Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Descubrimiento automático de conocimiento

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: José Ángel González Barba

Tutor: Lluís Felip Hurtado Oliver

Cotutor: Encarna Segarra Soriano

2015-2016

«*Vires acquirit eundo*»

-Publio Virgilio Marón

Resumen

En el presente proyecto se proponen, evalúan y estudian soluciones basadas en representaciones vectoriales continuas y discretas de palabras y frases a algunos de los problemas más significativos del descubrimiento automático de conocimiento aplicado al lenguaje natural y en general a lenguajes formales. Entre estos problemas destacan la detección de temática, identificación de idioma, análisis de sentimiento y detección de *malware*.

Además, debido a la complejidad que supone el aprendizaje y la utilización de dichas representaciones vectoriales, se ha desarrollado un sistema que facilita las tareas de evaluación, preprocesamiento, extracción de características y visualización de resultados; generalizando los aspectos comunes a todos los problemas abordados.

Destacamos, también, los buenos resultados obtenidos mediante el empleo de las representaciones mencionadas sobre el problema de detección de temática, que constituye el principal problema del proyecto, superando los mejores resultados conocidos, haciendo uso del mismo corpus, que han sido cosechados por investigadores de la Universidad Politécnica de Madrid.

Palabras clave: descubrimiento de conocimiento, representaciones vectoriales continuas, detección de temática, lenguaje natural, modelo de lenguaje.

Abstract

In this project, solutions based on continuous and discrete vector representations of word and sentences are proposed, evaluated and studied by using them in some of the most significant problems in automatic knowledge discovery applied to natural language and generally to formal languages. Among these problems, we highlight topic detection, language identification, sentiment analysis and malware detection.

Furthermore, due to the complexity of learning and use of vector representations, a system that facilitates evaluation tasks, preprocessing, feature extraction and results display has been developed; generalizing this way the common aspects to all the addressed problems.

We highlight too the good results obtained by means of using these representations on topic detection, which is the main problem of the project, surpassing the best known results that have been reached by Polytechnic University of Madrid researchers which uses the same corpus.

Keywords: knowledge discovery, continuous word vector representations, topic detection, natural language, language model

Resum

En el present projecte es proposen, avaluen i estudien solucions basades en representacions vectorials contínues i discretes de paraules i frases a alguns dels problemes més significatius del descobriment automàtic de coneixement aplicat al llenguatge natural i en general a llenguatges formals. Entre aquests problemes destaquem la detecció de temàtica, identificació d'idiomes, anàlisi de sentiment i detecció de *malware*.

A més, a causa de la complexitat que suposa l'aprenentatge i la utilització d'aquestes representacions vectorials, s'ha implementat un sistema que facilita les tasques d'avaluació, preprocessament, extracció de característiques i visualització de resultats; generalitzant els aspectes en comú a tots els problemes abordats.

Destaquem, també, els bons resultats obtinguts mitjançant l'ús d'aquestes representacions en el problema de detecció de temàtica, que constitueix el principal problema del projecte i se superen els millors resultats coneguts, fent ús del mateix corpus, que han sigut aconseguits per investigadors de la Universitat Politècnica de Madrid.

Paraules clau: descobriment de coneixement, representacions vectorials contínues, detecció de temàtica, llenguatge natural, model de llenguatge.

Índice general

Índice de figuras

Índice de tablas

Glosario

1-of-V: Caso particular de representación vectorial dispersa en el que cada elemento se representa mediante un vector con $V - 1$ elementos cero y un único elemento con valor uno, cuya posición identifica al elemento.

DAG: En matemáticas y computación, un grafo acíclico dirigido, es un grafo donde el conjunto de arcos está formado por pares ordenados y no existen caminos de longitud mayor que cero que permitan, partiendo de un vértice origen v, alcanzar dicho vértice inicial. Son las siglas de *Directed Acyclic Graph*.

EPPS: *European Parliament Plenary Sessions*, colección de intervenciones en el Parlamento Europeo ampliamente utilizada en sistemas de reconocimiento, traducción y síntesis del habla. Desarrollada e integrada en el proyecto *Technology and Corpora for Speech to Speech Translation* (TC-STAR).

LDA: Es un modelo generativo que, aplicado en NLP, presupone que cada documento es una mezcla de un pequeño número de categorías y la aparición de cada palabra en un documento se debe a una de las categorías que contiene el documento. Procede de *Latent Dirichlet Allocation*.

LSA: *Latent Semantic Analysis* es una técnica de procesamiento de lenguaje natural que analiza las relaciones entre un conjunto de documentos y los términos que contienen para producir una secuencia de conceptos representados en dichos documentos.

ML: Abreviatura de *Machine Learning* en inglés, es una disciplina de la computación dedicada al estudio del reconocimiento de patrones y a la teoría del aprendizaje computacional aplicado en inteligencia artificial.

Modelo de lenguaje: Mecanismo que permite definir la estructura de un lenguaje, restringiendo adecuadamente las secuencias de unidades lingüísticas más probables. Útiles en aplicaciones donde el lenguaje tratado emplea una sintaxis y/o semántica compleja.

NCE: Abreviatura de *Noise Contrastive Estimation*, es un modelo que pretende estimar una distribución de probabilidad desconocida P_d comparándola con una distribución conocida P_n, aproximando, para ello, la ratio $\frac{P_d}{P_n}$ mediante regresión logística no lineal.

NER: Tarea correspondiente al área de recuperación de información que identifica y clasifica elementos atómicos en texto sobre categorías predefinidas como nombres de personas, identificaciones, localizaciones, etc. Procede de *Name Entity Recognition*.

NLP: Abreviado del idioma inglés *Natural Language Processing*, es un campo de las ciencias de la computación que estudia la interacción entre las computadoras y el lenguaje humano.

POST: Proviene de *Part Of Speech Tagging*, es el proceso de etiquetar cada una de las palabras presentes en un texto con su categoría gramatical. La principal problemática del proceso recae en la variación de dicha categoría en función del contexto de una palabra dada.

Red neuronal: En aprendizaje automático son una familia de modelos matemáticos inspirados en las redes neuronales biológicas, se utilizan para aproximar funciones en base a un gran número de entradas.

TF-IDF: Siglas de *Term Frequency-Inverse Document Frequency*, es un método de pesado de palabras para determinar la relevancia de un término en un conjunto de documentos. Se obtiene mediante el producto de la frecuencia de término y la frecuencia inversa de documento.

VSM: Es un modelo algebraico para representar documentos como vectores de identificadores. Se emplea profusamente en recuperación de información, indexación y rankings de relevancia. Son las siglas de *Vector Space Model*.

Capítulo 1

Introducción

En este primer capítulo realizamos una introducción al descubrimiento automático de conocimiento, una enumeración de las razones que justifican la realización del proyecto, así como un planteamiento de los objetivos que se pretenden alcanzar, una descripción detallada de la estructura de la memoria en función de los capítulos desarrollados y una exposición de las asignaturas, cursadas a lo largo del grado, que han sido de ayuda durante el trabajo.

1.1 Motivación

La gran cantidad de información disponible y su variabilidad se han visto incrementadas en la última década, constituyendo dicha información una amplia fuente de conocimiento que podemos emplear para predecir o generalizar soluciones de problemas futuros. Es por ello que la extracción automática de conocimiento ha adquirido recientemente una importancia científica y económica inusual [1], siendo algunas de sus aplicaciones: gestión de información, marketing, finanzas y medicina.

Formalmente, una definición del tema tratado en el presente proyecto, según U. Fayyad, es: «proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y en última instancia comprensibles a partir de los datos». Entonces, de manera general, es posible afirmar que el proceso trata la explotación de datos en bruto, mediante algoritmos y procedimientos, cuyo fin es la obtención de información clave, relevante y útil, como patrones, correlaciones y reglas para cumplir determinados objetivos [2].

Este tipo de información clave es de vital importancia en el ámbito de las organizaciones, en la medida en que es posible ayudar a conocer y comprender el comportamiento de las mismas y de su entorno para favorecer la toma de decisiones [2]. Y no solo esto, también en el medio científico cobran importancia las técnicas de descubrimiento automático de conocimiento al estar fundamentadas en marcos de estudio relativamente asentados como la estadística, la inteligencia artificial, las bases de datos y la visualización gráfica.

Así pues, el descubrimiento automático de conocimiento implica un proceso iterativo e interactivo, involucrando la aplicación de técnicas de minería de datos para extraer o identificar conocimiento, teniendo como meta el procesamiento de grandes cantidades de datos *crudos*, intratables manualmente por humanos, para identificar patrones significativos y relevantes, al mismo tiempo que se presentan dichos patrones como conocimiento que permite satisfacer los objetivos y fines del usuario final.

Con todo ello, es posible vislumbrar cómo la tecnología promete procesar con facilidad amplias cantidades de datos y reconocer patrones subyacentes en estos, de forma que se soporte la toma de decisiones y se construya conocimiento de alto nivel de abstracción. Además, esta misma tecnología de minería de datos y descubrimiento automático de conocimiento, parece robusta y preparada para su aplicación debido a que se soporta sobre marcos teóricos bien estudiados, sin embargo, todavía se enfrentan grandes retos como falsas interpretaciones y relaciones temporales entre los datos [2].

Por último, es necesario mencionar que, en el presente proyecto, se tratan problemas relativos al descubrimiento automático de conocimiento aplicado sobre el lenguaje natural, como son la identificación de temática, detección del idioma, análisis de sentimiento y detección de *malware*, proponiendo y analizando soluciones que hacen uso de representaciones vectoriales continuas y discretas obtenidas mediante Word2Vec y DBpedia respectivamente.

1.2 Objetivos

Algunos de los métodos más sencillos para plantear sistemas de descubrimiento automático de conocimiento y de procesamiento del lenguaje natural (NLP) se basan en representaciones discretas construidas mediante modelos de bolsa de palabras o de n-gramas, sin embargo, estas no son las únicas que conocemos y es posible emplear otros tipos, como representaciones vectoriales continuas, que captan con una mayor precisión el contexto, y en cierto modo la semántica, de cada término.

Actualmente, algunos de los modelos más conocidos para producir este tipo de representaciones continuas son: bolsa de palabras continua (CBOW) y *skip-gram* (SG), basados en redes neuronales y creados por un grupo de investigadores de Google liderado por T. Mikolov, permiten generar y emplear dichas representaciones de palabras y frases en tareas típicas de descubrimiento automático de conocimiento como detección de temática o identificación del idioma.

Además, podemos verificar las hipótesis establecidas en [3] y [4] sobre los parámetros de aprendizaje de los modelos CBOW y SG empleando como banco de pruebas diferentes corpus, así como mejorar los resultados obtenidos de experimentaciones realizadas por otros investigadores sobre problemas, similares a los planteados en el proyecto, pero que no emplean representaciones vectoriales de espacio continuo (*word* embeddings) como [5].

Por otro lado, el uso de este tipo de representaciones no se limita únicamente a NLP y es por ello que proponemos, como objetivo adicional, demostrar las bondades de emplear representaciones vectoriales de espacio continuo de palabras en aplicaciones que han sido históricamente planteadas mediante enfoques diferentes v.g. detección de *malware* mediante sistemas expertos.

A pesar de lo mencionado, las representaciones vectoriales de espacio continuo pueden no ser adecuadas bajo determinadas circunstancias, por lo que otro de nuestros objetivos es hacer uso de representaciones discretas no convencionales, como la basada en propiedades extraídas de la ontología de DBpedia, dando así soluciones alternativas a los problemas tratados en el proyecto.

Como se ha enunciado en párrafos anteriores, se proponen una serie de objetivos parcialmente disjuntos complicados de alcanzar si no se lleva a cabo el desarrollo de un sistema capaz de simplificar las diversas tareas, así como de integrar y generalizar los aspectos comunes a todos los problemas esbozados. Debido a esto, sugerimos implementar dicho sistema que nos permita, al mismo tiempo que facilite la tarea, realizar una gran cantidad de funciones como extracción de características, evaluación, preprocesamiento y visualización sobre los corpus tratados.

Finalmente, cabe destacar, como objetivos personales y adicionales, focalizar los conocimientos adquiridos sobre *machine learning* (ML) en tareas específicas del descubrimiento automático de conocimiento, además de profundizar en la utilización de diversos tipos de redes neuronales para el aprendizaje de modelos de lenguaje y de representaciones vectoriales continuas de palabras y frases.

1.3 Estructura de la memoria

La memoria realizada sobre el trabajo de fin de grado está dividida en siete capítulos, así, en el presente apartado, realizamos una breve descripción de los aspectos tratados en cada uno de ellos:

- **Capítulo 1, Introducción**: en el primer capítulo se expone una introducción al descubrimiento automático de conocimiento justificando la motivación para la realización del proyecto, al mismo tiempo que se definen los objetivos que se desean abarcar, se enuncia la estructura de la memoria y se especifican las asignaturas, cursadas a lo largo del grado, que han sido de ayuda para desarrollar el trabajo.

- **Capítulo 2, Representación continua de palabras y frases**: se trata la obtención de vectores de representación continua de palabras mediante la utilización de diversas arquitecturas de redes neuronales, así como también se analizan varias optimizaciones que reducen la complejidad computacional del aprendizaje en estas redes. También se muestra cómo hacemos uso, de forma fácil e intuitiva, de dichos modelos mediante la herramienta Word2Vec y cómo es posible combinar la representación de palabras para obtener representaciones de unidades más complejas como frases o documentos.

- **Capítulo 3, DBpedia**: este capítulo se focaliza en la realización de una introducción al proyecto DBpedia, llevando a cabo una descripción detallada de cómo se ha extraído y caracterizado la información estructurada, proporcionada por dicho proyecto a través de su ontología, mediante un lenguaje de representación de conocimiento y un lenguaje de consulta de grafos, generando así representaciones vectoriales de palabras y frases que nos permiten proponer soluciones alternativas a algunos de los problemas tratados.

- **Capítulo 4, Algoritmos de aprendizaje**: este capítulo introduce y justifica la necesidad de utilización de algoritmos de aprendizaje para estimar clasificadores que permitan discriminar entre un conjunto de clases de un determinado problema. Además, se realiza una introducción a los marcos y límites del aprendizaje computacional, se describen y se formalizan con detalle todos los clasificadores empleados durante la realización del proyecto, se analizan las métricas que posibilitan la evaluación del comportamiento de dichos clasificadores y se expone el protocolo de experimentación seguido.

- **Capítulo 5, Detección de temática**: en el quinto capítulo proponemos la utilización, evaluación y comparación de representaciones vectoriales de palabras y frases para dar solución al problema de identificación de la temática en diferentes corpus mediante diversas experimentaciones. Concretamente, hacemos uso de representaciones de palabras obtenidas mediante modelos Word2Vec, entrenados con el propio corpus del caso y con artículos de Wikipedia en el idioma correspondiente, y mediante la ontología de DBpedia.

- **Capítulo 6, Otras aplicaciones de *word embeddings***: en este capítulo se introducen diversos problemas relacionados con el procesamiento del lenguaje natural como identificación del idioma, análisis de sentimiento y detección de *malware*, al mismo tiempo que se analizan, para dichos problemas, soluciones basadas en representaciones continuas de palabras y frases y se estudian los resultados obtenidos.

- **Capítulo 7, Conclusiones y trabajo futuro**: el último capítulo abarca las conclusiones, obtenidas tras haber realizado completamente el trabajo, así como 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.

1.4 Asignaturas relacionadas

Para el desarrollo del proyecto y la elaboración de la memoria se ha hecho uso de gran cantidad de material didáctico provisto por el profesorado de diversas asignaturas cursadas durante el grado en Ingeniería Informática. Además, a pesar de que en algunos casos dicho contenido no se ha utilizado explícitamente en el proyecto, ha proporcionado una base sólida que posibilita la adquisición de conocimiento en materias relativas.

En primer lugar, relacionadas con el área de aprendizaje automático y reconocimiento de formas, se encuentran las asignaturas «Sistemas Inteligentes», «Percepción» y «Aprendizaje Automático» que han facilitado la explicación de los diversos enfoques y algoritmos de aprendizaje empleados durante la realización del proyecto. Además, han permitido la comprensión de otros conceptos relativos como las arquitecturas de redes neuronales utilizadas para aprender modelos de lenguaje definidas en [3].

En segundo lugar, es necesario destacar las asignaturas «Agentes Inteligentes» y «Bases de Datos y Sistemas de Información». La primera de ellas ha sido útil para establecer una definición del concepto de ontología y de lenguajes de representación de conocimiento, mientras que la segunda, ha facilitado la familiarización con el lenguaje de consulta empleado en las experimentaciones con DBpedia, debido a que existen similitudes entre el lenguaje impartido en la asignatura y el utilizado en el proyecto.

Por último, la asignatura «Sistemas de Almacenamiento y Recuperación de Información» ha permitido una habituación a los modelos de espacio vectorial, tanto continuos como discretos, ampliamente utilizados en las experimentaciones llevadas a cabo en el proyecto, así como ha proporcionado los conceptos necesarios para evaluar sistemas de clasificación.

«Tenemos un plan estratégico. Se llama hacer las cosas bien.»

-Herb Kelleher

Capítulo 2

Representación continua de palabras y frases

Las representaciones de palabras y frases mediante vectores de espacio continuo tienen detrás una larga historia, habiéndose propuesto muchos tipos de modelos diferentes para estimar estos vectores incluyendo los ampliamente conocidos *Latent Semantic Analysis* (LSA) y *Latent Dirichlet Allocation* (LDA). En este capítulo, nos centramos en representaciones vectoriales continuas aprendidas mediante redes neuronales, las cuales poseen una mayor capacidad que LDA y LSA para preservar regularidades entre palabras [3].

2.1 Modelo de lenguaje basado en redes neuronales

Las redes neuronales hacia delante (*feedforward*), en las que únicamente pueden existir conexiones entre neuronas de una capa i y neuronas de siguientes capas $j: j > i$, tal como se muestra en la Figura 1, han sido propuestas en diversos artículos para obtener vectores de espacio continuo que representen palabras. Una arquitectura típica para obtener modelos de lenguaje basados en redes neuronales (NNLM), descrita en [3], consta de cuatro capas: capa de entrada, capa de proyección, capa oculta y capa de salida.

En la capa de entrada, dada una palabra x_i, todas las palabras $x_j : j < i$ son codificadas mediante la codificación *1-of-V* donde V es el tamaño del vocabulario. Posteriormente, se proyecta la capa de entrada sobre una capa de proyección P empleando una matriz de proyección compartida, cuya dimensionalidad es $N \cdot D$, siendo D la dimensión de los vectores de representación de espacio continuo de palabras que coincide con el número de neuronas en la capa de proyección.

Como solo hay N palabras o entradas activas en un mismo momento, siendo N el número de palabras en la frase a analizar, el cómputo realizado en la capa de proyección es relativamente barato, sin embargo, la arquitectura complica el cálculo de los pesos de las conexiones entre la capa de proyección lineal y la capa oculta no lineal, ya que los vectores de la capa de proyección son densos en comparación con los vectores dispersos de la codificación *1-of-V* de la capa de entrada.

Ya en la capa oculta, cuya dimensionalidad es H, se calcula la distribución de probabilidad sobre todas las palabras del vocabulario, comunicando finalmente estos resultados a una capa de salida con V neuronas. Con ello, la complejidad computacional para cada muestra de entrenamiento (conjunto de palabras representadas con vectores dispersos obtenidos mediante la codificación *1-of-V*) es: $Q = N \cdot D + N \cdot D \cdot H + H \cdot V$.

En la expresión anterior, el término dominante es $H \cdot V$ y su cálculo suele ser muy costoso, para ello, en la práctica, se propone evitar o reducir el cómputo necesario mediante la utilización de una versión jerárquica de *softmax* o de modelos que no normalicen los vectores durante el entrenamiento. Además, es posible representar el vocabulario mediante árboles binarios, reduciendo así el número de neuronas requeridas en la capa de salida a $\log_2 V$ unidades, lo que provoca que la complejidad radique en el término $N \cdot D \cdot H$.

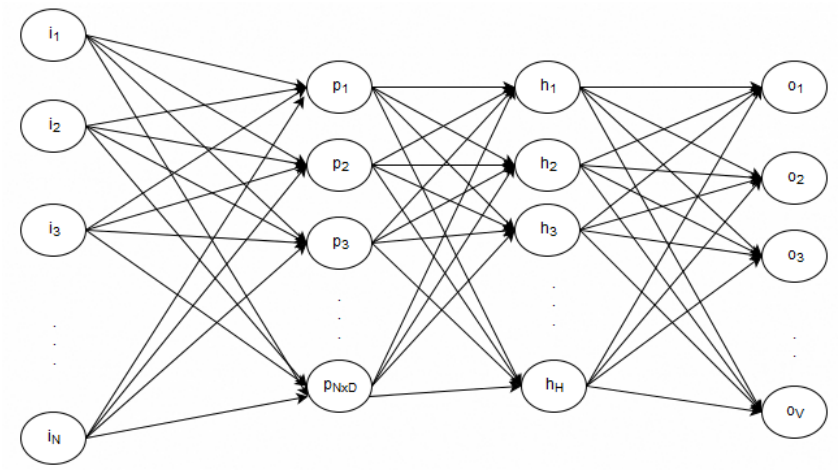

Fig. 1. Arquitectura típica de red neuronal hacia delante

Finalmente comentar que, este tipo de modelo de lenguaje planteado en 2003, mejoró entre un 10 % y un 20 % la perplejidad de las técnicas del estado del arte en ese periodo, así como ofreció una alternativa compacta a las tablas de probabilidad condicional entonces empleadas para representar modelos estadísticos de lenguaje, sin embargo, el obstáculo principal fue la complejidad computacional que, con el paso de los años, ha podido enfrentarse mediante arquitecturas más eficientes.

2.2 Modelo de lenguaje basado en redes neuronales recurrentes

Los modelos de lenguaje basados en redes neuronales recurrentes (RNNLM) se han propuesto para eliminar ciertas limitaciones de los modelos de lenguaje basados en redes neuronales hacia delante, tratados en el apartado anterior, como la necesidad de especificar el tamaño del contexto o ventana a considerar para cada palabra. Además, teóricamente, las redes neuronales recurrentes pueden representar, de forma eficiente, patrones más complejos que otros tipos de redes neuronales más simples [3].

Con respecto a la arquitectura, el modelo basado en este tipo de redes, no posee una capa de proyección; solo capa de entrada, una capa oculta y capa de salida. Otro de los aspectos esenciales en estas redes es la matriz de conexiones recurrentes que conecta neuronas de la capa oculta con otras neuronas de la misma capa mediante conexiones con retardo.

Este tipo de conexiones retardadas, permite al modelo recurrente poseer cierta memoria a corto plazo que emplea para almacenar información representada en el pasado en la capa oculta (estado anterior de la capa) y así poder actualizar el estado de dicha capa en función del estado anterior y de la entrada actual. Es posible observar estas conexiones en una arquitectura típica de red neuronal recurrente expuesta en la Figura 2.

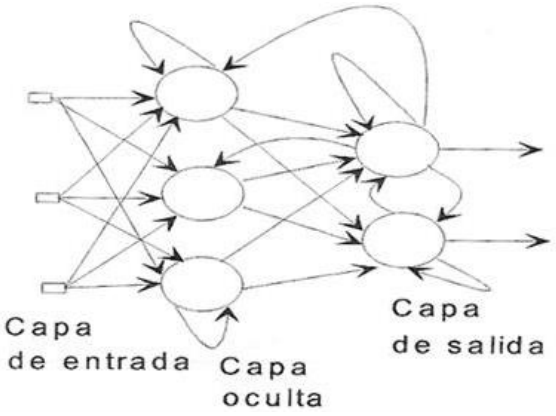

Fig. 2. Arquitectura típica de red neuronal recurrente

Por otro lado, atendiendo a la arquitectura, la complejidad para cada muestra de entrenamiento del modelo basado en redes neuronales recurrentes viene dada por la expresión: $Q = H \cdot H + H \cdot V$, donde la dimensión de la representación continua de cada palabra D, coincide con el número de neuronas en la capa oculta H.

Una vez conocida la complejidad computacional del método, si nos fijamos en la expresión del coste para cada muestra en este modelo y observamos las similitudes con la expresión del coste de la arquitectura anterior, del mismo modo que en NNLM, el término dominante es $H \cdot V$ que puede ser eficientemente reducido a $H \cdot \log_2 V$ usando *hierarchical softmax* y/o una representación en árbol binario del vocabulario V.

Finalmente, es posible observar que la mayor parte del coste de cómputo, tanto en NNLM como en RNNLM, recae en los cálculos asociados a los valores de salida de las neuronas de la capa oculta, por ello, en los siguientes apartados, se estudian arquitecturas de redes neuronales, propuestas en [3] y [4], que pretenden minimizar la complejidad, permitiéndonos así entrenar el sistema con una gran cantidad de datos de forma eficiente.

2.3 Modelo continuo de bolsa de palabras

La primera arquitectura propuesta por T. Mikolov en [3] para reducir la complejidad computacional de los métodos expuestos en los dos primeros apartados de este mismo capítulo, es el modelo continuo de bolsa de palabras (CBOW), similar a NNLM *feedforward*, tal como se puede observar en la Figura 3. En este caso, se elimina la capa oculta no lineal de las redes neuronales utilizadas en NNLM y la capa de proyección es compartida entre todas las palabras, no solo la matriz de proyección, provocando que las palabras sean proyectadas en la misma posición.

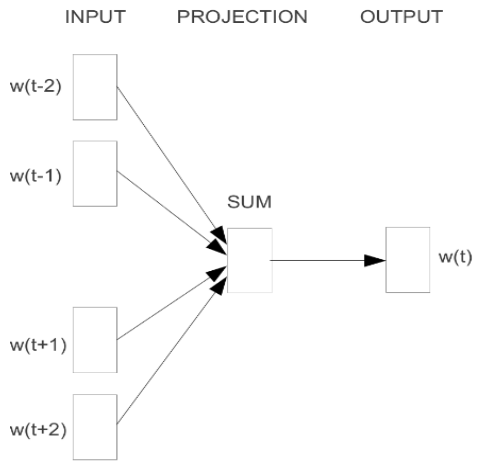

Fig. 3. Arquitectura CBOW [3]

Formalmente, considerando que w_t es la palabra analizada de la frase, la entrada de la red neuronal es $w_{t-k} \dots, w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}, \dots, w_{t+k}$, siendo k el tamaño de la ventana y la salida w_t, la tarea de la red consiste en predecir la palabra w_t de una declaración conociendo su contexto $w_{t-k} \dots, w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}, \dots, w_{t+k}$.

Así, en lo referente a la complejidad, sea N la dimensión de la capa de entrada, D el número de neuronas en la capa de proyección y V el tamaño del vocabulario, el coste temporal Q de entrenar el modelo CBOW está determinado por la expresión $Q = N \cdot D + D \cdot \log_2 V$.

Por último, comentar que este modelo de lenguaje suaviza mucho las distribuciones de probabilidad al promediar todos los contextos de las palabras [9], es por ello que, ante corpus de entrenamiento con grandes cantidades de muestras, los vectores de representación obtenidos no son de tanta calidad como los obtenidos mediante el modelo que se expondrá en el próximo apartado, en resumen, el comportamiento de CBOW es mejor con un pequeño número de muestras de entrenamiento donde este efecto regulador es beneficioso.

2.4 Modelo *skip-gram*

La segunda arquitectura tratada en [3] y en [4] es el modelo *skip-gram* (SG), que pretende predecir el contexto de una palabra w_t dada, de forma inversa a CBOW cuyo objetivo es adivinar w_t conociendo su contexto delimitado en ambos extremos por una ventana de tamaño k. En la Figura 4 se puede visualizar gráficamente la arquitectura mencionada.

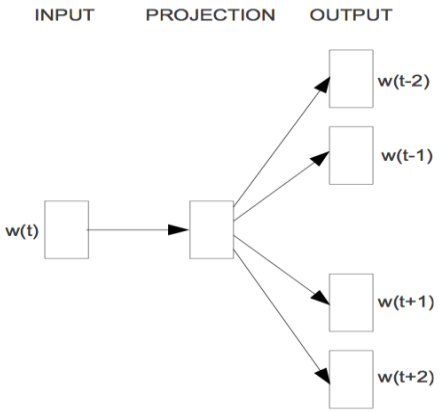

Fig. 4. Arquitectura skip-gram [3]

El objetivo de entrenamiento del modelo SG es encontrar representaciones vectoriales continuas que permitan predecir las palabras circundantes a un término w_t en una frase o documento. Más formalmente, dada una secuencia de palabras de entrenamiento $w_1, w_2, w_3, \dots, w_T$, el objetivo es maximizar el logaritmo de la probabilidad: $\frac{1}{T}\sum_{t=1}^{T}\sum_{-k\leq j\leq k, j\neq 0}\log p(w_{t+j}|w_t)$. Donde T es el tamaño del corpus de entrenamiento, 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 términos, permitiendo esto generar vectores de representación de una mayor calidad, con el inconveniente de un incremento en la complejidad computacional. Esto provoca también, un buen comportamiento del modelo ante corpus con un número reducido 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 conjunto de términos $w_{t+j} - k \leq j \leq k,\ j \neq 0$ dada una palabra w_t: $p(w_{t+j}|w_t)$, en el modelo SG se define dicha probabilidad mediante la función *softmax* de la siguiente manera, $p(w_O|w_I) = \frac{e^{({v'_{wO}}^T v_{wI})}}{\sum_{w=1}^{V} e^{({v'_w}^T v_{wI})}}$. Donde v_w y v'_w son los vectores de representación de entrada y salida respectivamente de w y V es el número de palabras en el vocabulario.

Sin embargo, es conveniente destacar que computar el gradiente de $p(w_O|w_I)$, $\nabla \log p(w_O|w_I)$ para entrenar los modelos, no es viable en la práctica debido a que el coste computacional es proporcional a V, cuyo valor debe ser alto, oscilando normalmente entre 10^5 y 10^7 términos. Con ello, de forma detallada, la complejidad del entrenamiento de la presente arquitectura es proporcional a $Q = C \cdot (D + D \cdot \log_2 V)$. Donde C es la máxima distancia entre palabras del corpus de entrenamiento y D la dimensión de los vectores representación, se observa que el coste temporal es mayor que en CBOW.

Para concluir, mencionar que este modelo ha sido empleado profusamente en el presente proyecto debido a que el autor de los artículos [3] y [4] recomienda su utilización, junto con formas alternativas de aproximar la probabilidad condicional $p(w_{t+j}|w_t)$ que se comentarán en el próximo apartado, y a que el modelo SG capta, mejor que todos los modelos expuestos hasta ahora, las propiedades y relaciones semánticas entre términos provocando que palabras con significado semejante aparezcan próximas en el espacio vectorial.

2.5 *Softmax* jerárquico y *negative sampling*

Como se ha visto en el apartado anterior, calcular el gradiente de la función softmax completa $\nabla \log p(w_O|w_I)$ para entrenar los modelos, no es viable en la práctica. Por ello, en [4] se proponen aproximaciones de la función *softmax* computacionalmente eficientes como son: *softmax* jerárquico (HS) y *negative sampling* (NS).

En primer lugar, tratamos la versión jerárquica de *softmax*, descrita en [4] permite, en lugar de evaluar V neuronas en la capa de salida de la red neuronal, calcular únicamente la salida de $\log_2 V$ nodos para aproximar la distribución de probabilidad condicional. Para poder llevar a cabo esta reducción en el número de neuronas, se representa la capa de salida mediante un árbol binario formado por V hojas, donde se conoce la probabilidad de transición de un nodo q a sus nodos hijos $\{q_L, q_R\}$, definiendo esto un recorrido aleatorio en el árbol que asigna probabilidades a las palabras. Un ejemplo se puede observar en la Figura 5, según la cual: $p(cat|context) = p(\delta(1,2)|context) \cdot p(\delta(2,5)|context) \cdot p(\delta(5,cat)|context)$.

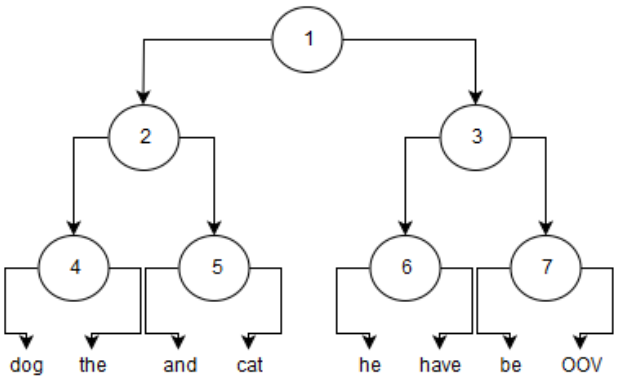

Fig. 5. Capa de salida jerárquica representada mediante un árbol binario

Más formalmente, cada palabra w puede ser alcanzada mediante un camino que recorre el árbol partiendo desde la raíz. Sea $n(w,j)$ el nodo j-ésimo en el camino desde la raíz hasta w, $L(w)$ la longitud de este camino, $ch(n)$ un nodo hijo arbitrario de n y $[[x]] = \begin{cases} 1 \; si \; x = True \\ 0 \; si \; x = False \end{cases}$, la formulación de *softmax* jerárquico define $p(w_O|w_I)$ como sigue: $p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left(\left[\left[n(w,j+1) = ch\big(n(w,j)\big)\right]\right] \cdot {v'_{n(w,j)}}^T \cdot v_{wI}\right)$. Donde σ es la función *sigmoide* definida como $\sigma(x) = \frac{1}{1+e^{-x}}$. Con ello, se consigue que el coste computacional de calcular $\log p(w_O|w_I)$ pase, de ser proporcional a V, a ser proporcional a $L(w)$ que en promedio no es mayor que $\log_2 V$.

Por otro lado, la segunda alternativa, planteada en [4], a *softmax* completo, es una simplificación de *Noise Contrastive Estimation* (NCE). NCE postula que un buen modelo debe ser capaz de distinguir información útil de ruido mediante métodos como regresión logística y pretende maximizar el logaritmo de la probabilidad obtenida mediante la función *softmax*. Por ello, debido a que el objetivo de NCE es maximizar la salida de la función *softmax,* pero SG únicamente persigue la obtención de representaciones vectoriales continuas de palabras, es posible simplificar el modelo NCE siempre y cuando se mantengan ciertos niveles de calidad en las representaciones.

Así, esta simplificación conduce al método NS para la aproximación de la distribución de probabilidad condicional, cuyo objetivo se define mediante la expresión $\log \sigma({v'^T_{wO}} v_{wI}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)}[\log \sigma(-{v'_{wI}}^T v_{wI})]$. Dicha formulación es usada en la función objetivo del modelo SG para reemplazar cualquier ocurrencia de $\log P(w_O|w_I)$, por lo que, ahora, la tarea consiste en distinguir la palabra objetivo w_O de las palabras extraídas de una distribución de ruido $P_n(w)$ usando regresión logística, donde hay k muestras negativas para cada muestra del conjunto de entrenamiento.

Con ello, la principal diferencia entre NCE y NS recae en que NCE requiere muestras de entrenamiento y probabilidades de la distribución de ruido $P_n(w)$, en cambio, NS únicamente necesita muestras de aprendizaje. Además, mientras que NCE pretende maximizar el logaritmo de la probabilidad obtenida mediante *softmax*, NS busca obtener vectores de espacio continuo de representación de palabras, por lo que el objetivo de NCE no es relevante para NS.

En resumen, se han estudiado dos métodos de optimización que permiten reducir el coste computacional que conlleva la obtención de vectores de salida mediante los modelos expuestos en los dos apartados anteriores, debido al cómputo de $\nabla \log p(w_O|w_I)$ que hace inviables dichos modelos al tener que iterar sobre todos los términos del vocabulario (una explicación y justificación detallada de las ecuaciones para calcular $\nabla \log p(w_O|w_I)$ y su coste asociado se realiza en [6]).

2.6 Word2Vec

Word2Vec es una herramienta que provee una implementación eficiente de los modelos CBOW y SG, permitiendo a su vez el uso de las optimizaciones mencionadas en el apartado anterior, *softmax* jerárquico y *negative sampling*, para la obtención de representaciones vectoriales continuas de palabras. Es por ello que Word2Vec no es un algoritmo único [10], sino una *suite* que facilita la generación de los modelos continuos expuestos en los apartados tres y cuatro de este mismo capítulo.

Uno de los aspectos más importantes a considerar es que Word2Vec no es aprendizaje profundo [10], ya que CBOW y SG son modelos de redes neuronales con un reducido número de capas ocultas. Esto no reduce la eficacia del método, de hecho, la idea principal comentada por los creadores de Word2Vec en [3], es demostrar la posibilidad de obtención de mejores representaciones continuas si se reduce la complejidad computacional en aras de la eficiencia i.e. evitar la utilización de redes neuronales profundas, con muchas capas ocultas, para permitir al modelo aprender de corpus muy grandes de forma eficiente.

Respecto a su utilización, la herramienta toma como entrada un corpus textual y produce vectores de representación para las palabras del vocabulario como salida. Para ello, primero se construye un vocabulario a partir de textos de entrenamiento y finalmente se aprenden los vectores de representación que pueden ser empleados como características en muchas tareas de NLP y *machine learning*. En nuestro caso, hacemos uso de dichos vectores para ofrecer soluciones a los problemas tratados en el proyecto como detección de temática, identificación de idioma etc.

Además, la herramienta nos permite realizar operaciones con los vectores obtenidos para investigar y determinar la calidad de estos mediante la identificación de regularidades sintácticas y semánticas [11]. Así, si el modelo está bien entrenado, *vector('Paris') - vector('France') + vector('Italy')* resultaría en un vector muy cercano a *vector('Rome')*, que puede ser visto como París es a Francia lo que Roma es a Italia. Sin embargo, para observar regularidades en el espacio vectorial donde se representan las palabras, es necesario entrenar los modelos con grandes corpus empleando los parámetros adecuados.

Entre estos parámetros, condicionantes de la eficacia del método, podemos encontrar:

- **Arquitectura:** SG o CBOW. SG es más lento, pero capta mejor las regularidades semánticas y representa con mayor calidad las palabras poco frecuentes con respecto a CBOW [10].
- **Algoritmo de entrenamiento:** *softmax* jerárquico o *negative sampling*. *Softmax* jerárquico se comporta mejor con palabras poco frecuentes mientras que *negative sampling* es preferible con palabas frecuentes y vectores con pocas dimensiones [7].

- **Submuestreo de palabras frecuentes:** En corpus de gran tamaño las palabras más frecuentes suelen ocurrir cientos de millones de veces aportando mucha menos información que palabras poco frecuentes. La idea del método consiste en que los vectores de representación de palabras frecuentes no cambian significativamente después de entrenar con varios millones de ejemplos, mejorando así la precisión y acelerando el entrenamiento cuando se utilizan grandes conjuntos de datos [3].
- **Dimensionalidad de los vectores de palabras**: normalmente, cuanto mayor sea la dimensión de los vectores de representación de palabras mejores resultados se obtienen [7].
- **Tamaño de ventana**: determina la cantidad de palabras circundantes a analizar dada una palabra w_i de una frase o documento de la que se desea obtener su vector de representación continuo. Ej., sea $k = 2$ el tamaño de la ventana y dada la frase «*coaches vulnerable when team fails*», el contexto de la palabra $w_3 =$ «*when*», es $\{w_{3-2}, w_{3-1}, w_{3+1}, w_{3+2}\} = \{$*coaches, vulnerable, team, fails*$\}$. Un estudio sobre el tamaño de ventana se realiza en [8], donde menciona que un valor reducido para el tamaño del contexto permite capturar más información sobre la palabra analizada (información de la palabra), mientras que un valor elevado proporciona un mejor comportamiento al determinar qué otras palabras están relacionadas con el término analizado (información del dominio).

Debido a la utilización, en el presente proyecto, de la versión en Python de Word2Vec implementada e integrada en la librería Gensim por Radim Řehůřek [12], podemos modificar parámetros adicionales del modelo como la tasa de aprendizaje, mínimo número de ocurrencias de una palabra para llegar a considerarla en el entrenamiento, número de palabras ruidosas extraídas de la distribución de ruido $P_n(w)$ al emplear *negative sampling,* cantidad de hilos a utilizar para entrenar el modelo (mayor eficiencia en máquinas con más de un núcleo), entre otros.

Finalmente, para resolver los problemas que se exponen en el proyecto, generamos los modelos Word2Vec de dos formas diferentes: mediante las frases o documentos de entrenamiento del propio corpus del problema o a partir de los artículos de Wikipedia en el idioma correspondiente al empleado en la tarea, aunque es posible recurrir a otras fuentes de información que contengan conjuntos de ejemplos de tamaño considerable, del orden de billones de palabras, como se puede ver en [11].

2.7 Representación de frases mediante combinación de palabras

Hasta ahora, únicamente se han tratado formas de obtener representaciones vectoriales continuas de palabras mediante modelos basados en redes neuronales, sin embargo, no se ha comentado ningún método para procesar secuencias de unidades lingüísticas como son las frases.

Para ello, no existe ningún método implementado en la versión de Word2Vec integrada en Gensim y debemos recurrir a técnicas de obtención de representaciones de frases mediante combinación de los vectores de palabras que conforman dichas sentencias. Esta área, que trata la combinación de vectores de palabras para la generación de representaciones vectoriales de frases, párrafos y documentos, es todavía un tema de investigación activo y abierto, por lo que no proponemos un único método, sino que realizamos una comparación de varios que pueden comportarse mejor o peor en función del problema analizado.

Así, existen diversos enfoques planteados en varias publicaciones como [13] y [14], que permiten llevar a cabo una composición de vectores de palabras para obtener vectores de calidad que son capaces de capturar gran parte de la semántica presente en secuencias de palabras con significado. Sin embargo, las alternativas que se han implementado, aunque no se han usado todas en los experimentos debido al incremento de tiempo que conlleva su completa utilización, en el sistema desarrollado para el proyecto son:

- **Suma:** el enfoque consiste, únicamente, en sumar los vectores de representación de todas las palabras w que conforman una frase s para obtener su representación vectorial continua, $vector(s) = \sum_{w \in s} vector(w)$. Adecuada ante casos en los que es posible afirmar que la semántica de una frase se puede expresar como la suma de las semánticas de los términos que la forman, si no es así, la suma de las palabras puede introducir gran cantidad de ruido que reduce la calidad de la representación de frases.

- **Centroide:** idéntico a la representación anterior, pero normalizando con respecto al número de palabras que se encuentran en la frase y han aparecido durante el entrenamiento del modelo empleado en Word2Vec, $vector(s) = \frac{1}{|\{w_1, w_2, \ldots, w_n\} : w_i \in s|} \sum_{w \in s} vector(w)$. De esta forma, se evita que el valor de los elementos de $vector(s)$ se incremente cuanto mayor sea el número de palabas en s.

- **Palabra próxima al centroide:** en este caso, se considera como representación de la frase el vector de representación de la palabra, presente en la sentencia, que más próxima se encuentre con respecto al centroide comentado en el punto anterior, $vector(s) = vector(w_i) : w_i \in s\ ,\ \nexists w_j \in s\ , dist(w_j, centroide) < dist(w_i, centroide), i \neq j$. 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 Word2Vec+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 $vector(s) = \sum_{i=1}^{n} k_i \cdot vector(w_i)$, considerando los vectores de palabras y un término que permita ponderarlas de una manera determinada. Entre los tipos de ponderaciones que es posible realizar, podemos destacar la aleatoria, donde cada palabra es multiplicada por un valor aleatorio o la ponderación basada en información de *Part of speech tagging* (POST), en la que no se ponderan de igual forma las categorías gramaticales. En el presente proyecto, se implementa una suma ponderada que hace uso del pesado TF-IDF de cada palabra para obtener vectores de representación continuos de frases, $vector(s) = \sum_{w \in s} vector(w) \cdot tfidf(w)$.

- **Producto**: el planteamiento es idéntico al de la suma de palabras, pero empleando la función producto en lugar de la suma, $vector(s) = \prod_{w \in s} vector(w)$. El principal inconveniente de esta representación es que los elementos v_i de los vectores $\boldsymbol{v}$ obtenidos mediante los modelos implementados en Word2Vec son reales pequeños cuyo producto da lugar a reales todavía más reducidos, con lo que se puede alcanzar una pérdida de precisión considerable.

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

No todos los métodos expuestos en la lista anterior han sido empleados en las pruebas sobre los problemas tratados, aunque sí que han sido implementados e integrados en el sistema desarrollado para su posterior uso en futuras experimentaciones. Además, en algunos de los problemas, se han empleado representaciones alternativas a Word2Vec como vectores obtenidos a partir de la ontología de DBpedia.

Por último, cabe recalcar que, con las representaciones empleadas, que no estén normalizadas por definición, se ha realizado una posterior normalización de los vectores mediante la norma L_2, $x' = \frac{x}{||x||_2}$, consiguiendo que $\|x'\|_2 = \sqrt{x'^T x'} = 1$, para evitar así que la longitud de las frases influya en el cálculo de sus representaciones. Además, la utilización de esta norma provoca que la representación suma y centroide resulten en los mismos vectores debido a la propiedad $\|v\|_2 = \left\|\frac{1}{|v|} v\right\|_2$, es por eso que únicamente se emplea en la otra representación utilizada, basada en la palabra más próxima al centroide.

«Debes hacer cosas que realmente sean importantes,
pero también debes divertirte, porque si no, no tendrás éxito.»

-Larry Page

Capítulo 3

DBpedia

Puesto que empleamos información semántica del proyecto DBpedia para, como ya se enunció en el apartado tres del capítulo uno, proponer soluciones alternativas a algunos de los problemas del proyecto sin tener que recurrir a representaciones continuas o a representaciones discretas tradicionales, este capítulo se centra en realizar una introducción a DBpedia y en llevar a cabo una descripción detallada de cómo se ha extraído y caracterizado la información estructurada obtenida de dicho proyecto.

3.1 Proyecto DBpedia

La recolección de toda la información estructurada actual y del conocimiento necesario para responder, de forma automática, consultas semánticamente ricas, es uno de los retos principales de la computación. Estos objetivos se han perseguido durante más de treinta años, y todavía se persiguen en la actualidad, mediante investigaciones sobre integración de la información y actualmente a través de tecnologías relacionadas con la web semántica, consiguiendo buenos resultados en dominios cerrados y especializados donde una ontología específica y robusta ante cambios en los datos puede ser desarrollada con facilidad, ya que estos cambios están controlados [15].

En cambio, en dominios amplios y generales, los planteamientos donde una ontología particular es diseñada antes de la introducción de la información no se comportan correctamente, debido al gran alcance de dichos dominios y a la constante evolución de la información y sus formatos, un ejemplo claro de este tipo de dominios es la web. Los movimientos recientes para la construcción de una web semántica se basan en enfoques colaborativos que requieren nuevos métodos de representación de información estructurada y una correcta gestión capaz de manejar inconsistencias, ambigüedades, incertidumbre y conocimiento implícito de una manera uniforme.

El principal proyecto que intenta lidiar con la problemática planteada es DBpedia. Iniciado en 2006, el proyecto DBpedia es una comunidad que centra sus esfuerzos en extraer información estructurada de Wikipedia, cuyas capacidades de búsqueda están limitadas a la identificación de palabras clave, empleando estándares como *Resource Description Framework* (RDF) y *SPARQL Protocol And RDF Query Language* (SPARQL), haciéndola accesible a través de la web y permitiendo consultas sofisticadas sobre las bases de datos que contienen el conocimiento estructurado derivado de Wikipedia [16], abriendo de esta manera la posibilidad de tratamiento de información mediante procesos de inteligencia artificial.

Por otro lado, enfatizar que, además de extraer información estructurada y procesar consultas sobre el conocimiento obtenido, se plantean una serie de contribuciones en los artículos del proyecto estudiado [15], [16] y [17] a destacar:

- Desarrollar un sistema de extracción automático que transforme información de Wikipedia en tripletas RDF, permitiendo así la manipulación de dichas tripletas para la extracción de información, *clustering* o procesado de consultas.
- Proveer todo el contenido de Wikipedia como una base de datos de tripletas RDF que puede ser usada en aplicaciones basadas en la web semántica.
- Interconectar DBpedia con otras bases de datos abiertas como FOAF, que describe personas y relaciones o DrugBank que contiene recursos detallados de bioinformática y bioquímica.
- Desarrollar un conjunto de interfaces de usuario y módulos de acceso que permitan acceder vía web a la base de datos y enlazarla desde otros servicios web.

Cerramos el presente apartado mencionando que es posible importar las bases de datos de DBpedia en aplicaciones de terceros o acceder *online* a ellas mediante el conjunto de interfaces propuestas por la comunidad que hacen uso de *Virtuoso* y MySQL. Además, en la Figura 6 podemos observar el proceso de extracción de información y cómo la información extraída es publicada en la web para poder ser accedida mediante los métodos expuestos.

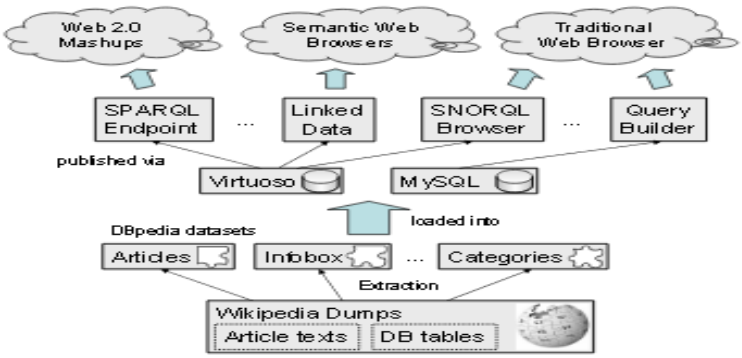

Fig. 6. Proceso de extracción y publicación de recursos [15]

3.2 Ontología

Una ontología es un sistema de representación del conocimiento resultado de aplicar, a un dominio de conocimiento, un método para obtener una representación formal de los conceptos que caracterizan el dominio y de las relaciones que existen entre dichos conceptos, convirtiendo así la información en conocimiento estructurado [18]. La construcción de las ontologías se realiza en relación a un contexto de utilización, es decir, la ontología especifica una conceptualización o una forma de ver el mundo, por lo que, cada ontología incorpora un punto de vista que, además, puede ser procesado por un computador.

Según [18] una ontología consta de:

- **Conceptos**: ideas básicas que se quiere formalizar y asociar semántica (clases de objetos, métodos etc.). Cada uno de estos conceptos tiene una serie de atributos o propiedades a las que se les puede asignar uno o varios valores.
- **Relaciones**: representan la interacción y enlace entre los conceptos de un dominio, ej. subclase-de o parte-de. Permiten obtener nuevo conocimiento mediante inferencia, partiendo del conocimiento que se encuentra en la ontología.
- **Instancias**: representan objetos determinados de un concepto, por ejemplo, Ana es una persona.
- **Axiomas**: teoremas sobre relaciones que deben cumplir los elementos de una ontología. Posibilitan la inferencia de conocimiento que no esté representado explícitamente en la taxonomía de conceptos.

La ontología diseñada en el proyecto DBpedia ha sido creada manualmente basándose en los atributos más utilizados en artículos de Wikipedia, un subconjunto de estos se muestra en la Figura 7. Es una ontología simple, que abarca un gran conjunto de dominios y actualmente cubre 685 clases mediante una jerarquía de subsunción, describiendo un total de 2 795 propiedades diferentes sobre 4 233 000 instancias [19].

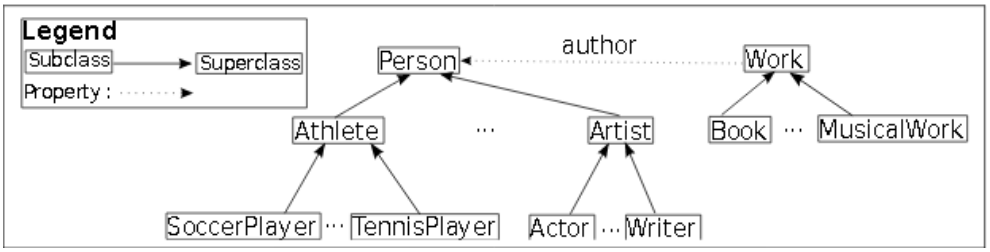

Fig. 7. Subconjunto de la ontología propuesta en DBpedia [16]

En 2007, la base de datos contenía información sobre más de 1 950 000 elementos, incluyendo al menos 80 000 personas, 70 000 lugares, 35 000 álbumes musicales, 657 000 imágenes, 1 600 000 enlaces a páginas web externas 180 000 enlaces a otras bases de datos RDF, 207 000 categorías de Wikipedia y 75 000 categorías de *Yet Another Great Ontology* (YAGO) [15]. Con todo ello, se formaron alrededor de 103 000 000 tripletas RDF. Se puede observar un extracto de la base de datos del mismo año en la Figura 8.

Dataset	Description	Triples
Articles	Descriptions of all 1.95 million concepts within the English Wikipedia including titles, short abstracts, thumbnails and links to the corresponding articles.	7.6M
Ext. Abstracts	Additional, extended English abstracts.	2.1M
Languages	Additional titles, short abstracts and Wikipedia article links in German, French, Spanish, Italian, Portuguese, Polish, Swedish, Dutch, Japanese, Chinese, Russian, Finnish and Norwegian.	5.7M
Lang. Abstracts	Extended abstracts in 13 languages.	1.9M
Infoboxes	Data attributes for concepts that have been extracted from Wikipedia infoboxes.	15.5M
External Links	Links to external web pages about a concept.	1.6M

Fig. 8. Extracto de la base de datos de DBpedia [15]

Con las sucesivas versiones del proyecto, la ontología y el sistema propuesto han ido enriqueciéndose [15]. Actualmente, algunas propiedades a destacar son: la ontología no es un árbol sino un grafo dirigido acíclico (DAG), las clases pueden estar relacionadas mediante herencia con más de una clase y usuarios externos pueden definir sus propios mapeos para atributos de Wikipedia en los que estén interesados, extendiendo así la ontología con clases y propiedades adicionales.

Finalmente, en el apartado actual, se ha visto una definición del concepto de ontología y se han concretado algunos detalles de la construida en DBpedia, sin embargo, es necesario un lenguaje lógico y formal que proporcione un alto grado de expresividad y uso para expresar las ontologías. Existen varios lenguajes de este estilo, pero nosotros comentaremos, en el próximo apartado, únicamente aquel empleado en la ontología de DBpedia (RDF).

3.3 *Resource description framework*

RDF no es propiamente un lenguaje de ontologías, pero está muy próximo a esta clase de lenguajes debido a que su principal objetivo es proporcionar un marco de representación de conocimiento estandarizado para la web [18]. Concretamente, RDF, es un lenguaje de representación de conocimiento estándar para el intercambio de datos en web, que nos permite describir cualquier tipo de recurso como recursos web, entidades y conceptos generales, que está formado por tripletas $(recurso, propiedad, valor) \equiv (sujeto, predicado, objeto)$.

Más detalladamente, la semántica de cada uno de los elementos de las tripletas, según [17] y [18], es como sigue:

- **Recurso**: es equivalente al sujeto de la declaración, ente del cual se habla. Comprende cualquier entidad concreta o abstracta, por ejemplo, un libro o una persona.

- **Propiedad**: su significado es equivalente al predicado de una oración. Define relaciones entre el sujeto y otros entes, valores, características o atributos.
- **Valor**: son recursos que simbolizan el objeto de las oraciones. Es la entidad a la que se refiere el predicado, utilizada para describir propiedades del recurso o sujeto.

A causa del enlace entre recursos y valores mediante propiedades, se genera un multigrafo dirigido y ponderado en el que las aristas representan un enlace etiquetado entre dos entidades relacionadas, tal como se puede ver en la Figura 9. $G = (V, E)\ dirigido\ ,\ ponderado, V = \{recursos\}, E = \{(x, y)\ /\ \exists R : xRy\}$. Usualmente, esta representación en forma de grafo, es empleada en explicaciones visuales para comprender, de la forma más sencilla posible, el modelo de datos de RDF.

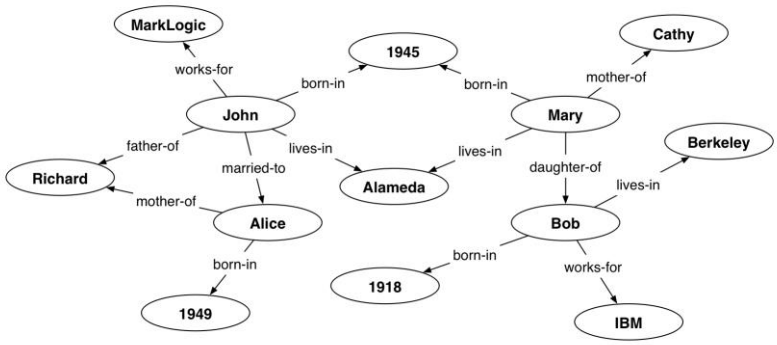

Fig. 9. Ejemplo de grafo RDF

De acuerdo al modelo de datos de RDF, los componentes de las tripletas: recurso, propiedad y valor están identificados con un único *Universal Resource Identifier* (URI). Para la ontología de DBpedia en inglés, estos URI de referencia son de la forma *http://dbpedia.org/resource/nombre* donde el valor de *nombre* se extrae automáticamente de la *Uniform Resource Locator (*URL) del artículo en inglés del recurso cuya URL es *http://en.wikipedia.org/wiki/nombre*. Esto proporciona, según [15], determinadas propiedades que son beneficiosas para los identificadores:

- Cubren un amplio rango de temas enciclopédicos.
- Son definidos mediante consenso por parte de la comunidad.
- Existen políticas claras para su gestión.
- El texto de definición del concepto es fácilmente accesible mediante el identificador.

Concluyendo, en este apartado se ha realizado una breve introducción al lenguaje de representación de conocimiento estandarizado RDF, pero no se ha comentado cómo es posible realizar consultas frente a los grafos generados por las relaciones especificadas en las tripletas. Para ello, empleamos SPARQL ya que DBpedia nos ofrece la posibilidad de consultar su base de datos mediante este lenguaje estandarizado y normalizado por el RDF Data Access Working Group (DAWG) del World Wide Web Consortium (W3C) para la consulta de grafos RDF.

3.4 SPARQL

SPARQL es un lenguaje de consulta estandarizado, normalizado por DAWG del W3C y específicamente diseñado para realizar consultas frente a grafos RDF [20]. Se considera como el lenguaje de consulta por referencia para la web semántica, permitiéndonos, entre otras acciones [21]:

- Extraer valores de información estructurada y semiestructurada.
- Explorar los datos mediante la consulta de relaciones desconocidas.
- Realizar uniones de diversas bases de datos mediante consultas sencillas.
- Transformar los datos RDF de un vocabulario a otro.

Del mismo modo que en *structured query language* (SQL) se suele distinguir entre *data definition language* (DDL), que se encarga de la modificación, eliminación y creación de la estructura de los objetos de la base de datos, y *data manipulation language* (DML), que permite la manipulación de los propios datos en la base de datos; en algunas de las implementaciones propuestas de SPARQL, es necesario discernir entre el lenguaje de consulta que incorpora funciones para recuperar sentencias RDF y las operaciones necesarias para el mantenimiento de los datos (creación, modificación y borrado).

En este proyecto, únicamente nos centramos en el lenguaje de consulta que nos permitirá extraer representaciones vectoriales discretas de palabras y frases. Debido a esto, consideramos necesario enunciar la estructura y los componentes de las consultas [21] que formamos cuando extraemos información para ofrecer soluciones a los problemas tratados en el proyecto:

- **Declaraciones de prefijos**: permiten abreviar URIs.
- **Definición de la base de datos:** se declara qué grafo o grafos serán tratados en la consulta.
- **Cláusula**: identifica qué información debe devolver la consulta. Existen cuatro tipos de cláusulas:
    - ***Select***: empleada para obtener valores en texto plano, el resultado se devuelve en forma tabular.
    - ***Construct***: permite, a partir de la información obtenida, transformar el resultado a un formato RDF válido.
    - ***Ask***: provee un resultado booleano cierto o falso para la consulta realizada.
    - ***Describe***: usado para extraer el grafo RDF resultado de la consulta.
- **Patrón**: especifica una serie de restricciones para la cláusula empleada, generalmente bloques *where* que restringen los resultados de la consulta. Su uso es obligatorio con todos los tipos de cláusulas menos con *describe*.

- **Modificadores**: detallan operaciones a realizar una vez han sido obtenidos los resultados de la consulta, ej. exclusión, ordenación y otras operaciones que permitan reordenar los datos.

Una cuestión de ejemplo mencionada en la página española de DBpedia [22] es: ¿qué toreros se casaron con cantantes de copla? Podemos dar respuesta a la cuestión lanzando la consulta SPARQL expuesta en la Figura 10, donde se identifican y detallan los componentes, mencionados en la enumeración anterior, que conforman las consultas.

Algoritmo 1: consulta de ejemplo SPARQL

```
1: PREFIX dcterms: http://purl.org/dc/terms/
2: SELECT ?torero ?cantante
3: WHERE{
4:    ?torero rdf:type dbpedia-owl:BullFighter .
5:    ?torero dbpedia-owl:spouse ?cantante .
6:   ?cantante dcterms:subject http://es.dbpedia.org/resource/Categoría:Cantantes_de_coplas
7: }
```

Fig. 10. ¿Qué toreros se casaron con cantantes de copla? [22]

Finalmente, mencionar que, para interactuar mediante SPARQL con un servicio, es necesario enviar las consultas del cliente a alguna interfaz conocida del servicio que permita dicha interacción. Para nuestro caso, DBpedia, es posible realizar consultas a través de la URL *http://dbpedia.org/sparql* , donde se aloja la interfaz que emplea *Virtuoso SPARQL Query Editor* para ejecutarlas y devolver los resultados asociados en función de las cláusulas, patrones y modificadores empleados en la consulta.

3.5 Representación vectorial de palabras y frases

En este apartado se comenta cómo obtenemos representaciones vectoriales de palabras empleando todo lo enunciado en el presente capítulo y cómo es posible combinar estas representaciones para generar vectores de las frases que aparecen en los problemas tratados. Todos los procesos han sido implementados en Python e integrados en el sistema propuesto en el punto tres del capítulo uno.

Por un lado, para representar palabras, se ha empleado la ontología de DBpedia en el proceso de extracción de todas las propiedades p relacionadas con el recurso que se corresponde con la entidad que representa una palabra w. Para ello, en primer lugar, es necesario disponer de una lista de propiedades que aparecen en la ontología [19], P, cuya talla determina la longitud de los vectores de representación. Posteriormente, mediante la consulta SPARQL mostrada en la Figura 11, extraemos el conjunto de propiedades del recurso *term* que ocurren en la lista previamente almacenada, P'.

Algoritmo 2: consulta SPARQL para extraer las propiedades de un recurso

```
1: PREFIX db: http://es.dbpedia.org/resource/
2: SELECT ?onto ?value
3: WHERE{
4:    db:term ?onto ?value .
5: }
```

Fig. 11. Extracción de todas las propiedades de un recurso

Estas propiedades pueden ocurrir una o más veces en P' y en función de esto proponemos dos formas diferentes de obtener la representación vectorial de una palabra:

- **Booleana**: sea V la representación vectorial de una palabra, $V_i = 1$ si $p_i \in P'$ ocurre en P y $V_i = 0$ en caso contrario.
- **Aditiva**: sea V la representación vectorial de una palabra, $V_i = ocurrencias(p_i)$, donde $ocurrencias(p_i)$ retorna el número de veces que ocurre p_i en P'.

Con ello, una vez se ha formalizado la representación de palabras, es necesario especificar cómo podemos construir vectores para las frases mediante la combinación de los vectores de los términos. Para esto, se han empleado las representaciones de frases mencionadas en el punto siete del capítulo dos, pero considerando en este caso, en lugar de los vectores de espacio continuo obtenidos mediante Word2Vec, las representaciones discretas extraídas mediante DBpedia.

Sin embargo, a pesar de conocer las representaciones que podemos emplear para las frases, es necesario averiguar sobre qué palabras de un documento es posible realizar el proceso de extracción de propiedades. Para solventar esto, hemos propuesto dos métodos (se han planteado más, como buscar únicamente palabras en mayúscula, pero no se han llegado a implementar):

- Intentar encontrar, por fuerza bruta, propiedades sobre todos los términos que aparecen en una determinada frase. El principal inconveniente del procedimiento es el alto coste temporal y la gran cantidad de tráfico ingenuo generado hacia el servidor de DBpedia debido a que, muchos de los términos que ocurren en las frases no tienen un recurso asociado.

- Determinar previamente con el sistema de selección DBpedia *Spotlight* [23], aquellos términos que, con un grado de confianza, tienen un recurso asociado en la ontología de DBpedia, anotándolos con la URL correspondiente. Como ventajas, destacar que se reduce el coste temporal del método y el tráfico generado, ya que, para una frase, N consultas en el procedimiento anterior donde N es el número de términos de la frase, se convierten en $|entidades|$ peticiones en este método, siendo $entidades$ el conjunto de entidades que detecta DBpedia *Spotlight* en la sentencia.

Así, mientras que en el primer método únicamente procesamos los documentos palabra por palabra ejecutando en el *endpoint* de DBpedia la consulta SPARQL de la Figura 11 y combinamos las representaciones de cada término mediante alguna de las operaciones propuestas para obtener el vector del documento, en el segundo, es necesario un filtrado previo de los términos, empleando la interfaz de programación de aplicaciones (API) mediante transferencia de estado representacional (REST) de DBpedia *Spotligh,* cuyas propiedades se quieren extraer. Sin embargo, debido a la limitada extensión del proyecto, las experimentaciones realizadas que emplean el sistema de selección han sido descartadas.

Por último, manifestar que se ha hecho un mayor énfasis en las representaciones vectoriales de espacio continuo, es por ello que, a pesar de que se han realizado algunas pruebas con las representaciones vectoriales discretas que se han tratado en este apartado, los experimentos utilizando DBpedia se han considerado, únicamente, para proponer soluciones alternativas con las que se intenta mejorar los resultados obtenidos con representaciones continuas.

«La naturaleza nos ha dado las semillas del conocimiento, no el conocimiento mismo.»

-Séneca

Capítulo 4

Algoritmos de aprendizaje

En el presente capítulo se justifica y se introduce, siguiendo a [24], la necesidad de utilización de algoritmos de aprendizaje para estimar clasificadores (se hará uso de los integrados en el módulo *scikit-learn* de Python) que permitan discriminar entre un conjunto de clases relativas a un determinado problema, en nuestro caso, problemas de detección de temática y otros relacionados como identificación del idioma o análisis de sentimiento. Además, se realiza una descripción detallada, de acuerdo a diversa literatura científica, de los clasificadores empleados a lo largo del proyecto, se analiza la utilización de diversas métricas que posibilitan la evaluación de dichos clasificadores y se expone el protocolo de experimentación seguido.

4.1 Introducción al aprendizaje computacional

El término aprendizaje computacional hace referencia al conjunto de tecnologías desarrolladas en el marco de diversas disciplinas relacionadas con los sistemas inteligentes como reconocimiento de formas, inteligencia artificial, estadística aplicada etc., considerándose recientemente, como un planteamiento integrador de todo este conjunto de disciplinas.

Típicamente, una taxonomía general de los tipos de aprendizaje considera, por un lado, el aprendizaje deductivo, en el que se asume que existe un agente que transfiere su conocimiento de alguna forma al sistema, y por otro, el aprendizaje inductivo, donde el sistema posee escaso conocimiento *a priori* sobre la tarea y construye su propio modelo mediante la observación de un conjunto de muestras de aprendizaje $S \subset \mathcal{X} \times \mathcal{Y}$. Nosotros nos centramos en el aprendizaje inductivo, a pesar de que existen enfoques para los problemas planteados que emplean aprendizaje deductivo mediante la utilización de sistemas basados en conocimiento [25].

En el aprendizaje inductivo, como ya se ha mencionado, se asume la existencia de datos de aprendizaje; normalmente información de entrada $x \in \mathcal{X}$ y de salida $y \in \mathcal{Y}$, y se plantea obtener una función $f : \mathcal{X} \rightarrow \mathcal{Y}$ que generalice adecuadamente los datos de entrenamiento, pudiendo predecir así, la salida de nuevas muestras de entrada diferentes a las de aprendizaje. Una vez el modelo f ha sido aprendido, podemos emplearlo para dos tipos de tareas:

- **Regresión**: tanto los datos de entrada $x \in \mathcal{X}$ como los de salida $y \in \mathcal{Y}$ pertenecen a dominios arbitrarios $(\mathcal{X}, \mathcal{Y})$, es decir, no hay ninguna restricción en cuanto al número de componentes en los vectores de entrada y salida. Un caso simple es aquel en el que se pretende predecir valores continuos dado un conjunto de muestras n-dimensionales $f : \mathbb{R}^n \rightarrow \mathbb{R}$.

- **Clasificación**: $\mathcal{X}$ es arbitrario, típicamente $\mathbb{R}^n$, pero $\mathcal{Y}$ es un conjunto generalmente pequeño de C elementos llamados clases, $\mathcal{Y} = \{1,2,\ldots,C\} \subset \mathbb{N}$ por tanto, los modelos predictores suelen ser funciones $f : \mathbb{R}^n \rightarrow \mathbb{R}^C$.Este tipo de problemas es muy común en reconocimiento de imágenes, texto manuscrito etc., y son problemas de clasificación los que se trataran en el presente proyecto.

Sin embargo, es posible que no se disponga de información relativa a la salida $y \in \mathcal{Y}$ de las muestras de entrenamiento y en función de dicha disponibilidad se distinguen dos tipos de aprendizaje dentro del aprendizaje inductivo:

- **Aprendizaje supervisado**: se dispone de información completa de entrada y salida de los datos de entrenamiento $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}: x_i \in \mathcal{X}, y_i \in \mathcal{Y}$ El objetivo es obtener un modelo que generalice dichos datos $f: \mathcal{X} \rightarrow \mathcal{Y}$.

- **Aprendizaje no supervisado**: los datos de aprendizaje solo contienen información de la entrada $S = \{x_1, x_2, \ldots, x_n\}: x_i \in \mathcal{X}$, por ello, en este caso, el objetivo es aproximar la estructura del dominio de salida $\mathcal{Y}$.

Existen muchos otros tipos de aprendizaje dentro del inductivo: semisupervisado, adaptativo, en línea, activo, por refuerzo etc., pero se han mencionado únicamente aquellos que se emplean en el proyecto para solucionar, mediante diversos enfoques, los problemas planteados.

Por otro lado, del mismo modo que se estableció una taxonomía general entre los tipos de aprendizaje i.e. inductivo y deductivo, se puede formular una clasificación similar para discriminar entre los diferentes planteamientos que nos permiten aproximar las funciones o modelos f en el aprendizaje inductivo, para ello, destacamos las aproximaciones que se emplean para implementar los algoritmos de clasificación utilizados en las experimentaciones:

- **Marco estadístico**: el modelo $f : \mathcal{X} \rightarrow \mathcal{Y}$ que se aprende, se considera una función de decisión y se asume que cada decisión tiene un coste, ej. cero si la decisión $f(x)$ es acertada y uno en caso contrario. Con este planteamiento, el criterio de éxito consiste en minimizar la probabilidad de error sobre todos los posibles datos de entrada $x \in \mathcal{X}$. Sabiendo que la decisión óptima es aquella que minimiza el error, $P_*(error|x) = \min_{y \in \mathcal{Y}} P_y(error|x) = 1 - \max_{y \in \mathcal{Y}} P(y|x)$, podemos descomponer la probabilidad *a posteriori* $P(y|x)$ mediante la regla de *Bayes,* resultando en la función óptima de decisión $f^*(x) = argmax_{y \in Y} P(y) \cdot P(x|y)$. Con esto, es posible aproximar la probabilidad condicional $P(x|y)$ mediante las distribuciones que más se ajusten a los datos de entrada: *Bernoulli,* Gaussiana etc.

- **Marco conexionista**: en este caso no se requieren asunciones probabilísticas sobre el dominio de las funciones a aprender por lo que el criterio de éxito se establece en términos de minimización del error de decisión sobre un conjunto de test (ya no sobre la probabilidad de error sobre toda posible muestra). Los modelos $f \in F$ se expresan mediante C funciones discriminantes $\phi_1, \phi_2, \ldots, \phi_C$, $\phi_i: \mathcal{X} \to \mathbb{R}$, $1 \leq i \leq C$, con ello, la función de decisión sobre una muestra x, $f(x)$, se define como la clase que maximiza el valor de su función discriminante $f(x) = argmax_{1 \leq i \leq C}\, \phi_i(x)$. A este enfoque pertenecen: máquinas de vectores soporte y redes neuronales entre otros.

- **Basado en proximidad**: es un enfoque no paramétrico en el que, aunque no es necesario que se empleen representaciones vectoriales, se asume que $\mathcal{X}$ es un espacio métrico (E, d) compuesto por un conjunto de puntos E y una función $d: ExE \to \mathbb{R}$ denominada métrica o distancia que cumple las siguientes propiedades $\forall\, p, q, r \in E$:

  - $d(p,q) \geq 0\,; \quad d(p,q) = 0 \; sii \; p = q$
  - $d(p,q) = d(q,p)$
  - $d(p,q) \leq d(p,r) + d(r,q)$

  Se espera que si dos objetos $\boldsymbol{x}$, $\boldsymbol{x}'$ son similares, $d(\boldsymbol{x}, \boldsymbol{x}')$ sea pequeña, por lo que el aprendizaje se realiza básicamente por memorización, almacenando las muestras de entrenamiento (llamadas prototipos en este planteamiento) y clasificando nuevos elementos mediante una función de clasificación f, basada en las distancias entre un dato de test y los prototipos de las distintas clases [26].

- **Árboles de decisión**: se enmarcan en la aproximación no paramétrica del reconocimiento de formas, constituyendo una forma de representación del conocimiento simple y efectiva, mediante la partición recursiva del espacio de representación a partir de una muestra de aprendizaje [27]. Los nodos internos de estos árboles contienen una pregunta y se ramifican en tantos hijos como posibles respuestas haya a esa pregunta, mientras que los nodos terminales contienen la etiqueta de la clase que se corresponde a una decisión final de clasificación. De esta manera, un dato de test se clasifica mediante una serie de preguntas sobre sus atributos que ayudan a definir fronteras paralelas a los ejes de un espacio n-dimensional.

- **Métodos de conjunto (*ensemble learning)*:** típicamente, el diseño de clasificadores persigue reducir dos fuentes de error, según [28]:

  - ***Bias****:* cuantifica el error de un clasificador debido a asunciones erróneas, por ejemplo, asumir que los datos son linealmente separables.
  - ***Variance****:* mide el error del clasificador debido a su dependencia en los datos de entrenamiento i.e. si el clasificador requiere muchos datos para poder aprender tendrá un *variance* alto al definir una frontera de decisión compleja, en cualquier otro caso, el *variance* será bajo.

Así, el error de un clasificador se define como la combinación de estas dos fuentes de error junto con el ruido inherente en los datos. Si tenemos en cuenta esta afirmación podemos pensar, por tanto, en combinar clasificadores para reducir ese error, esto es conocido con el nombre de *ensemble learning*.

En estos métodos se pretende, partiendo de diferentes clasificadores $G_1, G_2, \ldots, G_M$ y una serie de pesos $w_1, w_2, \ldots, w_M$, obtener un clasificador, por ejemplo, de la forma: $G(x) = \sum_{i=1}^{M} w_i G_i(x)$ cuyo error es $E = E' - D$ donde E' es el error promedio de todos los clasificadores a combinar y D es un término que representa la diversidad de los clasificadores. Por tanto, según la expresión del error, cuanto mejor y más diversos sean los clasificadores G_i empleados, menor será el error del clasificador final. Dos ejemplos muy conocidos de métodos de conjunto son *bagging* y *boosting*.

Con ello, se ha introducido un planteamiento formal del aprendizaje inductivo, donde el sistema posee escaso conocimiento *a priori* y debe ser capaz de obtener sus propios modelos mediante ejemplos supervisados o no supervisados, y se han detallado los límites y los paradigmas empleados en la obtención de dichos modelos.

4.2 K-vecinos más cercanos

A pesar de la existencia de algoritmos no supervisados y orientados a regresión basados en K-vecinos más cercanos (KNN), nos centramos en el método supervisado para clasificación que requiere información completa de la salida de las muestras [25]. Para hacer uso de él en los problemas planteados, se empleará el módulo *sklearn.neighbors* de la librería *scikit-learn*, llamando a las funciones *NearestNeighbors* [29] o *NearestCentroid* [30] en función de los prototipos que se quieran considerar.

Este clasificador, es un método basado en proximidad en el que se tienen en cuenta las etiquetas de clase de los k vecinos más próximos a una muestra dada $\boldsymbol{x}$, por lo que, para poder emplear dichos vecinos en la clasificación de la muestra se debe tener un conjunto de prototipos P_c de cada clase $1 \leq c \leq C$. Con ello, sea P_k el conjunto de los k vecinos más próximos bajo la métrica d, la regla de decisión de este tipo de clasificadores se define como $x \in c' \Leftrightarrow |P_k \cap P_c'| \geq |P_k \cap P_c|, 1 \leq c \leq C\ , c \neq c'$ [26], es decir, una vez calculados los k vecinos más próximos a $\boldsymbol{x}$, P_k , la clase predicha para $\boldsymbol{x}$ es aquella que más prototipos tiene dentro de P_k.

Se ha podido observar en la formalización del método, que un parámetro que puede condicionar en gran medida su funcionamiento es el valor de k, por ello, debido al interés en dicho parámetro, la comunidad científica ha demostrado que con un valor $k = \sqrt{n}$, cuando $n \to \infty$ (ya que $\frac{k}{n} \to 0$) el riesgo de error del clasificador tiende al error de *Bayes* (mínimo error posible para cualquier sistema decisor).

Un caso particular de este tipo de clasificador es cuando $k = 1$, en este caso, cada punto contribuye a la formación de la frontera de decisión y por ello se definen fronteras de clasificación lineales definidas a trozos como muestra la Figura 12. La regla de decisión se simplifica en comparación a la del caso general siendo ahora de la forma $x \in c' \Leftrightarrow \exists\, y \in P_{c'} : d(x,y) \leq d(x,y') \forall\, y' \in P_c\,,\ 1 \leq c \leq C\,, c \neq c'$ [27] y se puede acotar el error del clasificador cuando $n \to \infty$ (siendo P^* el error de *Bayes* y P el error del clasificador): $P* \leq P \leq P^* \cdot \left(2 - \frac{c}{c-1} \cdot P^*\right) \leq 2P^*$.

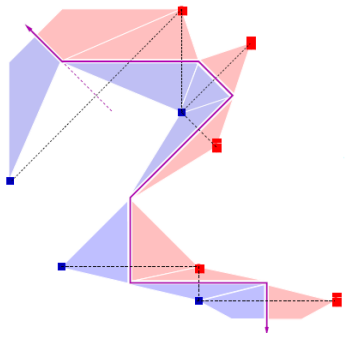

Fig. 12. Definición de una frontera de decisión definida a trozos [27]

Por otro lado, debido al alto coste espacial y temporal de este tipo de métodos basados en memorización, son necesarios algoritmos de edición y condensado de muestras que permitan efectuar una selección adecuada de aquellos prototipos más discriminantes, entre este tipo de algoritmos podemos encontrar el algoritmo de edición de D.L. Wilson y el algoritmo de condensado de P.E. Hart, sin embargo, estos no son los únicos métodos para reducir el número de elementos. Una forma muy eficiente para considerar un único prototipo por clase es calcular el punto central de todas las muestras de cada clase, con ello, el problema se reduce a calcular el *centroide* más cercano a una muestra $\boldsymbol{x}$ dada, este clasificador es *NearestCentroid* de *scikit-learn* [30].

Además, aparte del conjunto de prototipos de cada clase y de la variable k, existe otro meta-parámetro que puede condicionar la eficiencia del método, la función de distancia empleada. Esta función de distancia se debe adecuar a la distribución de los prototipos en cada una de las clases, así como eliminar los efectos de escala en las distintas componentes, llevando a cabo una normalización de los datos. Entre estas funciones podemos encontrar en [31]: euclídea normalizada, Mahalanobis-diagonal, Mahalanobis-diagonal por clase, Mahalanobis-local etc., ponderando, cada una de las distancias mencionadas, de forma diferente las muestras de cada clase.

Finalmente destacar que, a pesar de su simplicidad, el clasificador de vecinos más cercanos ha resultado exitoso en una gran cantidad de problemas de clasificación y regresión, incluyendo reconocimiento de çdígitos manuscritos o detección de objetos en imágenes por satélite. Además, al ser un enfoque no paramétrico, es a menudo muy útil en problemas de clasificación donde las fronteras de decisión son muy irregulares, ej. detección de la temática en el corpus TC-STAR planteado en este proyecto.

4.3 Árboles de decisión

Como ya se mencionó en el apartado uno del capítulo cuatro, este tipo de clasificadores se enmarca en la aproximación no paramétrica del reconocimiento de formas, clasificando mediante una partición recursiva del espacio de representación que define fronteras paralelas a los ejes [27]. Entre los posibles clasificadores basados en árboles podemos encontrar ID3, C4, C4.5, etc. pero nosotros nos centraremos en CART, cuya versión optimizada está implementada en *scikit-learn* y es accesible a través de la función *DecisionTreeClassifier* [32] del módulo *tree*.

Este algoritmo, construye árboles binarios empleando la función y el umbral que producen la mayor ganancia de información en cada nodo. Para la construcción de dichos árboles, se requerirá, según [27]:

- Un método para hacer particiones y seleccionar la mejor.
- Un criterio para determinar cuándo un nodo tiene la suficiente información como para considerarse terminal.
- Un criterio para asignar una etiqueta de clase a un nodo terminal.

En primer lugar, para formar particiones, se emplearán preguntas de la forma $¿y_j \leq r?$ donde $1 \leq j \leq D$ y r es el correspondiente umbral de la dimensión j, empleando, para evaluar las particiones, el concepto de impureza de un nodo $\chi(t)$ que se mide en función de las probabilidades estimadas de las clases en el nodo t. Existen varias aproximaciones para medir esta impureza, siendo una de las más interesantes la entropía $\chi(t) = -\sum_{c=1}^{C} P'(c|t) \cdot log_2 P'(c|t)$. El criterio para seleccionar la mejor partición consiste en elegir aquella que maximiza el decremento de impureza $max_{1\leq j\leq D, -\infty<r<+\infty} \Delta\chi(j, r, t)$.

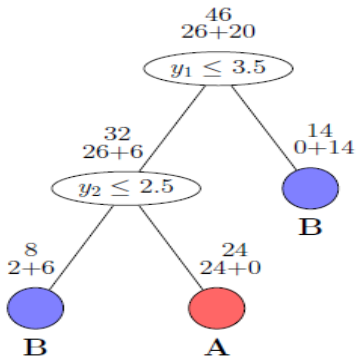

Fig. 13. Ejemplo de árbol de decisión para dos clases y dos dimensiones [27]

Posteriormente, para determinar cuándo un nodo *t* tiene la suficiente información como para considerarse terminal, uno de los criterios más simple es considerar si el máximo decremento de impureza posible es demasiado pequeño: $max_{1\leq j\leq D, -\infty<r<+\infty} \Delta\chi(j,r,t) < \epsilon$, donde ϵ es una constante pequeña a determinar empíricamente.

En último lugar, para asignar una etiqueta de clase a los nodos terminales, es posible integrar en cada uno de estos, la etiqueta de la clase que maximice el número de elementos en las particiones generadas por dichos nodos $c^*(t) = argmax_{1\leq c\leq C} P'(c|t)$. Es posible observar un ejemplo gráfico del resultado obtenido, tras todo el proceso mencionado, en la Figura 13.

4.4 Máquinas de vectores soporte y *kernels*

Citando a [33], las máquinas de vectores soporte o *support vector machines* (SVM) son un conjunto de algoritmos de aprendizaje supervisado basado en funciones discriminantes lineales del estilo $\phi : \mathbb{R}^d \to \mathbb{R} : \phi(\boldsymbol{x}; \boldsymbol{\Theta}) = \sum_{i=1}^{d} \theta_i x_i + \theta_0$ donde $\Theta = (\boldsymbol{\theta}, \theta_0)$: $\boldsymbol{\theta} \in \mathbb{R}^d$ es un vector de pesos y $\theta_0 \in \mathbb{R}$ se denomina umbral. Por tanto, la regla de clasificación, limitando las etiquetas de clase a +1 y -1, es como sigue: $f(x) = signo(\phi(\boldsymbol{x}; \boldsymbol{\Theta}))$.

Este tipo de métodos, se fundamenta en el hecho de que pueden existir múltiples funciones discriminantes lineales (FDL) que separen correctamente las muestras de aprendizaje, es decir, $c_n\big(\phi(x_n; \Theta)\big) > 0, 1 \leq n \leq N$ donde N es el cardinal del conjunto de muestras de entrenamiento. Sin embargo, no todas generalizan igual y nos interesan aquellas soluciones que tengan un margen considerable con respecto a las muestras de cada clase, esto es, que maximicen la distancia con respecto a cada una de las clases.

Por consiguiente, si definimos el margen del hiperplano separador H con respecto al conjunto de muestras S como $2r^* = \frac{2}{||\theta||}$ donde r^* es la distancia del vector $x^* \in S$, más próximo al hiperplano separador, a la frontera de decisión, el objetivo del aprendizaje será resolver un problema de maximización del margen $\frac{2}{||\theta||}$ sujeto a ciertas restricciones de canonicidad mediante la técnica de multiplicadores de *Lagrange* para optimización con restricciones. Un hiperplano definido tras la resolución del problema de maximización puede verse en la Figura 14.

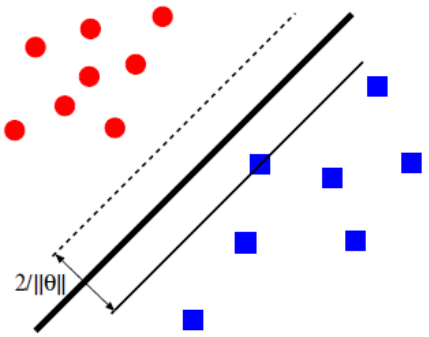

Fig. 14. Hiperplano que maximiza la distancia con respecto a ambas clases [33]

Tras la resolución del problema, el clasificador de máximo margen queda definido por la función discriminante lineal $\phi(\boldsymbol{x};\boldsymbol{\Theta}) = \boldsymbol{\theta}^{*t} \cdot \boldsymbol{x} + \theta_0^*$ donde $\boldsymbol{\theta}^*$ y θ_0^* se definen en función de los multiplicadores de *Lagrange* obtenidos en el proceso de resolución. Sustituyendo en la función de clasificación los términos mencionados se obtiene la función discriminante lineal que maximiza el margen: $\phi_{svm}(\boldsymbol{x};\boldsymbol{\Theta}) = \sum_{n\in v} \alpha_n^* c_n x_n^t \boldsymbol{x} + \theta_0^*$ donde v es el conjunto de multiplicadores de *Lagrange* y α_n^* es el vector soporte n-ésimo.

Pero, ¿y si no existe una FDL que separe correctamente el conjunto de muestras?, ante esto existen dos posibles soluciones ampliamente aceptadas:

- Modificar el problema de optimización original añadiendo un término que pondera cómo de mal clasificado se tolera que esté cada vector $x_n \in S$, esto da lugar a los llamados SVM con márgenes blandos.

- Emplear funciones *kernel* que modelan el producto escalar en un espacio de representación alternativo. Esto nos permitirá poder definir funciones discriminantes más complejas que las definidas por el planteamiento original de los SVM de una forma eficiente, sin la necesidad de transformar las muestras al espacio alternativo.

La implementación de la segunda solución mencionada, aprovecha la propiedad de que una SVM se puede expresar en base a productos escalares entre muestras de entrenamiento tal como se ha visto en este mismo apartado. Por esa razón, se pueden integrar las funciones *kernel*, sin más que modificar la función discriminante de la siguiente forma: $\phi_\kappa(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n c_n \kappa(\boldsymbol{x}_n, \boldsymbol{x}) + \theta_0$ donde κ es una función *kernel* válida.

Por otro lado, respecto a la utilización de SVM, en *scikit-learn* existen múltiples implementaciones que difieren, principalmente, en los métodos empleados a la hora de clasificar entre múltiples clases (se puede observar un ejemplo de clasificación multiclase en la Figura 15), a distinguir: *LinearSVC* [34] y *SVC* [35]. El primero de ellos está implementado en función de la libreria *liblinear* y emplea el enfoque uno contra todos en clasificación de múltiples clases, mientras que *SVC* está basado en *libsvm* y hace uso de la clasificación uno contra uno cuando existen más de dos clases.

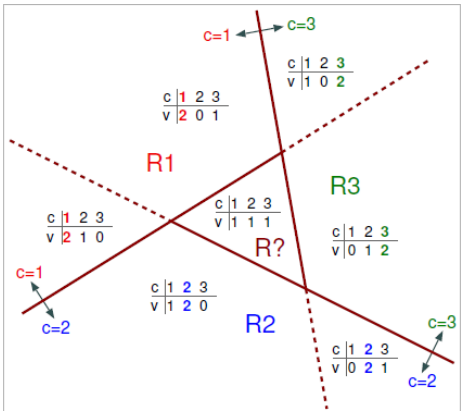

Fig. 15. Ejemplo de clasificación en múltiples clases [33]

Finalmente destacar, que los sistemas basados en máquinas de vectores soporte son, junto a las redes neuronales artificiales, uno de los métodos más utilizados en la actualidad debido a sus buenos resultados tanto en clasificación como en regresión, siendo eficientes y extremadamente robustos en la generalización de las muestras de aprendizaje [36].

4.5 *Naive Bayes*

El clasificador de *Bayes* es un método paramétrico y supervisado basado en la aplicación del teorema de *Bayes*, con la ingenua hipótesis de que todo par de características es independiente entre sí [37]. Al ser un método basado en probabilidades, se requiere realizar asunciones estadísticas sobre el dominio de las funciones a aprender, haciendo estas asunciones en función de densidades de probabilidad *a priori* y condicionales por cada clase [38], que rigen la distribución de las muestras en el espacio.

Formalmente, la regla de clasificación de *Bayes* consiste en asignar a una muestra $\boldsymbol{x}$ aquella etiqueta de clase c^* que maximice la probabilidad *a posteriori* de dicha clase dada la muestra $\boldsymbol{x}$: $c^*(x) = argmax_{c=1,\dots,C}\ p(c|x)$. Aplicando el teorema de *Bayes* a la expresión anterior se obtiene $c^*(x) = argmax_{c=1,\dots,C}\ \frac{p(c)\cdot p(x|c)}{p(x)}$, como $p(x)$ no depende de c la expresión resulta en $c^*(x) = argmax_{c=1,\dots,C}\ p(c)p(x|c)$.

Una vez definida la regla de clasificación, se deben estimar los parámetros $p(c)$ y $p(x|c)$. La estimación de $p(c)$ se puede realizar de forma muy sencilla teniendo en cuenta únicamente las muestras de la clase c mediante la expresión $p(c) = \frac{N_c}{N}$, sin embargo, la aproximación de la probabilidad condicional $p(x|c)$ requiere asunciones sobre la distribución de las muestras [38]. Para llevar a cabo dicha aproximación, podemos emplear distribuciones estadísticas conocidas, a destacar las utilizadas en el proyecto:

- ***Bernoulli***: se asume que la probabilidad condicional se distribuye como una distribución de *Bernoulli* $p(x|c) \sim Be_D(\boldsymbol{p_c})$ [39]. Es útil cuando los vectores de características son binarios i.e. $\boldsymbol{x} \in \{0,1\}^n$, penalizando la no ocurrencia de la característica i en la clase c según la siguiente expresión: $P(x_{\{i\}}|c) = P(i|c) \cdot x_i + (1 - P(i|y))(1 - x_i)$.

- **Gaussiana**: se asume que la probabilidad condicional se distribuye como una distribución normal $p(\boldsymbol{x}|c) \sim \mathcal{N}_D(\mu_c, \Sigma_c), c = 1, \ldots, C$ [40]. Eficaz cuando las muestras son representadas con vectores reales de características, esto es, $\boldsymbol{x} \in \mathbb{R}^n$. Los clasificadores que hacen uso de esta distribución definen fronteras de decisión cuadráticas, sin embargo, debido al coste de calcular las matrices de covarianza para cada clase, se suele emplear una matriz de covarianzas común obteniendo así un clasificador gaussiano lineal.

Por último, únicamente han sido mencionadas las distribuciones que se emplearán en los clasificadores usados en el proyecto: *BernoulliNB* [41] para la distribución de *Bernoulli* y *GaussianNB* [42] para la Gaussiana, sin embargo, existen muchas otras distribuciones que se pueden considerar para aproximar las probabilidades condicionales como por ejemplo la distribución multinomial, *Fisher* o *Poisson*, entre otras.

4.6 K-medias

K-medias es un método de aprendizaje no supervisado que pretende encontrar agrupamientos naturales en un conjunto de objetos, de forma que la descripción de estos se realice en términos de clases o grupos de objetos con fuertes semejanzas internas [43]. Para conseguir esto, se intentan separar las muestras en grupos de igual varianza, minimizando a su vez la suma de los cuadrados de las diferencias de cada muestra con respecto al *centroide* del agrupamiento al que pertenecen (criterio de inercia).

Formalmente, asumimos disponible una función criterio J para evaluar la calidad de cualquier partición de N datos en C clases, definiendo, por tanto, el problema de la agrupación como la búsqueda del conjunto de particiones que minimicen el valor de J: $\Pi^* = argmin_{\Pi=\{X_1,\ldots,X_C\}} J(\Pi)$ [43]. Como ya se ha comentado en la introducción de este mismo apartado, la función criterio será la suma de errores cuadráticos (SEC) que se define de la siguiente forma: $J(X_1, \ldots, X_C) = \sum_{1 \leq c \leq C} J_c$ donde $J_c = \sum_{\boldsymbol{x} \in X_c} ||\boldsymbol{x} - \boldsymbol{m}_c||^2$ y $m_c = \frac{1}{|X_c|} \cdot \sum_{\boldsymbol{x} \in X_c} \boldsymbol{x}$. Se muestra un ejemplo de agrupamiento óptimo en la Figura 16.

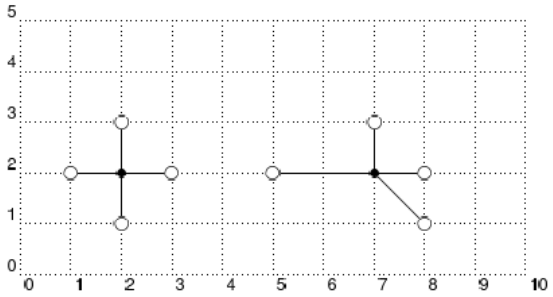

Fig. 16. Agrupamiento óptimo con SEC<12 [43]

Una vez formalizado el problema, únicamente resta implementar el algoritmo. Nosotros haremos uso, llamando a la función *KMeans* del módulo c*luster* [44], del método ya integrado en *scikit-learn* que implementa la versión de S. Lloyd, sin embargo, existen otras versiones como la de R.O. Duda y P.E. Hart que no están implementadas en dicha librería.

Finalmente, respecto a las propiedades de este tipo de métodos, es conveniente destacar, que los algoritmos K-medias pueden quedar estancados en mínimos locales y ninguna de las versiones ideadas garantiza la obtención de un mínimo global, no es así con los mínimos locales, cuya obtención sí está garantizada mediante los métodos de S. Lloyd y de R.O. Duda y P.E. Hart [43]. Además, destacar que el criterio SEC empleado solo es apropiado si los datos forman agrupaciones hiperesféricas de tamaño similar, si los tamaños de dichas agrupaciones son muy distintos, es posible que la agrupación natural no tenga el mínimo SEC.

4.7 *Random forest*

Los *random forest* son un ejemplo típico de los métodos de promedio del *ensemble learning,* que pretenden construir un conjunto de clasificadores independientes y promediar sus predicciones, siendo el clasificador final obtenido mejor que cualquier clasificador individual utilizado en la combinación debido a la reducción del *variance* [28]. Un ejemplo de este tipo de árboles se representa en la Figura 17.

En este caso, los clasificadores aplicados son árboles de decisión construidos a partir de subconjuntos aleatorios de las muestras de entrenamiento. Además, el proceso de construcción de cada árbol, concretamente el criterio para realizar y seleccionar las particiones, difiere del procedimiento ya explicado en el apartado cuatro del presente capítulo en que, en cada nodo, no se escoge la mejor partición teniendo en cuenta todas las características sino únicamente un subconjunto aleatorio de estas [45].

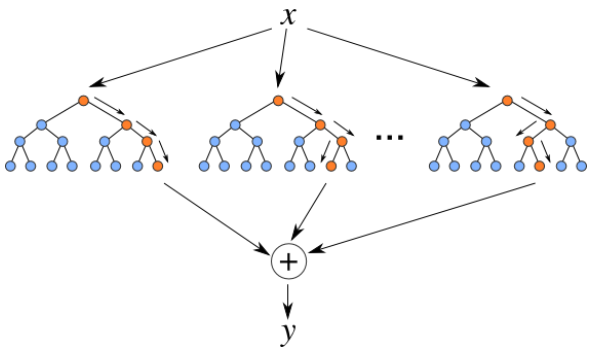

Fig. 17. Ponderación de varios árboles de decisión [45]

Como resultado de esta aleatoriedad, el *bias* del *random forest* se incrementa ligeramente con respecto a los árboles de decisión individuales no aleatorios empleados en la combinación, sin embargo, debido al promedio de todos los clasificadores empleados, el *variance* se reduce lo suficiente como para compensar el incremento del *bias,* obteniendo así un mejor modelo general.

Para concluir, acentuar el uso en la actualidad de este tipo de modelos ya que han sido empleados con éxito en tratamiento y análisis de imágenes para detección de tumores cerebrales, obteniendo mejores resultados que redes neuronales convolucionales específicas para este tipo de tareas.

4.8 Protocolo de experimentación

Debido a la utilización de modelos, basados en redes neuronales, que requieren ajustar una gran cantidad de parámetros condicionantes de su funcionamiento [6], es necesario establecer un protocolo o esquema que nos permita aproximar los valores óptimos para cada uno de estos parámetros. Además, la combinación de todos estos valores supone un problema de explosión combinatoria que dificulta la configuración mediante métodos de barrido y obliga a recurrir a hipótesis del funcionamiento de dichos modelos. Entre los factores variables que es posible combinar para solucionar las diferentes tipologías de problemas tratadas, destacamos:

- Arquitectura de la red neuronal.
- Optimizaciones para el entrenamiento del modelo Word2Vec.
- Tamaño del contexto.
- Dimensión de los vectores de representación.
- Método de normalización.
- Método de representación de frases.
- Clasificadores y sus parámetros.

Estos parámetros ya han sido comentados en el presente documento y es posible notar que una experimentación exhaustiva de las combinaciones entre todos los parámetros mencionados puede no ser factible si los valores de dichos parámetros varían mucho y efectivamente, así es. Debido a esto, durante la realización del proyecto, se han intentado elaborar una serie de pasos que, con el mínimo coste posible, nos permitan probar algunos de los casos más significativos en las experimentaciones.

En un primer momento, se planteó hacer un barrido con valores razonables de los parámetros para cubrir un mínimo espectro de posibilidades: dos arquitecturas diferentes, dos optimizaciones distintas, seis tamaños de ventana, nueve tamaños de representación, dos métodos de normalización, dos métodos de representación de frases y once clasificadores. Sin embargo, este barrido puede no ser suficiente y no es factible generar todos los modelos posibles, ya que un único modelo, dependiendo del corpus, puede tardar una gran cantidad de tiempo en obtenerse. Además, la extensión del presente documento es limitada y no es conveniente exponer una explicación detallada.

Como no es posible llevar a cabo el barrido de parámetros mencionado, es necesario considerar ciertas hipótesis recopiladas de [3] y [4] que nos permitan fijar unos parámetros iniciales y variarlos mínimamente para reducir el número posible de combinaciones a contemplar. Entre estas hipótesis destacamos:

- **Dimensión**: cuanto mayor sea la dimensión, dentro de unos límites razonables de acuerdo al tamaño del corpus, mayor calidad tendrán los vectores obtenidos [7].
- **Tamaño de contexto:** en función del tipo de problema se emplea un tamaño u otro. Si nos interesa conocer detalladamente el ámbito de los términos debemos emplear una ventana de mayor tamaño, en cualquier otro caso, el tamaño de la ventana puede ser reducido [8].
- **Arquitectura***:* debido a la recomendación de T. Mikolov en [3] se emplea la arquitectura *skip-gram* siempre que el coste temporal que conlleva su utilización no sea un impedimento para la experimentación.
- **Optimización:** por la misma razón que la hipótesis anterior, se recomienda el uso de *skip-gram* junto con *negative sampling* en [3].

Una vez conocidas las hipótesis y fijados los parámetros: tamaño de contexto, arquitectura y optimización de la red neuronal, en primer lugar, se debe hacer un barrido para determinar la dimensión de los vectores, ya que hemos supuesto que cuanto mayor sea la dimensión mejor, pero ¿cuál puede ser un valor acertado? Para esto, probamos $100, 250, 500$ y 750 dimensiones con los demás parámetros ya fijados y asignamos el valor de dimensionalidad con el que mejor precisión obtengamos mediante un clasificador dado. A partir de este resultado, variamos el método de optimización de *negative sampling* a *hierarchical softmax* y fijamos la optimización que ofrezca un valor de precisión superior. La visualización gráfica del proceso se puede ver en la Figura 18.

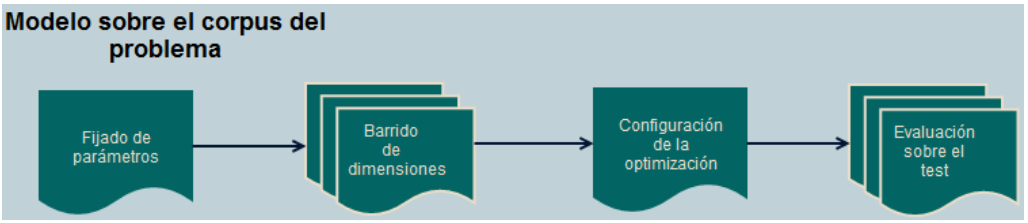

Fig. 18. Proceso de obtención del modelo a partir del corpus de un problema

Así, partiendo del modelo subóptimo alcanzado mediante los pasos mencionados en el párrafo anterior, podemos generar representaciones vectoriales de frases combinando los vectores de representación de las palabras presentes en una declaración dada. Sin embargo, en el presente proyecto, se han desarrollado seis métodos de representación de frases diferentes y la aplicación de todos en cada una de las experimentaciones conllevaría extender demasiado la memoria, además de un incremento del coste temporal, por lo que únicamente se ha hecho uso de tres de estas representaciones, aunque las demás se han expuesto con detalle en el apartado siete del capítulo dos por completitud.

Otro de los aspectos a considerar es la evaluación de los resultados obtenidos tras hacer uso de las representaciones de frases. Hemos realizado las evaluaciones de las soluciones propuestas a los problemas planteados mediante el método de partición, comentado en el siguiente apartado, estableciendo un conjunto de test y otro de entrenamiento y evaluando directamente los modelos obtenidos contra las muestras de test. Sin embargo, este enfoque puede ser optimista y no seguir el «método científico», por ello, en el apartado dedicado a la identificación de temática con representaciones continuas que constituye el principal problema del proyecto, se realiza un ajuste previo a la evaluación sobre un conjunto de muestras, denominado *tuning*, para determinar qué modelo y parámetros emplear sobre la partición de test.

Además, destacar que, no se han realizado las experimentaciones considerando únicamente los modelos Word2Vec para obtener representaciones de palabras y frases, sino que se han utilizado también propuestas alternativas como la ontología de DBpedia. En esos casos, el protocolo expuesto no es necesario debido a que no se requiere ajustar la dimensión de los vectores de representación, al venir prefijada por la talla de la ontología, ni otros parámetros propios de los modelos obtenidos con Word2Vec. A pesar de esto, sí se deben probar diversos métodos de representación de frases y varios clasificadores, ajustando correctamente sus parámetros.

Finalmente, por todo ello, el esquema expuesto en este apartado solo se utiliza para permitirnos reducir la explosión combinatoria que supone el entrenamiento de modelos Word2Vec en los problemas de identificación del idioma, análisis de sentimiento y detección de *malware*, quedando relegado del problema de detección de la temática donde se sigue otro método diferente, llevando a cabo un barrido más o menos sistemático de los parámetros de los modelos empleados, tal como indica la Figura 19.

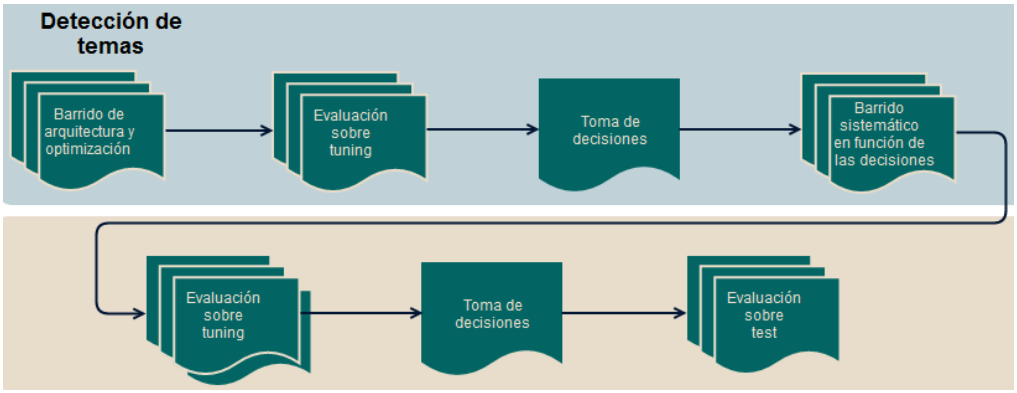

Fig. 19. *Workflow* para detección de temas

4.9 Validación

Una de las etapas más importantes en la implementación de un sistema de aprendizaje automático es la evaluación, por esa razón, se ha considerado de suficiente relevancia como para exponerla en detalle en el presente documento. Todo el contenido ha sido sintetizado de [24] y [46].

Sea $f: \mathcal{X} \to \mathcal{Y}$ el modelo aprendido mediante algún sistema de aprendizaje automático, la probabilidad de error de f es la esperanza estadística, sobre todas las muestras posibles, de que la salida $f(x)$ sea incorrecta. Supongamos que $\mathcal{X}$ es un dominio discreto y sea $P(x)$ la probabilidad de aparición *a priori* de una muestra $x \in \mathcal{X}$, la esperanza estadística de error de f es: $E[error(f)] = \sum_{x \in \mathcal{X}} error\big(f(x)\big) \cdot P(x)$.

Sin embargo, a pesar de que $E[error(f)]$ es la probabilidad de error exacta, no se suele conocer $P(x)$ o solo se conocen aproximaciones que no permiten calcular $E[error(f)]$. Para subsanar esto, en la práctica, es común estimar $E[error(f)]$ mediante datos de test etiquetados de igual forma que los datos de entrenamiento, pero no usados durante la etapa de aprendizaje del sistema de *machine learning*.

Haciendo uso de dichos datos de entrenamiento, es posible aproximar de forma empírica la verdadera esperanza de error de un sistema $p = E[error(f)]$, contabilizando el número de errores de decisión, N_e, que se producen en un conjunto de N muestras de test: $p' = \frac{N_e}{N}$. Si N es muy grande, p' se distribuye según una distribución normal $p' \sim \mathcal{N}(p, \frac{p(1-p)}{N})$ y por tanto p' se aproxima mucho a p (verdadera esperanza de error), por lo que cuantas más muestras de test se usen mucho mejor se estimará $E[error(f)]$.

Si según la conclusión anterior, cuantas más muestras de entrenamiento y de test se empleen, mejor se estima la esperanza de error, surge un problema, la correcta gestión del conjunto finito de datos disponible. Por este motivo, hay que dividir de forma razonable el conjunto de muestras en dos subconjuntos (entrenamiento y test) y para ello existen diversas formas de realizar dicha división:

- **Resustitución**: todos los datos disponibles se utilizan tanto para entrenamiento como para test. Siendo N la talla de entrenamiento y N_e el número de errores, la tasa de error r_e se calcula como $r_e = \frac{N_e}{N}$. El principal problema de este criterio es el optimismo.

- **Partición**: las muestras se dividen en un subconjunto para entrenamiento y otro para test. Sea N el cardinal del conjunto de muestras disponible, N' el cardinal del subconjunto de test, $N - N'$ la talla de entrenamiento y N'_e el número de errores de test, la tasa de error r_e es de la forma $r_e = \frac{N'_e}{N'}$. Como inconveniente, señalar el desaprovechamiento de datos.

- **Validación cruzada en B bloques**: los vectores se dividen aleatoriamente en B bloques y cada uno se utiliza como test para un sistema entrenado con el resto de bloques. En la Figura 20, se representa el caso $B = 4$.

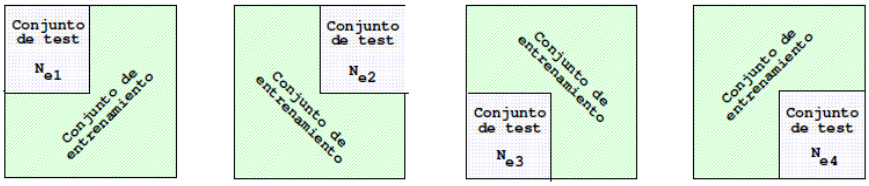

Fig. 20. Validación cruzada en B bloques con $B = 4$ [24]

La tasa de error se obtiene mediante la expresión $r_e = \frac{\sum_{i=1}^{B} N_{ei}}{N}$. En este caso, el coste computacional se incrementa con B.

- **Exclusión individual**: caso particular de validación cruzada en B bloques donde $B = 1$, i.e. cada dato individual se utiliza como dato único de test de un sistema entrenado con los $N - 1$ datos restantes. Al ser B mínimo, el coste computacional es máximo.

Por otro lado, con respecto a la evaluación de los clasificadores en los problemas de clasificación en múltiples clases, hacemos uso de la medida conocida en inglés como *accuracy*, sin embargo, en la memoria empleamos el término «precisión» como traducción de dicho término inglés. Esta métrica establece una relación entre el número de muestras correctamente predichas y el número total de muestras, definiéndose también como $1 - error$.

Además, en los problemas de clasificación binaria es posible utilizar otras métricas como verdaderos positivos (tp), falsos positivos (fp), falsos negativos (fn) y verdaderos negativos (tn). Un ejemplo de estas métricas en el problema de clasificación en dos clases («Recuperado» y «Relevante») se puede observar en la Figura 21.

	Relevante	No relevante
Recuperado	true positives (tp)	false positives (fp)
No Recuperado	false negatives (fn)	true negatives (tn)

Fig. 21. Definición de métricas de RI [46]

Estas métricas, nos permiten a su vez obtener otras mediciones como la precisión $p = \frac{tp}{tp+fp}$, distinta de *accuracy*, mide la fracción de instancias recuperadas que son relevantes; la cobertura $r = \frac{tp}{tp+fn}$, encargada de determinar las instancias relevantes que han sido recuperadas y la f-medida $f_\beta = \frac{(\beta^2+1)\cdot P\cdot R}{\beta^2\cdot P+R}$, que pondera la precisión y la cobertura mediante un parámetro β , entre otras.

Terminando, todas estas métricas han sido implementadas en los módulos de evaluación del sistema desarrollado, considerando la precisión (accuracy) en los problemas de clasificación en múltiples clases y las métricas de clasificación binaria en los problemas que consideren únicamente dos clases, como la detección de *malware*.

«Usamos los números todos los días, para predecir el tiempo, para decir la hora, al usar dinero. También los usamos para analizar el crimen, para buscar pautas y para predecir comportamientos. Con los números podemos solucionar los mayores misterios que se nos planteen.»

-Introducción de la serie Numbers

Capítulo 5

Identificación de temática

Según G. Brown y G. Yule en [47] el término *tema* se define como aquello de lo que se habla o escribe. La definición es clara e intuitiva, sin embargo, la complejidad de precisar un comportamiento formal para la identificación de la temática de un documento o conjunto de documentos es muy elevada (NP-duro según [48]). Así, en el presente capítulo, proponemos la utilización, evaluación y comparación de representaciones vectoriales de palabras y frases para dar solución al problema de detección de temática en diferentes corpus mediante diversas experimentaciones.

5.1 Introducción a la identificación de temática

La construcción de etiquetas que identifiquen la idea subyacente de un conjunto de documentos se denomina identificación de la temática (*topic detection* en el idioma inglés). Este tipo de sistemas cobra vital importancia en la tarea de clasificación y categorización de recursos en aplicaciones de búsqueda, proveyendo facilidades en cuestiones relativas a la recuperación de información, donde un gran conjunto de entidades debe ser retornado tras la realización de consultas especificadas por el usuario [49].

Las contribuciones del presente documento al área de identificación de temática son dos: la demostración de la utilidad de modelos Word2Vec en problemas de detección de temas y la propuesta de soluciones alternativas basadas en representaciones vectoriales obtenidas de la ontología de DBpedia. Además, experimentamos con las propuestas mencionadas sobre el corpus EPPS en español del proyecto TC-STAR, ampliamente utilizado en sistemas de reconocimiento, traducción y síntesis del habla, nos permitirá comparar los resultados con otras investigaciones de la misma índole [5] que hacen uso de dicho corpus.

A pesar de que normalmente se consideran enfoques no supervisados para la detección de temas, que permiten construir categorías al vuelo sin necesidad de definir previamente los temas ni de hacer uso de un conjunto de datos de entrenamiento, este proyecto abarca los dos planteamientos posibles: sistemas supervisados y no supervisados. Para ello, hacemos uso, como ya se ha mencionado, del corpus EPPS ya utilizado en otras investigaciones sobre detección de la temática como [5].

Finalmente, como resumen, empleamos y evaluamos representaciones vectoriales de palabras y frases, obtenidas mediante Word2Vec y DBpedia, aplicadas al problema de detección de temas subyacentes en documentos, siendo estos sistemas de gran utilidad en aplicaciones dedicadas a la búsqueda de recursos en grandes colecciones de documentos, permitiendo realizar así búsquedas más específicas, focalizadas únicamente en las categorías relevantes para una determinada consulta dada por parte del usuario.

5.2 Corpus EPPS TC-STAR

Las siguientes experimentaciones se realizan, tal como se ha expuesto en el apartado anterior, con el corpus EPPS del proyecto TC-STAR. Además, debido a que únicamente la partición de entrenamiento original se encuentra supervisada con los temas de cada sesión parlamentaria, todos los conjuntos necesarios de muestras, para evaluación y entrenamiento, han sido extraídos por los autores de [5] a partir de dicha partición.

El dominio de dicho corpus trata intervenciones del Parlamento Europeo en español y está formado por 15 399 participaciones para entrenamiento y 3 738 para test, de interventores tanto masculinos como femeninos (en proporción de $75\,\% - 25\,\%$), existiendo un total de 79 temas para evaluación y 73 para entrenamiento. Debido a esta diferencia entre el número de temas en las particiones del corpus, se han eliminado las categorías que están en el conjunto de test y no aparecen en entrenamiento, así como descartado los temas que no contienen muestras en la partición de evaluación. Con ello, se reduce el número de temas y frases en el conjunto de test a 61 y 3 417 respectivamente.

Uno de los aspectos a considerar es la complejidad del corpus. Como se puede observar en todas las figuras en las que se representa la distribución de las muestras de entrenamiento, los vectores de representación, tanto de espacio continuo como discreto, están muy solapados en el espacio, lo que complica la clasificación. Además, al tratar con 73 clases la tarea de detección de temática es todavía más compleja y debido a la limitada extensión de las figuras mostradas en la memoria, únicamente se indican en la leyenda las 23 primeras clases (to001, to002, etc.).

Otro aspecto clave está relacionado con la longitud de las frases. La identificación de temas en frases cortas puede ser compleja y ambigua debido a que una cantidad reducida de palabras no proporciona suficiente información semántica acerca del concepto tratado. Por ello, los autores del corpus decidieron, en la partición de evaluación, agrupar segmentos de duración menor a un minuto para formar intervenciones de mayor duración que solucionen los problemas de ambigüedad.

Así, siguiendo el criterio de agrupación de segmentos de reducida duración para el conjunto de evaluación, se plantean dos posibles agrupamientos con los que se obtienen 239 y 694 muestras de test. Sin embargo, únicamente se utilizará el primer conjunto, formado por 239 muestras de test, para llevar a cabo la evaluación del sistema que se expondrá en los siguientes apartados del presente capítulo dedicado a la identificación de la temática.

Finalmente, de cada categoría de entrenamiento se extrae un 20 % de las muestras, configurando así un conjunto de *tuning*, tal como se indicó en el apartado cuatro del capítulo uno, con el que determinar qué modelos emplear para llevar a cabo la evaluación sobre el conjunto de test. Con ello, el corpus EPPS tratado, queda segmentado en tres conjuntos, pudiéndose visualizar las características de dicho corpus en la Tabla I.

TABLA I
PROPIEDADES DEL CORPUS EPPS

Idioma	Español
Dominio	Intervenciones parlamentarias
Talla del vocabulario	18 780 palabras
Temas de entrenamiento	73
Temas de *tuning*	73
Temas de test	61
Conjunto de entrenamiento	12 292 frases
Conjunto de *tuning*	3 107 frases
Conjunto de test original	3 417 frases
Conjunto de test 1	3 417 frases agrupadas en 239 segmentos

5.3 Corpus Wikipedia

Aunque el entrenamiento y la evaluación de los clasificadores se realizan con las intervenciones del Parlamento Europeo presentes en el corpus EPPS, también se hace uso de un corpus formado por artículos de Wikipedia en español, extraído de [50], para obtener representaciones vectoriales continuas de palabras mediante Word2Vec.

Estos artículos están representados en *eXtensible Markup Language* (XML) y comprimidos con *GNU Zip* (gzip) por lo que es necesario, tras la descompresión, analizar el fichero XML para extraer los artículos en texto plano. Para ello, en el sistema desarrollado, hacemos uso de la clase *WikiCorpus* de Gensim y su método *get_texts* [51] que nos permite iterar sobre todos los textos del corpus.

De forma más detallada, el corpus contiene 1 060 000 artículos de Wikipedia en español que suponen un total de 542 283 968 palabras y 3 439 947 palabras diferentes, almacenadas en disco con un fichero de 3.24 GB. Finalmente, es posible observar las propiedades, ya mencionadas, de dicho corpus en la Tabla II.

TABLA II
PROPIEDADES DEL CORPUS WIKIPEDIA

Idioma	Español
Dominio	Wikipedia
Número de artículos	1 060 000 artículos
Talla del vocabulario	3 439 947 palabras
Número de palabras	542 283 968 palabras
Tamaño del corpus	3.24 GB

5.4 Experimentación con DBpedia

La primera experimentación sobre el corpus EPPS se ha realizado empleando las representaciones vectoriales discretas de palabras y frases obtenidas mediante la ontología del proyecto DBpedia detallado en el capítulo tres. Además, tal como se indicó durante la explicación del protocolo de experimentación, hacemos uso del método de partición para evaluar la calidad de estas representaciones contra el conjunto de test original, siendo inviables métodos como exclusión individual debido a su alto coste temporal.

Como primer paso, se realiza un preproceso sobre las muestras de entrenamiento y de test (en este caso no se emplea el conjunto de *tuning*), eliminando símbolos no alfanuméricos, no contemplados en la ontología, y convirtiendo los términos que representen entidades al formato de DBpedia. Para la detección de entidades se ha usado un método que consiste en juntar palabras adyacentes que comiencen por mayúscula. Un ejemplo de esta conversión es el paso de «Mariano Rajoy» a «Mariano_Rajoy».

Una vez realizado el preproceso mencionado, se obtienen las representaciones vectoriales siguiendo los métodos booleano y aditivo expuestos en el apartado cinco del capítulo tres. Con ello, podemos observar la distribución de las muestras de entrenamiento, cuya dimensión viene determinada por la talla de la ontología de DBpedia, normalizadas con la norma L_2 y proyectadas en dos dimensiones con *Principal Component Analysis* (PCA) según el método booleano en la Figura 22 y en función del método aditivo en la Figura 23.

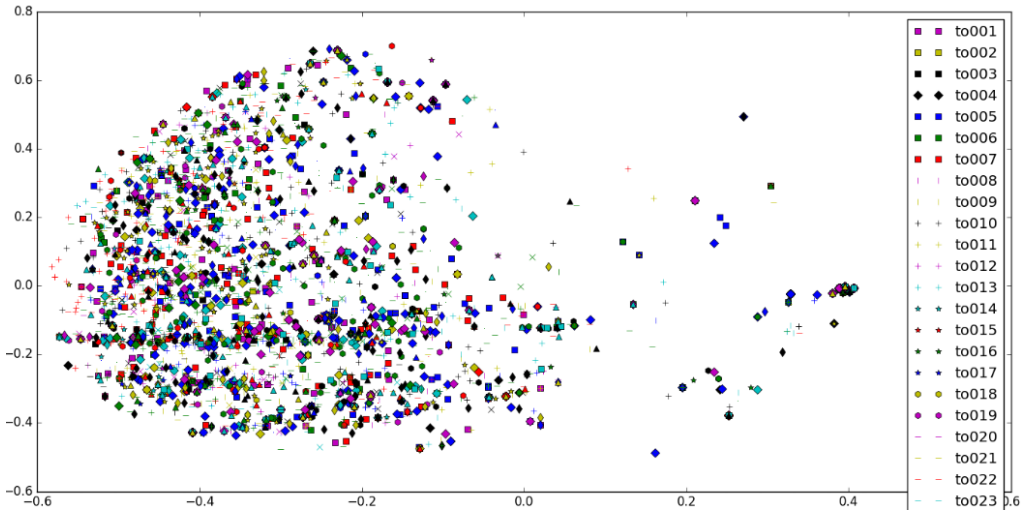

Fig. 22. Distribución de muestras de entrenamiento con método booleano

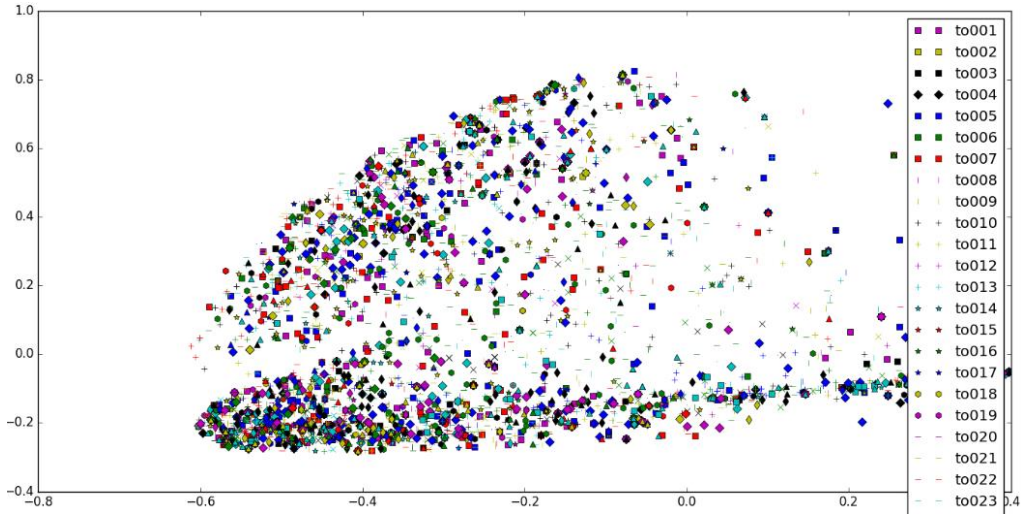

Fig. 23. Distribución de muestras de entrenamiento con método aditivo

Así, partiendo de los vectores de representación obtenidos para las muestras del conjunto de entrenamiento, podemos entrenar los clasificadores, detallados en el capítulo cuatro, para generalizar la distribución de dichos ejemplos e intentar predecir la salida de las muestras del conjunto de evaluación o test 1. Tras dicha evaluación se han obtenido los resultados expuestos en las tablas III y IV que se corresponden con los métodos booleano y aditivo respectivamente.

TABLA III
RESULTADOS DE EVALUACIÓN CON MÉTODO BOOLEANO

Clasificador	Precisión
Naive Bayes (*Gaussian)*	10.0%
Naive Bayes (*Bernoulli)*	6.1%
Árbol de decisión	**10.3%**
KNN (centroide, distancia correlación)	11.0%
KNN (centroide, distancia coseno)	3.5%
SVM (*kernel* lineal)	3.7%
SVM (*kernel* RBF)	4.3%
SVM (*kernel* sigmoide)	2.5%

TABLA IV
RESULTADOS DE EVALUACIÓN CON MÉTODO ADITIVO

Clasificador	Precisión
Naive Bayes (*Gaussian)*	10.0%
Naive Bayes (*Bernoulli)*	6.1%
Árbol de decisión	**10.3%**
KNN (centroide, distancia correlación)	3.2%
KNN (centroide, distancia coseno)	3.2%
SVM (*kernel* lineal)	3.7%
SVM (*kernel* RBF)	4.3%
SVM (*kernel* sigmoide)	2.5%

Como podemos observar, los resultados no varían prácticamente entre los dos métodos empleados debido al reducido número de ocurrencias de entidades en el corpus EPPS tratado, presentes en la ontología de DBpedia. Además, es posible detectar una precisión muy baja, entre un 2.5 % y un 10.3 %, por el motivo mencionado junto con otros inconvenientes como el solapamiento entre clases y la problemática de contar con la lista de propiedades *P* en el idioma inglés al no haberse podido conseguir en español.

Por otro lado, cabe destacar que aunque en la presente experimentación el comportamiento de este tipo de representaciones discretas no ha sido del todo correcto, es posible conseguir un mejor funcionamiento en corpus donde hay una gran cantidad de entidades no ambiguas consideradas en la ontología de DBpedia, habiéndose comprobado esto sobre un corpus de documentos en inglés elaborado a mano para el proyecto y cuyas experimentaciones se han descartado de la memoria, donde se obtiene un máximo del 65.7 % de precisión empleando SVM.

Por último, dicho corpus contiene documentos pertenecientes a tres temas o categorías diferentes (política, deportes y social), extraídos aleatoriamente de diversos medios electrónicos de contexto periodístico. Así, al tratar dominios en los que la ocurrencia de entidades es significativa y, además, el idioma coincide con el de la lista de propiedades de la ontología de DBpedia, los resultados obtenidos son mejores que con el corpus empleado en este apartado. Las características de este corpus se pueden observar en la Tabla V.

TABLA V
PROPIEDADES DEL CORPUS TOPICS

Idioma	Inglés
Dominio	Artículos periodísticos
Talla del vocabulario	3 116 palabras
Temas de entrenamiento	3
Temas de test	3
Conjunto de entrenamiento	132 documentos
Conjunto de test	81 documentos

5.5 Experimentación con Word2Vec

Mediante el apartado actual tratamos las experimentaciones que, con mayor profundidad, se han llevado a cabo para ofrecer soluciones, basadas en vectores de representación de espacio continuo para palabras y frases, al problema de detección de temática sobre el corpus EPPS del proyecto TC-STAR.

En primer lugar, debido a la complejidad del aprendizaje de modelos Word2Vec a partir de artículos de Wikipedia en español y del corpus del problema, se ha realizado un barrido inicial que nos permite determinar qué arquitectura y método de optimización emplear para, posteriormente, llevar a cabo un barrido más sistemático con el que decidir qué dimensión de vector y tamaño de ventana usar en el entrenamiento de los modelos.

Más concretamente, en dicho barrido inicial, se fijan algunos valores por defecto de Word2Vec, $d = 100$, $k = 5$ y se varía la arquitectura $a \in \{CBOW, SG\}$, el método de optimización $o \in \{NS, HS\}$, la técnica de representación de frases $f \in \{suma, centroide\}$ y el clasificador empleado C_i, $1 \leq i \leq 10$, evaluando las representaciones vectoriales obtenidas contra la partición de *tuning*. Los resultados de la evaluación, realizada con los diferentes clasificadores sobre la partición de *tuning* mencionada, se pueden observar en la Tabla VI para el caso de los modelos obtenidos a partir del corpus y en la Tabla VII para los modelos generados utilizando artículos de Wikipedia.

TABLA VI
PRECISIÓN EN TUNING CON MODELOS DEL CORPUS EPPS

Modelo	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
SGHS suma	**26.1%**	17.2%	7.7%	12.3%	20.9%	22.3%	15.5%	15.6%	16.4%	23.1%
SGNS suma	**22.3%**	11.1%	3.6%	9.7%	16.6%	16.4%	8.3%	8.3%	10.0%	16.3%
CBHS suma	12.5%	10.1%	4.0%	9.3%	**16.9%**	**16.9%**	7.4%	7.4%	10.7%	15.5%
CBNS suma	**13.5%**	7.6%	1.1%	7.0%	10.2%	10.9%	2.5%	2.4%	6.2%	11.4%
SGHS centroide	**26.2%**	17.1%	11.1%	11.4%	23.7%	25.5%	14.3%	14.3%	24.2%	11.5%
SGNS centroide	**19.9%**	11.1%	6.4%	10.5%	18.3%	19.4%	6.5%	6.5%	13.6%	11.4%
CBOW HS centroide	**19.7%**	10.1%	5.2%	9.7%	17.1%	18.8%	5.6%	5.6%	13.8%	11.4%
CBOW NS centroide	**12.6%**	7.6%	2.8%	7.0%	10.6%	11.8%	1.5%	1.5%	7.7%	9.4%

De izquierda a derecha los clasificadores utilizados C_1, ..., C_{10}, son: SVM lineal, *naive Bayes* (*Bernoulli*), *naive Bayes (Gaussian)*, árbol de decisión, *random forest* con 50 estimadores, ran*dom forest* con 100 estimadores, *nearest centroid* con distancia coseno, *nearest centroid* con distancia correlación, KNN $k = 1$ y SVM con *kernel* rbf. En la tabla siguiente se mantiene la misma notación.

TABLA VII
PRECISIÓN EN TUNING CON MODELOS DE WIKIPEDIA

Modelo	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
SGHS suma	**25.6%**	12.6%	5.4%	8.8%	15.5%	16.9%	12.1%	12.3%	15.1%	18.4%
SGNS suma	**25.8%**	13.1%	5.1%	7.8%	15.3%	17.2%	13.0%	13.0%	14.8%	18.3%
CBHS suma	**25.2%**	14.0%	7.6%	7.8%	15.4%	17.7%	12.5%	12.6%	17.4%	18.7%
CBNS suma	**25.1%**	13.7%	7.6%	8.1%	15.1%	17.8%	11.8%	11.9%	17.6%	18.6%
SGHS centroide	22.2%	12.6%	10.9%	8.4%	18.2%	19.5%	12.0%	12.6%	**24.1%**	9.0%
SGNS centroide	22.5%	13.1%	10.5%	8.8%	18.2%	20.8%	12.3%	12.2%	**25.3%**	9.0%
CBOW HS centroide	22.3%	14.0%	9.8%	8.7%	18.0%	19.1%	12.5%	12.4%	**24.1%**	9.0%
CBOW NS centroide	22.1%	13.7%	9.6%	8.2%	18.3%	20.1%	11.4%	11.5%	**22.8%**	9.0%

Así, en las tablas VI y VII, es posible observar que para los modelos Word2Vec generados a partir de las muestras de entrenamiento del corpus EPPS, la combinación que mejores resultados ofrece es SG con HS (precisión del 26.2 %) mientras que en el caso de los modelos obtenidos haciendo uso de Wikipedia, SG con NS muestra un mejor comportamiento (25.8 % de precisión). Por ello, se toma la decisión de fijar dichos parámetros para hacer un barrido más sistemático, considerando únicamente los clasificadores SVM lineal y KNN con $k = 1$, debido a que funcionan mejor que los demás, y las dos representaciones de frases (suma y centroide).

En este siguiente barrido sistemático, se varían los parámetros relacionados con la dimensión de los vectores de representación y con el tamaño del contexto a tener en cuenta para cada término, realizando, también en este caso, la evaluación contra el *tuning*. Para los modelos obtenidos a partir de EPPS, consideramos como valores de dimensión: 25, 50, 100, 150 y 200; y como tamaños de ventana: 5, 10 y 20. Para el caso de Wikipedia, los valores de dimensión considerados son: 100, 200, 300, 400, 500, 600, 700 y 800, mientras que, como tamaño de ventana se comprueban: 5, 10, 20 y 50.

Como se puede ver, es necesaria una gran capacidad de cómputo para poder entrenar los modelos mencionados en un tiempo coherente con la duración del proyecto. De este modo, para los modelos obtenidos a partir del corpus EPPS se ha hecho uso de una máquina con procesador Intel® Core™ i7-2670QM, 8 núcleos y 8 GB de RAM tomando aproximadamente 20 minutos de procesamiento. Con respecto a los modelos de Wikipedia, hemos empleado tres máquinas del departamento con procesadores Intel® Xeon® CPU E5-1660 v2 @ 3.70 GHz y Xeon® CPU E5-1620 v3 @ 3.70 GHz, 12 y 8 núcleos y 64 GB de RAM durante una semana, entrenando de forma paralela hasta cuatro modelos al mismo tiempo.

Con ello, se han elaborado ocho gráficas $3D$ en las que se representan los valores de precisión para cada combinación de dimensión, ventana, arquitectura, método de optimización y clasificador empleado, obtenidos mediante el barrido de parámetros ya mencionado, evaluando los modelos aprendidos contra el conjunto de *tuning*. Las cuatro primeras figuras se corresponden con los modelos extraídos de EPPS, mientras que las cuatro últimas hacen referencia a los modelos generados a partir de Wikipedia.

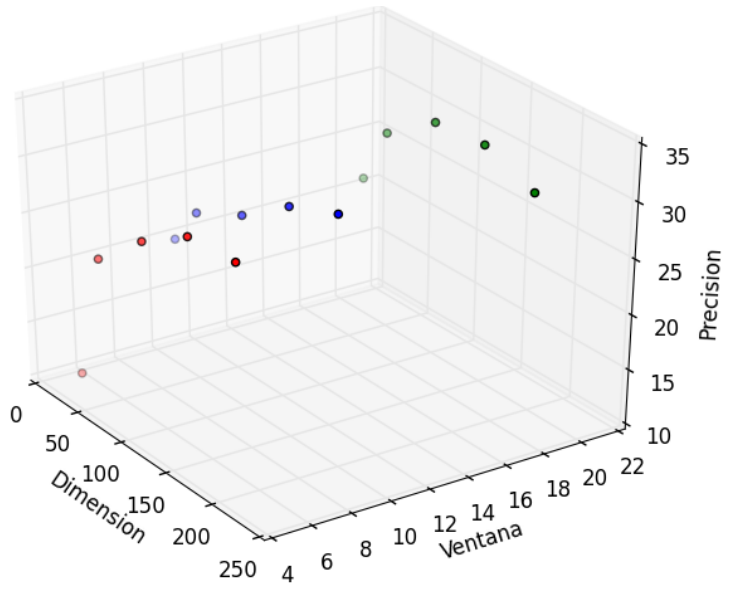

Fig. 24. Precisión obtenida en tuning con EPPS, SVM y representación suma

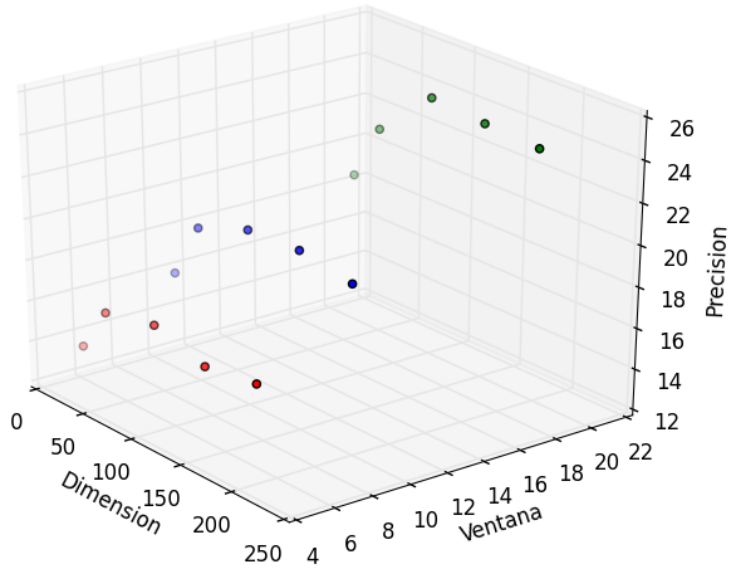

Fig. 25. Precisión obtenida en tuning con EPPS, 1NN y representación suma

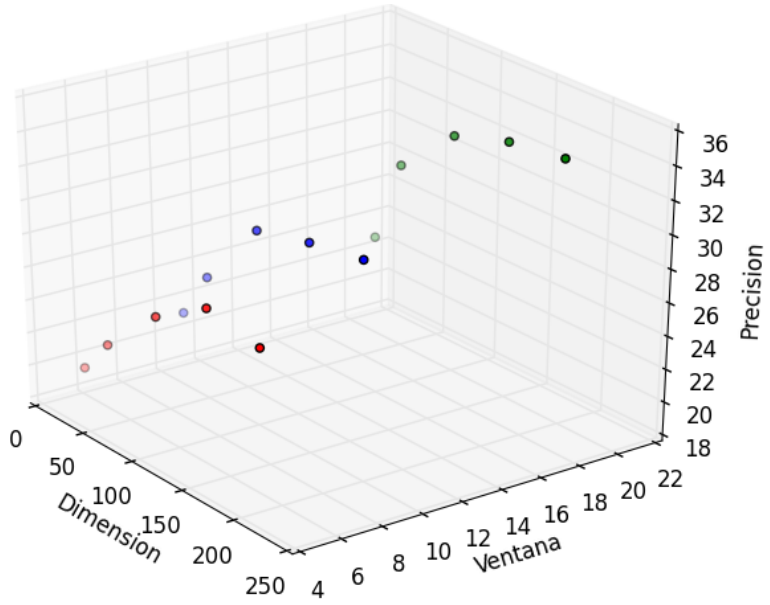

Fig. 26. Precisión obtenida en tuning con EPPS, SVM y representación centroide

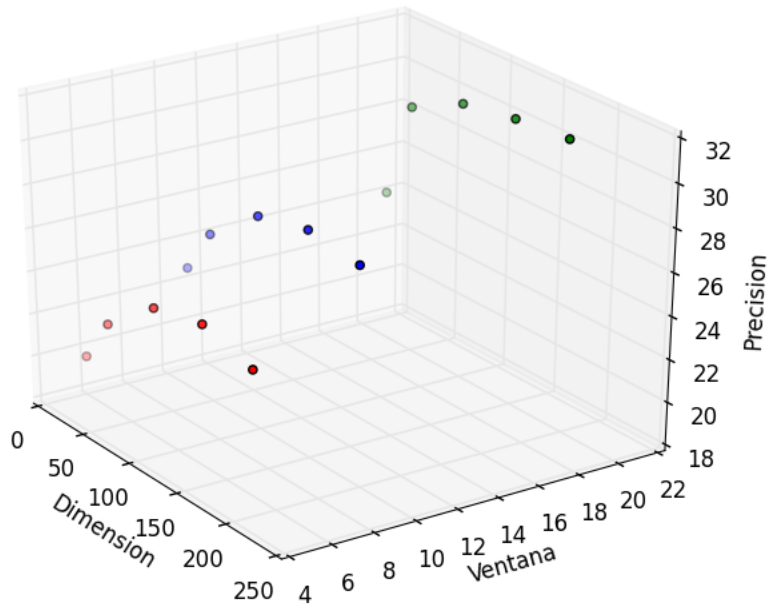

Fig. 27. Precisión obtenida en tuning con EPPS, 1NN y representación centroide

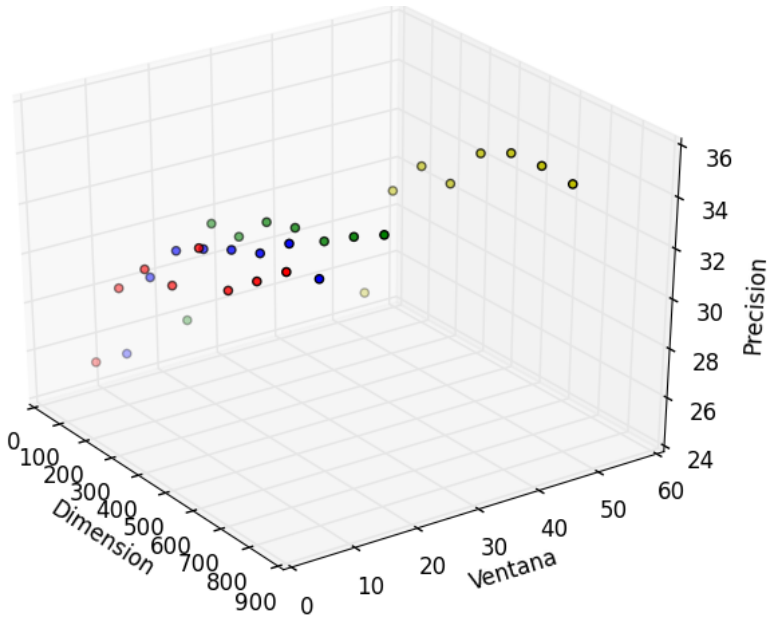

Fig. 28. Precisión obtenida en tuning con Wikipedia, SVM y representación suma

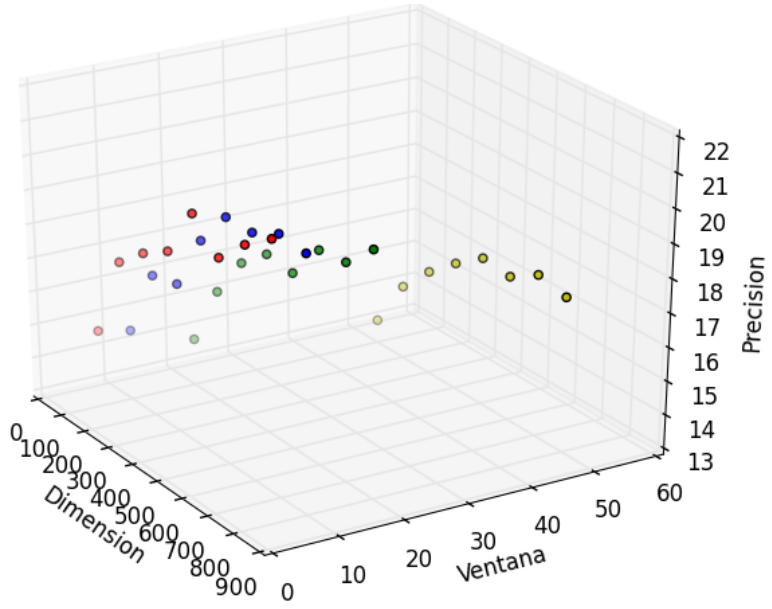

Fig. 29. Precisión obtenida en tuning con Wikipedia, 1NN y representación suma

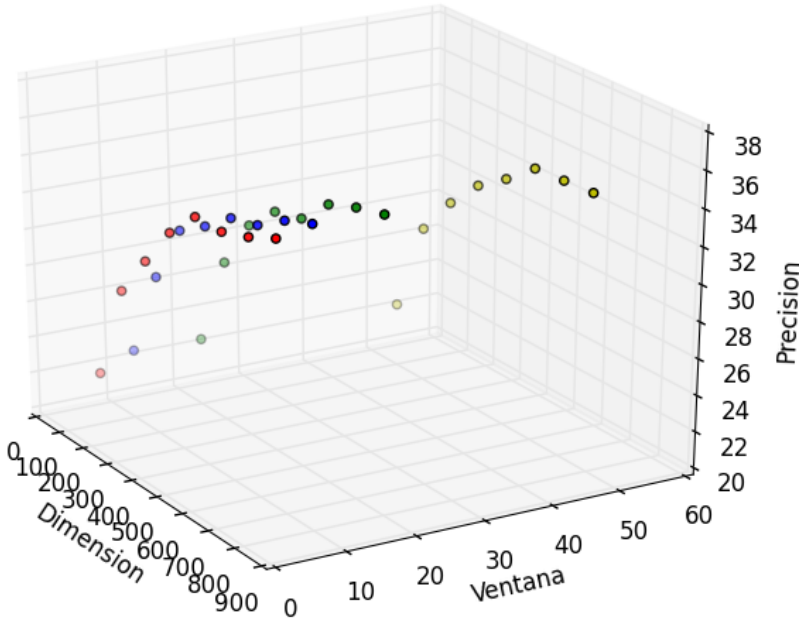

Fig. 30. Precisión obtenida en tuning con Wikipedia, SVM y representación centroide

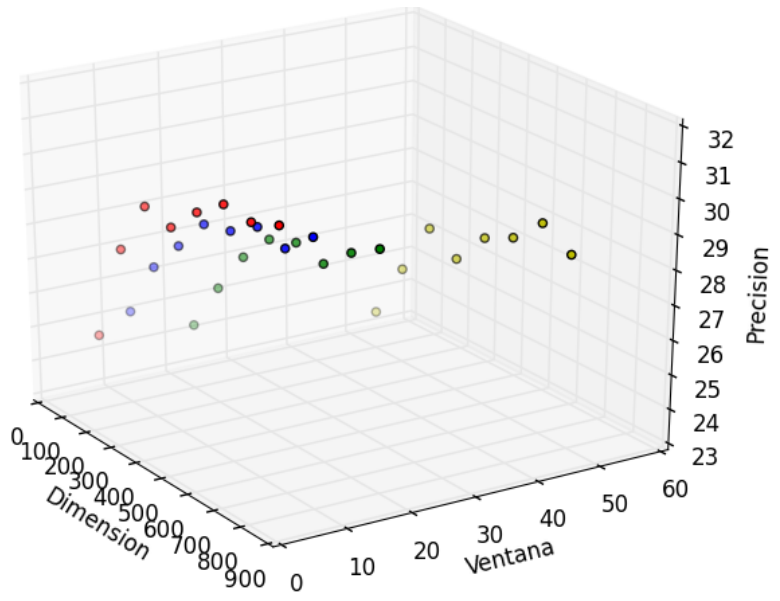

Fig. 31. Precisión obtenida en tuning con Wikipedia, 1NN y representación centroide

Antes de realizar consideraciones sobre qué modelo tiene un mejor comportamiento para evaluar sobre la partición de test, es importante notar que en los modelos obtenidos con las muestras de entrenamiento del corpus EPPS, un incremento en el tamaño de la ventana fijando la dimensión de los vectores de representación, generalmente, supone un aumento en la precisión obtenida. Ocurre lo mismo, hasta cierto valor, con un incremento en la dimensión de los vectores, cosa que podemos notar en las cuatro primeras figuras anteriores si observamos la distribución del valor precisión en función de la ventana y la dimensión a lo largo de los ejes X e Y.

Sin embargo, no ocurre lo mismo con los modelos generados a partir de Wikipedia, donde un incremento en el tamaño de la ventana no supone, con tanta frecuencia como en el caso anterior, un aumento en la precisión del sistema. Además, se puede observar cómo la dimensionalidad de los vectores tiene un efecto de mayor importancia que el tamaño de contexto sobre el valor de precisión, al obtener, habitualmente, un acrecentamiento en la precisión mediante el incremento de dicho parámetro. Así, en principio, si continuamos generando modelos que amplíen tanto el tamaño de contexto en EPPS como la dimensión de los vectores en el caso de Wikipedia, podemos incrementar la precisión hasta alcanzar un determinado máximo.

Con respecto a la elección de modelos, según los resultados conseguidos en el último *tuning,* los que exhiben un mejor comportamiento son SGHS, $d = 200$, $k = 20$ con representación centroide (44.76 % de precisión) en el caso de EPPS y SGNS, $d = 800$, $k = 10$ con representación centroide (precisión del 51.88 %) para Wikipedia, por lo que estamos en disposición de realizar la evaluación final sobre la partición de test haciendo uso de dichos modelos. Además, en esta etapa de toma de decisiones, hemos podido corroborar una de las hipótesis de T. Mikolov en [3], según la cual, la arquitectura SG combinada con el método de optimización NS ofrece vectores de representación de mayor calidad, con corpus de gran tamaño, que los obtenidos con otras combinaciones posibles de arquitectura y optimización.

Con ello, ya es posible realizar la evaluación contra las muestras del conjunto de test haciendo uso de los mejores modelos obtenidos y ya estudiados, sin embargo, con la intención de ampliar la experimentación sobre la partición de test se ha incluido la representación con suma en la evaluación. Así, se indica en la Tabla VIII la precisión obtenida tras llevar a cabo dicha evaluación, haciendo uso únicamente de SVM se puede observar cómo con los modelos generados a partir de la Wikipedia y representación suma se obtienen los mejores resultados.

TABLA VIII
RESULTADOS DE LA EVALUACIÓN CONTRA EL TEST EN EPPS

Modelo	Precisión con suma	Precisión con centroide
EPPS	44.7%	42.2%
Wikipedia	**69.9%**	51.8%

Si nos fijamos en los valores de precisión obtenidos mediante la evaluación contra el conjunto de test y realizamos una comparación con los resultados obtenidos durante la etapa de evaluación en la partición de *tuning*, es posible contemplar una gran diferencia o gap entre dichos valores de precisión. Esto puede ser debido, entre otros motivos que no se han logrado determinar, a que las muestras de test han sido agrupadas siguiendo el criterio mencionado en el segundo apartado del presente capítulo, mientras que en la partición de *tuning* no se ha realizado unión alguna de las muestras.

Por otro lado, uno de los puntos importantes de este apartado consiste en realizar una comparación con los resultados obtenidos en la misma tarea por investigadores de la Universidad Politécnica de Madrid (UPM) en el artículo [5]. Para dicha tarea, utilizan aproximaciones basadas en modelo de espacio vectorial, *latent semantic analysis* y TF-IDF sobre el corpus EPPS preprocesado, haciendo uso en algunas experimentaciones de *stemming* y de varias listas de *stopwords*. Sus resultados vienen dados en la Figura 32.

Topic identification approach	T.I.E. for *Set 1*
GVM + TF–IDF + SW (*List-1*) – Baseline	35.71 ± 5.91
GVM + TF–IDF + SW (*List-2*)	34.13 ± 5.85
GVM + TF–IDF + SW (*List-2*) + Stemming	36.11 ± 5.93
GVM + TF-Entropy + SW (*List-2*)	34.52 ± 5.87
LSA + TF–IDF + SW (*List-1*)	32.54 ± 5.78
LSA + TF–IDF + SW (*List-2*)	30.56 ± 5.68
LSA + TF–IDF + SW (*List-2*) + Stemming	34.13 ± 5.85
LSA + TF-Entropy + SW (*List-2*)	30.16 ± 5.66

Fig. 32. Resultados en identificación de la temática por investigadores de la UPM [5]

Como es posible observar según la Figura 32, sus resultados vienen dados por la métrica *Topic Identification Error* (T.I.E), por tanto, para poder compararlos con los nuestros, es necesario considerarlos como precisión, $p = 1 - error$. Con ello, se identifica el mejor resultado obtenido con LSA, TF-*Entropy* y la lista 2 de *stopwords* alcanzando un 69.84 % de precisión. En nuestro caso, se ha obtenido una precisión de 69.95 %, por lo que, los resultados están en la misma línea y se deben realizar más experimentaciones para poder superarlos, tal como se verá en próximos apartados donde se propone una mejora en la utilización de modelos Word2Vec que alcanza una precisión del 80.3 %.

Como resumen, se han empleado representaciones vectoriales de espacio continuo de palabras y frases para proponer soluciones al problema de detección de temática, se han comprobado algunas de las hipótesis establecidas por los autores de Word2Vec en [3] y se ha alcanzado una precisión máxima de 69.95 % tras la realización de diversos barridos que nos han permitido aproximar los parámetros condicionantes de un buen funcionamiento de los modelos.

5.6 Experimentaciones adicionales

Además de las experimentaciones realizadas en el apartado anterior, se proponen experimentos adicionales sobre el problema de detección de la temática. Concretamente, en los próximos apartados planteamos la evaluación de modelos Word2Vec alternativos, obtenidos a partir de artículos en español de Wikipedia en función de un compromiso entre las hipótesis expuestas en el apartado dedicado al protocolo de experimentación y el coste computacional que conlleva el entrenamiento; la combinación de dichos modelos mediante concatenación y ponderación, así como la extracción no supervisada de temas subyacentes en el conjunto de documentos del corpus EPPS mediante técnicas de *clustering*.

5.6.1 Modelos Word2Vec alternativos

En la presente subsección se plantea la utilización de algunos de los modelos descartados en el barrido realizado en el apartado cinco de este mismo capítulo. Mediante dicho barrido se habían considerado ciertas hipótesis que descartaban el empleo de modelos Word2Vec con arquitectura CBOW, es por ello que se han generado algunos modelos de este tipo para llevar a cabo experimentaciones que hagan uso de representaciones vectoriales continuas de palabras y frases obtenidas por medio de esta arquitectura.

Además, debido a que no se requiere tener en cuenta consideraciones para determinar qué modelos emplear porque ya están prefijados, la evaluación se realiza directamente sobre la partición de test, solo se usa el clasificador SVM, la representación suma y el método de optimización NS. De esta forma, mediante el fijado de los parámetros mencionados, que han demostrado tener un buen comportamiento en experimentos anteriores, reducimos el coste temporal y la complejidad de las experimentaciones.

Con respecto a los modelos Word2Vec con arquitectura CBOW, una justificación de su utilización, la precisión obtenida mediante SVM y una visualización gráfica, en dos dimensiones, de las muestras de entrenamiento representadas mediante vectores de espacio continuo generados por medio de dichos modelos, se destacan en la siguiente lista:

1. **Modelo CBOW, 800 dimensiones y tamaño de ventana 5:** se hace uso de este modelo porque consideramos la hipótesis según la cual una dimensión mayor resulta en vectores de mayor calidad [7] al mismo tiempo que se tiene en cuenta un compromiso entre la complejidad del aprendizaje y el coste temporal, es por ello que se ha utilizado un tamaño 5 de ventana.
2. **Modelo CBOW, 800 dimensiones y tamaño de ventana 50:** seguimos considerando la hipótesis que trata la calidad de las representaciones en función de la dimensión de los vectores, sin embargo, el tamaño de ventana se incrementa a expensas de un incremento en el coste temporal.
3. **Modelo CBOW, 800 dimensiones y tamaño de ventana 100:** se genera por la misma razón que el modelo anterior, con el objetivo de incrementar el conocimiento sobre el ámbito de cada término.
4. **Modelo CBOW, 500 dimensiones y tamaño de ventana 10:** en este caso no se considera ninguna de las hipótesis tenidas en cuenta en los modelos anteriores, reduciendo a 500 la dimensión de los vectores y a 10 el tamaño de contexto de cada término.

En la Tabla IX es posible visualizar los resultados obtenidos con los modelos mencionados, además, se grafica también como ejemplo la Figura 33, que representa la distribución de las muestras de entrenamiento del corpus EPPS, en dos dimensiones normalizadas con L_2, obtenidas mediante el modelo con el que se obtiene un mejor comportamiento según la tabla siguiente.

TABLA IX
PRECISIÓN CON MODELOS WORD2VEC ALTERNATIVOS

Modelo 1	Modelo 2	Modelo 3	Modelo 4
69.8%	70.7%	**71.1%**	65.2%

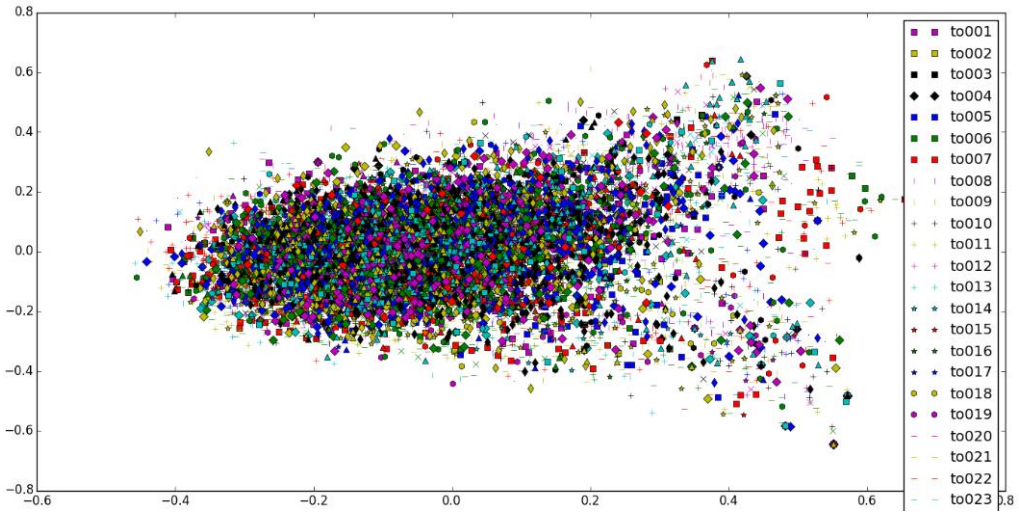

Fig. 33. Representación de muestras de EPPS con modelo tres

Finalmente, como se ha comprobado en la Tabla IX, los mejores resultados obtenidos en la experimentación se alcanzan mediante la utilización del modelo Word2Vec tres, sin embargo, debido a que se desea incrementar todavía más la precisión del método, en el próximo apartado se expone una vuelta de tuerca de la técnica básica empleada en todos los experimentos realizados hasta ahora, combinando y ponderando un conjunto de modelos Word2Vec.

5.6.2 Combinación de modelos Word2Vec

En el método básico para obtener representaciones vectoriales continuas de frases, empleado en todos los experimentos anteriores, se hace uso de un único modelo Word2Vec, sin embargo, es posible realizar una generalización de la técnica combinando N modelos para obtener dichas representaciones.

El procedimiento es muy simple, del mismo modo que con un único modelo se obtienen las representaciones de palabras y posteriormente se combinan mediante una función f para formar vectores que representen frases, con N modelos, podemos obtener la representación de una frase s como resultado de aplicar una función generalista a las representaciones vectoriales continuas de s generadas con cada uno de estos modelos. En nuestro caso, se utiliza la función de concatenación para obtener la representación de s. En la Figura 34 se puede observar la arquitectura de la nueva propuesta en comparación con el método básico.

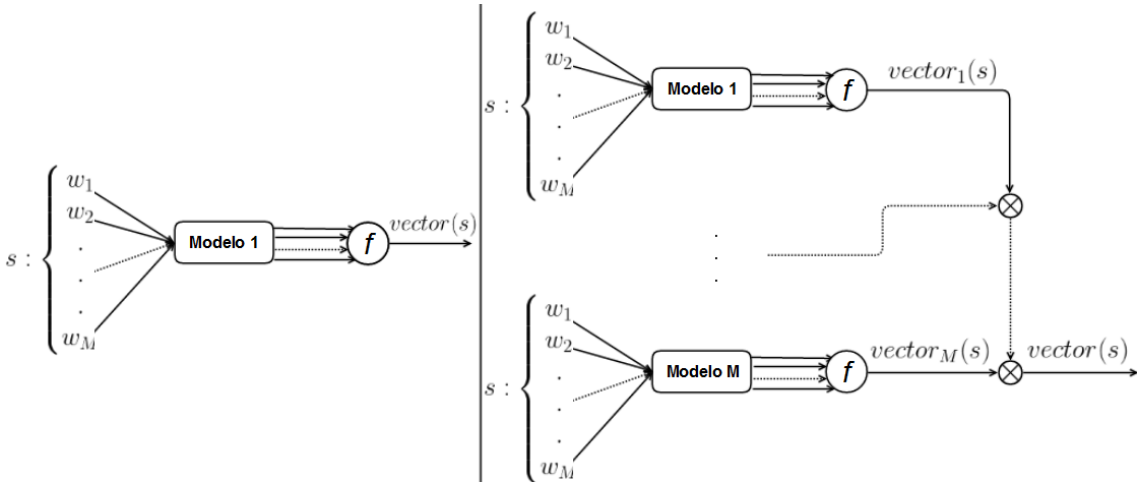

Fig. 34. Comparación entre método básico y combinación de modelos

Además, es posible extender la generalización ponderando cada vector de representación de s, $vector_i(s)$ $1 \leq i \leq M$, en función de algún criterio como, por ejemplo, priorizar un modelo en función del comportamiento que exhibe por separado en el método básico. Esto es, si el modelo m_i se comporta mejor que el modelo m_j en el método básico, este se pondera por un factor $k_i \in \mathbb{R}$ más elevado que el factor $k_j \in \mathbb{R}$ empleado para ponderar m_j. Nótese que el caso trivial de aplicar únicamente la operación sobre los modelos se da cuando $k_i = 1$ $1 \leq i \leq M$.

En lo relativo a la evaluación y los parámetros empleados para entrenar los diferentes modelos a utilizar, se realiza dicha tarea directamente sobre la partición de test usando SVM con representación suma, igual que en la experimentación del apartado anterior. Además, hay que tener en cuenta que se requeriría, mediante un proceso de barrido, comprobar una gran cantidad de combinaciones de ponderaciones de los M modelos, sin embargo, esto no es posible y por ello únicamente se expone un pequeño subconjunto de estas, establecido en función del comportamiento de cada modelo:

- **CBOW NS 800 dimensiones y tamaño de ventana 5, CBOW NS 800 dimensiones y ventana 50 y SG NS 400 dimensiones ventana 150**: se combinan los modelos CBOW mencionados con un modelo SG para comprobar si las diferentes informaciones sobre los contextos, obtenidas con las dos arquitecturas, se complementan correctamente generando representaciones vectoriales de mayor calidad. La precisión obtenida es 72.3 % y si se normalizan con L_2 los vectores de representación, 75.7 %.
- **$0.3 *$ CBOW NS 800 dimensiones y tamaño de ventana 5, $0.5 *$ CBOW NS 800 dimensiones y ventana 50 y $0.2 *$ SG NS 400 dimensiones ventana 150**: la idea es la misma que para la combinación anterior, ponderando en este caso cada uno de los vectores en función de su eficacia en el método básico. Véase que el modelo que mejor se comporta está ponderado por 0.5 y los demás por valores decrecientes en función de su posición en el *ranking* de precisión. La precisión es 78.6 % y normalizando, 74.8 %.

- **CBOW NS 800 dimensiones y tamaño de ventana 5, CBOW NS 800 dimensiones y ventana 50 y CBOW NS 500 dimensiones ventana 10:** en este caso, se combinan modelos con arquitectura CBOW de varias dimensiones con distintos tamaños de ventana. De esta forma podemos determinar si entrenar considerando contextos y dimensiones diferentes resulta en vectores de representación de mayor calidad. Se obtiene una precisión del 74 % y si se normaliza, 73.2 %.
- **$0.3 *$ CBOW NS 800 dimensiones y tamaño de ventana 5, $0.5 *$ CBOW NS 800 dimensiones y ventana 50 y $0.2 *$ CBOW NS 500 dimensiones ventana 10 :** semejante a la combinación anterior, ponderando por orden de relevancia los modelos en función de su comportamiento en el método básico. Es la combinación con la que se obtiene la máxima precisión, 79.7 %. Además, si se emplea una lista de *stopwords* alternativa con 429 términos, se incrementa dicha precisión hasta un 80.3 %.

Para concluir, mencionar que se han excluido algunas combinaciones de la memoria ya que no se pretende hacer un estudio intensivo de su comportamiento. También, esta técnica de combinación de modelos ha permitido incrementar hasta en un 10.35 % la precisión con respecto al método básico, suponiendo una mejora del 10.46 % de los resultados obtenidos por investigadores de la UPM con el mismo corpus. Por todo ello, la combinación de modelos, constituye una interesante área de exploración e investigación para trabajos o estudios futuros que hagan uso de representaciones vectoriales continuas de palabras y frases obtenidas mediante Word2Vec.

5.6.3 *Clustering* de temas

Otra de las experimentaciones propuestas en este capítulo consiste en emplear técnicas de *clustering* no supervisadas que nos permitan, una vez definidas las agrupaciones o clústeres, extraer la idea subyacente en el conjunto de documentos de cada agrupación para determinar así una etiqueta posible que identifique o proporcione una aproximación de la temática en este caso.

La técnica propuesta es muy sencilla, en primer lugar, partiendo del conjunto de muestras de entrenamiento (empleamos esta partición como si no estuviese supervisada), se extraen las representaciones vectoriales continuas de las frases a evaluar haciendo uso de uno de los modelos entrenado con los artículos en español de Wikipedia. En este caso, hemos empleado un modelo Word2Vec con arquitectura SG, 400 dimensiones, ventana de tamaño 150 y representación suma ya que estos parámetros ofrecían los mejores resultados cuando se realizó la presente experimentación.

Posteriormente, se agrupan los vectores de representación de frases mediante algún método de *clustering* como K-medias y se extrae el *k* -grama más frecuente o representativo en los documentos de cada agrupación. Así, consideramos el mencionado *k*-grama como un indicador o aproximación de la idea subyacente o tema. Podemos visualizar las agrupaciones, aunque no se representan claramente debido al método empleado para el graficado de las muestras, en la Figura 35 . Además, es posible identificar algunos de los *k*-gramas recuperados para un pequeño subconjunto de los clústeres, formado por las siete primeras agrupaciones obtenidas, en la Tabla X.

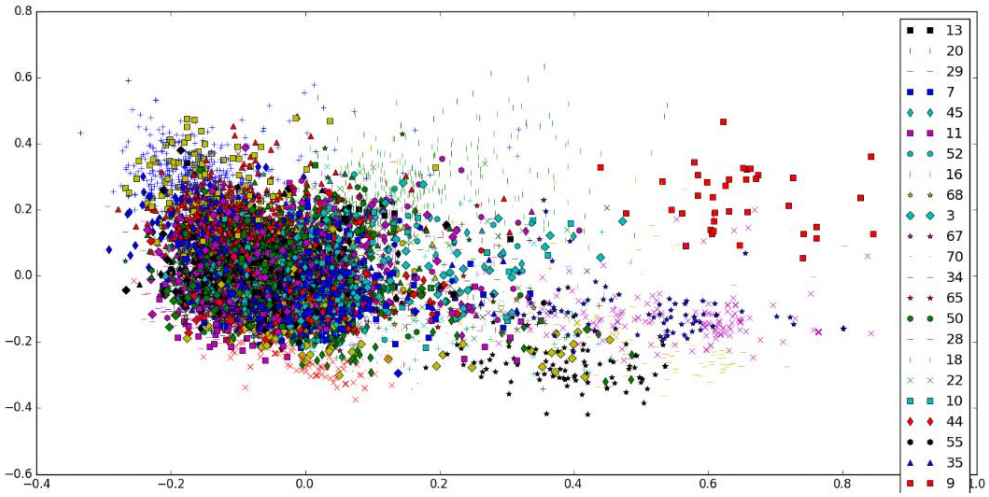

Fig. 35. *Clustering* de las muestras de entrenamiento en corpus EPPS

TABLA X
IDENTIFICACIÓN DE TEMAS MEDIANTE K-GRAMAS

Agrupación	1-grama	2-grama	3-grama
1	Palabra	usted palabra	ningún país puede
2	Aquí	Parlamento Europeo	presidente presidente consejo
3	Gracias	Unión Europea	nuevos países miembros
4	También	Presidente comisión	someto votación enmienda
5	Consejo	Grupo socialista	consejo asuntos generales
6	Nora	Unión Europea	muchas gracias Nora
7	Países	Constitución Europea	excede número puestos

Los *k*-gramas expuestos en la Tabla X, nos permiten inferir una aproximación al tema tratado en los documentos de cada agrupación, por ejemplo, en el clúster tres podemos intuir que el tema es la gratitud hacia los nuevos países miembros de la Unión Europea o que la agrupación cinco contiene los documentos que refieren asuntos generales del grupo socialista tratados en el consejo. Sin embargo, existen algunos clústeres de documentos cuyo tema es ambiguo y complejo de determinar como en el caso de la primera agrupación, además, este problema de ambigüedad se agrava si se dan ciertas condiciones como la presencia de *stopwords*.

Esta complejidad a la hora de determinar la temática mediante la técnica expuesta se debe principalmente a que la etiqueta del tema no tiene por qué ser el término o conjunto de términos que aparezca con mayor frecuencia en el conjunto de documentos analizado, por ejemplo, si el tema subyacente es «política», el término «política» no tiene por qué ser y, de hecho, generalmente no es, el término que más se repite en el conjunto de documentos cuya temática se quiere identificar. Es por ello que el método utilizado no se comporta bien ante problemas con temas muy generales cuya etiqueta no aparece en los documentos, pero sí puede tener un buen comportamiento cuando los temas son específicos y aparecen en las frases analizadas.

Finalmente, destacar la extensión de esta técnica a otro tipo de problemas, ya que nos puede permitir realizar aproximaciones de otros conceptos, diferentes de la temática, asociados a las agrupaciones encontradas como por ejemplo el idioma, tal como se verá en próximos apartados del capítulo seis, dedicado a experimentar con otras posibles aplicaciones de representaciones vectoriales continuas de palabras y frases.

«La política no es una ciencia exacta.»

-Otto von Bismarck

Capítulo 6

Otras aplicaciones de *word embeddings*

Además de para el problema de identificación de temática, las representaciones vectoriales continuas de palabras y frases, obtenidas mediante los modelos basados en redes neuronales expuestos en el capítulo uno, se han empleado en otras cuestiones relacionadas con el procesamiento del lenguaje natural como identificación del idioma, análisis de sentimiento y detección de *malware*. Por ello, en el presente capítulo, se introducen los problemas, se analizan soluciones basadas en representaciones vectoriales continuas y se estudian los resultados obtenidos tras el proceso de resolución.

6.1 Detección del idioma

La detección del idioma es una tarea que consiste en la identificación del lenguaje empleado a través de un canal mediante el que se establece una conexión entre el emisor y el receptor para posibilitar el intercambio de información v.g. documentos escritos o habla. Definido formalmente por E.M. Gold, gran cantidad de investigadores han trabajado desde entonces con diferentes aproximaciones, llegando a considerarse el problema generalmente resuelto [52].

Sin embargo, a pesar de considerarse resuelto, existen contextos particulares que incrementan la complejidad de la tarea de identificación del idioma como textos cortos, elevada presencia de ruido, mezcla de idiomas y uso de expresiones de jerga. Por ello, es necesario diseñar estrategias alternativas que permitan ampliar el espectro de experimentaciones que es posible realizar cuando las condiciones no son idóneas.

Proponemos enfocar el problema de detección del idioma como un problema de clasificación, donde el idioma determina la categoría, haciendo uso de representaciones vectoriales continuas de palabras y frases, de algunos de los clasificadores expuestos en el capítulo cuatro y de un corpus paralelo en múltiples idiomas específicamente diseñado para la experimentación realizada.

Dicho corpus ha sido diseñado automáticamente a partir de frases de artículos de Wikipedia traducidas a todos los idiomas considerados: alemán, catalán, español, francés, inglés e italiano, haciendo uso de la API REST del traductor de Bing. Con esto, traduciendo las mismas frases a múltiples lenguajes, evitamos la problemática que surge cuando hay palabras pertenecientes a un subconjunto reducido de idiomas, al tener estas un mayor peso discriminatorio.

Además, es importante eliminar: nombres propios, datos numéricos que inducen ruido y signos de puntuación no asignables a palabras (signos de exclamación, de interrogación, puntos etc.), al mismo tiempo que se conservan las *stopwords*, ya que, al ser diferentes entre muchos de los idiomas considerados, permiten incrementar la eficacia en la tarea de clasificación. Así, el conjunto de entrenamiento del corpus desarrollado cuenta con 23 frases para cada idioma, un total de 138 frases con una media de 25 palabras y 147 caracteres por frase, mientras que en el conjunto de test encontramos 15 frases por idioma (90 frases de test), tal como muestra la Tabla XI.

TABLA XI
PROPIEDADES DEL CORPUS IDIOMAS

Idiomas	inglés, alemán, catalán, español, francés, italiano
Dominio	Artículos de Wikipedia
Conjunto de entrenamiento	138 frases
Conjunto de test	90 frases

Es posible observar cómo se distribuyen los vectores de representación de las muestras de entrenamiento en un espacio bidimensional y analizar así la relación existente entre los diversos idiomas mediante la gráfica de la Figura 36. Según la gráfica, el alemán y el inglés son fácilmente distinguibles con respecto a los idiomas «mediterráneos» (español, francés, catalán e italiano), sin embargo, los idiomas del sur de Europa están muy solapados al compartir subconjuntos de palabras como por ejemplo algunas *stopwords*.

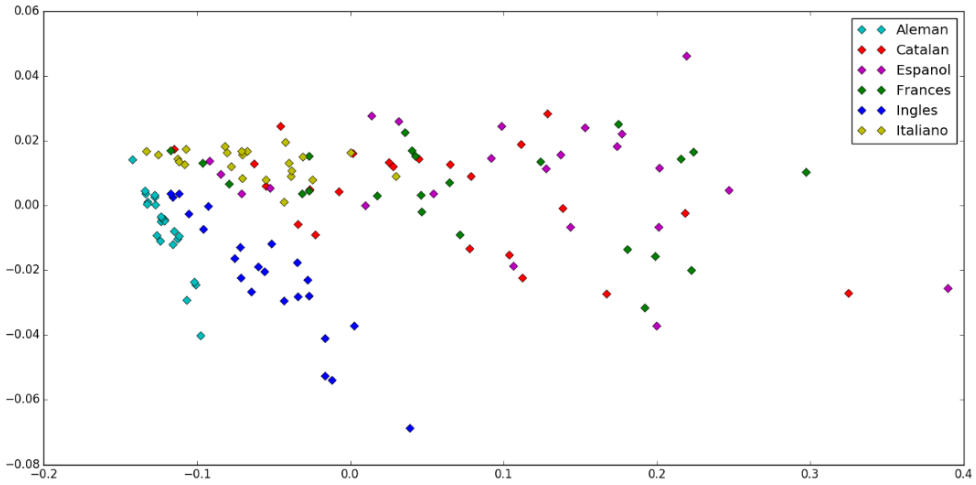

Fig. 36. Distribución 2*D* de diversos idiomas

Por otro lado, para la tarea tratada, no es necesario tener información de todo el contexto de las palabras, por lo que podemos emplear una ventana de tamaño reducido que permita capturar más información de la palabra analizada que del entorno de esta. Con ello, tal como se menciona en el apartado dedicado al protocolo de experimentación, primero se ha estimado la dimensión de los vectores y determinado el uso de HS o NS fijando una serie de parámetros: representación suma, tamaño de ventana=3, SG con NS y árbol de decisión como clasificador.

TABLA XII
ANÁLISIS DE DIMENSIONALIDAD EN DETECCIÓN DE IDIOMAS

Dimensionalidad	Precisión
100	62.2%
250	56.6%
500	64.4%
750	**67.6%**

La mayor precisión, según la Tabla XII, se obtiene con 750 dimensiones empleando los parámetros ya mencionados, sin embargo, si cambiamos NS por HS, la precisión se incrementa a un 76.6 %, por lo que aplicamos HS en las experimentaciones. En este caso se han realizado dos experimentos, usando clasificadores supervisados, variando la forma de representación de las frases entre suma y palabra más próxima al centroide.

En el primer experimento se clasifican los vectores de representación continuos de frases, generados a partir del modelo Word2Vec del corpus mediante la representación de suma de palabras. La precisión obtenida para cada clasificador está indicada en la Tabla XIII y se puede observar cómo los clasificadores SVM y K-vecinos más cercanos presentan el mejor comportamiento con un 93.3 % de precisión.

TABLA XIII
PRECISIÓN CON REPRESENTACIÓN SUMA EN DETECCIÓN DE IDIOMAS

Clasificador	Precisión
Naive Bayes (*Gaussian)*	91.1%
Naive Bayes (*Bernoulli)*	91.1%
Árbol de decisión	76.6%
Random forest 50 estimadores	91.1%
Random forest 100 estimadores	92.2%
KNN (centroide, distancia correlación)	**93.3%**
KNN (centroide, distancia coseno)	**93.3%**
KNN ($K = 1$, distancia *Minkowski)*	91.1%
KNN ($K = 2$, distancia *Minkowski)*	88.8%
SVM (*kernel* lineal)	**93.3%**
SVM (*kernel* RBF)	78.8%

El segundo experimento hace uso de la representación en función de la palabra más próxima al centroide. En este caso los resultados son peores que los obtenidos en la experimentación anterior, siendo la mayor precisión un 67.7 %, alcanzado mediante el clasificador *random forest* con cien estimadores, según la Tabla XIV.

TABLA XIV
PRECISIÓN CON REPRESENTACIÓN PALABRA CENTROIDE EN DETECCIÓN DE IDIOMAS

Clasificador	Precisión
Naive Bayes (*Gaussian)*	62.2%
Naive Bayes (*Bernoulli)*	63.3%
Árbol de decisión	63.3%
Random forest 50 estimadores	65.5%
Random forest 100 estimadores	**67.7%**
KNN (centroide, distancia correlación)	64.4%
KNN (centroide, distancia coseno)	64.4%
KNN ($K = 1$, distancia *Minkowski)*	64.4%
KNN ($K = 2$, distancia *Minkowski)*	62.2%
SVM (*kernel* lineal)	66.6%
SVM (*kernel* RBF)	61.1%

Así, se justifica mediante la Figura 37 por qué se comporta peor la representación basada en la palabra más cercana al centroide que la obtenida mediante la suma de las palabras que conforman la frase. En ella, observamos cómo los vectores obtenidos mediante la primera representación se solapan entre todos los idiomas i.e. debajo de un punto graficado en rojo, puede haber una gran cantidad de vectores de frases pertenecientes a otros idiomas diferentes del lenguaje que representa dicho color.

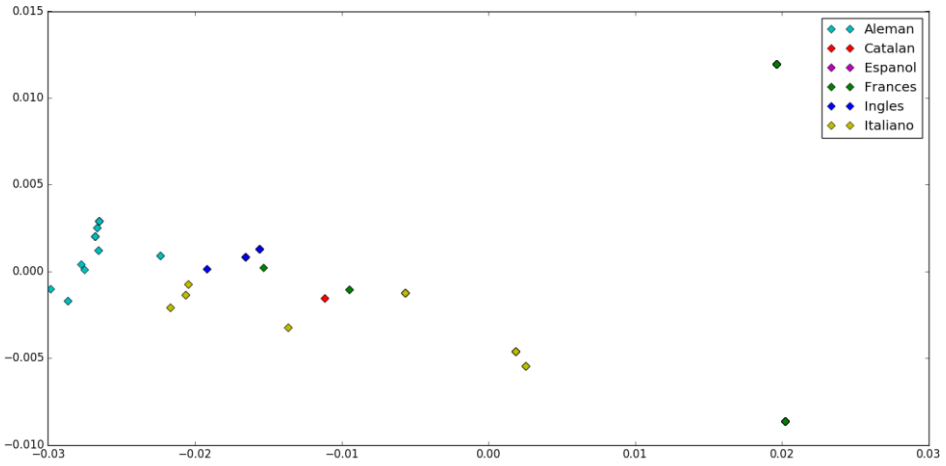

Fig. 37. Representación 2*D* de palabras representadas con palabra centroide

Por otro lado, también se han realizado experimentos de forma no supervisada empleando el método K-medias, tal como se hizo en el capítulo dedicado a la identificación de temática, para obtener un k-grama representativo de cada clase. En este caso, consideramos k -gramas, $1 \leq k \leq 3$, y la representación suma, que nos ha permitido obtener un mejor resultado en experimentaciones anteriores. La Figura 38 muestra las agrupaciones obtenidas mediante K-medias.

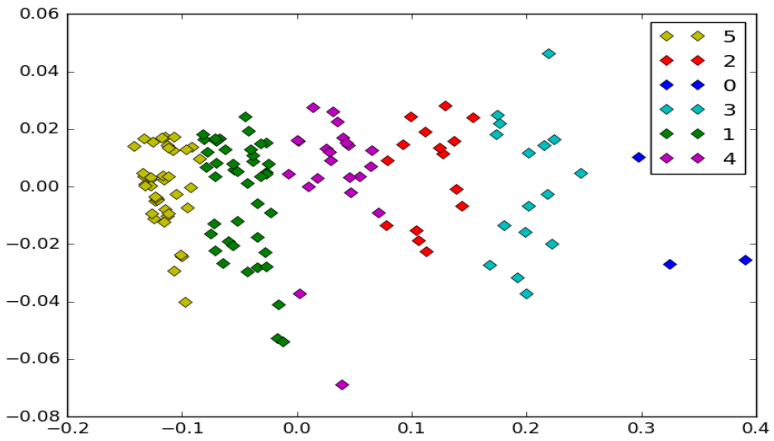

Fig. 38. *Clustering* con representación suma para detección de idiomas

Los *k*-gramas obtenidos aportan una visión de la clasificación no supervisada realizada en el experimento. Así, se extraen tantos *k*-gramas como idiomas a identificar y cada uno se asocia con un lenguaje, entre ellos encontramos:

- **1-gramas:** *que, the, que, de, die, a*.
- **2-gramas:** *dass es, de la, un equip, a un, de la, it is*.
- **3-gramas:** *les cristaux de, das salte zirkon, one is willing, che potrebbe essere, cràters d'impacte, vols arribar a*.

A partir de estos, considerando los trigramas que proporcionan más información, es posible observar cómo se identifican cinco de los seis idiomas, a destacar: francés (*les cristaux de*), alemán (*das salte zirkon*), inglés (*one is willing*), italiano (*che potrebbe essere*) y catalán (*cràters d'impacte* y *vols arribar a*). El catalán es confundido con el idioma español, al aparecer este en dos agrupaciones diferentes.

En resumen, se ha propuesto la utilización de representaciones vectoriales continuas de palabras y frases para ofrecer soluciones al problema de la detección de idioma, que, a pesar de considerarse generalmente resuelto, no es de fácil resolución en determinados contextos. Además, se han realizado experimentaciones, supervisadas y no supervisadas, sobre un corpus paralelo en múltiples idiomas, analizando y discutiendo los resultados obtenidos.

6.2 Análisis de sentimiento

El análisis de sentimiento o minería de opinión se refiere al uso de procesamiento de lenguaje natural para identificar y extraer información subjetiva, determinando así la polaridad: positiva, negativa, neutral etc., de recursos, generalmente textuales, donde se vierten opiniones [53]. Una herramienta de uso común en los sistemas de análisis de sentimiento son los lexicones de polaridad, que asocian un valor determinado de polaridad a cada palabra.

Sin embargo, a pesar del uso tan extendido de este tipo de diccionarios, la generación de lexicones de forma supervisada conlleva un gran esfuerzo humano cuando el dominio de aplicación del sistema no está acotado, es por ello que, siguiendo el trabajo realizado en [53], planteamos la utilización de representaciones vectoriales continuas de palabras para inferir automáticamente, de forma no supervisada, la polaridad de los términos.

Más concretamente, proponemos predecir la polaridad de una palabra w cuya representación vectorial es obtenida de acuerdo a un modelo Word2Vec, generado a partir de artículos de Wikipedia en español, con 400 dimensiones usando SG, NS y un valor de 150 como tamaño de ventana. Así, medimos la similitud con respecto a un conjunto de términos semilla de cada una de las polaridades tratadas: positivo (POS) y negativo (NEG). De esta forma, si $polaridad(w) < 0$ diremos que su clase es NEG, mientras que en el caso contrario $polaridad(w) \geq 0$ la clase será POS (nótese que podríamos tratar el caso $polaridad(w) = 0$ como neutro, sin embargo, únicamente consideramos polaridad positiva o negativa).

Por otro lado, uno de los principales inconvenientes que incrementan la complejidad del problema tratado es la polaridad de k-gramas con $k > 1$. En el presente trabajo, únicamente consideramos el caso $k = 1$, por lo que no será posible predecir la polaridad de términos multipalabra, ej. merece la pena. Una posible solución a este problema sería entrenar k modelos Word2Vec de k-gramas y consultar un modelo u otro en función del término del que se quiere conocer la polaridad (esto conllevaría establecer k-gramas semillas para cada valor de k).

Retomando la definición de la función de polaridad, para determinarla, se calcula la polaridad de una palabra w como la diferencia entre la similitud con respecto a la clase POS y la semejanza con los términos semilla de la clase NEG, normalizada por el número de términos en las clases consideradas. Podemos emplear, como función de similitud, funciones conocidas como la distancia coseno o la distancia euclídea. Formalmente, la formulación es: $polarity(w) = \frac{1}{|POS|}\sum_{v \in POS} sim(w, v) - \frac{1}{|NEG|}\sum_{v \in NEG} sim(w, v)$.

Centrándonos en los requisitos del experimento realizado, se ha empleado, como términos semilla positivos y negativos, un subconjunto de palabras extraído del lexicón *ElhPolar*, diseñado por X. Saralegi e I. San Vicente, en el que los términos compuestos han sido eliminados para limitar la experimentación a 1-gramas. Dicho lexicón posee 1 711 lemas positivos y 3 006 términos negativos, lo que conlleva un total de 4 717 palabras cuya polaridad ha sido supervisada.

Con ello, se confecciona un conjunto aleatorio de test, formado por los lemas del lexicón que no han sido utilizados como términos semilla, con 470 muestras (10 % de los términos del corpus), de las que 235 son términos positivos y el resto son lemas negativos. Además, mediante este conjunto de test, es posible evaluar el lexicón generado de forma no supervisada, al conocerse las etiquetas de los términos cuya polaridad se ha inferido.

Una vez planteada la forma de realizar el experimento y tras llevar a cabo la tarea de evaluación del sistema, es necesario destacar que, empleando como medida de similitud la distancia coseno y un modelo Word2Vec SG, NS, $k = 150, d = 400$, generado a partir de artículos de Wikipedia, los resultados no son del todo satisfactorios ya que la precisión obtenida es de un 52.1 %, errando el sistema en términos como abrazo, apasionado, apreciado, apropiado, abatido, abdicar etc.

Sin embargo, existe una justificación para el valor de precisión obtenido. Si observamos la forma en que se obtienen los vectores de representación de palabras, se tiene en cuenta el contexto de cada uno de los términos, por lo que, palabras que en principio pueden tener polaridades diferentes v.g. bueno y malo, tendrán representaciones vectoriales muy similares debido a que sus entornos de ocurrencia son muy semejantes.

Y no solo esto, también en función de los textos con los que haya sido entrenado el modelo Word2Vec, los contextos de algunas palabras las harán más similares a términos positivos o a negativos, ej. abdicar. Es por ello que, para una determinada tarea, la eficacia del sistema puede ser mejor si se emplea un modelo Word2Vec aprendido del propio corpus del problema tratado.

En suma, en el presente apartado hemos introducido el problema de detección de polaridad, se ha realizado y formalizado una reproducción simplificada del experimento de generación no supervisada de un lexicón de polaridad tratado en [53], así como evaluado y analizado los resultados obtenidos mediante el enfoque planteado que hace uso de representaciones vectoriales continuas de palabras.

6.3 Detección de *malware*

La tarea de detección de *malware* es muy compleja, más concretamente es un problema indecidible, según F. Cohen, para el que no existe un algoritmo que sea capaz de determinar siempre una respuesta lógica correcta. Además, la dificultad del problema crece exponencialmente cuando se pretende agrupar variantes de *malware* en sus respectivas familias, siendo los principales inconvenientes: la cantidad masiva de datos que deben ser evaluados para identificar ficheros potencialmente maliciosos y la capacidad de mutación de dichos programas malintencionados.

En este apartado, planteamos el problema simplificado de clasificación, únicamente, en dos clases: *malware* y no *malware*, haciendo uso de representaciones vectoriales de palabras y frases obtenidas mediante Word2Vec. Debido a que existen equivalencias entre el lenguaje natural y el ensamblador, ambos pueden ser considerados lenguajes incontextuales, es posible analizar ejecutables con técnicas propias de NLP para determinar propiedades como la presencia de código malicioso. En este caso, consideramos las instrucciones como palabras y obtenemos la representación vectorial de una aplicación en función de los vectores de sus instrucciones.

En lo referente al corpus con el que se han entrenado los modelos Word2Vec, este ha sido elaborado partiendo del repositorio theZoo, que contiene muestras de *malware* para investigación y análisis [54], y extrayendo ejecutables comunes en entornos Windows. Además, algunas de las muestras no maliciosas han sido minuciosamente seleccionadas para que sus propiedades (conexiones, cifrados, etc.) sean similares a *malware* tradicional, complicando así la tarea de clasificación. Con todo ello, el corpus contiene 179 muestras de entrenamiento, de las cuales 96 son maliciosas y 83 no lo son, y 91 ficheros de test, 50 *malware* y 41 no *malware*, constituyendo un total de 270 ejecutables con un tamaño de 61MB, tal como se indica en la Tabla XV.

TABLA XV
PROPIEDADES DEL CORPUS MALWARE

Lenguaje	Ensamblador X86
Dominio	Ejecutables
Conjunto de entrenamiento	179 ejecutables
Conjunto de test	91 ejecutables
Tamaño en disco	61 MB

Además, debido a las similitudes entre el problema tratado y la identificación de temática, los parámetros a emplear en la generación de los modelos Word2Vec serán semejantes a los establecidos en las experimentaciones del capítulo cinco. En este caso, conviene hacer uso de un tamaño de ventana k relativamente grande, sin embargo, no es posible configurarlo con el número total de instrucciones, ya que los programas pueden estar compuestos por decenas de miles de órdenes y se puede dar un incremento del coste computacional. Por este motivo se emplea $k = 150$.

Por otra parte, con respecto a la evaluación del sistema, empleamos métricas de clasificación binaria ampliamente utilizadas en el área de análisis de *malware*. Entre estas medidas, además de las mencionadas en el apartado nueve del capítulo cuatro, encontramos: valor predictivo positivo $\left(vpp = \frac{tp}{(tp+fp)}\right)$, representa la probabilidad de que un programa sea malicioso si se ha predicho que lo es, y valor predictivo negativo ($vpn = \frac{tn}{(tn+fn)}$), que hace referencia a la probabilidad de que un fichero no sea de la clase *malware* si se ha pronosticado que no lo es.

De esta forma, no es lo mismo conseguir cierto valor de precisión (*accuracy*) habiendo predicho correctamente todas las muestras maliciosas y errado con todas las muestras no maliciosas, que conseguirlo habiendo acertado gran parte de ambos tipos de muestras. Por ello, en la tarea, es importante considerar otras métricas de clasificación binaria diferentes de la precisión, utilizada en este proyecto en los problemas que conllevan clasificación en múltiples clases. Además, en este caso, la reducción de falsos positivos y de falsos negativos es un factor clave para la eficacia del método, evitando, en gran parte, situaciones que tienen un alto coste asociado como identificar un ejecutable como *malware* cuando no lo es.

Con ello, tal como se indica en el apartado dedicado a introducir el esquema de experimentaciones seguido en el presente proyecto, se debe aproximar la dimensión de los vectores y determinar las optimizaciones para el cálculo de la distribución de probabilidad condicional del modelo Word2Vec. El proceso de aproximación se puede visualizar en la Tabla XVI.

TABLA XVI
PRUEBA DE DIMENSIONALIDAD EN DETECCIÓN DE MALWARE

Dimensionalidad	Precisión
100	68.1%
250	74.7%
500	**78.1%**
750	64.8%

En este caso, los mejores resultados se obtienen con 500 dimensiones empleando *negative sampling*. Si consideramos la utilización de *hierarchical softmax*, la precisión alcanzada es 76.9 %, por tanto, se fijan $d = 500$ y *negative sampling*. A continuación, variamos el método de representación de frases (en esta tarea ejecutables) entre palabra más próxima al centroide y suma, representando gráficamente las muestras de entrenamiento y generando los valores de las diversas métricas ya mencionadas en párrafos anteriores de este mismo apartado.

Por un lado, con la representación suma, podemos observar cómo se distribuyen las muestras de entrenamiento en un espacio $2D$ en la Figura 39. En esta situación, es posible observar la segmentación de las muestras no *malware* (etiqueta«0») en tres densas agrupaciones en los extremos, aunque existen algunos vectores en zonas intermedias que se solapan con determinadas muestras de *software* malicioso y no permiten clasificar sin error los ejemplos de entrenamiento.

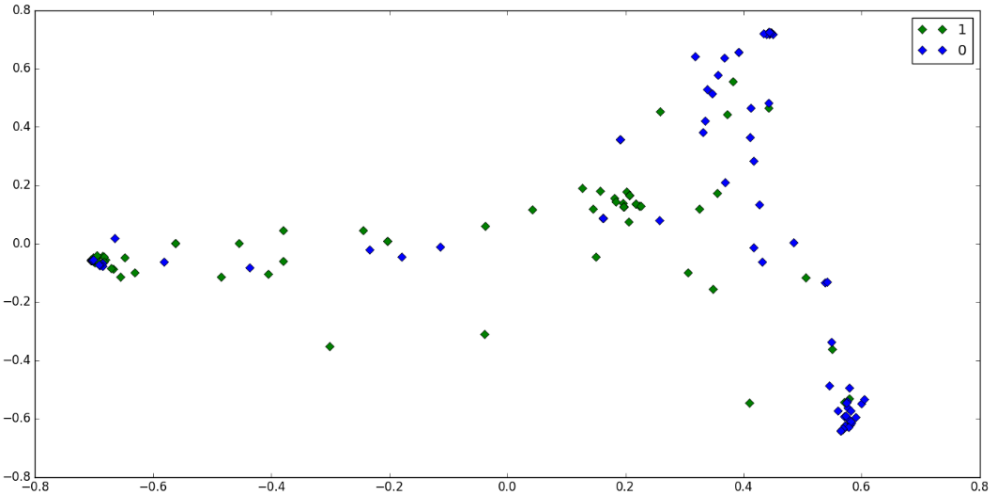

Fig. 39. Representación 2*D* muestras theZoo con suma

Además, mediante esta representación vectorial de frases, los resultados varían entre los diferentes clasificadores utilizados, según la Tabla XVII, alcanzando una precisión del 76 % con árboles de decisión. Sin embargo, a pesar de que la cobertura, la precisión, los valores predictivos positivo y negativo y la tasa de falsos negativos tienen valores muy adecuados, los falsos positivos constituyen el principal inconveniente del método empleado, variando en el intervalo [0.31,0.63]. Esto indica que, en el peor de los casos (clasificador KNN), un 63 % de los ejecutables predichos como *malware* realmente no lo son.

TABLA XVII
MÉTRICAS CON REPRESENTACIÓN SUMA EN DETECCIÓN DE MALWARE

Clasificador	r	vpp	vpn	fp	fn	p	$f1$
Naive Bayes (*Gaussian)*	94.0%	70.0%	**87.0%**	48.0%	**6.0%**	74.0%	**82.0%**
Naive Bayes (*Bernoulli)*	86.0%	74.0%	78.8%	36.0%	14.0%	75.0%	80.0%
Árbol de decisión	90.0%	73.0%	83.0%	39.0%	9.0%	**76.0%**	**82.0%**
Random forest 50 estimadores	94.0%	67.0%	85.0%	56.0%	**6.0%**	71.0%	80.0%
Random forest 100 estimadores	94.0%	64.0%	83.0%	63.0%	**6.0%**	68.0%	78.0%
KNN (centroide, distancia correlación)	86.0%	71.0%	77.0%	41.0%	14.0%	73.0%	78.0%
KNN (centroide, distancia coseno)	90.0%	71.0%	77.0%	41.0%	14.0%	73.0%	78.0%
KNN ($K = 1$, distancia *Minkowski)*	**98.0%**	63.0%	75.0%	63.0%	9.0%	65.0%	75.0%
KNN ($K = 2$, distancia *Minkowski)*	78.0%	**75.0%**	71.0%	**31.0%**	21.0%	73.0%	75.0%
SVM (*kernel* lineal)	92.0%	68.0%	83.0%	51.0%	7.0%	72.0%	80.0%

Por otro lado, con la representación de frases mediante la palabra más próxima al centroide, los ejemplos de entrenamiento parecen más solapados en la Figura 40, sin embargo, debido a su distribución, es posible que con un número mayor de dimensiones sean fácilmente separables. Ahora, los resultados obtenidos, mostrados en la Tabla XVIII, son sensiblemente mejores que en el caso anterior en gran parte de las métricas, pero no existen diferencias relevantes que destacar entre las experimentaciones, a excepción del resultado obtenido con KNN $k = 2$, cuyos valores de precisión, cobertura y valores predictivos positivo y negativo son muy reducidos en comparación con el resto.

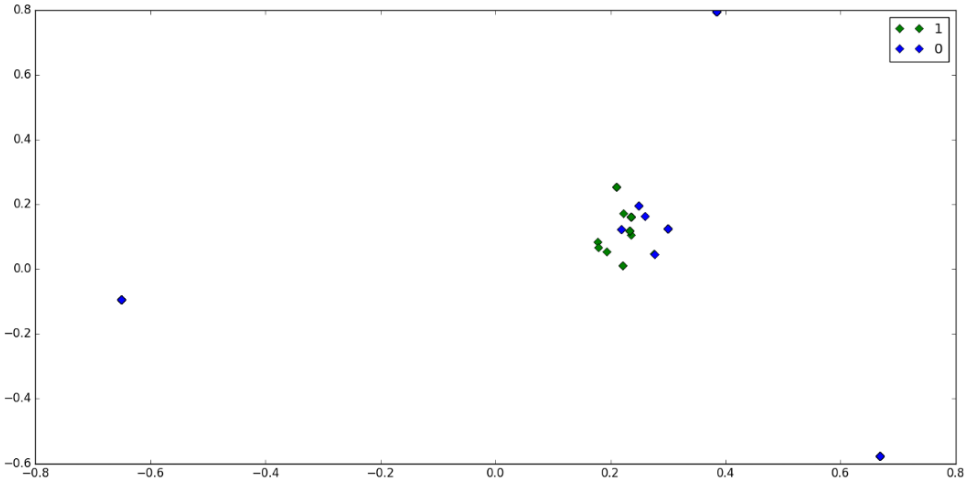

Fig. 40. Representación $2D$ muestras theZoo con palabra centroide

TABLA XVIII
MÉTRICAS CON REPRESENTACIÓN PALABRA CENTROIDE EN DETECCIÓN DE MALWARE

Clasificador	r	vpp	vpn	fp	fn	p	$f1$
Naive Bayes (*Gaussian)*	94.0%	69.0%	**87.0%**	51.0%	6.0%	73.0%	82.0%
Naive Bayes (*Bernoulli)*	92.0%	**74.0%**	78.8%	39.0%	7.0%	**78.0%**	84.0%
Árbol de decisión	94.0%	72.0%	83.0%	43.0%	6.0%	**78.0%**	85.0%
Random forest 50 estimadores	96.0%	72.0%	85.0%	43.0%	4.0%	**78.0%**	**86.0%**
Random forest 100 estimadores	96.0%	72.0%	83.0%	43.0%	4.0%	**78.0%**	**86.0%**
KNN (centroide, distancia correlación)	82.0%	73.0%	77.0%	41.0%	7.0%	76.0%	83.0%
KNN (centroide, distancia coseno)	92.0%	73.0%	77.0%	41.0%	7.0%	76.0%	83.0%
KNN ($K = 1$, distancia *Minkowski)*	**98.0%**	60.0%	75.0%	78.0%	**2.0%**	63.0%	76.0%
KNN ($K = 2$, distancia *Minkowski)*	38.0%	61.0%	71.0%	**29.0%**	62.0%	52.0%	43.0%
SVM (*kernel* lineal)	96.0%	69.0%	83.0%	51.0%	4.0%	74.0%	83.0%

En resumen, se ha propuesto una solución al problema de detección de ficheros maliciosos haciendo uso de representaciones vectoriales de palabras, comprendiendo los programas como la suma de las semánticas de las instrucciones que los forman. Además, se han evaluado, mediante un conjunto de métricas, diversos clasificadores aplicados a la tarea tratada, obteniendo buenos resultados con las muestras de la colección theZoo y ejecutables comunes empleados en sistemas Windows.

«Si conoces al enemigo y a ti mismo, no debes temer el resultado a un ciento de batallas.»
-Sun Tzu

Capítulo 7

Conclusiones y trabajo futuro

En el último capítulo exponemos las conclusiones y consideraciones, obtenidas tras haber realizado completamente el trabajo, así como un conjunto de propuestas y posibles mejoras a aplicar, sobre los métodos desarrollados en este proyecto, para llevar a cabo trabajos futuros que planteen o estudien soluciones a problemas similares a los tratados en el presente documento.

7.1 Conclusiones

Durante el presente proyecto, se ha logrado la implementación de un sistema que facilita el desarrollo de todas las pruebas realizadas sobre los diferentes problemas planteados. Este sistema nos ha permitido realizar las etapas de preprocesamiento, extracción de características, evaluación y visualización, siendo imprescindible para la consecución, de forma eficiente, de los demás objetivos planteados al permitirnos generalizar los aspectos comunes a todos los problemas esbozados

Por un lado, se han empleado con éxito representaciones vectoriales continuas de palabras y frases, generadas mediante Word2Vec, en problemas significativos del descubrimiento automático de conocimiento, obteniendo muy buenos resultados especialmente en el problema de detección de temática con el corpus EPPS, donde hemos mejorado en un 10.46 % los mejores resultados conocidos, obtenidos por investigadores de la Universidad Politécnica de Madrid.

Además, se han demostrado las ventajas e inconvenientes de la aplicación de este tipo de representaciones en las demás tareas tratadas a lo largo del proyecto como son la identificación del idioma, la detección de *malware* y el análisis de sentimiento; ofreciendo resultados prometedores en los dos primeros problemas mencionados, mientras que, en el último caso, debido a que palabras cuyas polaridades son diferentes no tienen por qué ocurrir en contextos diferentes, los resultados no manifiestan un buen comportamiento.

Por otro lado, también se ha hecho uso de representaciones vectoriales discretas de palabras y frases, obtenidas a partir de una lista de propiedades en inglés presentes en la ontología del proyecto DBpedia. En este caso, los resultados expuestos en su aplicación al problema de detección de temática no muestran evidencias sobre la eficacia del método, consiguiendo un pico máximo de precisión del 10.3 % en el corpus EPPS. Sin embargo, se ha comprobado su correcto funcionamiento en un corpus cuyo idioma coincide con el de la lista de propiedades y la ocurrencia de entidades en él es significativa, obteniendo un 65.7 % de precisión empleando SVM.

En lo referente a los objetivos personales, se ha logrado adquirir conocimiento sobre diversos tipos de redes neuronales aplicadas al aprendizaje de modelos de lenguaje, así como se han especializado las competencias, adquiridas durante el grado, sobre *machine learning* en aplicaciones dedicadas al procesamiento del lenguaje natural. Y no solo esto, también ha sido posible mejorar la capacidad de pensamiento y trabajo en equipo mediante las reuniones realizadas con los tutores del proyecto y adquirir competencias transversales relativas a la investigación de artículos científicos.

Para concluir, debido a la capacidad de cómputo requerida en alguna de las experimentaciones realizadas, se han adquirido habilidades para la coordinación de ejecución de tareas de aprendizaje automático en paralelo, haciendo uso de tres máquinas del Departamento de Sistemas Informáticos y Computación con procesadores Intel Xeon CPU E5-1660 v2 @ 3.70 GHz y Xeon CPU E5-1620 v3 @ 3.70 GHz, 12 y 8 núcleos y 64 GB de RAM durante un periodo de tiempo de aproximadamente una semana, lo que conlleva entre 144 y 168 horas de procesamiento solo para el entrenamiento de modelos Word2Vec.

7.2 Trabajo futuro

Debido a la gran cantidad de problemas tratados, pertenecientes al área de descubrimiento automático de conocimiento, y experimentaciones realizadas sobre dichos problemas, proponemos distintas vías de expansión del trabajo realizado que permitirán, en un futuro, desarrollar nuevos métodos y soluciones para las cuestiones abarcadas.

En primer lugar, un nuevo campo para experimentar en identificación de temática con el corpus EPPS consiste en hacer uso de representaciones vectoriales continuas obtenidas mediante Doc2Vec [55] y Lda2Vec [56]. También, es interesante la exploración de combinaciones de modelos continuos, uniéndolos mediante diversas técnicas como combinaciones lineales, que han demostrado ofrecer las mejores soluciones en la experimentación realizada.

En segundo lugar, sobre el mismo problema de detección de temática utilizando representaciones vectoriales discretas de palabras y frases, se podrían obtener mejores resultados empleando el método de selección de entidades provisto por el servicio DBpedia *Spotlight*. Así, en el sistema desarrollado, se han implementado las funciones necesarias para la extracción de características haciendo uso de dicho servicio, lo que puede ser de ayuda en estudios futuros.

En tercer lugar, con respecto a la identificación del idioma, sería conveniente realizar las experimentaciones con corpus más complejos con los que poder llevar a cabo comparaciones con resultados conocidos, además, proponemos también la extensión del modelo utilizado para ofrecer soluciones al problema de traducción automática, por ejemplo, mediante un enfoque simple que convierta términos de un idioma a otro en función de distancias en un espacio métrico.

En cuarto lugar, de la misma forma que, en el presente proyecto, se han empleado los *k*-gramas más representativos para intentar extraer una aproximación de la idea o concepto subyacente en un conjunto de representaciones vectoriales de documentos, se sugiere la definición, en futuros trabajos, de nuevos métodos que, partiendo de agrupaciones de este tipo de vectores, permitan extraer dicha idea, por ejemplo, el idioma o la temática.

En quinto lugar, en lo concerniente a la detección de ejecutables maliciosos, del mismo modo que en identificación del idioma, es interesante la realización de las experimentaciones sobre corpus más complejos como el propuesto en *Microsoft Malware Classification Challenge* para poder comparar con resultados conocidos. Además, es posible extender el método utilizado mediante la combinación con otras características como grafos de flujo, entropía, DAG semántico etc., para desarrollar sistemas que pueden ser incluso llevados a producción.

Por último, ha quedado pendiente la evaluación de los lexicones generados mediante el método no supervisado basado en términos semilla, así como la utilización de representaciones vectoriales continuas de palabras y frases, tal como se han empleado en las demás tareas, sobre problemas de detección de polaridad en documentos. Por ello, proponemos extender las experimentaciones en tareas de análisis de sentimiento en trabajos o estudios futuros.

"Sabiduría es saber qué hacer a continuación, habilidad es saber cómo hacerlo, y virtud es hacerlo".

–David Starr Jordan

Bibliografía

[1] J. Hernández Orallo y C. Ferri Ramírez, «Minería de Datos y Extracción de Conocimiento de Bases de Datos,» de *Extracción Automática de Conocimiento en Bases de Datos e Ingeniería del Software*, Valencia, 2005.

[2] M. E. Acosta Aguilera, «Minería de datos y descubrimiento de conocimiento,» La Habana, 2016.

[3] T. Mikolov, I. Sutskever, K. Chen, G. Corrado y J. Dean, «Distributed representations of words and phrases and their compositionality,» de *Advances in neural information processing systems*, 2013.

[4] T. Mikolov, K. Chen, G. Corrado y J. Dean, «Efficient Estimation of Word Representations in Vector Space,» *arXiv preprint arXiv:1301.3781,* pp. 1-6, 2013.

[5] J. D. Echeverry Correa, J. Ferreiros López, A. Coucheiro Limeres, R. Córdoba y J. M. Montero, «Topic identification techniques applied to dynamic language model adaptation for automatic speech recognition,» *Elsevier,* nº 41, pp. 101-112, 2014.

[6] X. Rong, «word2vec Parameter Learning Explained,» *arXiv preprint arXiv:1301.3781,* 2014.

[7] W. contributors, «Word2Vec,» 2016. [En línea]. Available: en.wikipedia.org/wiki/Word2vec. [Último acceso: 17 Junio 2016].

[8] O. Levy y Y. Goldberg, «Dependency-Based Word Embeddings,» de *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, 2014.

[9] C. Nicholson, «Quora,» 17 Agosto 2015. [En línea]. Available: https://www.quora.com/Is-skip-gram-negative-sampling-better-than-CBOW-NS-for-word2vec-If-so-why. [Último acceso: 17 Junio 2016].

[10] O. Levy, «Quora,» 20 Octubre 2014. [En línea]. Available: https://www.quora.com/How-does-word2vec-work. [Último acceso: 17 Junio 2016].

[11] T. Mikolov, «Google Code,» 30 Julio 2013. [En línea]. Available: https://code.google.com/archive/p/word2vec/. [Último acceso: 17 Junio 2016].

[12] R. Řehůřek, «Gensim,» 2011. [En línea]. Available: https://radimrehurek.com/gensim/models/word2vec.html. [Último acceso: 17 Junio 2016].

[13] W. Blacoe y M. Lapata, «A Comparison of Vector-based Representations for Semantic Composition,» de *ACL*, 2012.

[14] J. Mitchell y M. Lapata, «Vector-based Models of Semantic Composition,» de *Proceedings of ACL-08: HLT*, 2008.

[15] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak y I. Zachary, «DBpedia: a nucleus for a web of open data,» de *Proceedings of the 6th International Semantic Web Conference*, 2007.

[16] M. Freudenberg y M. Ackermann, «DBpedia Extraction of Knowledge from Wikipedia,» de *DBpedia community meeting*, Dublin, 2015.

[17] M. Morsey, J. Lehmann, S. Auer y S. Hellman, «DBpedia and the live extraction of structured data from Wikipedia,» *Program,* vol. 46, nº 2, pp. 157-181, 2012.

[18] V. Botti, C. Carrascosa y V. Julián, «Tema 4: Capacidad Social,» de *Agentes inteligentes*, Valencia, 2015.

[19] DBpedia, «Ontology Classes,» 17 Noviembre 2008. [En línea]. Available: http://mappings.dbpedia.org/server/ontology/classes/. [Último acceso: 17 Junio 2016].

[20] W. contributors, «SPARQL,» 2016. [En línea]. Available: en.wikipedia.org/wiki/SPARQL. [Último acceso: 17 Junio 2016].

[21] L. Feigenbaum, «SPARQL By Example,» 23 Mayo 2010. [En línea]. Available: cambridgesemantics.com/semantic-university/sparql-by-example. [Último acceso: 17 Junio 2016].

[22] M. Rico y Ó. Corcho, «DBpedia,» Junio 2011. [En línea]. Available: http://es.dbpedia.org/. [Último acceso: 17 Junio 2016].

[23] P. N. Mendes, M. Jakob, A. García-Silva y C. Bizer, «DBpedia Spotlight: Shedding Light on the Web of Documents,» de *Proceedings of the 7th International Conference on Semantic Systems*, Graz, Austria, 2011.

[24] E. Vidal Ruiz y F. Casacuberta Nolla, «Tema 1. Introducción al Aprendizaje Automático,» de *Aprendizaje Automático*, Valencia, 2015.

[25] R. Bhatnagar, M. K. Ansari, S. Bhatnagar y H. Barik, «An Expert Anti-Malware Decision System,» *International Journal of Soft Computing and Engineering,* vol. 2, nº 5, pp. 144-147, 2012.

[26] J. Civera Saiz, R. Paredes Palacios y C. D. Martínez Hinarejos, «Tema 8: Clasificación basada en distancias,» de *Percepción*, Valencia, 2015.

[27] E. Vidal Ruiz, «Tema 3: Inducción de reglas y patrones, Árboles de decisión,» de *Sistemas Inteligentes*, Valencia, 2015.

[28] J. Civera Saiz, C. D. Martínez Hinarejos y R. Paredes Palacios, «Tema 9: Bagging y Boosting,» de *Percepción*, Valencia, 2015.

[29] D. Cournapeau, M. Brucher, J. Millman y a. et, «Nearest Neighbors Scikit-learn,» 2010. [En línea]. Available: http://scikit-learn.org/stable/modules/neighbors.html. [Último acceso: 18 Junio 2016].

[30] D. Cournapeau, M. Brucher, J. Millman y a. et, «Nearest Centroid Scikit-learn,» 2010. [En línea]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestCentroid.html. [Último acceso: 18 Junio 2016].

[31] R. Paredes Palacios, J. Civera Saiz y C. D. Martínez Hinarejo, «Tema 8.1: Aprendizaje de distancias y prototipos,» de *Percepción*, Valencia, 2014.

[32] D. Cournapeau, M. Brucher, J. Millman y a. et, «Decision Tree Scikit-learn,» 2010. [En línea]. Available: http://scikit-learn.org/stable/modules/tree.html. [Último acceso: 18 Junio 2016].

[33] E. Vidal Ruiz y F. Casacuberta Nolla, «Tema 4: Máquinas de vectores soporte,» de *Aprendizaje Automático*, Valencia, 2015.

[34] D. Cournapeau, M. Brucher, J. Millman y a. et, «Linear SVC Scikit-learn,» 2010. [En línea]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html. [Último acceso: 18 Junio 2016].

[35] D. Cournapeau, M. Brucher, J. Millman y a. et, «SVC Scikit-learn,» 2010. [En línea]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html. [Último acceso: 18 Junio 2016].

[36] D. Cournapeau, M. Brucher, J. Millman y a. et, «Support Vector Machines,» 2010. [En línea]. Available: http://scikit-learn.org/stable/modules/svm.html. [Último acceso: 18 Junio 2016].

[37] D. Cournapeau, M. Brucher, J. Millman y a. et, «Naive Bayes,» 2010. [En línea]. Available: http://scikit-learn.org/stable/modules/naive_bayes.html. [Último acceso: 18 Junio 2016].

[38] J. M. García Gómez y S. Tortajada, «Tema 8: Métodos de Aprendizaje Estadístico y Geométrico,» de *Bioinformática*, Valencia, 2012.

[39] J. Civera Saiz, R. Paredes Palacios y C. D. Martínez Hinarejos, «Tema 5: Distribución de Bernoulli,» de *Percepción*, Valencia, 2015.

[40] J. Civera Saiz, R. Paredes Palacios y C. D. Martínez Hinarejos, «Tema 7: Distribución Gaussiana,» de *Percepción*, Valencia, 2015.

[41] D. Cournapeau, M. Brucher, J. Millman y a. et, «Bernoulli NB Scikit-learn,» 2010. [En línea]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html. [Último acceso: 18 Junio 2016].

[42] D. Cournapeau, M. Brucher, J. Millman y a. et, «Gaussian NB Scikit-learn,» 2010. [En línea]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html. [Último acceso: 18 Junio 2016].

[43] E. Vidal Ruiz, «Tema 4: Aprendizaje no-supervisado: algoritmo k-medias,» de *Sistemas Inteligentes*, Valencia, 2015.

[44] D. Cournapeau, M. Brucher, J. Millman y a. et, «KMeans Scikit-learn,» 2010. [En línea]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html. [Último acceso: 18 Junio 2016].

[45] D. Cournapeau, M. Brucher, J. Millman y a. et, «Ensemble methods Scikit-learn,» 2010. [En línea]. Available: http://scikit-learn.org/stable/modules/ensemble.html. [Último acceso: 18 Junio 2016].

[46] L. F. Hurtado Oliver, E. Segarra Soriano, E. Sanchís Arnal y e. a. , «Tema 5: Evaluación de sistemas de RI,» de *Sistemas de almacenamiento y recuperación de información*, Valencia, 2015.

[47] G. Brown y G. Yule, «Topic and the representation of discourse content,» de *Discourse Analysis*, Cambridge, Cambridge University Press, 1983, pp. 68-73.

[48] J. K. Seppänen, E. Bingham y H. Mannila, «A Simple Algorithm for Topic Identification in 0-1 Data,» de *7th Europea Conferencei on Principles and Practice of Knowledge Discovery in Databases*, Cavtat-Dubrovnik, 2003.

[49] B. Stein y S. Meyer zu Eissen, «Topic Identification: Framework and Application,» *Journal of Universal Computer Science,* pp. 353-360, 2004.

[50] W. contributors, «Wikipedia Dumps,» 1 Junio 2016. [En línea]. Available: https://dumps.wikimedia.org/eswiki/. [Último acceso: 2 Julio 2016].

[51] R. Řehůřek, «Gensim,» 2011. [En línea]. Available: https://radimrehurek.com/gensim/corpora/wikicorpus.html . [Último acceso: 2 Julio 2016].

[52] C. Y. Almeida, S. Estébez Velarde y A. Piad Morffis, «Detección de Idioma en Twitter,» *GECONTEC: Revista Internacional de Gestión del Conocimiento y la Tecnología.,* vol. 2, nº 3, pp. 35-45, 2014.

[53] A. García Pablos, G. Rigau y M. Cuadros, «Unsupervised Word Polarity Tagging by Exploiting Continuous Word Representations,» *Procesamiento del Lenguaje Natural,* nº 55, pp. 127-134, 2015.

[54] L. F. Hurtado Oliver, E. Segarra Soriano, E. Sanchís Arnal y e. a. , «Tema 4: Modelo de espacio vectorial,» de *Sistemas de almacenamiento y recuperación de la información*, Valencia, 2015.

[55] Q. Le y T. Mikolov, «Distributed Representations of Sentences and Documents,» de *Proceedings of ICML 2014*, 2014.

[56] C. Moody, «Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec,» *arXiv:1605.02019 [cs.CL],* 2016.