareas en inglés, se ha utilizado como modelo *out-domain* un modelo *Word2Vec* entrenado con 400 millones de *tweets* en inglés recogidos por Fréderic Godin [23, 63]. Para las tareas en árabe, se ha entrenado un modelo *Word2Vec* usando artículos de Wikipedia en árabe [82]. Con respecto a los modelos *in-domain*, un modelo *Word2Vec* ha sido entrenado para cada subtarea a partir del corpus proporcionado.

Además de las dos representaciones basadas en *embeddings*, hemos incluido información sobre polaridad como una representación adicional de la entrada. Para la inclusión de dicha información, se ha considerado una representación de los *tweets* basada en una secuencia de vectores *one-hot* que indican la polaridad de cada palabra de acuerdo a un determinado lexicón de polaridad.

De esta forma y, una vez más, los recursos dependen del idioma. Concretamente, hemos utilizado el lexicón NRC [56] tanto para las subtareas en árabe como en inglés y el lexicón *Afinn* [31] solo para las subtareas en inglés.

4.5. Resultados

En esta sección se presentan algunas de las modificaciones realizadas en la arquitectura general de nuestro sistema para cada subtarea en la que se ha participado. También, discutimos los resultados obtenidos en las diferentes subtareas.

Estas modificaciones, como ya se ha comentado, han consistido en la reducción o el incremento del número de parámetros a estimar en función del tamaño del corpus en cada subtarea. Esto nos ha permitido obtener modelos mejor estimados manteniendo una arquitectura genérica.

4.5.1. Subtarea A

En primer lugar, la subtarea A consiste en clasificar un mensaje o *tweet* como positivo, negativo o neutro. Nuestro modelo para esta subtarea consta de tres *stacks* de redes convolucionales y recurrentes cuyas salidas son concatenadas y tomadas como entrada para un MLP de tres capas (Figura 4.7).

Los resultados obtenidos por nuestro sistema en la subtarea A se muestra se muestran en la tabla 4.1, donde también se incluye, para cada medida, la posición alcanzada por nuestro sistema con respecto al resto de participantes. La métrica empleada para clasificar los resultados de los participantes es *macroaveraged recall* (ρ). Además, se consideran dos medidas adicionales: F_1 promediada entre las clases positivo y negativo (F_1^{PN}) y *Accuracy* (*Acc*).

Subtask A	English		Arabic	
ρ	63.20	(14/38)	47.80	(3/8)
F_1^{PN}	61.90	(12/38)	46.70	(4/8)
Acc	59.90	(24/38)	50.80	(3/8)

Tabla 4.1: Resultados de la subtarea A de SemEval 2017 Task 4.

Nótese las diferentes posiciones obtenidas por nuestro sistema considerando ρ y Acc para el idioma inglés. En función de ρ se alcanza la posición 14 mientras que según Acc el puesto obtenido es el 24º. Una posible explicación a ésto se fundamenta en la estrategia que hemos utilizado para abordar el problema del desbalanceo entre las clases del corpus (eliminación de muestras de las clases más pobladas).

De forma contraria, para la subtarea en árabe, los resultados de Acc no parecen verse influenciados por la manera en la que manejamos el problema del desbalanceo, alcanzando posiciones similares en todas las métricas consideradas.

4.5.2. Subtarea B

En la subtarea B, dado un mensaje y un tema, los participantes deben clasificar el mensaje en una escala de dos puntos (positivo y negativo) considerando el tema dado. Desafortunadamente, no disponemos de recursos para incluir información sobre el tema en el modelo y, en consecuencia, se ha usado el modelo genérico mostrado anteriormente.

En este caso, la capa de *max pooling* se reemplaza por otra capa convolucional, el número de neuronas en las capas *fully-connected* del MLP se reduce y se utiliza ruido gaussiano sobre las activaciones de las neuronas del MLP, debido a que esta configuración obtiene los mejores resultados sobre el conjunto de validación.

Para el idioma árabe también se ha utilizado la misma topología pero reduciendo el número de parámetros (disminuyendo d_Y en LSTM, número de filtros en las capas convolucionales y el número de neuronas en las capas del MLP) debido al tamaño del corpus de entrenamiento en árabe.

Los resultados obtenidos por nuestro sistema en la subtarea B se muestran en la tabla 4.2. Además, las métricas consideradas son las mismas que en la subtarea A.

Subtask B	English		Arabic	
ρ	76.60	(17/23)	72.10	(2/4)
F_1^{PN}	77.30	(16/23)	72.40	(2/4)
Acc	79.00	(13/23)	73.40	(2/4)

Tabla 4.2: Resultados de la subtarea B de SemEval 2017 Task 4.

Los valores de todas las métricas son mejores que los obtenidos en la subtarea A, sin embargo, esta subtarea quizás es más simple debido a que solo se consideran dos clases (sin tener en cuenta el tema hacia al que van dirigidos los mensajes). Aún así, comparados con el resto de participantes nuestro sistema no es tan competitivo. Una posible explicación es la no inclusión en el modelo de información sobre los temas (ninguno de los temas de test aparece en entrenamiento). Nótese que el comportamiento obtenido por el sistema en ambas lenguas es similar.

4.5.3. Subtarea C

En esta subtarea, dado un mensaje y un tema, los participantes deben clasificar el mensaje en una escala de cinco puntos hacia el tema. Como en la subtarea B, nosotros no hemos incluido información sobre los temas en el modelo.

Así, nuestro modelo es una extensión del modelo genérico con dos capas convolucionales y dos capas de *max pooling* en cada red convolucional del *stack*. Además, para el idioma árabe, también se ha utilizado el modelo genérico con menos parámetros debido al tamaño del corpus disponible.

Las métricas utilizadas para evaluar a los participantes son *macroaveraged Mean Absolute Error* (MAE^M) y una extensión de *macroaveraged recall* para regresión logística (MAE^μ). Los resultados obtenidos por nuestro sistema en la subtarea C se muestran en la tabla 4.3.

Subtask C	English		Arabic	
MAE^M	0.806	(7/15)	1.264	(2/2)
MAE^μ	0.586	(11/15)	0.787	(2/2)

Tabla 4.3: Resultados de la subtarea C de SemEval 2017 Task 4.

Como se puede observar, nuestro sistema obtiene una 7ª posición (0.806), con una amplia diferencia en comparación al primer clasificado (0.481) para el inglés. Una vez más, la inclusión de información sobre el tema en el modelo puede ser un factor decisivo en el rendimiento del sistema.

4.5.4. Subtarea D

La subtarea D consiste en una cuantificación de los *tweets* de acuerdo a una escala de dos puntos. Dado un conjunto de *tweets* sobre un tema dado, los participantes deben estimar la distribución de los *tweets* en una escala de dos puntos (positivo y negativo). Con ello, nosotros empleamos la salida de la subtarea B para estimar, por máxima verosimilitud, la distribución de los *tweets*.

La principal métrica utilizada para evaluar a los participantes es la divergencia de Kullback-Leibler (KLD), además, se consideran métricas adicionales como *Absolute Error* (AE) y *Relative Absolute Error* (RAE). Los resultados obtenidos por nuestro sistema en la subtarea D se muestran en la tabla 4.4.

Subtask D	English		Arabic	
KLD	1.060	(14/15)	1.183	(3/3)
AE	0.593	(15/15)	0.537	(3/3)
RAE	7.991	(15/15)	11.434	(3/3)

Tabla 4.4: Resultados de la subtarea D de SemEval 2017 Task 4.

A la vista de los resultados, exceptuando posibles errores en la implementación del código para llevar a cabo la tarea, una posible explicación a los malos resultados obtenidos se fundamenta en que la estimación se realiza por máxima verosimilitud sobre resultados poco competitivos extraídos de una tarea previa.

4.5.5. Subtarea E

De forma similar a la subtarea D, la subtarea E consiste en una tarea de cuantificción pero en una escala de cinco puntos. Para esta subtarea, también se ha empleado la salida de la subtarea C para estimar, por máxima verosimilitud, la distribución de los *tweets*.

En este caso, la métrica utilizada para evaluar a los participantes es *Earth Mover's Distance* (EMD). Con ello, los resultados obtenidos por nuestro sistema en la subtarea E se muestran en la tabla 4.5.

Subtask E	English	Arabic
EMD	0.306 (4/12)	0.564 (2/2)

Tabla 4.5: Resultados de la subtarea E de SemEval 2017 Task 4.

Destacar que nuestro sistema obtiene una 4ª posición (0.306) para el idioma inglés con poca diferencia en relación al primer clasificado (0.245), lo que llama la atención, debido a que el enfoque utilizado es el mismo que para la tarea anterior, donde los resultados eran bastante peores.

Capítulo 5

IberEval 2017

> Si A es el éxito en la vida, entonces A = X + Y + Z. Donde X es trabajo, Y es placer y Z es mantener la boca cerrada
>
> Albert Einstein

5.1. Introducción

El *workshop* IberEval 2017 se enmarca dentro del XXXIII Congreso Internacional de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN), donde participan expertos en lenguaje natural que proponen soluciones a los problemas planteados, relacionados con las tecnologías del lenguaje humano.

Los principales objetivos del congreso SEPLN consisten en estudiar las líneas de investigación más actuales en el ámbito del PLN y en contrastar dichas investigaciones con las necesidades del mercado. Además, el congreso constituye un entorno que favorece la participación y colaboración entre investigadores del área de PLN.

El taller IberEval 2017, tal como su nombre indica, plantea problemas relacionados con las tecnologías del lenguaje humano en idiomas de la península ibérica (español, portugués, catalán, vasco y gallego). Dentro de dicho taller, se proponen una serie de tareas, a destacar: *Classification Of Spanish Election Tweets* (COSET) [22] y Stance and Gender detection in *tweets* on Catalan Independence [75].

Es en estas dos tareas en las que nuestro equipo ha participado. En el taller se propusieron tres tareas más: Biomedical Abbreviation Recognition and Resolution,

Collective Elaboration of a Coreference Annotated Corpus for Portuguese Texts y Multilingual Web Person Name Disambiguation (M-WePNaD).

En este capítulo, se presenta el trabajo realizado para el taller IberEval 2017, concretamente, para las tareas COSET, Stance y Gender. Además, se incluye una introducción a los problemas que tratan las tareas, los sistemas utilizados y los resultados obtenidos en la competición.

5.2. COSET

La tarea *Classification Of Spanish Election Tweets* (COSET) [22] ha sido organizada mediante una colaboración entre el centro de investigación *Pattern Recognition and Human Language Technology* (PRHLT) y el grupo de investigación *Mediaflows*.

Dicha tarea plantea el problema de la detección de temas en conversaciones políticas, extraídas de *Twitter*, sobre la precampaña y campaña de las elecciones generales de 2015. Estas conversaciones políticas se incrementan conforme las elecciones se acercan y analizar los temas de discusión tratados por los usuarios proporciona información relevante para algunos problemas relacionados con las elecciones (predicción de resultados, influencia en la población, etc.).

COSET proporciona un corpus que permite entrenar sistemas de detección de temas en cinco categorías: cuestiones políticas relacionadas con la confrontación electoral; políticas sectoriales, cuestiones personales sobre los candidatos, temas de la campaña electoral y otras cuestiones.

La tarea comenzó el 20 de marzo de 2017 con la liberación del conjunto de entrenamiento y un pequeño subconjunto para validación. El 24 de abril se liberó el conjunto de test y el 8 de mayo se enviaron los resultados. Posteriormente, el 15 de mayo fueron publicados los resultados de la competición, donde nuestro mejor sistema alcanzó el primer puesto [25].

La evaluación se llevó a cabo mediante la métrica macro F_1 (5.1), que es la media de las F_1 (5.2) de cada clase. Ésto permite evaluar dichos sistemas sin tener en cuenta el desbalanceo entre las clases (el corpus está muy desbalanceado, tal como se verá

en el siguiente apartado) ya que todas las clases influyen igualmente en el resultado, independientemente de su número de muestras.

$$F_{1-macro} = \frac{1}{|L|} \sum_{l \in L} F_1(y_l, \bar{y}_l) \tag{5.1}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.2}$$

$$precision = \frac{1}{|L|} \sum_{l \in L} Pr(y_l, \bar{y}_l) \tag{5.3}$$

$$recall = \frac{1}{|L|} \sum_{l \in L} R(y_l, \bar{y}_l) \tag{5.4}$$

5.2.1. Corpus

Como se ha comentado anteriormente, el corpus COSET está formado por *tweets* pertenecientes a cinco categorías: cuestiones políticas relacionadas con la confrontación electoral; políticas sectoriales, cuestiones personales sobre los candidatos, temas de la campaña electoral y otras cuestiones.

Estos *tweets* están escritos en español y fueron recogidos durante las elecciones generales españolas del 2015. Se proporciona un conjunto de entrenamiento y otro de validación, formados por 2242 y 250 muestras, respectivamente. Es necesario considerar que el corpus está desbalanceado, tal como se muestra en la Figura 5.1, existiendo un sesgo claro entre las tres primeras clases (1, 2 y 9) y las dos últimas (10, 11).

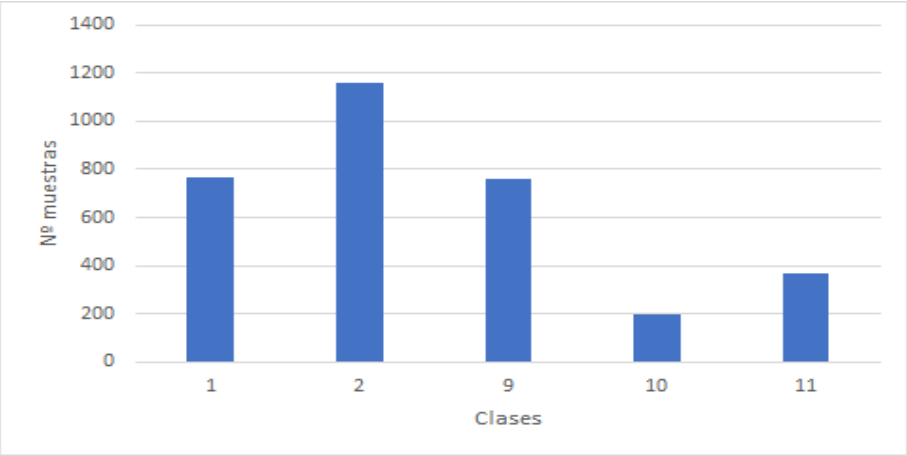

Figura 5.1: Número de muestras por clase en el corpus de COSET.

En la tabla 5.1 se muestran características adicionales del corpus como el número de palabras por clase, el número promedio de palabras por *tweet*, la talla del vocabulario y las palabras fuera del vocabulario.

Longitud media por *tweet*	22.34				
Talla del vocabulario	7241				
Palabras fuera de vocabulario	458				
Clases	**1**	**2**	**9**	**10**	**11**
Palabras por clase	12456	18363	11010	3297	4955
Media de palabras por clase	23.50	23.36	21,54	21.69	18.84

Tabla 5.1: Estadísticas del corpus COSET.

Con respecto al preproceso realizado, en primer lugar se convierte el texto a minúsculas y se eliminan los acentos. Las urls, menciones de usuario, números, exclamaciones e interrogaciones se sustituyen por tokens especiales para normalizar el valor de dichos tokens e.g. #VotaPP, #VotaPSOE y cualquier *hashtag* se convertiría en <hashtag>.

Por último, tras este preproceso se realiza un tipo de tokenización que varía en función de la experimentación, generalmente mediante una expresión regular, por ejemplo, *(?u)\b\w\w+\b*, que está implementada en librerías como *Scikit-Learn* [59] y *Keras* [14].

5.2.2. Características de los modelos

Para las experimentaciones se han evaluado distintos tipos de representación de los *tweets*, determinando dicha representación el modelo a utilizar. Por un lado, se han empleado representaciones vectoriales como *bag-of-ngrams* o *tf-idf* a nivel de palabras y de caracteres. También se ha experimentado con métodos de colapsado de *embeddings*, mediante la suma de los *embeddings* de las palabras de un *tweet*. Este tipo de representaciones es procesada por modelos que toman como entrada un único vector de representación, e.g. SVM o MLP.

Por otro lado, también se han usado representaciones secuenciales compuestas por una secuencia de *embeddings* obtenidos con *Word2Vec* [62]. En este caso (también en el colapsado de *embeddings* mencionado), se han utilizado *embeddings* de Wikipedia en español [83] de 128 dimensiones, ventana 10, *skipgram* y *negative sampling*.

Además, como experimentación adicional, se han combinado los *embeddings* de Wikipedia con *embeddings* entrenados con el propio corpus [65].

Destacar que la representación basada en *bag-of-1grams* sobre palabras ha obtenido los mejores resultados para la tarea, lo que parece indicar que, en este problema, es más importante la frecuencia de aparición de ciertas palabras clave que otro tipo de información contextual e.g. *embeddings* o a nivel de documento e.g. *tf-idf*.

5.2.3. Sistemas desarrollados

Todos los sistemas con los que se ha experimentado varían en función del tipo de representación empleada. Así, se han utilizado diversos tipos de sistemas (y ajustes de un sistema concreto) para un mismo tipo de representación con el objetivo de maximizar la métrica de evaluación (macro F_1) sobre el conjunto de validación proporcionado.

En primer lugar, se experimentó con una representación *bag-of-words*, *tf-idf* sobre palabras y colapsado de *embeddings* mediante suma, probando con varios clasificadores tal como se indica en la tabla 5.2.

	MLP	SVM (Lineal)	Random forest	Adaboost (DT)
Bag-of-words	68.23	58.66	51.69	35.05
TF-IDF	59.31	55.16	50.97	33.49
Colapsado de *embeddings*	46.73	48.85	39.02	34.64

Tabla 5.2: Resultados obtenidos en la fase de ajuste con representaciones vectoriales.

Como se puede observar, los mejores resultados se alcanzan mediante una representación *bag-of-words* utilizando un MLP como clasificador. Dicho MLP está compuesto por 3 capas ocultas con 128 neuronas y funciones de activación ReLU, excepto la capa de salida, con función de activación Softmax.

El método de optimización utilizado ha sido Adagrad y se ha optimización la entropía cruzada (función de *loss*). Además, se ha realizado un escalado de dicha función de *loss*, con un factor de suavizado a ajustar, para tratar con el problema del desbalanceo entre clases (véase la sección 2.5.3.1).

Sobre el MLP, otros métodos para mejorar la generalización y regularización como la adición de ruido gaussiano [30], *dropout* [71] o *Batch Normalization* [37] también se han empleado, peró no han conseguido mejorar los resultados obtenidos por dicha red.

Por otro lado, también se ha experimentado con representaciones secuenciales de los *tweets*, compuestas por vectores *one-hot* y *embeddings* (el modelo de *embeddings* ha sido comentado en la sección 5.2.2). En este caso, la red utilizada es una combinación de capas convolucionales, recurrentes y *fully-connected*, similar a la utilizada en [27]. Los resultados de la red con ambos tipos de representación se muestra en la tabla 5.3.

	One-Hot	Embeddings
CNN+LSTM	40.56	55.43

Tabla 5.3: Resultados obtenidos en fase de ajuste con representaciones secuenciales.

La red utilizada para los dos tipos de representaciones secuenciales es la misma y consiste en dos capas convolucionales 1D de 32 y 64 kernels con anchura 3, una capa LSTM, una capa *max pooling* de tamaño 3, una capa LSTM con 128 neuronas (función de activación ReLU) y tres capas *fully-connected* de 64 neuronas con funciones de activación ReLU.

Además, entre las capas *fully-connected* se emplea *dropout* con $p = 0{,}15$ para reducir el sobre ajuste y se emplea la misma técnica que en el caso anterior, basada en el escalado de la función de *loss*, para manejar el desbalanceo entre clases. Estas redes han sido entrenadas hasta que la macro F_1 de validación no varía o se reduce.

Como se puede observar comparando las tablas 5.2 y 5.3, las representaciones vectoriales se comportan mejor que las representaciones secuenciales en este problema, llegando a incrementar, en el mejor caso, hasta 22 puntos la macro F_1. Por este motivo, uno de los sistemas enviados para participar en la competición es el MLP ya comentado, entrenado a partir de una representación *bag-of-words* de los *tweets*.

Por último, aparte de dicho sistema, se enviaron tres más para comprobar el efecto del escalado de la función de pérdida sobre el resultado final con un corpus desbalanceado como es el propuesto por COSET. La lista de sistemas enviados es la siguiente:

Sistema 1. entrenado con todos los datos y escalado ajustado sobre validación.

Sistema 2. entrenado sin la partición de desarrollo y escalado ajustado sobre validación.

Sistema 3. entrenado con todos los datos sin escalado.

Sistema 4. votación de los tres runs anteriores.

5.2.4. Resultados

El 15 de mayo del 2017, la organización de COSET liberó los resultados de los diferentes sistemas enviados por los participantes de la competición. La evaluación de todos los sistemas se muestra en la tabla 5.4.

Sistema	Macro F_1
ELiRF-UPV-run1	**64.82**
ELiRF-UPV-run4	**64.00**
LuSer-run1	63.37
ELiRF-UPV-run3	**63.33**
ELiRF-UPV-run2	**62.33**
Puigcerver-run1	61.76
atoppe-run3	61.57
atoppe-run2	60.65
LTRC_IITH-run2	60.54
LTRC_IITH-run3	59.60
LTRC_IITH-run1	59.59
atoppe-run5	59.52
Carl os duty-run1	59.02
CD_team-run1	58.59
MiVal-run2	58.52
Carl os duty-run2	58.22
Electa-run1	57.84
atoppe-run1	57.84
MiVal-run1	57.33
Citripio-run1	56.76
ConradCR-run1	56.39
LichtenwalterOlsan-run1	55.90
UT text miners-run3	55.41
atoppe-run4	54.76
Puigcerver-run3	52.76
ivsanro-run1	52.34
LTRC_IITH-run5	44.35
Baseline-TF-IDF+RF	42.36
slovak-run1	42.33
UT text miners-run1	36.31
UT text miners-run2	33.41
UC3M-run3	27.55
UC3M-run4	27.55
Baseline-BOW+SVM	26.44
UC3M-run5	26.15
UC3M-run1	25.71
UC3M-run2	25.58
Team 17-run2	24.46
Team 17-run1	24.10
Baseline- Most frequent	10.70

Tabla 5.4: Resultados oficiales de COSET.

Como se puede observar en la tabla 5.4, el modelo entrenado con todos los datos y un escalado de la función de *loss* ajustado sobre validación proporciona los mejores resultados, seguido del cuarto sistema que clasifica un *tweet* en la clase más votada por los otros 3 sistemas. Con ello, dos de los sistemas propuestos han alcanzado las dos primeras posiciones del ranking.

Los run2 y run3 han conseguido un cuarto y quinto puesto respectivamente, superados por *LuSer-run1*, también basado en MLP entrenado con una representación *bag-of-words* de las muestras. El siguiente mejor modelo, del participante *Puigcerver*, también basado en MLP, ya se encuentra casi un punto por debajo de nuestro peor sistema y tres puntos por debajo del mejor sistema propuesto.

Por último, tal como indica la clasificación final y las diversas experimentaciones realizadas, los enfoques basados en representaciones vectoriales procesadas por redes neuronales han conseguido los mejores resultados en la tarea, lo que parece indicar que con los datos disponibles, representaciones demasiado complejas no consiguen mejorar los resultados.

5.3. Stance and Gender

El objetivo de la tarea *Stance and Gender Detection in Tweets on Catalan Independence* [75], es detectar el género (masculino y femenino) y la orientación hacia la independencia de Cataluña (a favor, neutral y en contra) de *tweets* escritos en español y en catalán. Nuestra participación se ha limitado al español, debido principalmente a que no disponíamos de recursos para el catalán.

Para el caso de Stance, es importante notar que no es exactamente un problema clásico de *sentiment analysis* que pretende analizar la polaridad subyacente en un texto determinado. En este caso, se pretende extraer la opinión favorable o desfavorable hacia un tema determinado, que puede no estar expresado explícitamente en el texto.

Este problema surge debido a que, normalmente, los comentarios van dirigidos hacia un tema en particular y puede ser más interesante explorar el posicionamiento

respecto a dicho tema en lugar de obtener una etiqueta general. Además, es de especial interés en el estudio político de temas controvertidos como, en este caso, la independencia de Cataluña.

La tarea comenzó el 21 de marzo de 2017 con la liberación del conjunto de entrenamiento. El 24 de abril se liberó el conjunto de test y el 8 de mayo se enviaron los resultados. Posteriormente, el 17 de mayo fueron publicados los resultados de la competición, donde alcanzamos el primer puesto en Gender y un 4º puesto en Stance [26].

La evaluación se realizó de acuerdo a métricas estándar [75]. En particular, para evaluar la tarea de stance, la organización propuso la métrica macro F_1 (5.1) sobre las clases *Favor* y *Against*, siguiendo la evaluación propuesta en la tarea 6 de *SemEval 2016*. Para Gender, la métrica a evaluar es el *Accuracy*, de acuerdo a las métricas propuesta en la tarea *Author Profiling* de *PAN@CLEF*.

5.3.1. Corpus

El corpus está compuesto por *tweets* que tratan sobre el debate de la independencia de Cataluña [75]. extraídos de la red social *Twitter* durante las elecciones regionales de septiembre de 2015, que fueron interpretadas por muchos políticos y ciudadanos como un referéndum sobre la posible independencia de Cataluña.

Con respecto al número de muestras, es necesario tener en cuenta que el corpus está desbalanceado en cuanto a las clases sobre la independencia de Cataluña, existiendo un claro sesgo, tal como se muestra en la Figura 5.2, entre las clases *AGAINST* y *NEUTRAL* con respecto a la clase *FAVOR*. Este desbalanceo no ocurre en la tarea de Gender, como se puede observar en la Figura 5.3.

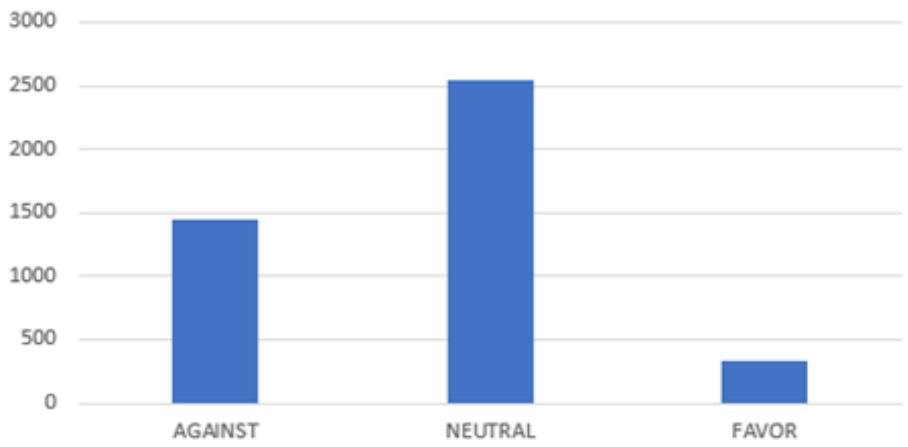

Figura 5.2: Número de muestras por clase para la tarea Stance.

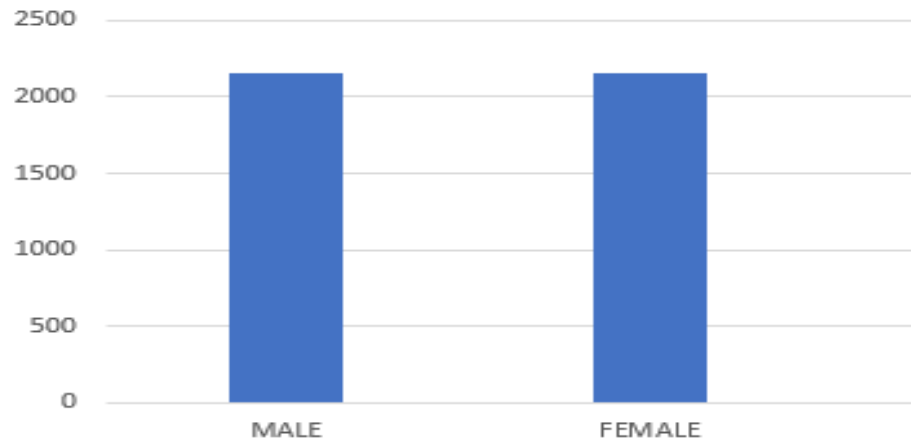

Figura 5.3: Número de muestras por clase para la tarea *Gender*.

En ambos casos se ha utilizado la misma partición del corpus para entrenamiento y desarrollo, 80 % para el conjunto de entrenamiento, y 20 % para validación, escogidos de forma aleatoria de entre todo el conjunto de muestras para reducir sesgos en el conjunto de validación debido al orden de las muestras.

Considerando dicha partición del conjunto de muestras en entrenamiento y validación, algunas características del corpus se muestran en la tabla 5.5.

	Palabras	Caracteres
Longitud media por *tweet*	18.09	108.32
Talla del vocabulario	7221	68
Fuera de vocabulario	866	2
Longitud media por clase (STANCE)	AGA $\|\|NEU\|\|FAV$ 17.85 ‖18,10‖19,05	AGA $\|\|NEU\|\|FAV$ 108.83 ‖107,42‖112,97
Longitud media por clase (GENDER)	MAL $\|\|FEM$ 18.70 ‖17,47	MAL $\|\|FEM$ 111.03 ‖105,62

Tabla 5.5: Estadísticas del corpus Stance and Gender.

En el preproceso realizado para la tarea de detección de género, se eliminan acentos y se convierten a minúsculass los *tweets* del corpus. También se sustituyen urls, emoticonos, menciones, *hashtags* y números por tokens especiales (se asume que es más importante el hecho de poner *hashtags*, emoticonos, menciones y números que el valor concreto de dichos tokens). Además, se usa el mismo tokenizador que en la tarea COSET, una expresión regular del estilo de *(?u)\b\w\w+\b*.

En el preproceso para la tarea de Stance, también se eliminan acentos y se convierten a minúsculas los *tweets*. En este caso, se sustituyen urls y números y se mantienen emoticonos, menciones y *hashtags* (para esta tarea sí interesa más el valor concreto de los emoticonos, menciones y *hashtags* que el hecho de poner algún tipo de estos tokens en el *tweet*). Además, se emplea el tokenizador de *Keras* [14], el cual separa por espacios y no elimina signos de puntuación.

5.3.2. Características de los modelos

Para llevar a cabo la representación de los *tweets* e intentar capturar la información más relevante de éstos, se han considerado distintos tipos de características. Entre las características utilizadas, destacamos:

- **Embeddings**: representación secuencial de *tweets*, formada por *embeddings Word2Vec* de las palabras que las componen. Estos *embeddings* han sido entrenados con el *framework* Gensim [62] y artículos de Wikipedia en español [83].
- **Bag-of-ngrams**: unigramas y bigramas de palabras y caracteres usando una representación basada en bolsa de ngramas.

- **One-hot vectors**: representación secuencial de *tweets*, formada por vectores *one-hot* de las palabras que las componen.

Por último, debido a que la tarea de Stance está estrechamente relacionada con el análisis de sentimientos, también hemos probado el uso de lexicones de polaridad. Concretamente, se ha hecho uso del lexicón NRC traducido al español [56] como característica adicional a las planteadas en esta sección.

5.3.3. Sistemas desarrollados

Hemos explorado diferentes modelos, que varían en función del tipo de representación de los *tweets*. De esta manera, para tratar con las representaciones secuenciales formadas por vectores que representan palabras o caracteres (*embeddings* y vectores one-hot respectivamente), se ha empleado un *stack* de redes recurrentes, concretamente LSTM y CNN.

Por otro lado, para tratar con las representaciones basadas en *bag-of-ngrams* (tanto a nivel de palabra como a nivel de caracter), hemos empleado SVM [16] y MLP.

Con respecto a las características adicionales, extraídas del lexicón de polaridad NRC, éstas solo se han empleado con la representación secuencial que hace uso de *embeddings*. Con esto, la topología de la red neuronal usada como modelo, es muy similar a la descrita en [27] pero sin la subred dedicada a procesar la secuencia de *embeddings* obtenidos con el corpus de entrenamiento.

Una vez se han determinado los modelos, se debe realizar una etapa de ajuste para seleccionar la representación y los modelos (incluyendo sus hiper-parámetros) más adecuados para cada subtarea, considerando las métricas de evaluación como criterio de optimización. Este proceso de ajuste se realiza con la partición entrenamiento y validación.

Además, debido a la imposibilidad de evaluar todas las posibles combinaciones de modelos y representaciones, únicamente se han considerado aquellas que tienen más sentido o se han comportado correctamente en problemas abordados anteriormente. La tabla 5.6 muestra las combinaciones más relevantes de características y modelos,

así como los resultados obtenidos durante la fase de ajuste.

Sistema	Características	Stance (macro F_1)	Gender (Accuracy)
CNN+LSTM	*Embeddings*	51.84	64.47 %
CNN+LSTM	*Embeddings*+NRC	48.80	-
CNN+LSTM	*One-Hot*	**55.10**	-
MLP	*Word* 1-2gramas	-	59.72 %
MLP	*Char* 2-gramas	-	63.81 %
SVM	*Word* 1-2gramas	-	58.30 %
SVM	*Char* 2-gramas	-	**66.92** %
SVM	*Char* 1-2-gramas	-	**66.99** %

Tabla 5.6: Resultados obtenidos en la fase de ajuste.

Con respecto a la tarea Stance, tal como se puede ver en la primera fila de la tabla 5.6, con la representación secuencial a nivel de *embeddings* de palabras, los resultados obtenidos son de 51.84 macro-F_1. Desafortunadamente, se obtienen peores resultados si esta representación se combina con la secuencia de polaridades de las palabras de la frase (obtenidas mediante el lexicón NRC).

Como ya se ha comentado, también se ha empleado una representación secuencial formada por vectores one-hot a nivel de caracter, la cual es procesada con la misma red de experimentos anteriores (*stack* de LSTM y CNN). Esta representación junto con dicho *stack* de redes neuronales ha obtenido los mejores resultados en la etapa de ajuste, 55.10 macro-F_1, tal como se puede ver en la tercera fila de la tabla 5.6.

Por otro lado, con respecto a la tarea de Gender, aunque la representación secuencial a nivel de palabras (con *embeddings* de Wikipedia) procesada usando el *stack* de LSTM y CNN, ha obtenido buenos resultados, 64.47 % *Accuracy*, los mejores resultados en la fase de ajuste se han alcanzado haciendo uso de representaciones basadas en *bag-of-ngrams* de caracteres. La misma representación (*bag-of-ngrams*) pero a nivel de palabra ha obtenido peores resultados en esta tarea.

Los modelos que mejores resultados han obtenido son SVM con *kernel* lineal. Más concretamente, el modelo SVM que usa *bag-of-unigrams* de caracteres como representación de *tweets* ha alcanzado 66.92 % de *Accuracy*; mientras que, si se concatena la representación *bag-of-bigrams* de caracteres al modelo *bag-of-unigrams*, se incrementa ligeramente el *Accuracy* a 66.99 %. Estos resultados se pueden observar en las dos

últimas filas de la tabla 5.6.

5.3.4. Resultados

En vista de los resultados obtenidos durante la etapa de ajuste y debido a las limitaciones de la tarea, decidímos enviar únicamente los dos siguientes *runs* a la competición:

- **Run1**
  - **Stance detection**: CNN + LSTM + *char-one-hot*
  - **Gender detection**: SVM + *bag-of-2grams* (caracteres)
- **Run2**
  - **Stance detection**: CNN + LSTM + *char-one-hot*
  - **Gender detection**: SVM + *bag-of-1grams* (caracteres) + *bag-of-2grams* (caracteres)

Las tablas 5.7 y 5.8 muestran los resultados oficiales obtenidos por nuestros sistemas en las tareas de Stance y Gender respectivamente. La posición obtenida por nuestros sistemas en la competición se incluye entre paréntesis.

run	Sistema y características	macro F_1
run1/run2	CNN+LSTM + *char-one-hot vectors*	46.37 (4/35)

Tabla 5.7: Resultados oficiales en la tarea Stance.

run	Sistema y características	Accuracy
run1	SVM + *bag-of-2grams of chars*	68.55 % (1/23)
run2	SVM + *bag-of-1grams + bag-of-2grams of chars*	58.74 % (16/23)

Tabla 5.8: Resultados oficiales en la tarea Gender.

Una vez analizados los resultados obtenidos, tanto en la etapa de ajuste como en la competición oficial sobre el test final, consideramos necesario destacar algunas situaciones interesantes.

En ambas tareas, los métodos basados en *deep-learning* ofrecen resultados competitivos. Sin embargo, en el caso de la tarea Gender, los mejores resultados se han obtenido con un modelo basado en SVM con representación *bag-of-chars* de los *tweets*. Otro punto a destacar es que nuestro sistema 1 es el único de la competición que supera el mejor *baseline* propuesto por los organizadores, además, con cierta holgura (3 puntos de *Accuracy*).

Siguiendo con el problema del desbalanceo en el caso de Stance, si dicho desbalanceo es muy pronunciado, puede causar que los modelos basados en redes neuronales asignen a todas las muestras la clase mayoritaria (las probabilidades a priori dominan la clasificación). Esto se ha solucionado en la tarea Stance mediante un escalado de la función de *loss* durante la fase de entrenamiento, de igual forma que en la tarea de COSET.

Así, abordar el problema del desbalanceo ha resultado ser un factor clave para entrenar correctamente los modelos basados en redes neuronales, lo que ha permitido prevenir que dichos modelos clasifiquen todos los *tweets* en las clases *AGAINST* y *NEUTRAL* (clases mayoritarias con mucha diferencia en el corpus español de Stance).

Continuando con la tarea Stance, la representación secuencial a nivel de caracteres con vectores *one-hot* ha sido utilizada debido al creciente interés que este tipo de representaciones está teniendo en el área del *deep-learning* aplicada a problemas de lenguaje natural y a los buenos resultados que permite obtener en problemas similares [86].

Por último, hemos podido comprobar como, efectivamente, este tipo de representación (en conjunto con un *stack* de LSTM y CNN que permite manejar secuencias de vectores) provee resultados competitivos en problemas de *text classification* como la tarea de Stance, alcanzando un cuarto puesto con 46.37 de macro-F_1.

Capítulo 6

Conclusiones

> La vida es el arte de sacar conclusiones suficientes de premisas insuficientes.
>
> Samuel Butler

Durante la realización del presente proyecto se han alcanzado con éxito los objetivos propuestos, entre los que destacan la formación del alumno en áreas como el *deep learning* y el desarrollo e implementación de técnicas para problemas relacionados con el área de PLN, de forma que éstas, han sido empleadas en el proyecto ASLP-MULAN [84] y sirven como base para algunos de los sistemas a desarrollar en próximos proyectos.

Con respecto a la formación del alumno en el área del *deep learning*, se ha llevado a cabo un proceso de estudio de la bibliografía más relevante, que propone modelos que constituyen el estado del arte en algunos de los problemas tratados, y se han implementado algunos de dichos modelos con el objetivo de ser utilizados en trabajos futuros para tareas de PLN.

Por otro lado, en lo referente a las representaciones de texto, se ha realizado un estudio de distintos tipos de representaciones, procesadas en las tareas abordadas mediante modelos de *deep learning*, con el objetivo de analizar propiedades de éstas para determinar su interés en determinados tipos de tareas, como por ejemplo *sentiment specific word embeddings* [74] en tareas de *sentiment analysis*.

Así, gran parte de las técnicas de *deep learning* aplicables a problemas de PLN y las representaciones de texto, comentadas en los capítulos dos y tres de la presente

memoria respectivamente, han sido evaluadas en diversas tareas propuestas en competiciones nacionales e internacionales como SemEval2017@ACL [65], IberEval@SEPLN [75] [22] y TASS@SEPLN, obteniendo resultados competitivos en ellas [25] [26].

Algunas tareas mencionadas a lo largo de la memoria han sido excluidas de ésta, debido a ciertas limitaciones relacionadas con el espacio y el tiempo necesario para detallarla, a destacar: *Fine-Grained Sentiment Analysis on Financial Microblogs and News* de SemEval2017@ACL o *Sentiment Analysis at Tweet level* y *Aspect-based Sentiment Analysis* de TASS@SEPLN.

También, para el proyecto ASLP-MULAN en el cual se enmarca este trabajo de fin de máster, se han desarrollado e implementado con éxito otros sistemas que no han sido expuestos en la memoria, como bots conversacionales mediante modelos *sequence-to-sequence* [78] o modelos de lenguaje recurrentes para generación de texto [2].

En lo referente a los recursos computacionales que han sido utilizados a lo largo del proyecto, en algunas ocasiones de forma paralela, por ejemplo para entrenar varios modelos de *embeddings* simultáneamente, destacar el empleo de dos tarjetas gráficas GeForce GTX 1080 y una GeForce GTX TITAN X, instaladas en tres máquinas con procesadores Intel Xeon CPU E5-1660 v2@3.70 GHz y Xeon CPU E5-1620 v3@3.70 GHz.

Se ha logrado cierta especialización en áreas de interés para el alumno como son el *deep learning* y PLN, aplicando los conocimientos adquiridos en el desarrollo e implementación de sistemas de interés, evaluados en tareas de *workshops* nacionales e internaciones y con aplicabilidad en el proyecto ASLP-MULAN, para problemas de gran repercusión en la actualidad como *sentiment analysis*, *stance detection* o *author profiling*.

Y no solo ésto, también se ha mejorado la capacidad de trabajo en equipo mediante las reuniones realizadas con los investigadores del proyecto ASLP-MULAN y los tutores del trabajo de fin de máster, adquiriendo así competencias transversales relacionadas con comunicación efectiva y trabajo en equipo.

Capítulo 7

Trabajo futuro

> Dejen que el futuro diga la verdad y evalúe a cada uno de acuerdo a sus trabajos y a sus logros. El presente es de ellos, pero el futuro, por el cual trabajé tanto, es mío.
>
> Nikola Tesla

El trabajo futuro a realizar se enmarca en un nuevo proyecto, pendiente de aprobación, continuación de ASLP-MULAN, que abarca diversas áreas como *Knowledge Processing*, *Multimedia Analytics* y *Inclusive and Natural communication*. Además, el alumno está pendiente de la beca para formación de profesorado universitario (FPU), del Ministerio de Educación Cultura y Deporte (MECD), habiendo superado ya la primera fase, para llevar a cabo la tesis de doctorado en este tipo de temas. Más concretamente, en el nuevo proyecto se explorarán con detalle tareas del área de *Multimedia Analytics*.

Entre estas tareas destacan *sentiment analysis* a nivel de aspectos y *stance*, como continuación de lo ya realizado en este proyecto. Además, se abordarán nuevas tareas como seguimiento de tendencias (aplicado a tendencias políticas, radicalización, etc., comprendiendo los temas que influyen en los picos de las sucesiones), predicción y seguimiento de reputación en redes sociales o detección de emociones, sarcasmo, ironía y otros aspectos clave de las opiniones en dichas redes sociales.

En lo referente a tareas de *Inclusive and Natural Communication*, como extracción de resúmenes, generación de lenguaje o bots conversacionales, se pretende hacer uso de modelos basados en *deep learning*, que constituyen el estado del arte en las tareas mencionadas, algunos de los cuales han sido mencionados en la presente memoria

(*sequence-to-sequence*, modelos de lenguaje recurrentes, etc.).

Con respecto a técnicas más específicas, una de las tareas futuras consiste en la generación de muestras sintéticas (*data augmentation*) con la técnica basada en modelos generativos, comentada en el apartado dedicado a *data augmentation* del capítulo dos. Además, también se plantea la utilización de métodos de optimización bayesiana [70] [4] para estimar los hiperparámetros de los modelos basados en redes neuronales para tareas similares a las ya abordadas.

Por último, con el objetivo de facilitar las realización de futuras tareas de *text classification*, se pretende implementar un *framework* que reúna y simplifique todas las técnicas de PLN comentadas en esta memoria, así como facilite la utilización de modelos basados en *deep learning* aplicables a tareas más generales.

Acknowledgements

This work has been partially supported by the Spanish MINECO and FEDER founds under project ASLP-MULAN: Audio, Speech and Language Processing for Multimedia Analytics, TIN2014-54288-C4-3-R.

Bibliografía

[1] Chatbots with seq2seq. `http://suriyadeepan.github.io/2016-06-28-easy-seq2seq/`. Accedido el 30/06/17.

[2] The unreasonable effectiveness of recurrent neural networks. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`. Accedido el 30/06/17.

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[4] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

[5] William Blacoe and Mirella Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 546–556. Association for Computational Linguistics, 2012.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[7] Tony Cai and Wen-Xin Zhou. A max-norm constrained minimization approach to 1-bit matrix completion. *The Journal of Machine Learning Research*, 14(1):3619–3647, 2013.

[8] Francisco Casacuberta. Chapter 2. multilayer perceptron. Slides of Articifial Neural Networks subject. Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, 2016.

[9] Francisco Casacuberta. Chapter 6. recurrent neural networks. Slides of Articifial Neural Networks subject. Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, 2016.

[10] Akemi Takeoka Chatfield, Christopher G. Reddick, and Uuf Brajawidagda. Tweeting propaganda, radicalization and recruitment: Islamic state supporters multi-sided twitter networks. In *Proceedings of the 16th Annual International Conference on Digital Government Research*, dg.o '15, pages 239–249, New York, NY, USA, 2015. ACM.

[11] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[12] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *CoRR*, abs/1606.03657, 2016.

[13] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[14] François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[15] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[16] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[17] Holk Cruse. *Neural Networks As Cybernetic Systems*. Thieme Medical Publishers, Incorporated, 1996.

[18] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[19] Adit Deshpande. A beginner's guide to understanding convolutional neural networks. https://adeshpande3.github.io/adeshpande3.github.io/A-Beginnerjul 2016.

[20] Patrick Doetsch, Pavel Golik, and Hermann Ney. A comprehensive study of batch construction strategies for recurrent neural networks in mxnet. *CoRR*, abs/1705.02414, 2017.

[21] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[22] M. Giménez, T. Baviera, G. Llorca, J. Gámir, D. Calvo, P. Rosso, and F. Rangel. Overview of the 1st Classification of Spanish Election Tweets Task at IberEval 2017. In *Notebook Papers of 2nd SEPLN Workshop on Evaluation of Human Language Technologies for Iberian Languages (IBEREVAL)*, Murcia, Spain, 2017. CEUR Workshop Proceedings. CEUR-WS.org.

[23] Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. Multimedia lab@ acl w-nut ner shared task: named entity recognition for twitter microposts using distributed word representations. *ACL-IJCNLP*, 2015:146–153, 2015.

[24] José-Ángel González, Lluís-F. Hurtado, and Encarna Segarra. Trabajo fin de grado. Descubrimiento automático de conocimiento. Grado en Ingeniería Informática. Universidad Politécnica de Valencia, 2016.

[25] José-Ángel González, Ferran Pla, and Lluís-F. Hurtado. ELiRF-UPV at IberEval 2017: Classification Of Spanish Election Tweets. In *Notebook Papers of 2nd SEPLN Workshop on Evaluation of Human Language Technologies for Iberian Languages (IBEREVAL)*, Murcia, Spain, 2017. CEUR Workshop Proceedings. CEUR-WS.org.

[26] José-Ángel González, Ferran Pla, and Lluís-F. Hurtado. ELiRF-UPV at IberEval 2017: Stance and Gender Detection in Tweets. In *Notebook Papers of 2nd SEPLN Workshop on Evaluation of Human Language Technologies for Iberian Languages (IBEREVAL)*, Murcia, Spain, 2017. CEUR Workshop Proceedings. CEUR-WS.org.

[27] José-Ángel González, Ferran Pla, and Lluís-F. Hurtado. ELiRF-UPV at SemEval-2017 Task 4: Sentiment Analysis using Deep Learning. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, pages 722–726, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[28] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014.

[29] Constantino Grana. Neighborhood operators. Slides of Computer Vision subject. University of Modena and Reggio Emilia, may 2017.

[30] Yves Grandvalet, Stephane Canu, and Stephane Boucheron. Noise Injection: Theoretical Prospects. *Neural Computation*, 9(5):1093–1108, 1997.

[31] Lars Kai Hansen, Adam Arvidsson, Finn Årup Nielsen, Elanor Colleoni, and Michael Etter. Good friends, bad news-affect and virality in twitter. In *Future information technology*, pages 34–43. Springer, 2011.

[32] David Harris and Sarah Harris. *Digital Design and Computer Architecture, Second Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2012.

[33] Zellig S. Harris. Distributional structure. 10(2-3):146–162, 1954.

[34] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[35] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[36] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM.

[37] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[38] Kam Jim, Bill G. Horne, and C. Lee Giles. Effects of noise on convergence and generalization in recurrent networks. In *Proceedings of the 7th International Conference on Neural Information Processing Systems*, NIPS'94, pages 649–656, Cambridge, MA, USA, 1994. MIT Press.

[39] Michael I. Jordan. Artificial neural networks. chapter Attractor Dynamics and Parallelism in a Connectionist Sequential Machine, pages 112–127. IEEE Press, Piscataway, NJ, USA, 1990.

[40] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759, 2016.

[41] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *CoRR*, abs/1703.05192, 2017.

[42] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[43] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 950–957. Morgan-Kaufmann, 1992.

[44] Andrew Lavin. Fast algorithms for convolutional neural networks. *CoRR*, abs/1509.09308, 2015.

[45] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[46] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 5 2015.

[47] I. J. Leontaritis and S. A. Billings. Input-output parametric models for non-linear systems part ii: stochastic non-linear systems. *International Journal of Control*, 41(2):329–344, 1985.

[48] Bing Liu. *Sentiment Analysis and Opinion Mining. A Comprehensive Introduction and Survey.* Morgan & Claypool Publishers, 2012.

[49] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

[50] Warren S. McCulloch and Walter Pitts. Neurocomputing: Foundations of research. chapter A Logical Calculus of the Ideas Immanent in Nervous Activity, pages 15–27. MIT Press, Cambridge, MA, USA, 1988.

[51] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[52] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

[53] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[54] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[55] Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *ACL*, pages 236–244, 2008.

[56] Saif M. Mohammad and Peter D. Turney. Crowdsourcing a Word-Emotion Association Lexicon. 29(3):436–465, 2013.

[57] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *In Proceedings of EMNLP*, pages 79–86, 2002.

[58] Roberto Paredes. Chapter 5-2. convolutional neural networks. Slides of Articifial Neural Networks subject. Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, 2016.

[59] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[60] Ferran Pla and Lluís-F. Hurtado. Political tendency identification in twitter using sentiment analysis techniques. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 183–192, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.

[61] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[62] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[63] Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1524–1534, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[64] Frank Rosenblatt. The perceptron, a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory (CAL), jan 1957.

[65] Sara Rosenthal, Noura Farra, and Preslav Nakov. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[66] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[67] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997.

[68] Terrence J. Sejnowski and Charles R. Rosenberg. Neurocomputing: Foundations of research. chapter NETtalk: A Parallel Network That Learns to Read Aloud, pages 661–672. MIT Press, Cambridge, MA, USA, 1988.

[69] Hava T. Siegelmann and Eduardo D. Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77 – 80, 1991.

[70] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[71] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[72] David Stutz. Understanding convolutional neural networks. *InSeminar Report, Fakultät für Mathematik, Informatik und Naturwissenschaften Lehr-und Forschungsgebiet Informatik VIII Computer Vision*, 2014.

[73] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[74] Duyu Tang. Sentiment-specific representation learning for document-level sentiment analysis. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 447–452, New York, NY, USA, 2015. ACM.

[75] M. Taulé, M.A. Martí, F. Rangel, P. Rosso, C. Bosco, and V. Patti. Overview of the task of Stance and Gender Detection in Tweets on Catalan Independence at IBEREVAL 2017. In *Notebook Papers of 2nd SEPLN Workshop on Evaluation of Human Language Technologies for Iberian Languages (IBEREVAL)*, Murcia (Spain), September 2017. CEUR Workshop Proceedings. CEUR-WS.org, 2017.

[76] Fredy Rodríguez Torres, Jesús A. Carrasco-Ochoa, and José Fco. Martínez-Trinidad. *SMOTE-D a Deterministic Version of SMOTE*, pages 177–188. Springer International Publishing, Cham, 2016.

[77] Peter D. Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *ACL*, pages 417–424, 2002.

[78] Oriol Vinyals and Quoc V. Le. A neural conversational model. *CoRR*, abs/1506.05869, 2015.

[79] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014.

[80] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L. Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 1058–1066. JMLR Workshop and Conference Proceedings, May 2013.

[81] Jingjing Wang, Wen Feng Lu, and Han Tong Loh. P-smote: One oversampling technique for class imbalanced text classification, 2011.

[82] Wikipedia. Wikipedia arabic dumps, 2017. [Online; accessed 22-February-2017].

[83] Wikipedia. Wikipedia spanish dumps, 2017. [Online; accessed 18-May-2017].

[84] Javier Ferreiros y José Manuel Pardo y Lluís-F Hurtado y Encarna Segarra y Alfonso Ortega y Eduardo Lleida y María Inés Torres y Raquel Justo. Aslp-mulan: Audio speech and language processing for multimedia analytics. *Procesamiento del Lenguaje Natural*, 57(0):147–150, 2016.

[85] Jun Yan. *Text Representation*, pages 3069–3072. Springer US, Boston, MA, 2009.

[86] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 649–657, Cambridge, MA, USA, 2015. MIT Press.

[87] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1):239 – 263, 2002.