Text Mining

Principios Básicos, Aplicaciones, Técnicas y Casos Prácticos

Rosa María Carrillo García

Trabajo Fin de Grado

Supervisado por Dr. Rafael Blanquero Bravo

Doble Grado en Matemáticas y Estadística

Universidad de Sevilla

Junio de 2021

Publicado en junio de 2021 por
Rosa María Carrillo García

Dr. Rafael Blanquero Bravo,

HACE CONSTAR

que Rosa María Carrillo García, ha realizado bajo mi supervisión el trabajo titulado

Text Mining:
Principios Básicos, Aplicaciones, Técnicas y Casos Prácticos

Una vez revisado, se autoriza el comienzo de los trámites para su presentación como Trabajo Fin de Grado al tribunal que ha de juzgarlo.

Dr. Rafael Blanquero Bravo
Sevilla, a 29 de junio de 2021

Yo, Rosa María Carrillo García con DNI número 26824253C,

DECLARO

mi autoría del trabajo que se presenta en la memoria de este Trabajo Fin de Grado que tiene por título:

Text Mining:
Principios Básicos, Aplicaciones, Técnicas y Casos Prácticos

Rosa María Carrillo García
Sevilla, a 29 de junio de 2021

RESUMEN

Es evidente que actualmente el *Text Mining* es considerado como un ámbito de gran importancia y progresión. En este trabajo, comenzando con la descripción de su composición y relación con otros campos, se analizan sus principios básicos, de preprocesamiento, representación y aplicación, así como las técnicas características para su implementación. De igual forma, con el objetivo de ilustrar algunos de los desarrollos teóricos expuestos, se plantea, como caso práctico, un problema de clasificación múltiple sobre temáticas de libros a través de su sinopsis. No obstante, para poder alcanzar dicho propósito, previamente, resultan esenciales las etapas de tratamiento y resumen de información. Finalmente, para su confirmación, tras la construcción y evaluación de los modelos correspondientes, también resultó de interés llevar a cabo una identificación de temáticas, comprobando de esta forma si realmente es posible captar los conceptos y relaciones existentes entre los géneros preestablecidos.

Palabras clave: *Text Mining*, *Bag Of Words*, *Words Embeddings*, *Word2Vec*, *GloVe*, *Resumen*, *TextRank*, *LexRank*, *Clasificación*, *SVM*, *Topic Modeling*, *Latent Dirichlet Allocation*.

ABSTRACT

At present, it is evident that *Text Mining* is contemplated as an important and continuing field. This project, starting off by its structure and relation with others disciplines, analyses its basic principles, of preprocessing, representation and application, just as the main techniques in order to its implementation. Furthermore, with the purpose to illustrate some of the theoretical descriptions expouned, a multiple classification task about genres through books' synopses is presented as practical approach. Nevertheless, to achive this central objetive, previuos tasks of data processing and summarization for the original information were essential. Finally, so as to its confirmation, after classification models were evaluated, it was also interesting to consider an identification of topics, that which allows to verify if they truly grasp the meanings and connections between the pre-established genres.

Keywords: *Text Mining*, *Bag Of Words*, *Words Embeddings*, *Word2Vec*, *GloVe*, *Summarization*, *TextRank*, *LexRank*, *Classification*, *SVM*, *Topic Modeling*, *Latent Dirichlet Allocation*.

AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a mi familia, pero, en especial, a mis padres, por su constante esfuerzo y sacrificio hacia nosotros, mi hermano y yo. Sin duda, me siento orgullosa de vosotros y, por ello, que seáis mi referencia e inspiración diaria. Habéis sido fundamentales para mi crecimiento como persona y sé que sin vuestro apoyo incondicional no me habría sido posible llegar hasta aquí. Gracias, hoy y siempre.

A mi hermano, por demostrarme que aún siendo el más pequeño es posible conseguir grandes cosas. Tu seguridad y dedicación hacia tus pasiones hacen que esté convencida de que serás capaz de alcanzar todas tus metas y objetivos. Por ello, solo te pido que, en dicho camino, continúes apreciando la vida con la misma alegría e ilusión con las que disfrutas ahora.

A mi segunda familia, porque a pesar de que tan solo hayamos disfrutado juntas algunos años de nuestras vidas, sabemos que vamos a continuar haciéndolo durante el resto de las mismas. Gracias, Alisede, Ana, Bea, ElenaMa, ElenaPa, Eli, Margarita y Raquel, por apoyarme, animarme y entenderme, pero, sobre todo, por permitirme formar parte de esta maravillosa amistad. Ojalá podamos retomar la normalidad cuanto antes y celebrar todas de nuevo lo que en este último año deberíamos haber vivido.

A otra persona vital, Ángela, gracias por saber estar, escuchar y creer en mí. Pese a que todo comenzó por simple casualidad, en poco tiempo se transformó en confianza y amistad. Espero que estos años solo hayan sido el comienzo de todo lo que nos queda por vivir.

A mis compañeros, porque "*sí, estoy segura*". Si no llega a ser por vuestra ayuda, esta etapa universitaria habría sido muchísimo más compleja y desapacible. Dejando atrás todo tipo de diferencias o rivalidades, rápidamente nos convertimos en un gran equipo y, en mi opinión, esa ha sido una de las claves que nos han permitido alcanzar este final.

Por último, pero no por ello menos importante, a mi tutor, Rafael, gracias por ofrecerme esta oportunidad, aconsejarme y confiar en mí desde el primer momento, incluso sin conocernos. Incuestionablemente, sin sus recomendaciones y sinceridad no habría sido posible llevar a cabo este proyecto.

Índice general

Índice de figuras

Lista de Abreviaturas

ANN Artificial Neural Network.

BDM Big Data Mining.

BOW Bag-Of-Words.

CBOW Continuous Bag-Of-Words.

CFS Correlation based Feature Selection.

CNN Convolutional Neural Network.

DL Deep Learning.

DM Data Mining.

FCBF Fast Correlation-Based Filter.

FFNN FeedForward Neural Network.

GRU Gated Recurrent Units.

HMM Hidden Markov Models.

IE Information Extraction.

IR Information Retrieval.

LDA Latent Dirichlet Allocation.

LSA Latent Semantic Analysis.

LSTM Long Short-Term Memory.

ML Machine Learning.

MLP Multi-Layer Perceptron.

MSSA Most Suitable Sense Annotation.

MSSG Multi-Sense Skip-Gram.

MUC Message Understanding Conferences.

NER Named Entity Recognition.

NLG Natural Language Generation.

NLP Natural Language Processing.

NLU Natural Language Understanding.

NN Neural Network.

NNF Non-Negative Factorization.

NNLM Neural Network Language Model.

NP-MSSG Non-Parametric Multi-Sense Skip-Gram.

NS Negative Sampling.

PLSA Probabilistic Latent Semantic Analysis.

PMI Pointwise Mutual Information.

POS Position Of Speech.

RDM Relational Data Mining.

RNN Recurrent Neural Network.

SA Sentiment Analysis.

SVD Singular Value Decomposition.

TF-IDF Term Frequency-Inverse Document Frequency.

TM Text Mining.

VSM Vector Space Model.

WM Web Mining.

Introducción

Nadie duda que los textos escritos en lenguaje natural representan una de las mayores fuentes asociadas a la comunicación y actividad humana. No obstante, el hecho de que la capacidad de análisis para este tipo de información dependa fundamentalmente de la disponibilidad de herramientas y métodos computacionales aplicables a datos de carácter no estructurado, ha conllevado que haya sido en estas últimas décadas cuando este ámbito ha sufrido una mayor progresión.

Los seres humanos procesamos la información textual a través de diferentes patrones de reconocimiento y, como consecuencia de nuestra limitación cognitiva, centrando una minúscula parte de nuestra atención en la capacidad de acceder a amplias cantidades de dicha información. Puesto que cada documento es creado con algún propósito de comunicación, el objetivo principal del análisis de textos es poder comprender, a través del estudio del lenguaje, el sistema más complejo del universo, el pensamiento humano (pg. xxxi, [1]). Evidentemente, dicha aspiración no estaría siendo posible sin el desarrollo tecnológico y la inmensa suma de recursos textuales que este ha permitido generar, tanto en cantidad como en relevancia. Es por ello por lo que, en la actualidad, la minería de textos, *Text Mining* (*TM*) , también denominada por algunos autores a lo largo de su progreso como *Text Data Mining* o, más recientemente, como *Text Analytics*, es considerada como una de las áreas de mayor potencial.

Gracias a los avances previos en ámbitos como la *Extracción de Información (Information Extraction, IE*, Sec. 2.3.2) o resumen de documentos (Sec. 5.4), y haciendo frente a la vía tradicional, la cual se caracterizaba por recurrir a amplios conjuntos de reglas gramaticales, el enfoque empírico fue ganando aceptación y permitió al TM evolucionar drásticamente, a partir de 1990, hacia un enfoque más cuantitativo, dando lugar al comienzo de una nueva era y desarrollando técnicas que lo fusionaron completamente con el *Data Mining (DM*, Sec. 2.2) y el análisis estadístico. Desde su aparición, el DM generó gran interés en la sociedad y esto conllevó que fuese desarrollado y utilizado sólidamente en la industria tecnológica. En cambio, históricamente, el TM comenzó a observarse desde una perspectiva complementaria y experimental, sin considerarse esencial o necesario y con un número reducido de profesionales interesados en el mismo (Sec. 1, [1]).

Afortunadamente, hoy día esta visión ha cambiado por completo y, a causa, por

ejemplo, del aumento de la conectividad o crecimiento de las redes sociales, existe un aliciente real por mejorar y explotar los recursos textuales disponibles lo máximo posible. A través de múltiples técnicas, como la clasificación, reconocimiento de entidades o análisis de sentimientos, entre otras, la minería de textos se extiende a un amplio abanico de campos de aplicación: gestión de riesgos, prevención de ciberdelincuencia y fraude, mejora y automatización de respuestas al consumidor, filtrado de correos no deseados, determinación de la autoría de documentos, etc [2].

No obstante, pese a la gran utilidad y aplicabilidad de estos métodos, sus resultados siempre deben ser interpretados con cierto nivel de autocrítica y precaución. Mas allá de las restricciones técnicas o computacionales que impedían la evolución y desarrollo de este ámbito, las dificultades implícitas al análisis del lenguaje, como ambigüedades y sutilezas, hacen que este se convierta en un proceso realmente complejo y necesario de previos procesamientos lingüísticos. Generalmente estos procesos corresponden a técnicas asociadas al campo tradicional del *Procesamiento del Lenguaje Natural (Natural Language Processing, NLP*, Sec. 2.3.3), que, entre otros, mantiene el objetivo de proporcionar un primer enfoque estructural para conseguir "convertir el texto en números" y poder así recurrir a poderosos algoritmos que permitan obtener conocimiento.

Entre los numerosos avances que se llevaron a cabo para estos ámbitos, diferentes autores defienden que una de sus mayores innovaciones se corresponde con la incorporación del *Aprendizaje Automático (Machine Learning, ML*, Sec. 8) para el desarrollo de técnicas y algoritmos. Esta integración permitió una gran mejora en el análisis, procesamiento y representación de información textual, dando lugar rápidamente a aplicaciones alternativas a los métodos y reglas básicas tradicionales, como fueron las asociadas al análisis sintáctico, *parsing*, o reducción al término raíz, *stemming* (pg. 8, [1]).

Tradicionalmente estas técnicas se combinaban con métodos clásicos, como la representación *Bag-Of-Words (BOW*, Sec. 4.1) y la ponderación *Term Frequency-Inverse Document Frequency (TF-IDF*, Sec. 3.2.2), pero ciertas dificultades, como la alta dimensionalidad (*curse of dimensionality*) y la dispersión (*sparsity*), motivaron la búsqueda de modelos más eficientes lingüísticamente. Por ello, bajo la idea de obtener representaciones *densas* basadas en las similaridades semánticas y contextuales a través de un espacio vectorial continuo, *Vector Space Model (VSM*, Sec. 4.2), se fomentaron distintas vías de desarrollo: (i) continuar con el enfoque ususal centrado en el cálculo de medidas estadísticas relativas a la ocurrencia de palabras, *count-based*, o (ii) centrarse en el modelado del lenguaje con la predicción de palabras o secuencias según su propio contexto, *métodos predictivos* (pg. 232, [3]).

Text Mining

Tal y como se recoge en la introducción, inicialmente, este ámbito puede parecer confuso y desordenado, como consecuencia de la amplia variedad de técnicas y aplicaciones relacionadas con el análisis de los textos escritos. Comúnmente, el *Text Mining (TM)* engloba el proceso completo para poder llevar a cabo el estudio práctico de interés, por lo que necesita partir de la conversión o estructuración de los datos originales y continuar, a través de múltiples etapas intermedias (como pueden ser la derivación de patrones y relaciones), hasta la evaluación e interpretación final de los resultados obtenidos.

En esencia, como propósito fundamental del análisis de textos se establece el hecho de descubrir información originalmente desconocida e implícita, pero no inmediatamente obvia (pg. 4, [1]). Por tanto, desde una perspectiva general, el concepto de *TM* puede definirse como un proceso intensivo de aprendizaje en el que el usuario interactúa con una colección de documentos (*corpus*) a través del uso de herramientas analíticas, permitiéndole así identificar y explorar patrones relevantes para la aportación de nuevo conocimiento (pg. 2, [4]). Las técnicas o aplicaciones del *TM* van más allá de la simple extracción, búsqueda o clasificación de documentos; también tienen como objetivo la abstracción y construcción de relaciones a priori no observables, de forma que no solo se reconocen nuevos aspectos, sino que, además, se permite confirmar percepciones ya intuidas previamente a través de la información disponible.

2.1 Desarrollo Histórico

Para comprender plenamente el significado del TM actual y poder garantizar mejoras y avances futuros, es necesario contemplar su evolución desde el contexto histórico en el que comenzó a desarrollarse (Sec. 1, [1]).

Desde finales del último siglo el TM comenzó a surgir como disciplina de gran progresión. Sin embargo, pese a que en ese momento el término presentaba un elevado interés y expectación, generando curiosidad en la sociedad, gran parte de la misma también mostraba cierto escepticismo. Esta situación desembocó en un lento y duro proceso de aceptación en el que tan solo algunos profesionales decidían apostar por el desarrollo de estos métodos. Ciertos autores señalaban que el hecho de que Hearst

(1999) [5] excluyera la Recuperación de la Información (*Information Retrieval, IR*, Sec. 2.3.1) en su definición general para el concepto de TM, tampoco resultó favorable y, en su lugar, avivó la incertidumbre.

No obstante, sí es claro que el significativo aumento de información textual que comenzaba a generarse requería de procedimientos más avanzados y, por tanto, fue un destacado incentivo para su desarrollo. El gran desafío se encontraba en ser capaz de acceder a la información asociada a una amplia recopilación de textos procedentes de diferentes formatos. Por ello, con el objetivo de hacer frente a este problema, se establecieron como necesidades básicas y esenciales el hecho de poder resumir, para así reducir el cuerpo del texto a un tamaño procesable, y clasificar u ordenar de forma lógica.

Históricamente, las aplicaciones asociadas al resumen y clasificación de documentos comienzan con la catalogación de ejemplares atribuida a Thomas Hyde en 1674. Sin embargo, no fue hasta 1898 cuando comenzó el interés por la obtención de extractos para obras científicas, consiguiendo Luhn en 1958, a través de un novedoso enfoque y recurriendo a información obtenida a partir de datos no estructurados, adaptar el primer ordenador capaz de generarlos. En 1961 Doyle propuso un nuevo avance para la clasificación de documentos, basándose en técnicas de asociación y frecuencia para las palabras, y afirmando además, de forma acertada, que este se correspondía con un proceso más sistemático y automatizado, que, por tanto, permitía obtener mejores resultados. A partir de 1950 comenzaron también a desarrollarse las primeras aplicaciones relacionadas con la teoría de la información sobre textos escritos, como fue la evolución de la bibliometría junto a sus respectivos índices de medición y, una década más tarde, el desarrollo tecnológico de los ordenadores permitió el desarrollo del NLP.

El intento de llevar estas técnicas, como las asociadas al NLP, hacia enfoques centrados en el contexto, alejándose del estudio individual de las palabras, permitió fomentar el desarrollo del TM actual y generar nuevas aplicaciones, como fue la categorización de documentos según ciertas categorías prefijadas o existentes, introducida por Dörre en 1999 [6]. Por ello, junto a la extensión de la representación clásica BOW hacia espacios de variables y características de alta dimensión, uno de los elementos clave para hacer posible esta evolución fue, al igual que en el ámbito del DM, la contribución gradual de la tecnología y algoritmos de ML entre los años 1980 y 2000, destacando como, a pesar de que fueron aplicados fuertemente en los años sucesivos, ya en estos se recogen los primeros casos prácticos para la categorización de documentos, como fueron los presentados por Li et al. en 1991 [7] para redes neuronales o Apte et al. en

1994 [8] para árboles de decisión. De igual forma, paralelamente a dicha incorporación, fueron surgiendo diferentes enfoques estadísticos, como la aplicación de *Modelos Ocultos de Markov (Hidden Markov Model, HMM)* a la extracción de entidades por Bickel et al. en 1997 [9], así como diferentes alternativas centradas en la semántica presente en la información textual.

La primera gran técnica asociada a esta última visión se corresponde con el *Indexado Semántico Latente (Latent Semantic Indexing, LSI)*. Utilizado desde 1988 por Deerwester et al. [10], inicialmente para tareas asociadas a IR, se centra en relacionar semánticamente los términos ocultos en las colecciones de documentos. Profundizando en este planteamiento y analizando la co-ocurrencia de términos, Landauer et al. en 1998 [11] presentaron el *Análisis Semántico Latente (Latent Semantic Analysis, LSA*, Sec. 6.1), rápidamente extendido fuera del ámbito usual del LSI. Con el objetivo de acercar estos razonamientos hacia unos sólidos fundamentos estadísticos, en 1999 Hofmann [12] propuso como modelo generativo la variante probabilística asociada al *Indexado Probabilístico de la Semántica Latente (Probabilistic Latent Semantic Indexing, PLSI)*, respectivamente también denominado *Análisis Probabilístico (Probabilistic Latent Semantic Analysis, PLSA*, Sec. 6.2). No obstante, aunque supuso un gran comienzo hacia la modelización probabilística del texto, distintas dificultades, como el crecimiento lineal del número de parámetros frente al tamaño del corpus, hicieron que en 2003 Blei et al. [13] dieran un paso más aplicando la *Asignación Latente de Dirichlet (Latent Dirichlet Allocation, LDA*, Sec. 6.3). En la actualidad, este progreso junto a la mejora de las estructutras de representación para el corpus, a través de *Words Embeddings* (Sec. 4.3) , han permitido desarrollar numerosas variantes y combinaciones, como es el caso de W2V-LSA [14], basado en las técnicas de *Word2Vec* (Sec. 10.2.1) [15, 16] y *k-means Esférico* [17].

Finalmente, en los últimos años, el continuo afán de investigación por mejorar la eficiencia de las distintas aplicaciones prácticas ha conllevado que, en su mayor parte y al igual que en otros ámbitos, el TM se enfocase, además de en las técnicas generales de ML, hacia las correspondientes al *Aprendizaje Profundo (Deep Learning, DL*, Sec. 10), como son las *Redes Neuronales Recurrentes (RNN*, Sec. 10.1.2) o *Convolucionales (CNN*, Sec. 10.1.3).

2.2 Relación con Data Mining, Web Mining y Big Data Mining

Como se puede observar en la representación por niveles (Fig. 2.1), es posible relacionar el ámbito del TM con varias disciplinas (Sec. 1, [18]). Inicialmente, el análisis recaía sobre datos relacionados o estructurados, *Relational Data Mining (RDM)*, pero

el avance y desarrollo hacia otros formatos de datos, de carácter no estructurado, ha provocado cierta segmentación y división en torno a este campo, permitiendo así diferenciar entre TM y *Web Mining (WM)*. Generalmente, se destaca cómo las técnicas y metodología asociada al ámbito del RDM, han servido de base para la evolución del TM y WM, y estos de forma respectiva para el *Big Data Mining (BDM)*. Todos ellos se engloban dentro del ámbito del DM, definiéndose este como un proceso automático o semiautomático de búsqueda y descubrimento de conocimiento, implícito pero a priori desconocido, a través de datos almacenados electrónicamente, siendo posible establecer patrones o predicciones de utilidad futura (pg. 3, [4]).

A diferencia del TM, el cual hace frente a mayores dificultades técnicas y datos heterogéneos, los asociados al RDM siempre muestran carácter estructurado y homogéneo, i. e., permiten una representación en formato tabular como en el caso de las bases de datos relacionales. Como consecuencia, el RDM suele conllevar tareas de preprocesamiento, como la normalización, ajuste o codificación, de mayor sencillez que las asociadas al TM, *text indexing* (Sec. 3.1) y *text encoding* (Sec. 3.2), y puesto que se centra en datos de fácil acceso, una vez construidos los algoritmos, la solución puede implementarse rápidamente.

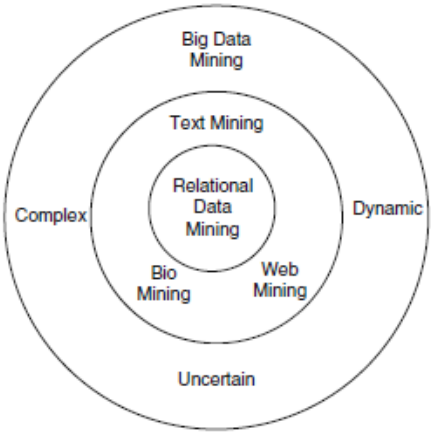


Figura 2.1: Relación con DM, WM y BDM. Fuente: pág. 15 [18].

Por su parte, el WM se centra en el análisis de datos derivados de la Web y en los que, además de la infomación relativa al documento, de carácter textual, son de importancia aspectos como la dirección URL o hipervínculos. Por tanto, es usual que estos no se identifiquen de forma exacta como datos no estructurados, sino que, al hacer referencia a etiquetas o marcas que sí permiten delimitar secciones, se consideran como datos *semi-estructurados*. A diferencia del TM, donde los elemenos son clasificados y organizados recurriendo puramente al contenido, en el WM se priorizan las características o formas, por lo que, en lugar de recurrir únicamente al posicionamiento y estructura gramatical de las palabras, también se tienen en cuenta como atributos, las etiquetas, las fuentes o los colores, entre otros.

Finalmente, el BDM es considerado como el gran desafío del DM recurriendo a los principios básicos que caracterizan al ámbito del Big Data, i. e., la regla de sus 5 dimensiones: Volumen, Velocidad, Variedad, Veracidad y Valor. En el TM se asume que

los datos son fijos o constantes, mientras que en este campo se considera que pueden actualizarse de forma continua o frecuente, siendo destacable además que las fuentes de recopilación de datos son mucho más extensas que las correspondientes al TM, incluyendo, por ejemplo, sensores o etiquetas RFID.

2.3 Estructura y composición del TM

Como consecuencia de la disparidad y amplitud en su rango de aplicación, el hecho de proporcionar una adecuada caracterización y estructura para el TM resulta de gran dificultad, incluso para expertos de este ámbito. Por ello, han sido varios los autores que, desde la coherencia y sentido práctico, han intentado establecer aproximaciones a dicho aspecto, consiguendo resultados generalmente aceptados.

Heart [5] consideraba que los textos son fuentes ampliamente ricas de información, pero con la desventaja de que esta viene codificada de forma altamente complicada para el descifrado automático. Además, señaló la importancia de estabecer las diferencias entre los conceptos de acceso o recuperación de información y "lingüística computacional" con el correspondiente al TM. Este autor estableció cómo el objetivo del acceso a la información es ayudar a los usuarios a satisfacer sus necesidades a través de la búsqueda de información previamente conocida e insertada en los documentos. Por su parte, centró la lingüística computacional en el estudio de resultados estadísticos obtenidos sobre colecciones de documentos con el objetivo de descubrir patrones útiles para desarrollar o mejorar diferentes algoritmos, como es el caso de los subproblemas del NLP correspondientes al proceso de etiquetado gramatical o la resolución del conflicto de la ambigüedad semántica. En cambio, como se ha comentado anteriormente, el TM tiene como propósito derivar nueva información examinando colecciones de documentos y descubrir aquella que no está contenida en ninguno de los mismos, siendo posible así determinar las relaciones ocultas y establecer vínculos visibles. Evidentemente, Heart también añadió que, pese a ello, en ciertas ocasiones los resultados de las aplicaciones del TM generan herramientas que sirven de ayuda en el proceso de recuperación de la información, como son la generación automática de asociaciones entre términos, para las consultas, o las técnicas de clustering, para generar agrupaciones. Finalmente, concluyó que el TM no puede corresponderse exclusivamente ni a IR, IE o clasificación de documentos, puesto que estos procesos no implican por sí mismos estrictamente el descubrimiento de nueva información.

Por consiguiente, a lo largo de su evolución se optó por plantear una perspectiva inclusiva y considerar que, en definitiva, el TM debe verse como el ámbito que engloba

todas aquellas tareas y técnicas relacionadas principalemnte con: la información textual y la lingüística computacional, y, por tanto, los campos que permiten su análisis y procesamiento, Interligencia Artificial, Machine Learning y Data Mining (Fig. 2.2). De esta forma, como resultado de dichos vínculos e interesecciones, G. Miner et al. (Sec. 2, [1]) propusieron un enfoque en el que consideraban 7 posibles áreas prácticas relativas al TM, incluida la asociada al WM, la cual, como se ha comentado anteriormente, según otros autores sí debería diferenciarse respecto a este. No obstante, cabe destacar que estas se encuentran altamente interrelacionadas y, por tanto, de forma general, un proyecto asociado a aplicaciones de TM requerirá técnicas asociadas a más de una de ellas.

Además, estos autores establecieron cómo a través de 5 preguntas directas orientadas al objetivo y recursos disponibles, es posible decidir sobre qué área del TM debería enfocarse la tarea de interés. Afirman que la cuestión principial es determinar el alcance del caso práctico: el documento completo, o bien estrucuturas con mayor nivel de desglose, como palabras, oraciones o párrafos. En el primer caso proponen continuar dependiendo de si el propósito es de búsqueda: *(A1.) Recuperación de Información*, u ordenación: *(A2.) Clasificación* o *(A3.) Clustering*, según se dispongan o no de categorías, respectivamente. En cuanto al enfoque alternativo, se procede distinguiendo si se desea identificar conceptos específicos: *(A4.) Extracción de Información*, o bien centrarse en la estructura: *(A5.) Procesamiento del Lenguaje Natural* o significado: *(A6.) Extracción de conceptos*.

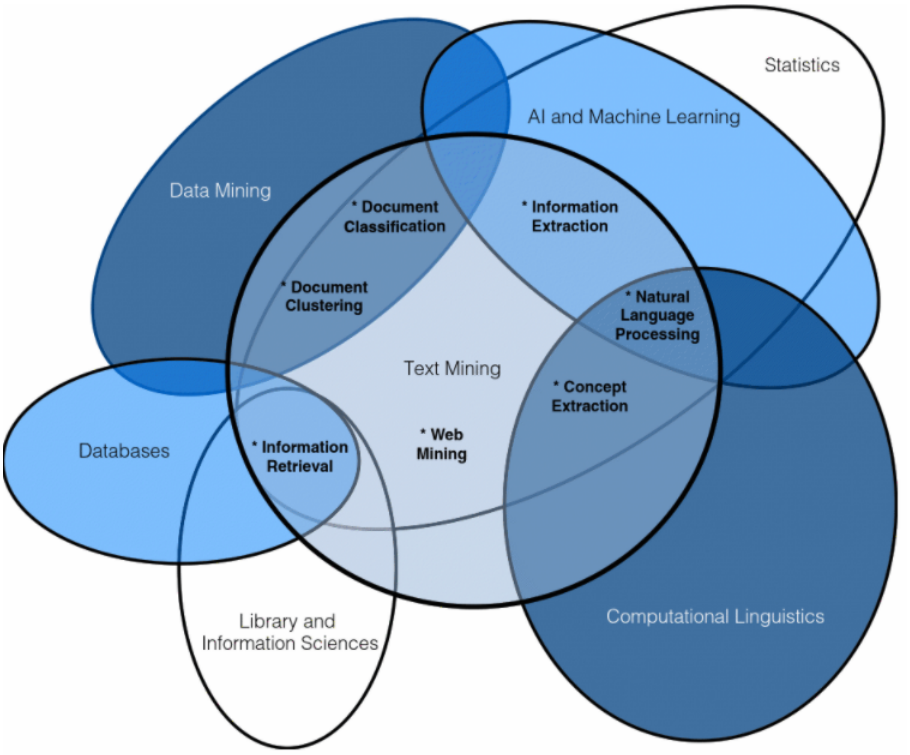


Figura 2.2: Relación: TM y otros campos. Fuente: pág. 31 [1].

2.3.1 Recuperación de Información

Como se ha comentado en la introducción previa, rápidamente se establecieron las diferencias entre TM e IR, aunque cabe destacar cómo la esencia de estas discrepancias se encuentra en los conceptos de "mining" y "retrieval". El resultado implícito del DM en general, y, por tanto, del TM, es conocimiento, siendo usual su aplicación en la toma de decisiones. Sin embargo, el resultado asociado a "retrieval" corresponde a los aspectos presentes y relevantes dada una determinada consulta (*query*). Por tanto, este ámbito, IR, se centra en distinguir como respuesta para una consulta lógica a un conjunto adecuado, "recuperando" así los que resultan de destacada importancia.

Si el objetivo es el documento completo, la aplicación recae en la recuperación de documentos, *document retrieval*, mientras que en la actualidad el término IR también incluye la recuperación de imágenes, audio, e incluso documentos que contienen anotaciones de otros textos (Sec. 1, pg. 8 y 20, [19]).

2.3.2 Extracción de Información

Inicialmente, puede establecerse que este ámbito tiene como objetivo "extraer" información específica del texto, de forma que esta permita un posterior análisis para así identificar tendencias y generar otros resultados de posible interés. Según Sanger y Feldman (pg. 9 [1]), la IE consiste en llevar a cabo un conjunto ordenado de etapas diseñadas expresamente para extraer términos, atributos, hechos o eventos, señalados y presentes en el texto.

Aunque las primeras aplicaciones fueron surgiendo en torno a 1970, el gran estímulo y evolución para la IE se estableció a partir de 1987 con las *MUC*s (*Message Understanding Conferences*), donde se comenzó a centrar la atención en el análisis de texto, en general para mensajes militares, permitiendo así iniciar nociones y conceptos de gran importancia actual, como es el reconocimiento de entidades (*Named Entity Recognition, NER*).

2.3.3 Procesamiento del Lenguaje Natural

A priori, este ámbito podría verse como una disciplina híbrida entre el desarrollo de elementos ligüísticos y la ciencia de la información, cuyo objetivo se centra en entender cómo el lenguaje humano natural es aprendido y cómo podría llegar a modelarse (pg. 7, [1]). Más concretamente, el NLP está formado por las funciones encargadas de analizar y sintetizar el lenguaje escrito o hablado. El epíteto "natural" se establece pa-

ra distinguir el lenguaje humano de otros formales, como son las notaciones lógicas o matemáticas (pg. 3, [19]).

La importancia del NLP radica en que, de forma rutinaria, el resto de aplicaciones o áreas relativas al TM necesitan de forma obligada metodologías pertenecientes al mismo. En definitiva, para el análisis lingüístico, los procesamientos asociados al NLP se encargan de ayudar al ordenador a "leer" el texto, de manera que, a través de las técnicas posteriores, sea posible, por ejemplo, llegar a la comprensión de su significado. No obstante, para conseguirlo es necesario poder descifrar las ambigüedades propias del lenguaje humano y, por ello, que resulte esencial disponer de consistentes bases de información, como son los diccionarios, de vocabulario y sinónimos, conjuntos de reglas lingüísticas y gramaticales, ontología o incluso listados actualizados de entidades.

A su vez, este no debe confundirse con los conceptos de *Compresión del Lenguaje Natural (Natural Language Understanding, NLU)* y *Generación del Lengaje Natural* (*Natural Language Generation, NLG*). El primero de ellos, *NLU*, hace referencia al hecho de que un sistema informático sea capaz de *comprender* el lenguaje natural tal y como lo hace un ser humano. Por ello, se considera que este ámbito se corresponde con una componente del NLP pero centrada en identificar el significado de la comunicación del usuario para poder, así, proporcionar respuestas adecuadas en base a sus intenciones [20]. Por su parte, el concepto de NLG va asociado a la capacidad de producir información textual en base a la disponible, centrándose, por tanto, en la *escritura* del lenguaje natural.

Parte I

Principios Básicos

Preprocesamiento en TM

Dado el carácter no estructurado de la información textual, toda aplicación asociada al ámbito del TM necesita de procesos previos que permitan la transformación de los datos originales para su posterior análisis y estudio.

No obstante, previamente a estas etapas de estructuración, resulta esencial disponer de los recursos sobre los que llevar a cabo dicho tratamiento, i. e., obtener los datos e información. Más notablemente que para el caso de los datos estructurados, es evidente que existe una amplia heterogeneidad en relación al formato para el almacenamiento de esta clase de información. Como consecuencia, además de la implementación de todos los procedimientos posteriores, más cercanos a enfoques lingüísticos y estadísticos, es fundamental tener un adecuado conocimiento de las propiedades y configuraciones asociadas a dicho tipo de archivos, como son JSON, XML o HTML, para conseguir así una correcta lectura inicial de los datos a procesar (pg. 6, [18] y pg. 76, [4]).

3.1 Text Indexing

La primera etapa asociada al procesamiento del texto se centra en poder descomponerlo y desglosarlo en unidades de estudio de interés: párrafos, oraciones, expresiones o, más usualmente, palabras, consiguiendo así una indexación adecuada y eficiente para la información textual inicial. El resultado de este proceso es la base para las posteriores opciones de representación y estructuración de los datos.

Como consecuencia de la amplia variedad presente en el lenguaje, distinguida desde la trascendente obra de Noam Chomsky (1957) [21], en la que ya se diferenció entre oraciones con correcta e incorrecta estructura sintáctica e introduciendo así la gramática generativa, se establece que la forma y significado de estas unidades de estudio son aspectos separables, i. e., primero debe analizarse la estructura sintáctica asociada a la oración y, posteriormente, recurrir a su análisis semántico. Por ello, es habitual seguir un orden de etapas lingüísticas: análisis sintáctico (análisis gramatical), análisis semático (análisis del significado) y análisis pragmático (análisis del efecto e influencia del contexto en la interpretación del significado) (pg. 4, [19]).

Actualmente esta etapa concentra parte de los procesos clásicos y tradicionales, de entre los que destacan (Sec. 2, [4] y pg. 570-571, [3]):

Tokenization: consiste en la segmentación del texto original, generalmente a través de marcas delimitadoras, como el espacio en blanco o signos de puntuación, hacia unidades simples, individuales y carentes de significado, *tokens*. No obstante, pese a su sencillo planteamiento, este proceso también presenta diversas dificultades. Para ciertos lenguajes, como los orientales, estas consideraciones para la delimitación y separación no son suficientes y es necesario un estudio más profundo orientado hacia un análisis morfológico. Además, los propios signos de puntuación pueden generar confusión, puesto que no necesariamente deben indicar el fin de oración o párrafo. Más concretamente, el punto ya presenta ambigüedad puesto que podría ser parte de algún valor numérico decimal o alguna abreviatura.

Stemming: tiene como objetivo procesar cada *token* a través de las reglas y diccionarios considerados para asociarlo a su forma raíz. Al igual que la mayoría de estos procesos, el hecho de no tener en consideración el contexto de la palabra impide resolver las ambigüedades, conllevando que, en ocasiones, resulten múltiples raíces para un mismo término.

Stop-Words Removal: busca la eliminación de las palabras comunes al lenguje y no relevantes para el análisis, denominadas *stop-words*. Generalmente corresponden a artículos, conjunciones, preposiciones o pronombres, y el objetivo es establecer una lista con la que comprobar la original y eliminar de esta toda aquella que se encuentre en la considerada como referencia. Esta idea parte del hecho de que ciertas palabras aparecen de forma usual en todo tipo de textos, son comunes, y, por tanto, no aportan distinción y capacidad de información para el estudio en cuestión.

Eliminación de caracteres especiales: se centra en un razonamiento similar al correspondiente a la tarea anterior, pero desde una visión más específica. El objetivo radica en excluir o tratar, de forma particular, los tokens o términos que añaden errores o ruido como consecuencia de presentar caracteres y símbolos propios del idioma en cuestión. Este procedimiento permite facilitar el posterior análisis y mejorar el control de la coherencia de la información disponible.

Position Of Speech Tagging (POS tagging): se centra en la clasificación de las palabras según su categoría gramatical (sustantivos, verbos, adjetivos, adverbios,etc). Generalmente existen dos enfoques para afrontar este etiquetado gramatical (pg. 13, [19]): (i) basándose en reglas lingüísticas e intentando aplicar conocimiento

para descartar secuencias sintácticamente incorrectas, y (ii) llevar a cabo un etiquetado estocástico, recurriendo a datos de entrenamiento y considerando probabilidades y frecuencias para eliminar ambigüedades. Algunos métodos usualmente utilizados se basan en HMM [22] o máxima entropía [23], puesto que permiten tener en cuenta el contexto tomando como indicadores y referencias las palabras del entorno.

Normalización: como consecuencia de la amplia variedad lingüística, en la mayoría de las ocasiones, ya bien al disponer de infomación múltiple, i. e., diversas fuentes, o dentro de un mismo documento, podría resultar necesario llevar a cabo una unificación de la nomenclatura para los términos y conceptos a los que se hace referencia en los mismos. Este proceso, además de disminuir el número de unidades a considerar y, por tanto, conseguir una reducción de la redundancia, permite mejorar la coherencia en el estudio. Algunos ejemplos pueden ser la expansión de contracciones en diversos idiomas, o el renombrado uniforme de entidades propias.

No obstante, por lo general, salvo para la primera tarea indicada, *tokenization*, su aplicación no es estrictamente necesaria. En cuanto al orden y complejidad para el resto de las mismas, se tiene que estos pueden variar según el carácter del caso práctico de interés. En ocasiones, con el objetivo de evitar confusiones básicas, como las que ocurren entre los sustantivos acabados en "s" y formas en plural, la etapa de *stemming* necesita de un previo etiquetado gramatical, *POS Tagging*. Por otra parte, según el propósito y localización en el texto, también es usual que la eliminación de *stopwords* requiera de un estudio previo para determinar qué terminos específicos podrían resultar relevantes para la aplicación y, por tanto, deberían o no incluirse en dicha lista. Un ejemplo para el caso particular anterior, correspondería a una tarea de análisis de sentimientos, donde palabras como "pero" o "no" sí deberían ser consideradas pese a que son frecuentes en la mayoría de las colecciones y no tienen excesiva relevancia para otro tipo de tareas.

Otras alternativas basadas en el cálculo de pesos o ponderaciones para las palabras, en lugar del simple y previo conocimiento del lenguaje, consisten en la eliminación de las no relevantes a través del filtrado según su grado de importancia, *index filtering*, o recurrir a herramientas de búsqueda para la expansión hacia otros términos altamente relacionados con las palabras de mayor peso o ponderación, *index expansion*, recibiendo la combinación de ambas el nombre de *index optimization*. En cuanto a los criterios de ponderación y cuantificación de la relevancia para las unidades a añadir o eliminar,

puede recurrirse a los procesos de asignación descritos en la siguiente sección (Sec. 3.2.2) pero desde el enfoque adecuado, i. e., centrándose en las que obtienen valores mayores o inferiores, respectivamente.

3.2 Text Encoding

La segunda etapa en el proceso de preparación asociado al TM consiste en estructurar por completo la información textual partiendo de la disponible tras el proceso previo de indexado, *text indexing*. Ahora el objetivo se centra en seleccionar algunas de las unidades resultantes como características o atributos para el estudio, determinando así la dimensión del espacio vectorial de representación para el texto o documento, y asignar a cada una de ellas el valor numérico correspondiente.

Independientemente de la complejidad de los atributos y, como consecuencia de las numerosas posibles combinaciones entre palabras o relaciones gramaticales, los estudios relativos al TM se enfrentan a una alta dimensionalidad del espacio ambiente. De igual forma, esta alta dimensionalidad respecto al propio tamaño del documento provoca una gran escasez de componentes no nulas para la representación vectorial asociada, i. e., son representaciones *sparse*, lo que conlleva una gran dificultad a la hora de la identificación de patrones o relaciones en los datos. Además, al ser producciones humanas, es usual encontrar también errores en los textos, como de escritura o carácter gramatical, por lo que inevitablemente se presentan grandes cantidades de errores o perturbaciones, *ruido* (pg. 4, [4]).

En el inicio de este proceso, las leyes de *Heaps* y *Zipf*, dos de las leyes estadísticas más importantes para el análisis del lenguaje natural, permiten obtener una perspectiva inicial sobre los datos y sus características ayudando así a una adecuada selección de atributos (Sec. 3.3, [4]). La *Ley de Heap* relaciona el tamaño total de la colección, N, y el número de palabras únicas, M, estableciendo como cuánto mayor sea el tamaño de la colección más difícil será encontrar palabras distintas a las anteriores. Por su parte, la *Ley de Zipf* afirma que, teóricamente, la frecuencia de una palabra (unidad lingüística de interés), t_i, en el texto d_j, i. e., tf_{ij}, resulta inversamente proporcional a su rango, i, determinado según la ordenación descencende de frecuencias.

$$\text{Ley de Heap:} \qquad M = k \cdot N^b$$

$$\text{Ley de Zift:} \qquad tf_{ij} = \frac{C}{i^a}$$

donde k, b, C y a son constantes a determinar a partir de los datos.

Esta última ley, determina cómo las palabras relativas al contexto aparecen en un número significativo de ocasiones, pero la gran mayoría restante presentan baja frecuencia, indicando, por tanto, que la probabilidad asociada a que cada palabra aparezca en el documento se encuentra fuertemente sesgada. Según la *Ley de Zipf* las 10 palabras de mayor importancia del corpus representan una tercera parte del total, de manera que su identificación y consideración para el estudio resulta de vital importancia, puesto que, en el caso de no considerarse, para conseguir representar el mismo nivel de frecuencia y relevancia, debería seleccionarse una amplia cantidad de atributos, lo que a su vez conllevaría, en comparación, una alta dimensionalidad. Por otro lado, además de evitar un aumento innecesario de dimensión, la elección de estas características también permite un menor número de valores nulos a lo largo de su representación, lo que hace mejorar el problema de la dispersión, *sparsity*.

3.2.1 Selección y Derivación de características

Puesto que estas dificultades, alta dimensionalidad y dispersión, suponen serios problemas de complejidad a la hora de recurrir a los algoritmos de ML (como la lentitud y sobreajuste) empeorando así su aprendizaje y resultado, es necesario conseguir reducir el número de características representativas de los documentos. Para ello resulta fundamental eliminar aquellas que presentan redundancia, correlación respecto al resto, irrelevancia, y, por consiguiente, baja importancia para el análisis. Para conseguirlo, son usuales dos posibles enfoques: *Derivación o Extracción* y *Selección* de variables.

La extracción de variables tiene como objetivo derivar nuevas características a través del análisis y estudio de las disponibles inicialmente. Pese a que proporciona buenos resultados, uno de sus inconvenientes es la dificultad y baja interpretabilidad asociada a las nuevas variables creadas. Por su parte, la selección de variables consiste en la elección del subconjunto que mejor se adecúa al documento según cierta medida de efectividad y, por tanto, al no alterar la representación original de las mismas mantiene la posibilidad de una correcta interpretación (Sec. 14, [4]).

Algunas de las técnicas más usuales para la extracción de características recaen en métodos basados en la descomposición de la matriz de representación, como pueden ser *Descomposición en Valores Singulares (Singular Value Decomposition, SVD)*, *Análisis de Componentes Principales (Principal Component Analysis, PCA)*, *Non-Negative Matrix Factorization (NMF)*, *T-distributed Stochastic Neighbor Embedding (t-SNE)* o técnicas propiamente más específicas para este contexto (lenguaje natural) como son el *Análsis Semán-*

tico Latente (Latent Semantic Analysis, LSA) y la *Asignación Latente de Dirichlet, (Latent Dirichlet Allocation, LDA)* (Sec. 3.2, [18]). Por ejemplo, algunas aplicaciones de estas técnicas corresponden a los estudios de Wang et al. [24] y Barman et al. [25] para la clasificación de documentos basados en PCA y NMF, respectivamente.

En cuanto a las técnicas de selección de atributos [26, 27], estas tratan de seleccionar el subconjunto de atributos que proporciona un mejor resultado. Se trata, en esencia, de un problema de naturaleza combinatoria, por lo que podría plantearse la obtención de tal subconjunto mediante una exploración exhaustiva, generando todas las posibles combinaciones de atributos y evaluándolas según un cierto clasificador. Sin embargo, la alta dimensionalidad del problema hace que este planteamiento sea inviable desde el punto de vista práctico; en su lugar, la exploración se realiza mediante alguna técnica heurística. Este enfoque se corresponde con los denominados métodos *wrappers*. Frente a estos, se sitúan los denominados filtros, *filters*, que son técnicas basadas en la ordenación de los atributos según cierta medida que indique la relación de cada uno de ellos con la variable respuesta, sin la intervención, por lo general, de ningún clasificador. Finalmente, es preciso mencionar los denominados métodos empotrados, *embedded methods*, que se caracterizan por incorporar dentro del clasificador el procedimiento de selección de atributos, constituyendo una unidad.

Filters

Llevan a cabo la evaluación de los atributos antes del entrenamiento del modelo y, por tanto, son independientes del clasificador considerado. Sin embargo, pese a presentar gran rapidez y simplicidad, no permiten detectar ni la redundancia ni interacción entre atributos. Según la contribución de los atributos a la variable respuesta se considere de forma individual o colectiva, es posible, a su vez, distinguir entre:

Univariantes: determinan la importancia de los atributos a partir de distintas medidas, como el estadístico χ^2 o la entropía, para atributos discretos, y de correlación de atributos con la variable respuesta, para los de carácter continuo.

Respecto a las relativas al reflejo de la entropía es usual considerar: la ganancia de información, *information gain* (cantidad en la que decrece), el ratio de ganancia, *gain ratio* (proporción de ganancia de la información respecto a la entropía original) e incertidumbre simétrica, *symmetrical uncertainty* (propuesta para corregir el sesgo de las anteriores en favor de los atributos que presentan un mayor número de modalidades) (Fig. 6.1).

Algunos ejemplos empíricos a través de las medidas χ^2 e *information gain*, entre

otras, para las tareas de clasificación y clustering de documentos son los correspondientes a los estudios de Forman [28] y Liu et al. [29], respectivamente.

Multivariantes: basados en la idea de que un buen subconjunto de atributos debe presentar alta correlación con la variable respuesta pero escasa entre los atributos que lo componen, dos posibles alternativas corresponden a *CFS (Correlation based Feature Selection)* [30] y *FCBF (Fast Correlation-Based Filter)* [31].

La técnica *CFS* recurre, según la naturaleza de cada atributo, a la medida de correlación más adecuada (*Pearson, Spearman o simmetrical uncertainty*) y se centra en la exploración del espacio de atributos según la técnica heurística denominada *Best First Search*. Por su parte, *FCBF* trata de elegir, a través de la medida *simmetrical uncertainty*, los atributos relevantes a medida que elimina los redundantes según el concepto de correlación predominante.

Un ejemplo de estudio comparativo entre técnicas basadas en la correlación de variables para el campo de la clasificación de texto, es el presentado por S. Sharma y A. Jain [32] en relación a un análisis de sentimientos para tweets.

Función	Fórmula
Entropy	$H(X) = -\sum_i P(x_i) log(P(x_i))$
Conditional Entropy	$H(X\|Y) = -\sum_j P(y_j) \sum_i P(x_i\|y_j) log(P(x_i\|y_j))$
Information Gain	$IG(X\|Y) = H(X) - H(X\|Y)$
Symmetrical Uncertainty	$SU(X,Y) = 2\left[\frac{IG(X\|Y)}{H(X)+H(Y)}\right]$

Figura 3.1: Funciones relativas a la Entropía. Fuente: Elaboración Propia.

Wrappers

Los algoritmos tipo *wrapper* exploran el espacio formado por las posibles combinaciones de atributos, tratando de encontrar aquella que optimice el rendimiento de un cierto clasificador. En general, proporcionan mejores resultados que los métodos anteriores, *filters*. Dada la naturaleza combinatoria del problema, en la exploración del espacio de atributos suelen emplearse técnicas metaheurísticas, entre las que pueden destacarse las siguientes:

Forward y Backward Search: Son técnicas tipo *greedy* que, de forma iterada, añaden

(respectivamente, suprimen) el atributo que proporciona una mayor incremento (respectivamente, menor disminución) en el rendimiento del clasificador. El proceso continúa hasta que la adición o eliminación, respectivamente, no supone un cambio significativo en la capacidad predictiva del modelo.

Best First Search: técnica heurística similar a la búsqueda hacia delante pero permitiendo la expansión con atributos individuales y retroceso hacia pasos anteriores en caso de que esta (la expansión del subconjunto tras la incorporación del atributo) no mejore el subconjunto de partida. Con el objetivo de evitar la exploración del espacio completo de posibles combinaciones, es usual también limitar el número de subconjuntos expandidos sin mejora.

Un caso de estudio para la categorización de documentos basado en esta técnica es el presentado por Hawashin et al. [33], donde se utiliza junto a un clasificador SVM.

Recursive Feature Elimination: se trata de un algoritmo de tipo *backward* propuesto en 2002 por Guyon et al. [34] para SVM, pero que puede generalizarse para cualquier clasificador. La clave se centra en ordenar los atributos según cierta medida de importancia de forma que, en cada iteración, únicamente se retiene un determinado número de los mismos, S_i. A partir de los considerados, se lleva a cabo el ajuste del modelo así como la medición de su rendimiento, seleccionando, finalmente, el valor para S_i que mejor valor proporcione.

Un ejemplo de aplicación para el ámbito del NLP corresponde al estudio sobre la autoría de documentos realizado por Bedo et al. [35].

Algoritmos Genéticos: se corresponden con procedimientos de optimización basados en los principios de la evolución biológica donde las distintas combinaciones de atributos son representadas mediante *cromosomas*.

Partiendo de una población inicial de cromosomas, esta se va desarrollando según los principios biológicos básicos de evolución: *selección* (evaluando la aptitud de cada cromosoma y seleccionando los más válidos para dar lugar a la siguiente generación), *cruzamiento* (operando sobre dos cromosomas da lugar a los dos descendientes en los que se combinan las características de los progenitores) y *mutación* (modificación aleatoria por parte de los cromosomas), para finalmente seleccionar los mejores individuos resultantes de este proceso y dando lugar, así, a la siguiente generación.

Algunos ejemplos de aplicación en la tarea de clasificación de documentos corresponden a los propuestos por Labani et al. [36] y Basiri et al. [37].

3.2.2 Criterios de asignación numérica

Una vez seleccionados los atributos con los que representar el documento, o teniendo en cuenta el total de unidades en el caso inicial del proceso de indexado, deben establecerse los valores numéricos asociados al grado de importancia para cada uno de los mismos (Sec. 3.3.1 y 2.2.4, [18] y Sec. 3.14, [4]).

De forma genérica, el peso o ponderación, ω_{ij}, asociado al término t_i del documento d_j está determinado por tres componentes:

- *Peso local*, lw_{ij}: está asociado a la frecuencia de dicho término en el propio documento y, por tanto, tiene en cuenta únicamente la importancia de este en el mismo.
- *Peso global*, gw_{ij}: refleja la capacidad discriminatoria del término basándose en la distribución del mismo a lo largo de toda la colección o corpus. Se centra en disminuir la importancia asociada a términos generales y potenciar la correspondiente a aquellos de aparición más específica.
- *Factor normalizador*, n_j: corrige el impacto de documentos de diferentes longitudes.

Finalmente, estos componentes se combinan para dar lugar a la fórmula:

$$\omega_{ij} = \frac{lw_{ij} \cdot gw_{ij}}{n_j}$$

La particularización de las distintas componentes da lugar a diferentes criterios. Entre la gran variedad existente, los más usuales corresponden a:

Ocurrencia o Modelo Binario: es la forma que presenta mayor sencillez y, aunque no proporciona demasiada información, es considerada la base para cuantificar la utilidad del término. Corresponde a asociar *valores binarios* según se tenga presencia o no en el documento, indicándose $\omega_{ij} = 1$, clase positiva, si el término asociado aparece en el texto y $\omega_{ij} = 0$ en caso contrario.

Frecuencia absoluta: con este procedimiento se toma como medida *Term Frequency* (*TF*, haciendo referencia a la representatividad del término en el documento) el número de ocasiones en las que aparece el término en el texto, $\omega_{ij} = tf_{ij} = f_{ij}$.

No obstante, este razonamiento se basa en el hecho de que, a mayor frecuencia, mayor importancia, por lo que en determinadas ocasiones, una elevada relevan-

cia para el documento podría ser consecuencia de la alta frecuencia individual obtenida por un único término. En dicho caso, se reflejaría una falsa adecuación global frente a otros documentos que, pese a no superar dicha importancia máxima, sí obtienen valores uniformes para un mayor número de términos y, por tanto, podrían permitir un mejor resultado futuro.

Frecuencia relativa: puesto que los valores anteriores podrían estar altamente influenciados por la longitud del texto, otra opción para llevar a cabo el cálculo de la medida *TF* consiste en recurrir a la normalización usando frecuencias relativas, como son los casos de $\omega_{ij} = tf_{ij} = \frac{f_{ij}}{f_{maxj}}$ donde f_{maxj} hace referencia a la máxima frecuencia en el documento, d_j, o $\omega_{ij} = tf_{ij} = \frac{f_{ij}}{\sum_{k \in d} f_{kd}}$, según el número total de ocurrencias en el documento, i. e., su extensión o longitud. De igual forma, también existen otras muchas variantes como alternativas a las anteriores, como es la obtenida al considerar un parámetro de doble normalización, $k + (1+k)\frac{f_{ij}}{f_{maxj}}$ [38].

Inverse Document Frequency (IDF): consiste en calcular el logaritmo del inverso de la probabilidad relativa de que el término aparezca en un documento, de forma que mide si dicho término es o no común en la colección de documentos. Si N es el número total de documentos o textos de la colección y df_i el número de textos de la misma que incluyen al término t_i, se establece la medida como $\omega_i = idf_i = log\left(\frac{N}{df_i}\right)$ o, más usualmente, con el objetivo de evitar la división por valores nulos, $\omega_i = 1 + log\left(\frac{N}{1+df_i}\right)$.

Term Frequency-Inverse Document Frequency (TF-IDF): es el resultado de la combinación de los criterios anteriores. Los pesos asociados son proporcionales de forma directa a las ocurrencias de dicho término en el texto en cuestión, pero de forma indirecta a las correspondientes al corpus. Siendo tf_{ij} la frecuencia asociada a t_i en el texto d_j, se obtienen los pesos *TF-IDF* originales, ω_{ij}, así como sus variantes, $\overline{\omega}_{ij}$:

$$\omega_{ij} = tf_{ij} \times idf_i \qquad \overline{\omega}_{ij} = \Big(1 + log\big(tf_{ij}\big)\Big)\left(1 + log\left(\frac{N}{1+df_i}\right)\right)$$

$$\overline{\omega}_{ij} = \Big(1 + log\big(tf_{ij}\big)\Big)\left(1 + log\left(\frac{N-df_i}{1+df_i}\right)\right)$$

REPRESENTACIONES

Tal y como se ha indicado en secciones previas, los formatos de representación para la información textual también han evolucionado a lo largo de los distintos años de exploración para el TM. Partiendo de enfoques clásicos y consolidados, gracias al afán por mejorar las estructuras conocidas, fue posible el desarrollo de técnicas más innovadoras, capaces no solo de recoger mayor cantidad de información, sino también de ampliar dicho rango de accesibilidad hacia relaciones o vínculos no disponibles hasta el momento, i. e., imposibles de capturar con los procedimientos anteriores, como eran, por ejemplo, las correspondencias semánticas.

4.1 BAG-OF-WORDS/BAG-N-GRAMS

Es la representación tradicional utilizada en el ámbito del NLP para conseguir una aproximación inicial a la estructuración del texto. Consiste en la descomposición o segmentación de la información original disponible, de forma que sea posible obtener una lista o conjunto, *bag*, que recoja las diferentes unidades de interés sin tener en cuenta su orden y significado. Cada documento o texto se representa por medio de una lista considerando la multiplicidad de cada uno de los términos. Para ello es fundamental el proceso de *tokenization* y, dada la amplia extensión de las producciones textuales, suele procederse también a la eliminación de *stop-words*.

De forma general, se opera sobre las unidades más simples y, por tanto, se consideran las palabras individualmente para la construcción de la lista, dando lugar a la "bolsa de palabras", *Bag-Of-Words (BOW)*. Sin embargo, esta representación conlleva una pérdida total del contexto, por lo que, en ocasiones, en lugar de llevar a cabo el análisis término a término, también resulta de interés considerar secuencias de palabras consecutivas, obteniéndose así a la "bolsa de N-gramas", *Bag-N-grams*.

Aunque usualmente se hace referencia a los N-gramas de palabras como secuencias de N palabras consecutivas, en realidad, pueden utilizarse otro tipo de N-gramas, como los asociados a letras o caracteres, los cuales han sido utilizados en multitud de aplicaciones, como son la detección de código malicioso o la identificación de autoría, mostrando como no dan lugar a una excesiva dimensionalidad y mejoran la dispersión de los datos. No obstante, el concepto de N-gramas también puede verse como

un simple modelo estadístico del lenguaje que se centra en determinar las probabilidades asociadas a secuencias de palabras o caracteres, ayudando así por ejemplo a la predicción del siguiente término dada una secuencia previa (Sec. 3.3, 3.4 [4]).

Puesto que suele proporcionar buenos resultados, esta representación se considera la base para la derivación de otros tipos más elaborados capaces de recoger la información de forma más exacta y estructurada.

4.2 Vector Space Model

Una vez que se dispone de la lista o conjunto de características seleccionadas para la representación del documento, el objetivo es obtener para este una representación vectorial de forma que cada componente haga referencia a una de dichas características. De esta forma, el vector asociado al documento puede verse como la combinación de los vectores, *one-hot*, asociados a cada uno de los términos, presentando estos como no nula tan solo la componente correspondiente. Para completar este proceso es necesario considerar algún criterio de asignación numérica de la relevancia (Sec. 3.2.2) así como aspectos de referencia comunes para poder dotar de uniformidad al proceso en caso de considerar una colección de documentos. Por lo general, al presentar un corpus, cada uno de los documentos estará representado por características específicas, por lo que para poder realizar un análisis conjunto es necesario considerar un espacio de representación global que recoja todas y cada una de las mismas, i. e., *vocabulario*. El tamaño asociado a este conjunto genérico corresponde a la *dimensión global* del espacio vectorial de representación.

Puesto que cada documento se representa de forma vectorial, llevando a cabo la unificación de los conjuntos de características para que todos ellos pertezcan al mismo espacio ambiente, es posible establecer una representación conjunta que recoja toda la información relativa a los documentos de interés incluidos en la colección. Esta corresponde a la matriz término-documento, *term-document matrix*, donde las filas representan los términos o características, las columnas los documentos y los valores de sus componentes el peso o indicador correspondiente a la importancia de cada una de las combinaciones. Si el punto de partida para este proceso son las listas obtenidas para cada documento a través de BOW o Bag-N-gram, el espacio de representación común correspondería a la unión de los conjuntos y, puesto que el orden de los elementos no es influyente en el mismo, la ordenación de las filas y columnas correspondientes a la matriz de asociación tampoco resultaría de relevancia, siendo, por tanto, intercambiables (Sec 3.3 [4]).

Para ilustrar con un ejemplo simple este procedimiento, he considerado un corpus formado por dos documentos y donde cada uno de los mismos está constituido únicamente por una oración:

Doc. 1: "*El TM es un ámbito de gran extensión.*"

Doc. 2: "*La representación BOW es una de las alternativas clásicas del TM y junto a la medida TF-IDF determinan la extensión usual hacia la representación VSM.*"

De esta forma, se obtiene la siguiente representación BOW para las frecuencias asociadas, Fig. 15.7 (por comodidad se simula la matriz documento-término, i. e., la matriz traspuesta de la correspondiente matriz término-documento):

ID	ámbito	de	el	es	extensión	gran	tm	un
doc1	1	1	1	1	1	1	1	1
doc2	0	1	0	1	1	0	1	0

ID	a	alternativas	bow	clásicas	del	determinan	hacia	idf	junto
doc1	0	0	0	0	0	0	0	0	0
doc2	1	1	1	1	1	1	1	1	1

ID	la	las	medida	representación	tf	una	usual	vsm	y
doc1	0	0	0	0	0	0	0	0	0
doc2	4	1	1	2	1	1	1	1	1

Figura 4.1: Matriz documento-término (BOW). Fuente: Elaboración Propia.

Análogamente, llevando a cabo la eliminación de palabras *stop-words* y realizando el cálculo de las medidas indicadas en la sección 3.2.2 se obtienen los resultados correspondientes para cada una de las asociaciones documento-término (Fig. 4.3).

Por último, en cuanto al análisis de los valores más relevantes (Fig. 4.2), se observa que la palabra común a ambos documentos, *tm*, al presentar generalidad para el corpus, efectivamente, se corresponde con una medida IDF nula y, por consiguiente, también obtiene el mismo valor para la asociada a TF-IDF. De igual forma, se tiene

que, a pesar de la superioridad en la frecuencia absoluta y la igualdad en los valores IDF, las distintas extensiones para los documentos, es decir, el número de palabras (sin tener en cuenta las repeticiones) no *stop-words* representantes de los mismos, 4 y 15, respectivamente, hacen que se obtengan significativas diferencias para los valores TF-IDF asociados a los términos seleccionados, i. e., *ámbito*, *representación* y *vsm*.

ID	WORD	TF	TF_r	IDF	TF_IDF
doc1	ámbito	1	0.2500000	0.6931472	0.1732868
doc2	representación	2	0.1333333	0.6931472	0.0924196
doc1	tm	1	0.2500000	0.0000000	0.0000000
doc2	tm	1	0.0666667	0.0000000	0.0000000
doc2	vsm	1	0.0666667	0.6931472	0.0462098

Figura 4.2: Medidas TF, IDF, TF-IDF para la selección. Fuente: Elaboración Propia.

ID	ámbito	extensión	gran	tm	alternativas	bow	clásicas	determinan
doc1	0.173	0	0.173	0	0.000	0.000	0.000	0.000
doc2	0.000	0	0.000	0	0.046	0.046	0.046	0.046

ID	hacia	idf	junto	medida	representación	tf	usual	vsm
doc1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
doc2	0.046	0.046	0.046	0.046	0.092	0.046	0.046	0.046

Figura 4.3: Matriz documento-término (TF-IDF). Fuente: Elaboración Propia.

No obstante, este razonamiento clásico asociado a la matriz término-documento puede generalizarse hacia el concepto de matriz término-contexto, *term-context matrix*, donde cada columna hace referencia a un contexto lingüístico, cada fila a un término, y la correspondiente componente a la fuerza de asociación presentada entre ambos a lo largo del corpus, siendo posible así capturar las propiedades distribucionales entre los mismos, términos y contextos. Formalmente, sea V_w el vocabulario considerado y V_c el conjunto de posibles contextos (documentos, oraciones, párrafos,...), ambos indexados; se tiene que la matriz término-contexto, $\mathbf{M}^f \in \mathbb{R}^{|V_w|} \times \mathbb{R}^{|V_c|}$, queda determinada

según la medida de asociación considerada, $\mathbf{M}^{f}_{[i,j]} = f(c_i, w_j)$ (Sec. 10.4.1, [39]). No obstante, cuando el contexto no corresponde al documento completo, también es usual recurrir a la matriz de co-ocurrencia, *co-ocurrence matrix*, o matriz término-término, *term-term matrix*, donde se presenta la medida de asociación para la co-ocurrencia de los términos correspondientes. Puesto que ciertas palabras, generalmente stop-words, tienden a aparecer con mayor frecuencia con otros términos, pero estas no tienen capacidad discriminativa, con el objetivo de identificar este comportamiento y proporcionar valores adecuados y útiles para las asociaciones, suele recurrirse a medidas, f, como la *Información Mutua Puntual (Pointwise Mutual Information, PMI)* [40] (Sec. 13.2, [4]).

Sin embargo, factores como el hecho de no considerar el orden de las características, ya sean palabras o expresiones, siempre conllevan cierta pérdida de información y ambigüedad, provocando que numerosas oraciones con significados totalmente diferentes sean tratadas de manera similar. Por ello, aspectos como la posición, localización o significado sí podrían ser en ocasiones de interés y, por tanto, considerables. Aunque anteriormente se propusieron las secuencias, N-gramas, como alternativa al estudio individual de las palabras, existen otras opciones para la representación vectorial, como las que recaen en extender y crear nuevas características que recojan información relativa al entorno de cada término. No obstante, estas podrían agravar las dificultades ya presentes en las matrices de estas representaciones, relativas a la dimensionalidad y dispersión (Sec. 13, [4]).

4.3 Word Embeddings

Pese a las alternativas presentadas para los enfoques previos, donde las características seleccionadas son representadas de forma independiente al resto a través de una única dimensión y estableciendo como dimensión final del espacio vectorial de representación el total de las mismas, este tipo de procedimientos destaca por presentar el gran inconveniente de no ser capaz de capturar de forma adecuada las relaciones, es decir, correspondencias semánticas, sintácticas o gramaticales, existentes entre términos o documentos.

Esta nueva vía (Sec. 13, [4] y Sec. 10, [39]) centra el objetivo en conseguir una representación vectorial para cada palabra o término dentro de un espacio vectorial común de menor dimensión a la correspondiente al enfoque anterior, i. e., de dimensión fija e inferior al número total de términos considerados. Puesto que se permite así obtener una representación vectorial *densa* para cada palabra, *embeddings*, se establece que conceptualmente estas son "embebidas" hacia el espacio vectorial prefijado. Este razo-

namiento se desarrolla bajo la *Hipótesis Distribucional* del lenguaje y significado de las palabras, relativa a la *semántica distribucional* y estableciendo que palabras asociadas a un mismo contexto tendrán significados similares, por lo que se espera que a través de cierta medida de distancia o de similaridad, sea posible esclarecer cualquier tipo de relación. A diferencia de los métodos basados en conteo, *count-based methods*, el objetivo de las *representaciones distribuidas* es que las diferentes dimensiones capturen el "significado" del término de interés y puedan verse como características contextuales asociadas al mismo. No obstante, de forma general, estas no corresponden necesariamente a conceptos específicos o propiamente interpretables.

Los resultados obtenidos pueden utilizarse como datos de entrada para los diferentes modelos según las aplicaciones, o para la directa comparación, puesto que la representación vectorial permite identificar el cálculo de distancias entre embedding con el correspondiente a los términos asociados. Una de las ventajas de este enfoque, respecto a los anteriores tradicionales, alude al cálculo de la regla de asociación básica, la similitud entre documentos o términos. Al centrarse en reconocer los vínculos existentes a través de su entorno, contexto, en lugar de analizar la exactitud de los mismos, se permite una mejor identificación dado que, a diferencia de las medidas clásicas que buscan la coincidencia, no fallan a la hora de analizar contextos de significado similar reflejados a través de sinónimos y expresiones semejantes. Un ejemplo de estas medidas corresponde a la introducida por Kusner et al., *Word Movers Distance (WMD)* [41], con la que cuantifican la disimilitud entre dos documentos como la mínima distancia necesaria para que los embeddings asociados a los términos de uno de los mismos alcancen a los correspondientes al otro documento.

En cuanto a los procedimientos para el cálculo de vectores, embeddings, generalmente se consideran dos posibles alternativas: métodos basados en la factorización de la matriz término-contexto, o los métodos centrados en los modelos neuronales para el lenguaje, *neural language models*. Los primeros recaen en técnicas, también analizadas desde el punto de vista de reducción de dimensionalidad, cuyo objetivo final es obtener una aproximación para la matriz original pero permitiendo a lo largo del proceso conseguir representaciones densas de menor dimensión para cada uno de los términos. De entre los más usuales destacan las técnicas asociadas a *LSA* (Sec. 6.1) y *LDA* (Sec. 6.3). Por su parte, los asociados al segundo enfoque recurren al aprendizaje a través de redes neuronales. Para estos destacan los algoritmos de *Word2Vec* (Sec. 10.2.1), *GloVe* (Sec. 10.2.2) o *fastText*, desarrollados bajo la idea de que el modelado estadístico del lenguaje, *language modeling*, puede tratarse desde una perspectiva no supervisada donde la predicción para un término puede determinarse a partir del contexto formado por

sus k predecesores.

Aunque es posible proceder según un enfoque supervisado, el hecho de que, en dicho caso, el entrenamiento para los embeddings vaya dirigido a capturar información relevante para dicha tarea específica, i. e., preestablecido su etiquetado correspondiente, hace que los procedimientos más usuales correspondan a técnicas no supervisadas, de forma que el único objetivo sea obtener las representaciones según las consideraciones originales:

(i) predecir un término dado su contexto, o bien,

(ii) decidir si un término podría asociarse a un determinado contexto.

Por ello, de entre los aspectos de mayor influencia a la hora de obtener los embeddings destacan la determinación y tratamiento del contexto de referencia para cada uno de los términos.

Por lo general, aunque este puede venir representado por documentos completos, a la hora de establecer el contexto es usual considerar un conjunto de m palabras situadas en torno al término en cuestión, i. e., a cada lado del mismo, determinando así la *ventana* o *window context* de tamaño $2m + 1$. Como consecuencia, la amplitud y procesamiento considerados tendrán un alto impacto en el aprendizaje pudiendo afectar a los resultados. En cuanto a la extensión de la misma, a través de un rango de 1-3 palabras se favorecería el descubrimiento de relaciones sintáticas y funcionales, mientras que con mayores longitudes, 4-10 palabras, se permitiría encontrar similaridades semánticas.

Por otra parte, a la hora de su implementación, también es posible plantear dos enfoques alternativos:

(i) global, a partir de técnicas como la *concatenación* o *Representación Continua del Listado de Palabras, (Continuous Bag-Of-Words, CBOW*), con el que predecir el término a través de todos los recogidos en el contexto, o bien,

(ii) modelo *Skip-gram*, distinguiendo $2m$ tareas distintas asociadas a los pares determinados según el término de interés y cada uno de los incluidos en el contexto.

No obstante, pese a los grandes avances, el hecho de representar cada palabra a través de un único vector presenta limitaciones, como las asociadas a la polisemia y homonimia, motivando así la consideración de representaciones múltiples, *multi-sense embeddings*, generalmente caracterizados por los enfoques no supervisados y basados en conocimiento. Un primer método corresponde al *Multi-Sense Skip-Gram (MSSG)* que,

basado en las técnicas de *Word2Vec* y *Skip-gram*, combina la capacidad discriminatoria y embedding simultáneamente pero prefijando el número de significados, vectores, asociados a cada término. En cambio, la variante no paramétrica, *Non-Parametric Multi-Sense Skip-Gram (NP-MSSG)*, permite variar dicho número dependiendo del término. Por último, la asociada a *Most Suitable Sense Annotation (MSSA)* permite, a través de la combinación de conocimiento previo asociado a bases de datos léxicas (por ejemplo, WordNet o ConceptNet), una primera desambiguación de términos y una posterior construcción de embeddings, permitiendo así mejorar los resultados[42] [43].

APLICACIONES

Como ya se mecionó en la introducción inicial, las técnicas relativas al ámbito de *Text Analytics* destacan tanto por su extensa variedad como por su gran aplicabilidad. Junto a las tareas aparentemente más tradicionales o intuitivas asociadas a la información textual, como puede ser la clasificación de documentos según distintas categorías, la continua evolución de recursos y campos como el *Machine Learning* han permitido no solo mejorar y progresar en las ya existentes, sino también, dar lugar a nuevos retos y desafíos lingüísticamente más complejos, siendo capaces de obtener resultados considerablemente coherentes y aceptables para los mismos.

Generalmente es habitual que para una aplicación u objetivo final sea necesario recurrir a distintas tareas, no solo de preparación y estructuración, que son imprescindibles, sino relativas a diferentes aspectos o naturalezas, sirviendo así de ayuda para el caso principal en cuestión. Como consecuencia, aunque para las más clásicas sí existe concordancia, al igual que ocurría con la estructuración propia de este ámbito, también existe parte de confusión a la hora de establecer cierto orden o disposición.

En cuanto a su complejidad, pese a que a priori dichas aplicaciones pueden resultar sencillas, evidentemente, la dificultad de las mismas recae en la cantidad y extensión de los textos y colecciones a los que se recurre para el análisis. De forma general estas conllevan cientos e incluso miles de características provocando que los procesos de preparación y ponderación de variables, *feature engineering*, sean tan o más importantes que la elección del propio procedimiento, algoritmo o método con el que resolver el problema en cuestión.

5.1 CLASIFICACIÓN

Tradicionalmente, para el ámbito del TM, la tarea de *clasificación* hace referencia a la clasificación de documentos y, por tanto, el objetivo se centra en asignar una o varias categorías, previamente fijadas y establecidas, a los documentos de interés. Para conseguirlo se recurre al estudio de las propiedades o atributos inherentes a dichos documentos, lo que, en definitiva, hace que sea posible identificar las clases que mejor se adaptan y caracterizan a los mismos. No obstante, pese a que las tareas usuales radican en amplias recopilaciones de texto, incluyendo varios párrafos y documentos,

también es posible enfocar esta aplicación hacia unidades más simples, como palabras (Sec. 15.2.2, [18]).

Desde su aparación, esta tarea ha demostrado resultar de gran utilidad en multitud de campos de aplicación (pg. 40, [44]), como la industria, finanzas, ciencia, o, particularmente, el ámbito médico. Además, no solo ha permitido establecer la cuestión, a priori, usual de clasificación según cierta temática, como la asignación de géneros para libros o noticias, sino que también ha permitido establecer novedosos propósitos a través de enfoques no tan evidentes, como son la detección de ironía en el texto, selección y recomendación en base a los intereses del usuario (pg. 55-57, [45]), captación de clientes, o identificación de la autoría del documento [46].

A la hora de determinar los distintos tipos de clasificación, existen numerosos criterios que permiten establecer una dicotomía, siendo posible distinguir entre las siguientes clasificaciones (Sec. 5, [18]):

Basada en contenido vs Basada en sugerencias *(Content-based vs Request-oriented)*: en un primer caso la asignación se determina a través de los pesos o ponderaciones establecidas para las categorías en función de su presencia y relevancia, i. e., según el contenido, mientras que otra posible opción sería recurrir a sugerencias de usuarios que determinasen cómo los documetos deben ser clasificados.

Binaria vs Múltiple *(Binary vs Multiple)*: en el caso binario se consideran únicamente dos posibles categorías, donde cada elemento podría pertenecer exclusivamente a una de ellas o de forma simultánea a ambas (clasificación solapada, *fuzzy binary classification*). Por su parte, en la clasificación múltiple se consideran 3 o más categorías, de forma que esta pueda verse como una sucesión de clasificaciones binarias entre M clases. Algunos ejemplos tradicionales corresponden, respectivamente, a la clasificación de correo spam o al etiquetado gramatical, POS tagging, para las palabras. En ciertas ocasiones, como caso particular, estableciendo una visión simplificada, también se compara la tarea de IR como una clasificación binaria, considerando que el objetivo se encuentra en decidir si la información o texto a recuperar es o no relevante para la consulta.

Simple vs Solapada $(Hard\ vs\ Soft)$: como se indicaba en el caso anterior, cada documento puede ser asignado a una única categoría, simple, o en caso contrario se permite una asignación simultánea a distintas clases, solapada, siendo este último el más frecuente en las aplicaciones reales.

Plana vs Jerárquica $(Flat\ vs\ Hierarchical)$: todas las categorías se presentan a un mis-

mo nivel o, en cambio, es posible que alguna de ellas pueda ir anidada en otra. En este segudo caso, a causa del anidamiento, puesto que las asignaciones de etapas previas irán determinando las categorías factibles, la clasificación final resultante dependerá del camino y decisiones tomadas con anterioridad. Sumando el hecho de que cada nivel de desglose debe tener asociado un clasificador específico y la necesidad de considerar situaciones complejas, como una clasificación adecuada a niveles superiores pero errónea a niveles inferiores, es evidente que el proceso de evaluación de resultados presenta mayor dificultad que los correspondientes a otro tipo de clasificación.

Visión simple vs Visión Múltiple (*Single vs Multiple Viewed*): podrían permitirse uno o varios enfoques. Al considerar un único enfoque existe un grupo de categorías y, por tanto, una única asignación para cada documento. En cambio, al considerar diferentes alternativas, cada una de estas llevará asociado un conjunto o grupo de categorías, de forma que cada documento recibe una clasificación para cada uno de los mismos. La diferencia entre la asignación solapada y el visionado múltiple se encuentra en que los conjuntos de clases no tienen por qué coincidir y, por tanto, diferentes visiones darían acceso a distintas categorías.

Independiente vs Dependiente (*Independent vs Dependedent*): dependiendo de si los resultados de la clasificación actual están o no influenciados por otros resultados previos relativos a alguna otra clasificación. En el enfoque independiente la decisión de clasificación no tiene influencia posterior mientras que en el caso dependiente esta conllevará uno u otro camino y, por tanto, ciertos clasificadores posteriores específicos. La clasificación plana y jerárquica correspondería a un caso particular de esta división, independiente y dependiente, respectivamente.

Pese a la multitud de casos prácticos asociados a cada una de las posibles combinaciones, usualmente, la mayoría de las aplicaciones hacen referencia a una clasificación múltiple, solapada, de visión única, basada en el contenido, plana y, por tanto, independiente. En definitiva, puede considerarse como la asignación de uno o varios temas, *topics*, a cada documento, lo que conlleva que esta tarea también sea conocida a través del término *topic spotting* [47]. Es importante destacar las diferencias entre este, generalmente, centrado en temas objetivos, y los casos particulares del *análisis de sentimientos* (Sec. 5.3), donde el texto es clasificado según la opinión e intencionalidad del autor, y la tarea de *topic routing*, que se corresponde con su proceso inverso, i. e., dados ciertos temas o conceptos, el propóstio consiste en recuperar documentos asociados a dichas categorías, siendo habitual su ubicación dentro de IR (Sec. 15.2.4, [18]).

En cuanto a las técnicas asociadas al ML para poder llevar a cabo una adecuada implementación, es importante señalar que tanto las correspondientes a un enfoque más clásico y tradicional, como las relativas a procesos más avanzados, por ejemplo, *Deep Learning*, han demostrado ser capaces de proporcionar resultados altamente exitosos. Tal y como se comentaba inicialmente, la base se encuentra en disponer de una muestra de datos etiquetada, donde cada documento, $D = \{d_1, \ldots, d_n\}$, esté asignado a su/s clase/s o categoría/s, $C = \{c_1, \ldots, c_n\}$, determinando así el conjunto de entrenamiento $\{(d_1, c_{11}), \ldots, (d_n, c_{nt_n})\}$, de forma que, tras la elección del algoritmo correspondiente, sea posible obtener un modelo de clasificación útil para el estudio y permitir, tras su aprendizaje y validación, una correcta predicción para futuros documentos.

De entre los posibles modelos asociados al aprendizaje supervisado para la clasificación de documentos pueden destacarse las siguientes técnicas:

- *(Multinomial) Naïve Bayes Classifier*
- *Support Vector Machine (SVM*, Sec. 9)
- *Random Forest, Gradient Boosting*
- *RNN* (Sec. 10.1.2) ,*CNN* (Sec. 10.1.3)

5.2 Clustering

Al igual que en el caso anterior, implícitamente se hace referencia a los documentos, aunque también podría corresponderse con otro tipo de entidades (oraciones, párrafos, etc.), y, como consecuencia, suele identificarse como propósito principal la agrupación de los mismos en determinados grupos o *clusters*. Puesto que el hecho de disponer de datos etiquetados para el estudio puede resultar de gran complejidad, esta técnica es considerada como una poderosa alternativa en caso de no conocer las categorías en las que poder llevar a cabo la clasificación.

El *clustering* de documentos, *document clustering*, es la aplicación más usual relativa al aprendizaje no supervisado y se centra en analizar los atributos y características presentes en los recursos textuales para poder así establecer de forma automática subconjuntos de dicha colección que sean distinguibles entre sí e internamente homogéneos (Sec. 12, [4]). A partir del conjunto de textos proporcionado y, tras la elección del procedimiento más adecuado (algoritmo), los prototipos de clustering (grupos) son inicializados, generalmente de forma aleatoria, y optimizados para obtener el resultado más adecuado.

Pese a que en la mayoría de las ocasiones se hace referencia a datos completamente no etiquetados, podría ocurrir que algunos elementos sí lo estuviesen, obteniendo así un conocimiento inicial parcial para las agrupaciones. En definitiva, se corresponde

a una extensión del clustering usual donde algunos ejemplos vienen etiquetados de forma adicional, permitiendo al algoritmo evitar la inicialización aleatoria y llevarla a cabo en base a los ejemplos disponibles. Este recibe el nombre de *clustering restringido*, *constraint clustering*, y no debe confundirse con el aprendizaje semi-supervidado, puesto que, a diferencia de este último, que engloba tanto técnicas supervisadas como no supervisadas, únicamente se centra en el aprendizaje no supervisado (pg. 189, [18]).

Otros posibles criterios para establecer diferentes alternativas para el clustering son (Sec. 9.3, [18]):

Léxico vs Semántico (*Lexical vs Semantic*): se distingue si los elementos son agrupados según su contenido, i. e., según la similitud en base a su estructura y sintaxis o, en cambio, la agrupación se enfoca según la similitud entre contenido, englobando la determinación de significados. La dependencia gramatical (POS tagging) o el reconocimiento de entidades (*Named Entity Recognition*, *NER*) son herramientas que suelen facilitar este último proceso.

Estático vs Dinámico (*Static vs Dynamic*): hace referencia a si los elementos a agrupar se encuentran fijos o, en su lugar, son continuamente actualizados. En este último caso, dicha transformación podría ser consecuencia de añadir nuevos elementos o eliminar algunos ya existentes. Estos cambios supondrían una reordenación continua y podría realizarse desde una perspectiva completa, *hard*, organizando todos los elementos, o a través de una visión parcial, $soft$, partiendo de los clusters ya construidos y procediendo a fusionarlos entre sí, incluir nuevos elementos, o dividirlos y, a partir de dicha separación, dar lugar a nuevas agrupaciones.

Simple vs Solapado (*Crip vs Fuzzy*): según si el documento pertenece a un único cluster o en cambio, podría asociarse a varios, siendo esta la visión más frecuente.

Plano vs Jerárquico (*Flat vs Hierarchical*): análogamente al caso de clasificación, hace referencia a si todos los clusters se encuentran al mismo nivel o, por el contrario, es posible establecer cierta jerarquía o anidamiento entre los mismos.

En el primer caso el resultado correspondería a una lista de agrupaciones y la asignación entre elemento y cluster sería directa. El objetivo recae en encontrar una partición, $C = \{C_1, \ldots, C_k\}$, de documentos, $D = \{d_1, \ldots, d_n\}$, por tanto, $k \leq n$, i. e., $\bigcup_{i=1}^{k} C_i = D$, donde $C_i \neq \emptyset$ para $i = 1 \ldots k$ y, según se considere una aplicación simple o solapada, $C_i \cap C_j = \emptyset$ o $C_i \cap C_j \neq \emptyset$, respectivamente (pg. 243, [4]).

Por su parte, el segundo se asemejaría a una estructura ramificada, de árbol, según los anidamientos, y se establecería una construcción arriba-abajo o divisiva,

top-down, fragmentando y separando los de niveles superiores hasta llegar a las unidades individuales, o bien, abajo-arriba o aglomerativa, *bottom-up*, fusionando y agrupando los de etapas inferiores hasta alcanzar el conjunto total. El objetivo se centra en contruir una jerarquía, $H = \{H_1, \ldots, H_h\}$, de subconjuntos de documentos, $H_i \subset D$, de forma que $H_i \cap H_j = \emptyset$ o bien, $H_i \subset H_j$ o $H_j \subset H_i$, para cualesquiera $i, j = 1 \ldots h$, $i \neq j$.

En ocasiones el clustering jerárquico también se identifica como basado en la conexión, *connectivity-based* (pg. 498, [3]), puesto que los grupos se determinan agrupando, i. e., conectando, los elementos según cierta medida de similitud o distancia, siendo, por tanto, necesaria la elección de dicha medida y el criterio de conexión.

Visión simple vs Visión Múltiple (*Single vs Multiple Viewed*): como se indicó anteriormente, hace referencia al número de enfoques establecidos.

Es claro que esta tarea depende considerablemente de la relación o conexión entre documentos y, en concreto, de la similitud entre los mismos, resultando, por tanto, de vital importancia su determinación en el estudio. En relación a dicha medida, el objetivo se encuentra en maximizar la similitud entre los elementos que conforman los clusters, *intra-cluster*, y simultáneamente minimizar la similitud entre los propios clusters, *inter-cluster*, permitiendo así evaluar la adecuación del procedimiento. No obstante, aunque lo usual es recurrir a su cálculo a través de documentos, también sería posible analizarla entre términos individuales, tokens, lo que permitiría obtener información inicial para el caso práctico. De entre las posibles opciones más usuales destacan las asociadas a las distancias de *Hamming*, *Manhattan*, *Euclídea*, *Levenshtein*, *Chebyshev* o *Minkoeski*, así como la similaridad *Coseno*.

En cuanto a las técnicas y algoritmos asociados al aprendizaje no supervisado para la construcción de los clusters, destacan (Sec. 7, [3] y Sec. 10, [18]):

- *Métodos Jerárquicos (Aglomerativos y Divisivos)*
- *K-medias, K-medias Esférico, K-medoides*
- *Kohonen Networks*
- *Affinity Propagation*

Puesto que usualmente se enfoca desde la perspectiva de ordenar o agrupar según cierta temática común, esta aplicación ha encontrado gran utilidad en el campo de la IR facilitando la comparación y búsqueda de información, por ejemplo a través de técnicas *centroid-based*, donde los clusters están representados por un vector de referencia, evitando que esta deba realizarse sobre todos los documentos.

Por otra parte, gracias a la combinación con técnicas de clasificación, también ha permitido resolver con éxito tareas relativas a la creación de ontologías, análisis de patrones, resúmenes, detección y rastreo de eventos a lo largo de diferentes documentos, así como generar distintas tareas derivadas. Entre estas es posible destacar (Sec. 9.4, [18]):

Nombramiento de Clusters (*Cluster Naming*): consiste en identificar cada agrupación en relación a sus contenidos relevantes otorgándole un valor clave y simbólico, verificando este ser único, breve y capaz de reflejar la esencia del mismo. Un sencillo enfoque, propuesto por Jo en 2006, corresponde a indexar los términos de cada cluster según cierta ponderación y seleccionar como candidatos los asociados a mayor peso. En el caso de que dos o más coincidiesen, se repitiría el proceso de selección según el orden decreciente establecido.
La tarea asociada a la generación de taxonomías, relativa a la contrucción de listas, árboles o redes según los temas, relaciones y asociaciones entre textos correspondientes a un corpus, puede verse como una aplicación combinada de clustering y posterior nombramiento, así como extenderse a la construcción de ontologías.

Base para Clasificación: en caso de no disponer de datos etiquetados, es posible también recurrir a los clusters resultantes, así como a los obtenidos tras la etapa posterior de nombramiento, con el objetivo de establecer un etiquetado inicial que sirva de base para un posterior proceso de clasificación. No obstante, este enfoque presenta la desventaja de poder arrastrar al proceso de clasificación los errores producidos en el paso previo de agrupación.

5.3 Análisis de sentimientos

Esta aplicación surge a causa de la propia naturaleza de los datos textuales, la cual permite distinguir entre *documentos objetivos*, centrados en describir hechos o situaciones específicas, y *documentos subjetivos*, basados en los sentimientos, opiniones y emociones expresados por el autor respecto a cierta cuestión. Gracias a estos últimos, apareció como idea y, posteriormente, como necesidad, el hecho de poder analizar las reacciones de la población respecto a cierta entidad, temática o factor, de forma que, en consecuencia, fuese posible la toma de útiles y adecuadas decisiones.

El *Análisis de sentimientos (Sentiment Analysis, SA)*, también conocido como análisis de opinión, *opinion analysis* u *opinion mining*, constituye una de las aplicaciones más extendidas asociadas a los ámbitos de NLP y TM.

De forma inicial, este recae en identificar la intención emocional del autor en el documento de estudio y, por tanto, se establece como objetivo principal el hecho de analizar ciertos documentos o información textual para ser capaz, a través de su contenido, de predecir la opinión de los mismos (Sec. 9, [3]). No obstante, además de las dificultades propias asociadas al carácter multidisciplinar que presenta, por ejemplo englobando aspectos relativos a la lingüística y psicología, los desafíos para esta tarea también recaen en las diferencias culturales o demográficas de los propios autores, así como en sus percepciones de los grados o estados emocionales.

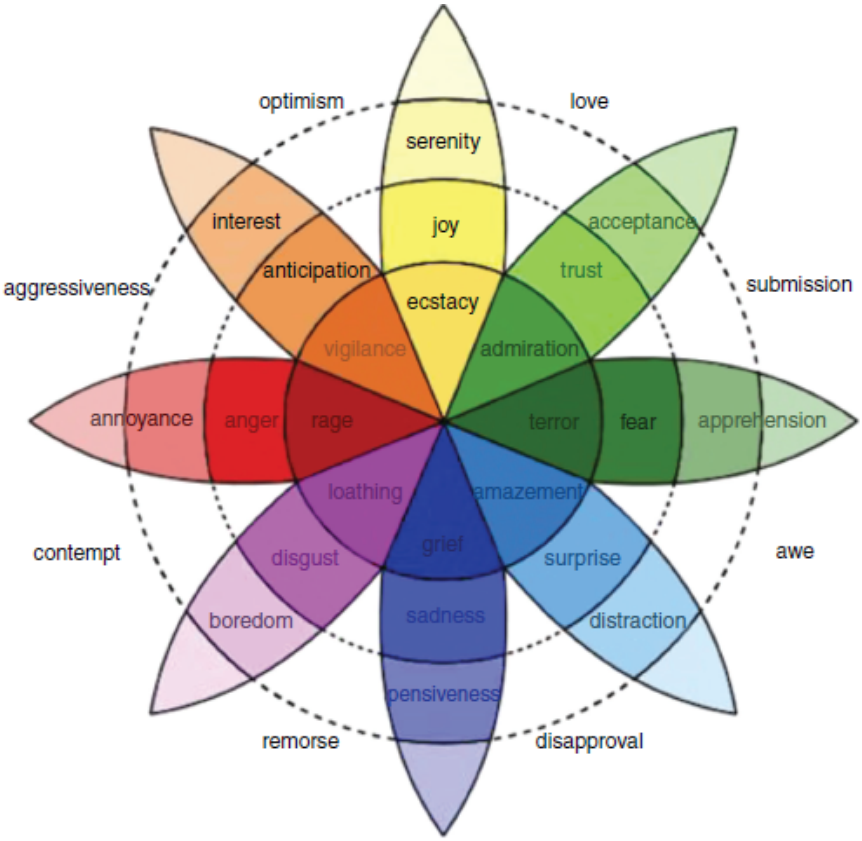


Figura 5.1: Estados emocionales. Relación y derivación. Fuente: pág. 87 [48].

A lo largo de su evolución se han desarrollado numerosas infraestructuras para poder implementearla y llevarla a cabo, como es el caso del sistema de clasificación propuesto por Robert Plutchik en 1980 (Sec. 4, [48]) donde distinguió 8 clases emocionales primarias o básicas: ira, miedo, tristeza, indignación, sorpresa, expectación, confianza y alegría, estableciendo que cualquier otra diferente estaría subordinada a las mismas (Fig. 5.1). Con este sencillo ejemplo se comprueba cómo es evidente que este proceso de identificación o distinción emocional está altamente condicionado por los grados o categorías establecidas y, por tanto, la subjetividad y la multitud de connotaciones asociadas a las mismas hacen que resulte de gran complejidad.

Por ello, con el objetivo de obtener resultados más acertados, usualmente la mayoría de los autores al hablar de SA hacen referencia a un enfoque más general, distinguiendo tres categorías globales asociadas al carácter de la opinión o emoción: negativa, neutral y positiva. Esto conlleva que, en definitiva, se proceda a un enfoque

más cercano al denominado *análisis de la polaridad* del documento, donde es frecuente establecer las clases negativa y positiva. Aunque es posible determinar el umbral específico para la clasificación en cada una de las clases según se requiera, generalmente se establecen las nociones de opinión positiva, neutral o negativa según si la polaridad obtenida es mayor, igual o menor que 0, respectivamente. De esta forma, el *SA* no solo permite obtener resultados cualitativos asociados al etiquetado, como es el grado de opinión global del documento, sino que también es posible el cálculo de medidas cuantitativas, como las asociadas al análisis de las proporciones de polaridad, objetividad o subjetividad del mismo.

Una de las alternativas no supervisadas tradicional para esta aplicación corresponde a las técnicas basadas en *lexicons* (pág. 572 [3]). Estas se centran en la construcción de modelos de manera que, a partir de diferentes fuentes de información externas, sea posible identificar la perspectiva global del documento. Estas fuentes hacen referencia a bases de datos, ontologías, diccionarios o vocabulario, donde se encuentren detallados aspectos relativos a la subjetividad, polaridad u objetividad de las palabras, quedando reflejadas las opiniones y emociones presentadas en los mismos. La clave de estos métodos está en recurrir a dichos conjuntos para el cálculo del sentimiento del documento según la correspondencia entre las palabras y puntuaciones disponibles, además de tener en cuenta otros factores, como la presencia de parámetros negativos en torno a la mismas. Entre los repertorios más populares y utilizados destacan: el creado por *Bing Liu* [49], el subjetivo *MPQA* [50] o *AFINN* [51].

Otra alternativa usual, que facilita la evaluación del modelo, corresponde a recurrir a las técnicas comentadas anteriormente (Sec. 5.1) asociadas a la clasificación, puesto que, en caso de disponer de datos etiquetados, se correspondería a una situación particular de dicho ámbito.

Finalmente, además de conocer la intención u opinión del autor, en ocasiones, también resulta de gran importancia identificar las razones o causas asociadas a dicho comportamiento. La idea es encontrar los factores clave que provocan el carácter de los sentimientos, positivos o negativos, siendo posible establecer varias alternativas. De forma inicial podría recurrirse a las probabilidades de pertenencia a cada una de las categorías para la predicción o resultado, con el objetivo de conocer el grado de adecuación a cada una de las mismas y obtener una orientación sobre dicha búsqueda. No obstante, en ciertos casos, esta visión resulta demasiado simple y, como consecuencia, se recurre a métodos más elaborados como son: (i) los *modelos de interpretación*, como la técnica de *LIME*, intentando y facilitando la comprensión del procesamiento interno

local asociado al modelo de clasificación y que permite llevar a cabo la toma de decisiones, o (ii) la extracción de los conceptos o temas presentes en el documento y que resultan de relevancia, *topic modeling* (Sec. 5.6).

5.4 Resumen (Summarization)

La excesiva sobrecarga de información junto a nuestras limitaciones humanas hacen que, hoy día, la capacidad de resumir automáticamente extensas cantidades de texto sea considerada como una de las aplicaciones, asociadas a este ámbito, de mayor importancia. Desde su inicio, ha permitido recoger con éxito numerosos casos prácticos, asociados, por ejemplo, a ámbitos generales como noticias, libros, emails o blogs, así como a campos más específicos, como los correspondientes a documentación biomédica, científica o legal. Es cierto que frente al *resumen manual*, donde el humano comprende el texto por completo y es capaz de reescribirlo a través de diferentes palabras y expresiones, el *resumen automático*, si se define como la selección literal de secciones, párrafos u oraciones esenciales, puede perder gran parte de la riqueza de producción y expresión (Sec. 13, [18]). No obstante, el desarrollo de diferentes técnicas, como las asociados al *deep learning*, han permitido mejorar y acercar esta tarea hacia dicho enfoque más "natural".

El propósito fundamental del resumen recae en descubrir los patrones y relaciones ocultas o intrínsecas en el texto desde un enfoque no solo estructural sino también semántico y lingüístico, de forma que, además de recoger los aspectos de mayor relevancia, se consiga un fragmento coherente y cohesionado capaz de reflejar las principales ideas del documento o colección. Como consecuencia, es usual relacionar esta tarea con el campo de IE donde parte del objetivo se centra en identificar y recoger las ideas y conceptos clave presentados en los documentos, para que así estos puedan ser comprendidos e interpretados. Esta relación hace referencia a técnicas que, en lugar de buscar una representación compacta y reducida de la amplia información inicial, se centran en el reconocimiento y extracción de unidades específicas o conceptos de importancia, como son las asociadas a *Key-Phrase Extraction* (Sec. 5.5) y *Topic Modeling* (Sec. 5.6).

Al igual que ocurría con las aplicaciones anteriores, existen numerosos criterios para distinguir el resumen a realizar [52]:

Entrada Individual vs Múltiple (*Single-document vs Multiple-document*): hace referencia al número de documentos utilizados para obtener el resumen. En el primer

caso, al disponer de un único documento, el objetivo principal suele recaer en la identificación de la información más importante, mientras que en el segundo, la clave se encuentra en evitar y eliminar una posible redundancia del contenido.

Genérico vs Específico (*Generic vs Query-based*): según se pretenda un resumen global o, en su lugar, dar respuesta a una determinada consulta u objetivo, reflejando así los contenidos de mayor relevancia para la misma.

Indicativo vs Informativo (*Indicative vs Informative*): depende de cúal sea el contenido deseado para el resumen. El primer enfoque se centra en la idea básica del documento, con el objetivo de determinar su propósito y ayudar así a futuros usuarios a decidir si este le resulta o no de interés. Por su parte, el resumen informativo se basa en recoger no sólo la temática o concepto principal, sino todos los contenidos relevantes del texto original.

Monolingüe vs Multilingüe (*Monolingual vs Multilingual*): según si los documentos y resúmenes se establecen para un único idioma o, por el contrario, para varios. El caso particular en el que la fuente de información (documento/s) y la salida (resumen) no coinciden en idioma recibe el nombre de lenguaje cruzado o *Cross-lingual*.

Global vs Dominio Específico (*General vs Domain-Specific*): dependiendo de si la técnica en cuestión puede aplicarse a cualquier tipo de documentos o, en su lugar, únicamente a los asociados a una clase o características determinadas. Dadas la estructura y composición propias para ciertos documentos, como los relativos a infomación médica o legal [53], pueden encontrarse procedimientos diseñados específicamente para textos de dicha naturaleza.

No obstante, además de las numerosas clasificaciones anteriores, uno de los criterios más extendidos hace referencia al enfoque con el que se genera el resumen para el documento. Desde una perspectiva más clásica destacan las técnicas basadas en la **extracción** de contenido, *Extraction-based*, con las que se pretende seleccionar los conceptos o unidades clave sin generar otros nuevos fragmentos. En cambio, desde una visión más compleja y sofisticada esta tarea puede plantearse según la *Generación Del Lenguaje Natural (Natural Language Generation, NLG)*, basándose en técnicas de **abstracción**, *abstraction-based*, donde, a través del aprendizaje, es posible crear un resumen propio, acercándolo así al razonamiento humano. De igual forma, con el objetivo de fusionar ambos planteamientos, algunos autores también señalan como posible alternativa intermedia una combinación *híbrida*.

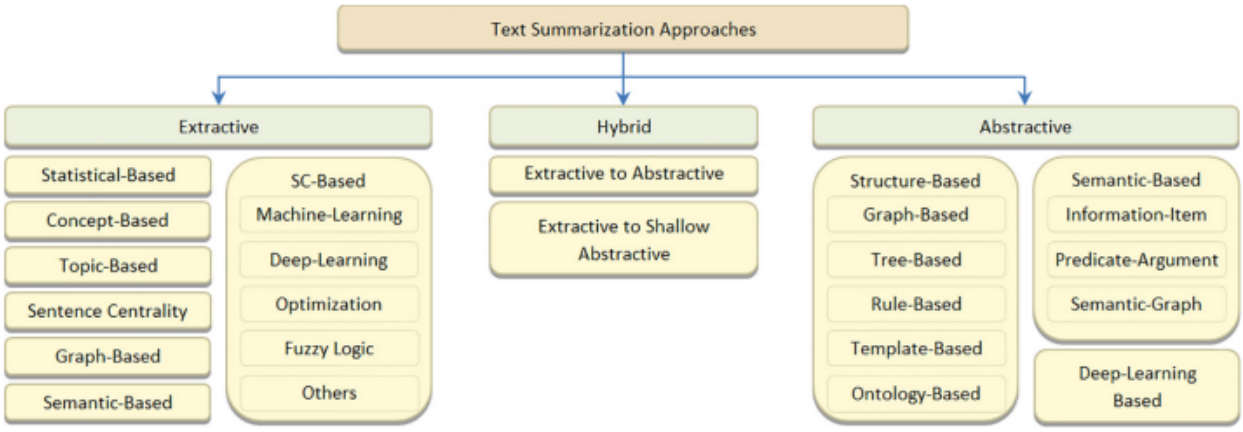


Figura 5.2: Enfoques y métodos para el resumen de documentos. Fuente: pág. 5 [52].

A pesar de que estos dos enfoques principales requieren tanto un preprocesamiento de la información inicial, como un postprocesamiento de los resultados, por ejemplo, reordenando o remplazando ciertas expresiones, la clave de los mismos se encuentra en sus diferencias, las tareas de procesamiento. De forma simplificada, el proceso de extracción se centra en: (i) crear una representación inicial para el texto, (ii) puntuar las unidades de selección, y (iii) extraer las correspondientes a una mayor relevancia. En cambio, el procesamiento asociado a la abstracción se caracteriza por: (i) construir una representación semántica interna y (ii) recurrir a las técnicas del NLG, para que, en base a aspectos como la comprensión, fusión y paráfrasis, sea posible generar un resumen lo más "humano" posible.

En cuanto a sus ventajas e inconvenientes, el enfoque clásico ha demostrado ser capaz de obtener resultados adecuados con notable precisión y rapidez. Sin embargo, este podría también presentar problemas relativos a la redundancia, pérdida de semántica y cohesión, o conflictividad entre expresiones procedentes de múltiples documentos. Por su parte, pese a que permiten resolver dichas dificultades, así como flexibilizar los resultados, las técnicas de abstracción se caracterizan por su gran complejidad, lo que tradicionalmente, hasta el desarrollo de herramientas y sistemas capaces de soportarlas, impidió su generalización y aplicación.

Tal y como se observa en el esquema (Fig. 5.2), también es posible considerar distintas alternativas a la hora de seleccionar las técnicas y algoritmos para llevar a cabo el resumen de documentos [52].

Desde una perspectiva de extracción destacan los métodos basados en:

Conceptos *(Concept-based)*: recurren a conceptos de bases de conocimiento externas,

como WordNet, y calculan la importancia de las unidades, fragmentos u oraciones, en base a los mismos. Se distinguen tres etapas: (i) selección de conceptos externos, (ii) construcción de las relaciones entre los conceptos anteriores y las unidades de interés, y (iii) puntuación y selección de las unidades.

Optimización *(Optimization-based)*: se basa en convertir el modelo de extracción, multi-document, en un problema de optimización multi-objetivo a través de distintos algoritmos, generalmente de tipo metaheurístico (por ejemplo, *Multi-Objective Artificial Bee Colony* [54]), estableciendo uno o varios criterios para los mismos, como podrían ser la cobertura del contenido, reducción de la redundancia, coherencia, relevancia, etc. Otras metaheurísticas utilizadas, también de carácter bio-inspirado, son los algoritmos genéticos, basados en la analogía con la evolución y genética de poblaciones, y la optimización por enjambre de partículas *(Particle Swarm Optimization)* [55].

Lógica Difusa *(Fuzzy Logic-based)*: intentando adaptarse a las distintas expresiones del documento se centra en: (i) seleccionar un conjunto de variables para cada unidad, como su longitud o ponderación, (ii) insertar las reglas para la base del sistema de conocimiento lógico, y (iii) puntuar según dicho sistema la importancia de cada unidad.

Estadística *(Statistical-based)*: centrados en el análisis estadístico de un conjunto de características, como frecuencia o posición, para determinar así los conceptos de mayor importancia y llevar a cabo la extracción.

Probabilidad *(Probabilistic-based)*: recaen en la búsqueda de relaciones y conceptos de relevancia a través de un estudio probabilístico. Algunas de las técnicas clásicas empleadas corresponden a los modelos Bayesianos o Modelos Ocultos de Markov (HMM) [56].

Grafos *(Graph-based)*: utilizan un grafo en el que los nodos representan las unidades a extraer y las aristas las relaciones entre las mismas. El objetivo radica en cuantificar su importancia analizando la influencia e interacción a lo largo del propio documento. Puesto que se centran en establecer las conexiones entre los distintos elementos, estos pueden verse a su vez como un tipo de representación para el contenido del texto. Además, según el carácter de las variables a tratar para su construcción sea léxico o semántico (como sinonimia, hiponimia o estructura sintáctica), suele distinguirse entre estas dos clases de grafos, *léxicos* y *semánticos*, respectivamente. No obstante, tradicionalmente, los asociados al primero de ellos, como los algoritmos de *LexRank* [57] y *TextRank* [58] (Sec. 7), han sido los

destacados.

Semántica *(Semantic-based)*: basados en un enfoque distribucional se han consolidado técnicas asociadas al *Análsis Semántico Latente* (LSA) así como sus variantes (Sec. 6).

Desde una visión más general, debe considerarse también el amplio conjunto de algoritmos asociados al *Machine Learning*, siendo así posible decidir entre un enfoque supervisado o no supervisado. Aunque habitualmente es más frecuente llevar a cabo esta aplicación desde una perspectiva no supervisada, también se proponen otras opciones (Sec. 13 de [18]) que consideran el resumen de documentos como una tarea de clasificación binaria, donde cada párrafo u oración debería asignarse de forma exclusiva a una de las clases correspondientes a *resumen* y *no-resumen*.

En cuanto a la alternativa de abstracción, pese a que se han desarrollado procedimientos asociados a algunas de las categorías anteriores, como la basada en grafos *Opiniosis* [59], o relativas a otros enfoques, como los correspondientes a un resultado jerárquico, *Hierarchical model*, y estructura arbórea, *Tree-based*, claramente esta se identifica con las técnicas correspondientes al ámbito del *Deep Learning* (*DL*, Sec. 10). Sin duda, el avance y mejora de dicho campo resultó esencial para la evolución de este tipo de aplicación, permitiendo que, hoy día, puedan establecerse numerosas posibilidades [60]. Una de las más destacadas corresponde al algoritmo de *Sequence-to-Sequence (Seq2Seq)*, asociado a un enfoque *Encoder-Decoder* y centrado en estructuras *RNN* (Sec. 10.1.2), desarrollado por Google originalmente como herramienta de traducción y rápidamente aplicado con éxito a tareas de reconocimiento de voz, sistemas de diálogo y resumen de documentos.

5.5 Key-Phrase Extraction

Se centra en extraer los términos clave, palabras o frases, *Key-word* o *Key-phrase extraction*, presentes en el cuerpo del documento, permitiendo, por tanto, que los temas de mayor relevancia queden identificados (Sec. 6, [3]). Como consecuencia, esta tarea es asimismo conocida con el nombre de *extracción de terminología* (*terminology extraction*), y es posible relacionarla a su vez con el ámbito del *Topic Modeling* (Sec. 5.6), en el caso de considerarse como una versión simplista del mismo. Además, de forma similar al caso anterior del resumen de documentos, en ocasiones, se señala como esta también podría identificarse como una tarea de clasificación binaria según la unidad correspondiente, a través de su importancia, se encuentre en la categorías *clave* y *no-clave*, respectivamente (pg. 321, [18]).

Esta tarea resulta de gran utilidad en áreas como la semática web, sistemas de recomendación y etiquetado, similitud entre documentos o traducción. Puesto que sus resultados podrían proporcionar información y conocimiento inicial, así como utilizarse para establecer las variables o características de futuras aplicaciones, es usual que se identifique como punto de partida para tareas de mayor complejidad. Por ejemplo, podría proporcionar las entidades clave para construir posteriormente el indexado de una colección de documentos o llevar a cabo una clasificación, así como las unidades de mayor relevancia para poder establecer un resumen breve y conciso de la información original (pg. 3, [58]).

En cuanto las técnicas para llevar a cabo el proceso, pueden distinguirse varias alternativas [61]:

Enfoque Estadístico: recae en el propio análisis de características estadísticas derivadas a partir de las unidades de interés, según su presencia en el documento. Se centran en las frecuencias de palabras, bien absolutas o a través de criterios como TF-IDF, y destaca el estudio de *colocaciones* (pg. 351, [3]). Haciendo referencia al término lingüístico, estas se corresponden a expresiones o secuencias que tienden a aparecer más frecuentemente a las correspondientes por aleatoriedad. Al tratarse de combinaciones de dos o más palabras, de forma similar a como se indicaba en capítulos previos, uno de los procedimientos que proporciona mejores resultados se basa en considerar las ocurrencias de *N-gramas*, y cuantificarlas a través de una determinada medida, como las comentadas anteriormente, TF-IDF, u otras alternativas, como PMI (Point-Mutual Information) [40].

Enfoque Lingüístico: se recurre a variables o conocimiento lingüístico para el estudio y extracción de expresiones. Generalmente requieren un amplio dominio del lenguaje o idioma asociado a la infomación, puesto que, entre otros, incluyen análisis léxico y sintáctico. Uno de los procedimientos más usuales corresponde a la extracción ponderada según el etiquetado gramatical (pg. 357, [3]), centrándose en las unidades catalogadas como expresiones o sintagmas nominales. Puesto que únicamente estos son los que resultan de interés, en lugar de llevar a cabo un etiquetado gramatical completo (POS tagging), sería suficiente realizar un análisis sintáctico superficial, *shallow parsing*, y, posteriormente, calcular las ponderaciones asociadas a cada uno de los sintagmas obtenidos, por ejemplo, a través del criterio TF-IDF, de manera que sea posible reconocer los de mayor relevancia.

Enfoque Supervisado: tras las técnicas supervisadas originales, como las asociadas a *Extractor*, basado en un algoritmo genético y *Keyphrase Extraction Algorithm*,

construido en base al teorema de Bayes, fueron desarrollándose sistemas de clasificación según las diferentes alternativas para este ámbito, como SVM (Sec. 9), Naïve Bayes, o Conditional Random Field.

Enfoque No Supervisado: análogamente al caso anterior, tras las técnicas iniciales, como las basadas en la *teoría de resonancia adaptativa* o *divergencia de Kullback-Leibler*, surgieron las más destacadas: *TextRank* [58] (Sec. A.5.3), desde un enfoque basado en grafos, y *Rapid Automatic Keyword Extraction*, basado en la observación de que las palabras clave, *keywords*, raramente presentan signos de puntuación o *stop-words*.

5.6 Topic Modeling

Es evidente que cuanta mayor variedad temática y lingüística se recoja en los documentos, mayor complejidad presentará la tarea de identificación. Como consecuencia, en ocasiones, la técnica anterior asociada a la extracción de oraciones o palabras clave resulta insuficiente, impidiendo así un correcto reconocimiento de los temas de relevancia.

Esta situación favoreció el desarrollo del *Topic Modeling*, diseñado específicamente con el propósito de extraer las peculiariedades, *topics*, que caracterizan al documento o corpus de forma múltiple y variada, permitiendo, por tanto, distinguir y diferenciar por completo la temática mostrada en los mismos. Puesto que estos factores a identificar podrían hacer referencia a cualquier aspecto, como pensamientos, opiniones, afirmaciones o hechos, para lograr reconocerlos es necesario recurrir a técnicas de modelado, matemático y estadístico, con las que sea posible esclarecer las estructuras semánticas ocultas y latentes en el corpus. Para ello, el objetivo en la construcción de las temáticas (más comúnmente, *topics*) recae en conseguir una combinación de los términos o unidades de estudio en distintos grupos o clusters, capaces de reflejar sus conexiones y, por consiguiente, proporcionar significado y sentido a las mismas.

Originalmente surgió como herramienta para el análisis de texto, TM, pero de igual forma, a lo largo de su evolución, este ha sido empleado con éxito en la detección de estructuras en campos como la bioinformática e información genética, visión artificial, o procesamiento de imágenes. Entre las técnicas asociadas a este ámbito más destacadas se encuentran el *Análisis Semántico Latente* (*LSA*), el *Análisis Probabilístico de la Semántica Latente* (*PLSA*), la *Asignación Latente de Dirichlet* (*LDA*) (Sec. 6), o la *Factorización No Negativa* (Non-Negative Factorization, *NNF*).

Parte II

Técnicas

Semantic Mapping

Tal y como se indicó en la sección inicial, el desarrollo de las técnicas asociadas al análisis y estudio semántico de la información textual resultó fundamental para el progreso y evolución del ámbito del TM. Estas fueron rápidamente aceptadas y ampliamente utilizadas, posibilitando así la mejora en un gran número de aplicaciones, como el resumen de documentos o la identificación y agrupación de temáticas, *topic modeling*.

Aunque en los últimos años se han ido obteniendo procedimientos más novedosos y complejos capaces de conseguir una mayor comprensión de las relaciones semánticas presentes entre los distintos documentos, como son los correspondientes a los modelos de *Pachinko Allocation Model*, *Correlated Topic Model* y *Dynamic Topic Model* para las tareas de *topic modeling* [62], claramente los originales son, aún, los más conocidos e implementados.

6.1 Latent Semantic Analysis

El *Análisis Semántico Latente (Latent Semantic Analysis, LSA)*, incialmente denominado *Indexado Semántico Latente (Latent Semantic Indexing, LSI)* en el campo de la IR [63], surge con la aspiración de poder solventar dificultades como los problemas derivados de la sinonimia o polisemia, manteniendo el objetivo de capturar la estructura oculta de los documentos, para representarlos, no a través de términos, como en los métodos convencionales, sino a partir de *conceptos subyascentes* referidos a los mismos.

Para conseguirlo se centra en técnicas asociadas al ámbito del álgebra lineal, concretamente basándose en el análisis espectral de la matriz término-documento a través de su *Descomposición en Valores Singulares (Singular Value Decomposition, SVD)*. El propósito fundamental se encuentra en modelar dichos conceptos, *topics*, como distribuciones probabilísticas sobre los términos presentes en los documentos, permitiendo así que estos puedan ser interpertados a través de la distribución resultante de la combinación convexa de un número reducido de topics.

Sea $A \in \mathcal{M}(n \times m, \mathbb{R})$ de rango r la matriz de representación término-documento para el corpus, con un total de n términos y m documentos, y sean $\lambda_1 \geq \cdots \geq \lambda_r$ los valores singulares asociados, la descomposición en valores singulares correspondientes

viene dada por:

$$A = UDV^t$$

donde $D = diag(\lambda_1,\ldots,\lambda_r) \in \mathcal{M}(r \times r,\mathbb{R})$, $U = (u_1,\ldots,u_r) \in \mathcal{M}(n \times r,\mathbb{R})$ y $V = (v_1,\ldots,v_r) \in \mathcal{M}(m \times r,\mathbb{R})$ matrices ortonormales, i. e., $U^tU = V^tV = I$. Fijada la dimensionalidad final, k, el LSA (respectivamente LSI) busca la representación de los documentos a través de un espacio vectorial de dimensión reducida, *latent semantic space*, a partir de los mayores valores singulares sobre dicha descomposición, permitiendo así establecer las aproximaciones: $D_k = diag(\lambda_1,\ldots,\lambda_k) \in \mathcal{M}(k \times k,\mathbb{R})$, $U_k = (u_1,\ldots,u_k) \in \mathcal{M}(n \times k,\mathbb{R})$ y $V_k = (v_1,\ldots,v_k) \in \mathcal{M}(m \times k,\mathbb{R})$, y, por consiguiente, dando lugar a la correspondiente para la matriz original:

$$A_k = U_k D_k V_k^t$$

Además, gracias al teorema de *Eckart-Youngz* [64], se garantiza que esta última, A_k, minimiza el error bajo la noma L_2 para las matrices de rango k, tratándose, por tanto, de la mejor aproximación posible.

Finalmente, estas aproximaciones permiten llegar a la clave del razonamiento, siendo posible identificar los documentos y términos a través de su proyección sobre el espacio k-dimensional según las filas de V_kD_k y U_kD_k, respectivamente.

En cuanto al resultado para cada topic, la combinación y ponderación de términos podría venir caracterizada tanto por pesos positivos como negativos. Puesto que dicho signo hace referencia al sentido de la dirección u orientación para el vector asociado al topic, cabe esperar que términos relacionados tengan signo o dirección similar, concluyendo además que, cuanto mayor valor en módulo les corresponda, mayor será la contribución asociada.

6.1.1 LSA para resumen de documentos

En concreto, dado un documento, para poder realizar su resumen desde el punto de vista de extracción, se establece la división del mismo según sus oraciones y se obtiene la descomposición SVD sobre la matriz término-oración, i. e., donde las filas hacen referencia a los términos base (palabras) y las columnas a las oraciones. Finalmente, para la elección de las de mayor relevancia, uno de los procedimientos usuales se corresponde con el método presentado por Gong y Liu [65] basado en la matriz de aproximación V_k. Puesto que esta recoge las relaciones entre las oraciones y los k topics generados, el objetivo radica en identificar, a través de sus valores, qué oración es más representativa o adecuada para cada uno de los conceptos subyascentes obtenidos, de

manera que estas conformarían las k oraciones de mayor relevancia y, por tanto, el resumen deseado.

6.2 Probabilistic Latent Semantic Analysis

Por su parte, el *Indexado Probabilístico de la Semántica Latente* [12], posteriormente consolidado como *Análisis Probabilístico de la Semántica Latente (Probabilistic Latent Semantic Analysis, PLSA)* tiene como objetivo establecer un modelo estadístico, también denominado *aspect model*, de variable latente centrado en la co-ocurrencia y asignación entre cierta categoría no observable, $z \in Z = \{z_1,\ldots,z_k\}$, y cada observación, i. e., con cada ocurrencia de una palabra, $w \in W = \{w_1,\ldots,w_n\}$, en un documento, $d \in D = \{d_1,\ldots,d_m\}$.

Desde el punto de vista de un modelo generativo, considerando: (i) $P(d)$ como la probabilidad de seleccionar el documento d, (ii) $P(z|d)$ como la probabilidad de escoger la clase z, y (iii) $P(w|z)$ como la probabilidad de generar la palabra w, se obtiene como resultado observable del modelo el par (d,w), mientras que la clase z es descartada. Considerando la suma sobre todos los posibles valores de z, a través de la probabilidad conjunta, y bajo las hipótesis esenciales de independencia, se obtiene el modelo estadístico de mixtura, *mixture model*, asociado:

$$P(d,w) \;=\; P(d)P(w|d) \;=^{1}\; \sum_{z\in Z} P(z)P(w|z)P(d|z)$$
$$\text{donde } P(w|d) = \sum_{z\in Z} P(w|z)P(z|d) \quad (^1 \text{ regla de Bayes})$$

modelando cada palabra de un documento como una muestra para el mismo y donde las componentes de mixtura se corresponden con variables aleatorias multinomiales. De esta forma, cada documento es representado como mezcla de dichas proporciones y, por tanto, se reduce a una distribución de probabilidad en base al conjunto fijado de topics, denominada descripción reducida (*reduced description*) [13]. Como consecuencia, Hofmann [12] detalló que, a diferencia del procedimiento de *clustering*, donde el objetivo recae en asignar cada documento a uno de los distintos grupos o clusters, bajo este razonamiento, las distribuciones asociadas a las palabras, $P(w|d)$, se obtienen como una combinación convexa de los factores $P(w|z)$, de forma que los documentos, en lugar de ser asignados, son caracterizados según una mixtura específica de factores con pesos $P(z|d)$.

En cuanto a los supuestos necesarios indicados previamente, estos asumen: (i) la independencia de los pares (d,w) al ser generados, consecuencia del enfoque genérico

basado en BOW en el que no se tiene en cuenta el orden de las palabras, y (ii) la independencia condicional, i. e., condicionando sobre la clase z, la palabra w es generada independientemente a la identidad del documento d.

Analizando esta última, debe destacarse cómo hace referencia a la hipótesis de intercambiabilidad entre palabras en un documento. Se dice que un conjunto finito de variables aleatorias, $\{z_1,\ldots,z_k\}$, es *intercambiable* si la distribución conjunta es invariante a cualquier permutación, i. e., $p(z_1,\ldots,z_n) = p(z_{\pi(1)},\ldots,z_{\pi(k)})$. Es importante señalar que esta no es equivalente al supuesto de independencia e idéntica distribución de variables aleatorias, sino que, en caso de relacionarse, debe interpretarse como idéntica distribución pero, tal y como se mostraba anteriormente, independencia condicional relativa al parámetro oculto subyascente, z. Por ello, siguiendo el teorema de Finetti que asegura la posibilidad de establecer una mixtura de distribuciones, en general infinita, para cualquier colección de variables aleatorias intercambiables, es necesario considerar modelos de mixtura capaces de capturar la intercambiabilidad asociada a ambas unidades, palabras y documentos.

Sin embargo, a pesar de que al considerar un número de estados inferior al total de documentos, $k \ll m$, la variable latente, z, actúa como un "cuello de botella" en la predicción de w condicionada a d, este razonamiento plantea dos serias dificultades: (i) no establece una forma natural de asignar probabilidad a un nuevo documento, y (ii) la búsqueda de k distribuciones multinomiales de tamaño n (número de palabras, i. e., tamaño del vocabulario), así como m mixturas sobre los k topics ocultos, provocan un crecimiento lineal del número de parámatros, $kn + km$, respecto al tamaño del corpus, m.

6.3 Latent Dirichlet Allocation

Con el objetivo de poder solventar dichos inconvenientes y profundizando en esta línea de pensamiento, se desarrolla la técnica asociada a la *Asignación Latente de Dirichlet (Latent Dirichlet Allocation, LDA)* [13]. Esta, en lugar de tratar los pesos asociados a la mixtura para los topics de forma individual como un gran conjunto, centra su objetivo en establecer un parámetro principal, k-dimensional y asociado a una variable aleatoria oculta (*latente*). Este plantemiento resuelve las dificultades correspondientes al enfoque anterior permitiendo generalizar adecuadamente a documentos externos, así como evitando la dependencia del número de parámetros, $kn + k$, al tamaño del corpus, m.

En lo que respecta a su razonamiento, el LDA busca la representación de documen-

tos a través de mixturas aleatorias sobre topics latentes donde, a su vez, cada uno de los mismos se caracteriza a través de una distribución sobre palabras. Por ello, esta técnica plantea dos objetivos simultáneos:

(i) Obtener para cada documento una mixtura sobre los topics ocultos (*per-document topic distribution*), de manera que posteriormente sea posible determinar la adecuación del documento a cada uno de los mismos.

(ii) Establecer cada topic como mixtura de palabras (*per-topic word distribution*), es decir, determinar las palabras que conforman o representan a los topics.

En cuanto al proceso generativo para cada documento del corpus, $d \in D$, en definitiva, este asume las siguientes etapas:

I. Elegir $n_d \sim Poisson(\xi)$ como el número de palabras por las que se representará al documento durante el proceso, $\mathbf{d} = (w_1, \ldots, w_{n_d})$.

II. Elegir $\theta_d \sim Dir(\alpha)$, según una distribución de *Dirichlet* k-dimensional, como la mixtura de topics asociada al documento.

Por tanto, se tiene que θ toma valores en el $(k\text{-}1)$-$simplex$, $\theta : \theta_i \geq 0$ y $\sum_{i=1}^{k} \theta_i = 1$, según la densidad de probabilidad:

$$P(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \ldots \theta_k^{\alpha_k - 1}$$

Esta puede verse como la generalización multivariante de la distribución Beta y se corresponde con una de las distribuciones usuales para este espacio (simplex). De igual forma, pertenece a la familia exponencial y de entre sus características destacan que es posible obtener estadísticos suficientes de dimensión finita, así como que resulta ser distribución a priori conjugada para las distribuciones multinomial y categórica [66], lo que facilita la inferencia y estimación de parámetros para el algoritmo.

III. Para cada palabra en d, es decir, $i = 1, \ldots, n_d$:

a) Elegir un topic $z_{di} \sim Multinomial(\theta_d)$, i. e., según una distribución multinomial acorde a la mixtura de probabilidades obtenida para d. De esta forma se establece una asignación inicial de un topic representativa (probabilística) a lo "observado" o "conocido" para el documento, es decir, θ_d.

b) Elegir una palabra w_{di} a partir de $p(w_{di}|z_{di}, \beta)$, es decir, la probabilidad multinomial condicionada al topic elegido en el apartado anterior, así como al

parámetro de adecuación de las propias palabras a cada uno de dichos topics. De esta forma se selecciona una palabra para "representar" al topic en cuestión, de manera que, posteriormente, pueda determinarse la mixtura asociada a z_{di}.

En cuanto al parámetro $\beta \in \mathcal{M}(k \times n, [0,1])$, se trata de una matriz donde cada fila representa un topic, cada columna una palabra del vocabulario y cuyas componentes hacen referencia a las probabilidades entre palabras y topics, i. e., $\beta_{ij} = p(w_j = 1 | z_i = 1)$, a priori, consideradas fijas y a ser estimadas.

Por tanto, dicho procedimiento se corresponde con un modelo Bayesiano que puede estructurarse en 3 niveles (Fig. 6.1, izquierda):

(i) Parámetros iniciales para las distribuciones, α y β.

(ii) Repeticiones para cada documento, $d \in D = \{d_1, \ldots, d_m\}$, determinándose para cada uno de los mismos la mixtura respecto a los topics, θ_d.

(iii) Iteraciones para las palabras de cada documento, determinándose el topic respectivo, z_{di}, y la palabra resultante en base al mismo, w_{di}.

En cuanto a la formalización matemática del proceso, dados los parámetros α y β, la distribución conjunta corresponde a:

$$P(\theta, \mathbf{z}, \mathbf{d} | \alpha, \beta) = P(\theta_d | \alpha) \prod_{i=1}^{n_d} P(z_{di} | \theta_d) P(w_{di} | z_{di}, \beta) \quad (\text{donde } P(z_{di} | \theta_d) = \theta_{dj} \; t.q. \; z_{dij} = 1)$$

Integrando sobre θ_d y sumando sobre $\mathbf{z}$ se obtiene la distribución marginal de un documento:

$$P(\mathbf{d} | \alpha, \beta) = \int P(\theta_d | \alpha) \left(\prod_{i=1}^{n_d} \sum_{z_{di}} P(z_{di} | \theta_d) P(w_{di} | z_{di}, \beta) \right) d\theta_d \tag{6.1}$$

de forma que tomando el producto de las probabilidades marginales asociadas a los documentos individualmente, $\mathbf{d}$, es posible obtener la probabilidad del corpus:

$$P(D | \alpha, \beta) = \prod_{d=1}^{m} P(\mathbf{d} | \alpha, \beta) = \prod_{d=1}^{m} \int P(\theta_d | \alpha) \left(\prod_{i=1}^{n_d} \sum_{z_{di}} P(z_{di} | \theta_d) P(w_{di} | z_{di}, \beta) \right) d\theta_d$$

En relación a dichas igualdades, destacar que este razonamiento extiende la hipótesis de independencia condicional asumiendo que dichos topics son *infinitamente intercambiables* dentro de un documento, i. e., toda subsecuencia finita es intercambiable, lo

que permite a través del teorema de Finetti y la marginalización de las variables hacia distribuciones Dirichlet, obtener Eq. 6.1.

Finalmente, respecto al cálculo de las distribuciones a posteriori asociadas a las variables ocultas dado un documento, $P(\theta, \mathbf{z}|\mathbf{d}, \alpha, \beta)$, y, por consiguiente, la correspondiente a $P(\mathbf{d}|\alpha, \beta)$, puesto que su obtención exacta es generalmente intratable, suele recurrirse a diferentes algoritmos de aproximación inferencial, como son el método de aproximación de Laplace, método de Monte Carlo basado en cadenas de Markov (usualmente utilizada la técnica de *Gibbs Sampling* [67]), o *variational inference*. En particular, esta última [13] consigue proporcionar una cota inferior para la función de log-verosimilitud asociada a la estimación de los parámetros α y β:

$$\ell(\alpha, \beta) = \sum_{d=1}^{m} log\ P(\mathbf{d}|\alpha, \beta)$$

lo que, junto al procedimiento de *EM (Expectation-Maximization)*, permite optimizarla, en primer lugar respecto los parámetros asociados al enfoque variacional y, posteriormente, respecto a los asociados al modelo.

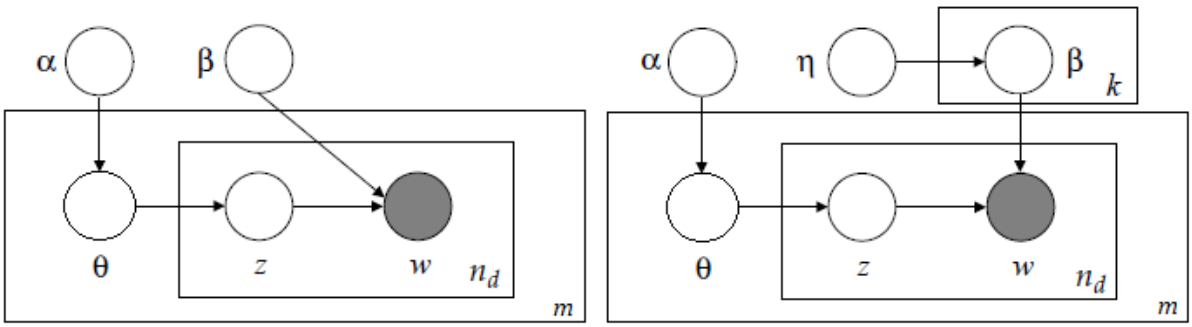


Figura 6.1: Modelos LDA: inicial y modificado. Fuente: pág. 997 y 1003 [13].

No obstante, el problema de la dispersión en los documentos también plantea serias dificultades para este razonamiento dado que las amplias dimensiones del vocabulario global provocan que sea altamente probable que un nuevo documento contenga palabras que no aparecen en algunos de los originales. Como consecuencia, esta situación conllevaría que, durante el proceso de estimación, se asignasen probabilidades nulas a dichas palabras y, por consiguiente, también a los documentos correspondientes. Por ello, con el objetivo de poder solventar esta restrictiva eliminación de contribuciones, los autores plantearon el ajuste de los parámetros multinomiales incluyendo una nueva distribución de *Dirichlet* n-dimensional (tamaño del vocabulario), de manera que, gracias a que esta verifica ser distribución a priori conjugada de la distribución multinomial, se añaden cantidades estrictamente positivas a las mismas (evitándose así la

nulidad) y β pasa a ser considerada como una matriz aleatoria de filas independientes, es decir, $\beta_{i.} \sim Dir(\eta)$ para $1 \leq i \leq k$ (Fig. 6.1, derecha).

Bajo esta última consideración, como consecuencia de la independencia, se tiene como formulación usual:

$$P(D|\alpha,\eta) = \prod_{d=1}^{m} P(\theta_d|\alpha) \prod_{j=1}^{k} P(\beta_j|\eta) \prod_{i=1}^{n_d} P(z_{di}|\theta_d) P(w_{di}|z_{di},\beta)$$

de manera que los objetivos inicialmente planteados son claramente identificables:

(i) $P(\theta_d|\alpha)$ y $P(\beta_j|\eta)$ hacen referencia a las distribuciones de *Dirichlet* asociadas a los topics y palabras, respectivamente.

(ii) $P(z_{di}|\theta_d)$ y $P(w_{di}|z_{di},\beta)$ se corresponden con las distribuciones multinomiales para determinar el topic asociado al documento y la palabra representativa del topic elegido según las probabilidades respectivas.

Con el objetivo de proporcionar una idea intuitiva sobre dicho razonamiento, en la siguiente imagen (Fig. 6.2) puede observarse una versión muy simplificada del mismo. En esta se ha representado el proceso correspondiente para un único documento suponiendo que tan solo se consideran 3 topics y 4 palabras para el vocabulario:

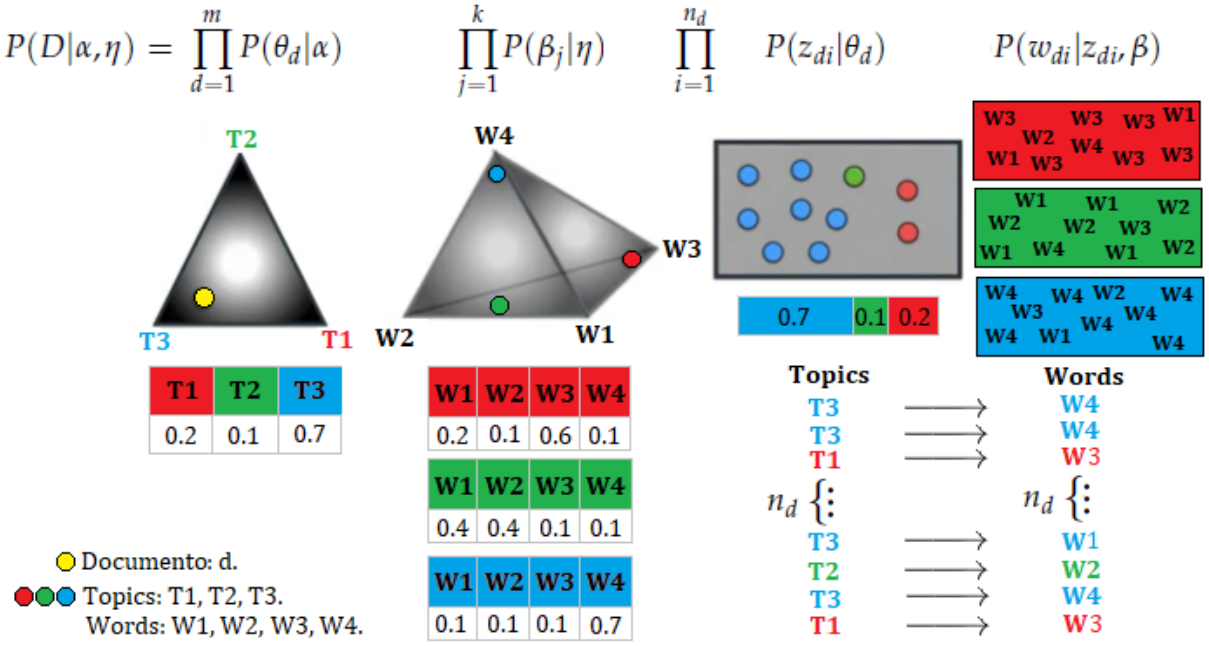


Figura 6.2: Idea intuitiva del modelo LDA. Fuente: Elaboración Propia.

Algoritmos Basados en Grafos

Tal y como se indicó en las secciones correspondientes a las aplicaciones del TM (Sec. 5.4 y Sec. 5.5), es posible, para las mismas, recurrir a técnicas basadas en grafos. Concretamente, los algoritmos de *TextRank* [58] y *LexRank* [57], presentados en 2004, han sido extensamente utilizados, para tareas relativas a la extracción de unidades clave (palabras, expresiones u oraciones) del documento original.

El objetivo principal de estos dos algoritmos de ordenación, *TextRank* y *LexRank*, recae en identificar, partiendo de una estructura de conexión entre términos, las unidades de mayor relevancia para la información textual disponible, permitiendo así ampliar el conocimiento sobre la misma y, por consiguiente, mejorar la posterior toma de decisiones. No obstante, al igual que se diferencia entre la extensión de las unidades, siendo posible considerar palabras, expresiones u oraciones, la finalidad de la aplicación en cuestión también permite distinguir el carácter de las conexiones a representar, como podrían ser relaciones léxicas, semánticas, contextuales o de coincidencia.

Por ello, la primera etapa del procedimiento se corresponde con la necesidad de establecer una adecuada representación capaz de interconectar las entidades (futuros vértices) a través de relaciones (futuras aristas) que reflejen su coherencia y significado. La clave de sus razonamientos se encuentra en la determinación de la importancia para las unidades representadas: en lugar de centrarse en el contexto local de la unidad textual, tienen en cuenta también, recursivamente, la relevancia global del documento/s, es decir, la obtenida para el grafo completo. Para conseguirlo, se basan en la puntuación de las unidades según la conectividad del grafo y el concepto derivado de *recomendación*. De esta forma, las unidades altamente recomendadas por el resto se corresponderán con las más informativas y relevantes, de manera que según el número de votos y la puntuación de los vértices que contribuyen a los mismos, i. e., emisores, se permite identificar las asociadas a una mayor influencia.

Tradicionalmente, las técnicas desarrolladas para la extracción de unidades se basaban en características simples, como su posición o frecuencia en el texto, recurriendo generalmente a la medida IDF para la determinación de la importancia de cierta palabra en la oración. Con el objetivo de mejorar dichas aproximaciones para definir la similitud entre unidades, los autores de *TextRank* proponen un primer enfoque para trabajar sobre un único documento, mientras que los correspondientes a *LexRank* am-

plían dicha visión hacia posibles técnicas y medidas a utilizar para un conjunto de múltiples documentos.

En general, para la representación del conjunto de documentos u oraciones, ambos métodos se basan en dos alternativas comunes: (i) establecer una matriz de similitud, según la medida establecida, en la que cada entrada se corresponda con el criterio o valor obtenido para cada par, y (ii) un grafo, $G = (V, E \subset V \times V)$, según la matriz de adyacencia correspondiente. Tal y como se indicó anteriormente, para este último, i. e., el grafo de representación, los nodos se corresponderían con las unidades de análisis y ordenación, de manera que las relaciones lingüísticas (cohesión, semántica, etc.) a capturar entre los mismos establecerían las aristas.

En cuanto a los métodos propuestos para el ranking de unidades, puesto que el propósito fundamental radica en identificar aquellas relaciones realmente significativas, se propone eliminar las correspondientes a valores de similitud insuficientes, es decir, aquellas aristas que no superen cierto umbral preestablecido. De esta forma, una manera inicial y simple de establecer la centralidad (relevancia) para las unidades es la denominada *centralidad de grado* (*degree centrality*), que consiste en el recuento de unidades adyacentes resultantes tras la eliminación, i. e., número de aristas salientes de cada nodo. Evidentemente, la elección del umbral es determinante para la interpretación de la centralidad, de manera que valores pequeños podrían considerar relaciones demasiado débiles e insignificantes, mientras que valores elevados podrían excluir relaciones relevantes.

Claramente, el cálculo bajo el enfoque anterior (*degree centrality*) implica que todas las relaciones que superan el umbral reciben la misma importancia, puesto que únicamente se tiene en cuenta la existencia de las mismas. En ocasiones, esta situación podría conllevar efectos negativos en la calidad del proceso de extracción u ordenación, ya que unidades no deseadas o menos relevantes podrían votarse entre sí e incrementar su centralidad. Por tanto, este resulta insuficiente y conlleva que sea necesario considerar, tal y como se comentó anteriormente, además de la existencia, la procedencia de dicho voto bajo la centralidad del emisor, i. e.:

$$S(v_i) = \sum_{v_j \in In(v_i)} \frac{1}{|Out(v_i)|} S(v_j)$$

donde $S(v_i)$ hace referencia a la puntuación o relevancia asociada al vértice $v_i \in V$, $In(v_i) = \{v_j \in V | e_{ji} \in E\}$ y $Out(v_i) = \{v_j \in V | e_{ij} \in E\}$. Equivalentemente, la formula-

ción matricial correspondería a:

$$\mathbf{S} = \mathbf{B}^t\mathbf{S} \iff \mathbf{S}^t\mathbf{B} = \mathbf{S}^t, \text{ con } \mathbf{B} = \frac{\mathbf{A}_{ij}}{\sum_k \mathbf{A}_{ik}}$$

donde $\mathbf{A}$ hace referencia a la matriz de adyacencia obtenida para la similaridad del grafo (cada unidad es al menos similar a sí misma, por lo que se garantiza que la suma de cada fila sea no nula).

De esta forma, puesto que $\mathbf{B}$ se corresponde con una matriz estocástica, esta puede tratarse como matriz de transición asociada a una *Cadena de Markov* y, por tanto, gracias al teorema de *Perron-Frobenius*, bastaría verificar su *irreducibilidad* (todo estado es alcanzable desde otro culaquiera) y *aperiodicidad* ($\forall i \ \text{mcd}\{n|\mathbf{B}_{ii}^n > 0\} = 1$) para así garantizar la unicidad de convergencia hacia una distribución estacionaria, $\mathbf{S}$.

Este razonamiento fue originalmente propuesto por Page et al. en 1998 [68], a través del algoritmo de *PageRank* (enfocado a tareas relativas a la navegación web y elemento básico del buscador Google), incorporando el coeficiente de amortiguación, $d \in [0,1]$ (*damping o jumping factor*), centrado en integrar en el modelo la probabilidad de transición ("saltar") de un vértice a otro de forma aleatoria. Formalmente, establecieron la puntuación asociada al vértice $v_i \in V$ de un grafo *dirigido* como:

$$S(v_i) = (1-d) + d \sum_{v_j \in In(v_i)} \frac{1}{|Out(v_j)|} S(v_j)$$

donde la representación matricial se corresponde con:

$$\mathbf{S} = [(1-d)\mathbf{U} + d\mathbf{B}]^t\mathbf{S}$$

siendo $\mathbf{U}$ una matriz cuadrada con todos sus elementos iguales a 1.

No obstante, estas definiciones iniciales suponen cierta pérdida de información (puesto que el filtrado bajo el umbral de decisión para la existencia de relevancia provoca la eliminación total de dichos elementos para el proceso), por lo que junto al hecho de que se corresponden (como consecuencia de dicha discretización) a enfoques sobre grafos *no ponderados*, se tiene que resultan demasiado simplistas, sobre todo para este ámbito en concreto, donde los documentos o información se encuentran recogidos en lenguaje natural y el hecho de poder cuantificar la "fuerza" de la conexión entre los propios vértices resulta fundamental.

Por ello, se introduce la versión ponderada del algoritmo como :

$$WS(v_i) = (1-d) + d \sum_{v_j \in In(v_i)} \frac{w_{ji}}{\sum_{v_k \in Out(v_j)} w_{jk}} WS(v_j) \tag{7.1}$$

donde la medida w_{ij} depende del método en cuestión.

7.1 TextRank

Tal y como se ha comentado, fueron los autores de *TextRank* [58] los primeros en introducir un algoritmo para la extracción de elementos a partir de información textual. Esta técnica no supervisada fue presentada como símil al algoritmo clásico de ordenación *PageRank*, comparando la aleatoriedad de la navegación en páginas web con el concepto de *text surfing*, asociado a la búsqueda e identificación de conceptos relacionados, léxica o semánticamente, capaces de mantener la cohesión del resultado.

Concretamente, propusieron su aplicación para dos tareas de relevancia: la extracción de *palabras* o *expresiones* y la extracción de *oraciones* clave.

Keyword-Keyphrase Extraction: en este caso el resultado final esperado es el conjunto de palabras o expresiones más representativas del texto original. Por ello, para conseguir la identificación de cualquier conexión potencialmente útil entre unidades, se recurre al análisis de la relación de *co-ocurrencia*, controlada por un margen o distancia máxima prefijados, es decir, dos vértices están conectados si las unidades léxicas que representan co-ocurren en una ventana de tamaño máximo N ($2 < N < 10$).

Puesto que se centra en la relación entre elementos sintácticos, con el objetivo de reducir el tamaño del grafo de representación, en ocasiones la incorporación de vértices podría restringirse a ciertos criterios, como por ejemplo la categoría gramatical, lo que supondría una previa identificación (POS tagging).

Sentence Extraction: esta alternativa recae en la ordenación de oraciones completas, de forma que cada vértice hace referencia a una de ellas y, por consiguiente, a diferencia del caso anterior, no tiene sentido llevar a cabo el estudio según la relación de co-ocurrencia. En su lugar, se recurre a la relación de similitud en base a la *coincidencia* y *semejanza* de contenido. Dadas dos oraciones, esta podría determinarse como el número general de tokens que presentan en común, o, análogamente a la aplicación previa, precisar dichos resultados según la limitación a ciertas categorías gramaticales.

Formalmente, dadas dos oraciones s_i y s_j, cada una de ellas representadas como un conjunto de tokens, $s_i = \{w_{i1}, \ldots, w_{in_i}\}$ y $s_j = \{w_{j1}, \ldots, w_{jn_j}\}$, se propone como medida de similaridad entre las mismas :

$$Sim(s_i, s_j) = \frac{|\{w_k \mid w_k \in s_i \cap s_j\}|}{log(|s_i|) + log(|s_j|)} \tag{7.2}$$

Aunque, de igual forma, también podría recurrirse a otras medidas, como funciones núcleo para cadenas (*String Kernels*) similaridad coseno, etc.

Finalmente, puesto que la representación resultante se correspondería a un grafo *no dirigido* y *ponderado* con vértices altamente conectados, debería recurrirse a la segunda formulación indicada para el cálculo de puntuaciones (Eq. 7.1) siendo la medida de ponderación (w_{ji}) la propuesta por los autores, $Sim(s_i,s_j)$ (Eq. 7.2), y, como consecuencia de su simetría, bastaría establecer la igualdad de valores para las relaciones contrarias entre vértices (puesto que el algoritmo original se establece para grafos dirigidos).

Por último, como ventajas destacables de este enfoque, los autores defienden que, además de permitir la ordenación, identificación y resumen de documentación desde el punto de vista de extracción, la gran capacidad de esta técnica es que, a diferencia de las técncias supervisadas, evita la necesidad de recurrir a fuentes de información externa para conseguir así un aprendizaje del modelo. Además, puesto que se centra únicamente en el propio texto objetivo, no requiere de un profundo conocimiento lingüístico, un lenguaje específico o anotaciones, permitiendo así fáciles y cómodas adaptaciones a otros géneros, dominios e idiomas.

7.2 LexRank

Los autores de *LexRank* [57] proponen una modificación de la medida clásica de determinación de relevancia, *IDF*, dando lugar a:

$$\text{idf-cos}(v_i,v_j) = \frac{\sum\limits_{w\in\{v_i,v_j\}} tf(w,v_i)\ tf(w,v_j)\ idf^2(w)}{\sqrt{\sum\limits_{w_\in v_i}(tf(w,v_i)\ idf(w))^2}\sqrt{\sum\limits_{w_\in v_i}(tf(w,v_j)\ idf(w))^2}} \tag{7.3}$$

donde $tf(w,v_i)$ se corresponde con la frecuencia del token (usualmente palabra), w, en la unidad superior de análisis, vértice (usualmente oración), v_i.

Por tanto, en definitiva, la versión inicial de este algoritmo, *LexRank*, (*Lex*ical Page*Rank*) radica en el razonamiento de *PageRank* sobre un grafo *no ponderado*, pero, al igual que la técnica anterior (*TextRank*) y a diferencia de este último (*PageRank*), aplicándose sobre un grafo *no dirigido*. Además, amplía la visión correspondiente al algoritmo previo de *TextRank*, puesto que permite el análisis sobre múltiples documentos. De igual forma, dada la simetría de la medida sugerida, *idf-cos* (Eq. 7.3), al tomar como matriz de adyacencia (tras la eliminación de las no relevantes) la resultante de los

valores obtenidos, es suficiente establecer la igualdad para los valores de relevancia asociados a la relación contraria entre unidades.

Sin embargo, tal y como se indicó anteriormente, este enfoque supone cierta pérdida de información, por lo que también se introduce la versión ponderada, denominada *LexRank Continuo* (*Continuos LexRank*), en la que se considera la formulación ponderada del algoritmo (Eq. 7.1) y la medida *idf-cos* (Eq. 7.3) como ponderación.

Por último, señalar que, además de las ventajas indicadas para la técnica anterior (*TextRank*), puesto que se basan en los mismos principios, los autores defienden que la medida sugerida permite prever qué altos valores *idf* no naturales incrementen la puntuación para una oración no relacionada con el tema del documento. No obstante, aunque se tienen en cuenta las frecuencias de palabras, también permite que una oración con palabras "raras", i. e., de baja frecuencia en la misma, con elevada valoración *idf* pueda alcanzar una alta relevancia.

Machine Learning

Desde la aparición e invención de los ordenadores ya se planteó la pregunta que resultó clave para la evolución de este ámbito, [69]: ¿es posible que las "máquinas" puedan aprender?.

Centrado en la construcción de programas capaces de mejorar automáticamente en base a la experiencia, el *Aprendizaje Automático* o *Machine Learning (ML)* se inspira en la habilidad de obtener nuevo conocimiento a través del desarrollo de algoritmos con el objetivo de simular el razonamiento humano desde el punto de vista del aprendizaje.

Como ya se mostró en la representación incial, Fig. 2.2, más allá de la parte computacional, se trata claramente de un ámbito interdisciplinar, que recoge, entre otros, aspectos estadísticos y sociales, y, por tanto, está consolidada como una de las áreas importantes relativas a la Inteligencia Artificial. No obstante, al igual que la mayoría, dado el dominio y dependencia de las necesidades prácticas reales, la mejora de recursos y algortimos disponibles hasta el momento provoca que este se encuentre en una continua investigación y crecimiento (Sec. 1.4, [4]).

El propósito fundamental radica en obtener, a través del aprendizaje o entrenamiento, la mejor relación posible entre los datos iniciales disponibles, *input*, y los algoritmos, permitiendo así dar lugar a un modelo, capaz de descubrir información y apto para futuras aplicaciones. A la hora su organización, las distintas técnicas de ML suelen clasificarse según el objetivo que permiten alcanzar, así como por el carácter de los datos e información inicial necesarios para los mismos.

Estos criterios permiten establecer la división entre los siguientes ámbitos de aprendizaje:

Supervisado *(Supervised)*: se corresponde con las técnicas que recurren a datos de entrada etiquetados, según la presencia, además de las variables explicativas correspondientes, de cierta variable objetivo, permitiendo así capturar los patrones asociados a las relaciones entre características y resultado. Los algortimos supervisados se dividen a su vez en técnicas de clasificación o regresión según sea el carácter de la variable de predicción, categórico o numérico, respectivamente.

No supervisado (*Unsupervised*): hace referencia a los algoritmos que no requieren de

datos etiquetados para la construcción del modelo. A diferencia del caso anterior, donde la atención recae en el análisis predictivo de la situación, con estas técnicas se pretende encontrar las subestructuras o patrones latentes, ocultos y significativos presentes en los datos, de forma que sea posible agrupar los elementos más similares y relevantes.

Semi-Supervisado *(Semi-Supervised)*: corresponde a una combinación de los dos anteriores para los casos en los que se dispone de una etiquetación parcial de los datos.

Por refuerzo *(Reinforcement)*: se centra en obtener como resultado una recompensa o penalización asociada a la acción o entorno proporcionado como entrada, de forma que las acciones sucesivas se deciden en función de dicho aprendizaje. Por ello, es usual recurrir a estas técnicas para la implementación de robots o agentes de juego.

Tal y como se ha desarrollado a lo largo de los capítulos previos, los procedimientos más usuales para el ánalisis de información textual recaen mayoritariamente en técnicas asociadas a los aprendizajes supervisado y no supervisado, siendo, quizás, los primeros, algoritmos supervisados, los más extendidos.

Gracias a sus favorables propiedades, como su facilidad, eficiencia o interpretabilidad, el enfoque clásico e incial predominante en los ámbitos del TM y NLP se corresponde con los *modelos lineales* (Eq. 8.1) y *log-lineales* .

$$f(\mathbf{x}) = \mathbf{x}\mathbf{W} + \mathbf{b} \text{ donde}$$
$$\mathbf{x} \in \mathbb{R}^{d_i} \ (input),\ \mathbf{W} \in \mathbb{R}^{d_i \times d_o} \ y \ \mathbf{b} \in \mathbb{R}^{d_o} \ (parámetros) \tag{8.1}$$

Sin embargo, la limitación hacia funciones lineales, i. e., restricción de búsqueda sobre la familia lineal (particularización de la *hypothesis classes*), junto a la hipótesis de *separación lineal* para los datos de entrada, determinan los posibles aspectos a representar por el algoritmo de aprendizaje, *learner*, y, por tanto, evitan parte de las situaciones complejas de la realidad (Sec. 2.1, [39]). Como consecuencia, con el objetivo de mejorar el tratamiento de los datos no separables linealmente, se desarrollaron distintas alternativas (Sec. 3, [39]):

Transformación no lineal: se centra en la aplicación de cierta función no lineal conocida, ϕ, que permite resolver la situación en cuestión: $\hat{\mathbf{y}} = f(\mathbf{x}) = \phi(\mathbf{x})\mathbf{W} + \mathbf{b}$. Aunque de forma general supone un aumento de la dimensionalidad y, por tanto, un incremento en el riesgo de sobreajuste. No obstante, como mayor inconveniente de este procedimiento se destaca el hecho de que su definición es de carácter

manual y particular para cada conjunto de datos, requiriendo gran intuición humana.

Métodos basados en funciones núcleo (*Kernel Methods*): como indica su nombre, recaen en la elección de cierta *función núcleo*, $K : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$, que permita evitar el problema de la no separabilidad lineal de las observaciones, $x \in \mathcal{X}$, para las distintas categorías. Este razonamiento, conocido como **kernel trick**, busca funciones, K, que verifiquen la *Condición de Mercer*:

- K es simétrica: $K(x, x') = K(x'x)$.
- Para cualquier conjunto de puntos $\{x_1, \ldots, x_n\}$ se verifica que la matriz $K_{ij} = (K(x_i, x_j))_{ij}$ es semidefinida positiva.

lo que garantiza que sus valores se corresponden con el producto escalar en cierto espacio, $\mathcal{V}$, $\langle \cdot, \cdot \rangle_{\mathcal{V}}$, para determinada transformación de los vectores, $\varphi(x)$ (con $\varphi : \mathbb{R}^m \longrightarrow \mathbb{H}$, función *base* con imagen en el espacio de Hilbert $\mathbb{H}$:

$$K(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{V}}$$

En definitiva, aunque de forma general supone un aumento de la dimensión para el espacio de características, al basarse en el cálculo de productos escalares entre vectores, no es computacionalmente costosa y el conocimiento de la función de transformación, $\varphi : \mathcal{X} \longrightarrow \mathcal{V}$, no resulta necesario, evitando así tener que hallar la correspondencia explícita para determinar la región de decisión [70].

De entre las funciones núcleo más utilizadas destacan las correspondientes a los *kernels* de *Fisher*, *polinómico*, *de base radial* y, particularmente, *gausiano*, o los de tipo "cadena", *string*. De igual forma, aunque existe gran variedad de algoritmos operando sobre esta clase de funciones, como son las técnicas *Kernel Principal Component Analysis* o *Kernel Perceptron*, el más conocido se corresponde con las *Máquinas de Vector Soporte (Support Vector Machine, SVM)* (Sec. 9).

Por su parte, el enfoque genérico para la tarea relativa a la categorización de documentos comenzó con el estudio de Joachims [71] en 1998 en el que se utilizaba el algoritmo SVM bajo la representación estándar BOW para las características textuales. No obstante, pese a los adecuados resultados, la pérdida de información como consecuencia de tal consideración (BOW) motivó su extensión hacia funciones que incorporasen otros aspectos, como la posición de las palabras en el documento, *word position-sensitive kernels*, o la relación semántica de los términos, *semantic kernels* [72]. Análogamente, también surgieron los métodos conocidos como *secuenciales*. En primer lugar con los expuestos por Lodhi et al. [73] en 2002

(relacionados con los presentados previamente por Haussler [74] y Watkins [75]) correspondientes a los *String Kernels* y, posteriormente, con los propuestos por Cancedda et al. en 2003 [76], *Word-Sequence Kernels*. El enfoque original, *string*, se basa en el cálculo de la similitud de documentos según las correspondencias entre sub-secuencias no consecutivas de *caracteres* para cada par de secuencias iniciales, penalizándose según el número de espacios o huecos presentados. En cuanto al segundo, *word-sequence*, con el objetivo de obtener resultados más precisos y captar secuencias más informativas, los autores plantean un razonamiento similar al anterior pero elevándolo hacia las *palabras*, lo que consigue mejorar la eficiencia computacional y las tareas de preprocesamiento.

Funciones no explícitas (*Mappings Entrenables*): se centran en recurrir a funciones entrenables altamente no lineales, de forma que sea el algoritmo, durante su entrenamiento, quien se encargue de encontrar y seleccionar la representación más adecuada. Esta es la idea principal que hay detrás de las redes neuronales y, en particular, de la metodología denominada *Deep Learning* (Sec. 10).

Kernel	Fórmula	Parámetros
Polinómico	$K_{pol}(x,x') = (x^t x' + c)^p$	$p \in \mathbb{Z}^+$, $c > 0$. En NLP suele elegirse $p = 2$ (cuadrático) para no sobreajustar el resultado.
Gaussiano	$K_{gau}(x,x') = exp\left(\frac{-\|x-x'\|^2}{2\sigma^2}\right)$	$\sigma^2 > 0$. Define un espacio de características de dimensión infinita.
BOW	$K_{bow}(x,x') = \sum_{t \in V} tf(t,x) \cdot tf(t,x')$	$x = (t_{-n},\ldots,t_{-1},t_1,\ldots,t_m)$ contexto de $t_0 \in V$, $tf(t,x)$: frecuencia de t en x
Word-Pos.	$K_{wpos}(x,x') = \sum_{t \in V} \sum_{p=-c}^{c} \sum_{q=-c}^{c} tf(t,p,x) \cdot tf(t,q,x') \cdot g(p,q)$ $g(p,q) = exp\big(-\theta_1(p^2+q^2) - \theta_2(p-q)^2\big) + \theta_3$	c: longitud del contexto, $g(p,q)$: distancia entre posiciones, θ_1, θ_2: control del efecto entre posiciones, θ_3: bias.
String	$K_{str}(x,x') = \sum_{u \in \Sigma^n} \sum_{i:u=x[i]} \sum_{j:u=x'[j]} \lambda^{l(i)+l(j)}$	u: subsecuencia de x, $\lambda \leq 1$, Σ^n: conjunto de secuencias de longitud n
Word-Seq.	$K_{str}(x,x') = \sum_{u \in \Sigma^n} \sum_{i:u=x[i]} \sum_{j:u=x'[j]} \prod_{i_1 \leq k \leq i_n} \prod_{j_1 \leq l \leq j_n} \lambda_{x_k} \lambda_{x'_l}$	$\lambda_{x_k}, \lambda_{x_l} \leq 1 \ \ \forall k,l$ (distintos factores de descenso para las sub-secuencias)

Figura 8.1: Algunas Funciones Núcleo. Fuente: Elaboración Propia.

Support Vector Machine

Puesto que, generalmente, suelen proporcionar resultados aceptables para la mayoría de los casos prácticos, su versatilidad y adecuadas propiedades hacen que las *Máquinas de Vector Soporte (Support Vector Machines, SVMs)* sean ampliamente utilizadas y, por consiguiente, consideradas como unas de las técnicas fundamentales para el ámbito del *Machine Learning*.

Introducidos por Boser et al. en 1992 [77], fueron mejorándose en los años sucesivos llegando a denominarse también *Support Vector Networks* [78]. Los SVMs [79] se corresponden con un conjunto de técnicas supervisadas que combinan el principio de *Minimización del riesgo estructural (Structural Risk Minimization)*, asociado a la Teoría del Aprendizaje Estadístico (centrada en la generalización de los modelos de ML), con técnicas de optimización y la idea de correspondencia basada en las funciones núcleo [71, 73].

En su enfoque más simple, dado un conjunto de datos etiquetados, con variable respuesta binaria, i. e., $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ con $\mathbf{x}_i \in \mathcal{X}$ e $y_i \in \{\pm 1\}$, donde las clases son linealmente separables, el razonamiento de los SVMs radica en identificar los hiperplanos paralelos de mayor amplitud o distancia capaces de distinguirlas, de forma que, a partir de estos, sea posible determinar el hiperplano de máxima separación entre las mismas.

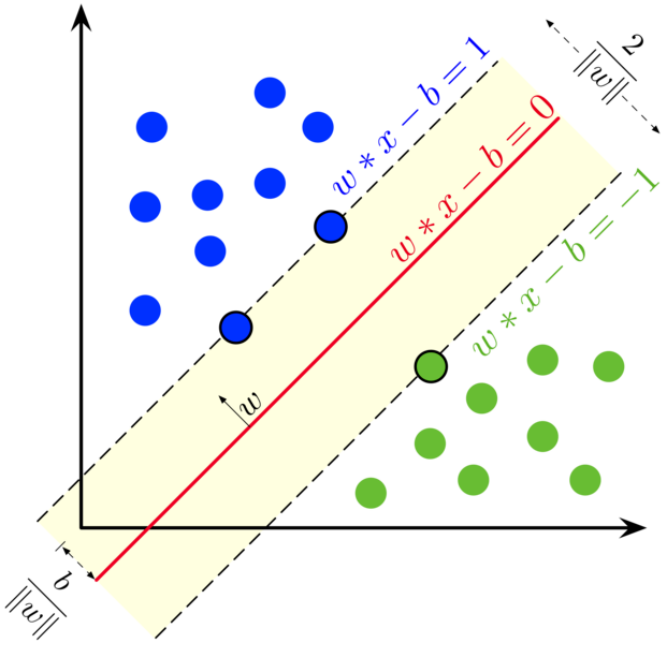


Figura 9.1: Separación lineal *SVM*. Fuente: [79].

En cuanto a las observaciones sobre las que se "apoyan" estos hiperplanos, que determinan la región de búsqueda, *margen*, reciben el nombre de *vectores soporte*, y, respectivamente, el hiperplano de máxima separación, $\mathbf{w}^t \cdot \mathbf{x} - b = 0$, es conocido como *hiperplano de margen máximo* (Fig. 9.1).

Teniendo en cuenta que debe evitarse la inclusión de las instancias dentro del margen y asegurarse la correcta ubicación de cada una de las mismas, se tiene la siguiente formulación:

$$\left.\begin{array}{lll} \mathbf{w}^t \cdot \mathbf{x}_i - b \geq 1 & \text{si} & y_i = 1 \\ \mathbf{w}^t \cdot \mathbf{x}_i - b \leq -1 & \text{si} & y_i = -1 \end{array}\right\} \Rightarrow y_i \cdot (\mathbf{w}^t \cdot \mathbf{x}_i - b) \geq 1 \ \ \forall i = 1, \ldots, N$$

Por tanto, considerando la distancia del hiperplano a cierto punto, $\mathbf{x}_i$, así como el margen del hiperplano, τ, se obtienen las siguientes equivalencias:

$$\frac{y_i \cdot (\mathbf{w}^t \cdot \mathbf{x}_i - b)}{||\mathbf{w}||} \geq \tau \ \Rightarrow \ y_i \cdot (\mathbf{w}^t \cdot \mathbf{x}_i - b) \geq \tau \, ||\mathbf{w}|| \quad \forall \, i = 1, \ldots, N$$

Esto permite, gracias a la elección del hiperplano proporcional que verifique $\tau||\mathbf{w}|| = 1$, garantizar que el hecho de maximizar el margen τ es esquivalente a minimizar $||\mathbf{w}||$ o, análogamente, minimizar $\dfrac{||\mathbf{w}||^2}{2}$.

De esta forma, para el cálculo del margen se obtiene el problema de optimización alternativo para la determinación del hiperplano de separación máxima, usualmente considerando la *norma euclídea* y resultando en un problema cuadrático convexo con restricciones lineales (se muestran a continuación las versiones primal y dual del problema de optimización asociado):

$$(P) \quad \left\{ \begin{array}{ll} \min\limits_{\mathbf{w},b} & \dfrac{1}{2}\mathbf{w^t} \cdot \mathbf{w} \\ s.a. & y_i \cdot (\mathbf{w}^t \cdot \mathbf{x}_i - b) \geq 1 \\ & \forall i = 1, \ldots, N \end{array} \right. \qquad (D) \quad \left\{ \begin{array}{ll} \max\limits_{\boldsymbol{\alpha}} & \displaystyle\sum_{k=1}^{N} \alpha_k - \frac{1}{2} \sum_{k=1}^{N} \sum_{l=1}^{N} y_k y_l \mathbf{x}_k^t \cdot \mathbf{x}_l \alpha_k \alpha_l \\ s.a. & \displaystyle\sum_{k=1}^{N} \alpha_k y_k = 0 \\ & \alpha_k \geq 0 \ \ \forall k = 1, \ldots, N \end{array} \right.$$

En cuanto al resultado, se tiene que la mayoría de los coeficientes α_k toman valores nulos puesto que tan solo los correspondientes a los vectores soporte obtienen valores positivos, i. e., $\alpha_k = 0$ sii $y_k \cdot (\mathbf{w}^t \cdot \mathbf{x}_k - b) > 1$ y, por tanto, $\alpha_k > 0$ implica que $\mathbf{x_k}$ es un vector soporte. Por consiguiente, se verifica que la solución únicamente depende de dicho subconjunto de observaciones, y, por ello, la eliminación de puntos no soporte no modificaría el modelo.

Finalmente, considerando $\mathbf{w} = \displaystyle\sum_{k=1}^{N} \alpha_k y_k \mathbf{x}_k$ y b, determinándose este a través de la elección de un vector soporte ($\alpha_i > 0$) y despejando en la igualdad $y_i \cdot (\mathbf{w}^t \cdot \mathbf{x}_i - b) = 1$,

se tiene que la regla de decisión para una nueva observación viene dada por la función:

$$d(\mathbf{x}) = \hat{y} = f(\mathbf{x}, \mathbf{w}, b) = \text{sgn}(\mathbf{w}^t \cdot \mathbf{x} - b) = \text{sgn}\left(\sum_{k=1}^{N} \alpha_k y_k \mathbf{x}_k^t \cdot \mathbf{x} - b \right)$$

No obstante, el procedimiento que acaba de ser descrito no es aplicable en la mayoría de las ocasiones, ya que, en general, las clases no resultan linealmente separables. De igual forma, en otras situaciones, pese a presentarse separabilidad lineal entre las mismas, la existencia de observaciones *outliers* podría alterar notablemente la elección del hiperplano de separación, perjudicando así la posterior clasificación para nuevos datos.

Para resolver el último de los problemas indicados en el párrafo anterior, se introducen *variables artificiales*, $\xi_i \;\; i = 1, \ldots, N$, en la formulación del problema de optimización del SVM. A diferencia del caso anterior, estas variables permiten considerar todo hiperplano que se adapte adecuadamente a la separación de las instancias disponibles, a pesar de que algunas de las correspondientes a la muestra de aprendizaje queden mal clasificadas. Se habla entonces de hiperplano de separación de *margen débil* (*soft-margin*) y para determinarlo se recurre al siguiente problema de optimización (versión primal y dual):

$$(P) \quad \begin{cases} \min\limits_{\mathbf{w},b} & \frac{1}{2}\mathbf{w}^{\mathbf{t}} \cdot \mathbf{w} + C \sum\limits_{i=1}^{N} \xi_i \\ s.a. & y_i \cdot (\mathbf{w}^t \cdot \mathbf{x}_i - b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \\ & \forall i = 1, \ldots, N \end{cases} \qquad (D) \quad \begin{cases} \max\limits_{\boldsymbol{\alpha}} & \sum\limits_{k=1}^{N} \alpha_k - \frac{1}{2} \sum\limits_{k=1}^{N} \sum\limits_{l=1}^{N} y_k y_l \mathbf{x}_k^t \cdot \mathbf{x}_l \alpha_k \alpha_l \\ s.a. & \sum\limits_{k=1}^{N} \alpha_k y_k = 0 \\ & 0 \leq \alpha_k \leq C \;\; \forall k = 1, \ldots, N \end{cases}$$

donde $C > 0$ hace referencia al parámetro de regularización sobre el coste de incrementar el tamaño del margen y una clasificación errónea para las observaciones. Cuanto mayor sea el valor para C, menor número de instancias de la muestra de aprendizaje serán clasificadas incorrectamente, ya que las variables asociadas, $\xi_i > 0$, dan lugar a una penalización mayor para la función objetivo. Análogamente, cabe destacar que la situación inicial (caso linealmente separable) se obtiene bajo la consideración límite de $C = \infty$. Por último, para el cálculo de los valores que permiten determinar el hiperplano, $\mathbf{w}$ y b, de nuevo, se recurre a las consideraciones indicadas para el caso anterior.

Ahora bien, pese a dicha consideración, existen otras muchas situaciones en las que la disposición de las observaciones hace que esta última versión (*soft-margin*) resulte insuficiente y no proporcione buenos resultados. Para evitarlo, tal y como se indicó

previamente, el enfoque central de los SVMs se encuentra en recurrir al concepto de *kernel trick*, el cual, a través de cierta función, denominada *función de inmersión* o también conocida como *función base*, tiene como propósito la inmersión de las instancias en un espacio de dimensión superior, denominado *espacio de características*, $\mathcal{F}$, donde ya sí sea posible considerar la separabilidad lineal para las mismas (Fig. 9.2).

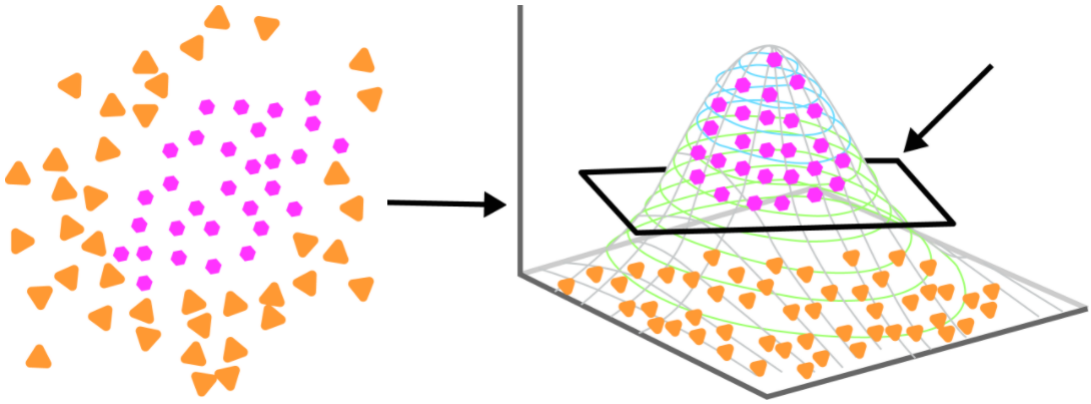


Figura 9.2: Efecto Kernel Trick. Hiperplano de separación.

Esta estrategia da lugar a clasificadores no lineales en el espacio de entrada, $\mathcal{X}$, y el objetivo radica en determinar el hiperplano de separación, en $\mathcal{F}$, para los datos transformados, $\varphi(\mathbf{x}_i)$, i. e., donde $\varphi : \mathcal{X} \longrightarrow \mathcal{F}$ se corresponde con la función de inmersión en el espacio de dimensión superior. Por tanto, deben determinarse $\mathbf{w} \in \mathcal{F}$ y $b \in \mathbb{R}$ de modo que el hiperplano $\mathbf{w}^t\varphi(\mathbf{x}) - b$ maximice el margen en $\mathcal{F}$.

En dicho caso, los productos escalares desconocidos, $\varphi^t(\mathbf{x}_k) \cdot \varphi(\mathbf{x}_l)$, son reemplazados por los valores correspondientes a la función núcleo, $K : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$, dando lugar a los siguientes problemas de optimización:

$$(P)\begin{cases} \min\limits_{\mathbf{w},b} \dfrac{1}{2}\mathbf{w^t}\cdot\mathbf{w} + C\sum\limits_{i=1}^{N}\xi_i \\ s.a.\ y_i\cdot(\mathbf{w}^t\cdot\varphi(\mathbf{x}_i) - b) \geq 1-\xi_i \\ \quad \xi_i \geq 0 \\ \quad \forall i = 1,\ldots,N \end{cases} \quad (D)\begin{cases} \max\limits_{\boldsymbol{\alpha}} \sum\limits_{k=1}^{N}\alpha_k - \dfrac{1}{2}\sum\limits_{k=1}^{N}\sum\limits_{l=1}^{N} y_k y_l \varphi^t(\mathbf{x}_k)\cdot\varphi(\mathbf{x}_l)\alpha_k\alpha_l \\ \quad = \sum\limits_{k=1}^{N}\alpha_k - \dfrac{1}{2}\sum\limits_{k=1}^{N}\sum\limits_{l=1}^{N} y_k y_l K(\mathbf{x}_k,\mathbf{x}_l)\alpha_k\alpha_l \\ s.a.\ \sum\limits_{k=1}^{N}\alpha_k y_k = 0 \\ \quad 0 \leq \alpha_k \leq C\ \ \forall k = 1,\ldots,N \end{cases}$$

donde la regla de decisión vendría dada por la función:

$$d(\mathbf{x}) \ = \ f(\mathbf{x},\mathbf{w},b) \ = \ \text{sgn}(\mathbf{w}^t\cdot\varphi(\mathbf{x}) - b) = \ \text{sgn}\left(\sum_{k|\alpha_k>0}\alpha_k y_k K(\mathbf{x}_k,\mathbf{x}) - b\right)$$

Respecto a su efectividad, se tiene que esta depende de la selección de la función núcleo y sus parámetros, pero por lo general, su buen funcionamiento hace que sea considerado como uno de los métodos de predicción más robustos del momento. Además, existe garantía de que el margen y, por consiguiente, la solución maximal, no depende de forma directa de la dimensionalidad del espacio de características, por lo que, a pesar del uso del *kernel trick*, y su correspondiente aumento de dimensión, permite evitar el posible sobreajuste tras la transformación. Por último, en cuanto a sus características, también destaca como su estructura se corresponde con la asociada a un perceptrón multicapa (Sec. 10.1.1) con tres niveles, donde cada vector soporte define un nodo oculto, y, por tanto, verifican ser *aproximadores universales* para ciertas funciones núcleo.

No obstante, pese a que el enfoque original de los SVMs presenta significativas ventajas para las aplicaciones relativas al TM, ya que se adaptan a gran parte de sus propiedades, como el hecho de no sufrir sobreajuste, de forma excesiva, ante el alto número de variables características o su eficiencia ante la dispersión de los vectores, también existen situaciones en las que resulta insuficiente.

Como consecuencia, a lo largo de su desarrollo se han ido presentando diferentes extensiones que, bajo un razonamiento y filosofía similares, permiten resolver problemas de distinta naturaleza a la original, como son las variantes correspondientes a *Support Vector Clustering* [80] (para datos no etiquetados), *Support Vector Regression* (para los casos de regresión) [81] o *Multiclass SVM* [82] (para problemas de clasificación multiclase), siendo estos últimos los más destacados.

Brevemente, en las situaciones en las que se presentan más de dos clases o categorías, $\{C_1, \ldots, C_k\}$ con $k > 2$, el procedimiento usual consiste en ir desagregando el problema original en múltiples problemas de clasificación binaria, según alguna de las siguientes alternativas:

Uno Contra Todos (*One vs. All*): se construyen k modelos, $SVM_i: \ C_i \ vs \ (C_j \ \forall j \neq i)$, de forma que dada una observación, o vector de características, $\mathbf{x}$, se le asocia la clase, $C_{i_\mathbf{x}}$, cuyo modelo le otorga el mayor margen.

Uno Contra Uno (*One vs. One*): se construyen $m = k(k-1)/2$ modelos SVM, $SVM_{ij}: \ C_i \ vs \ C_j, \ \forall j \neq i$, de forma que dada una observación, $\mathbf{x}$, se le asocia la clase, $C_{i_\mathbf{x}}$, más frecuente tras realizar el recuento sobre las elegidas por cada uno de los modelos, es decir, tras llevar a cabo una votación entre los k clasificadores.

Por último, cabe señalar que, en dichas situaciones de clasificación múltiple, también existe la posibilidad de acompañar cada decisión de clasificación, $d(\mathbf{x})$, de una probabilidad según las clases consideradas, $p_i = P(\mathbf{x} \in C_i) \ \ \forall\, i = 1,\ldots,k$, de manera que, además del propio problema original, también se plantee la estimación de dichas probabilidades (clasificación *probabilística* en SVM)[83].

Bajo un enfoque *One vs. One*, en primer lugar se estiman las probabilidades auxiliares, $r_{ij} = P(x \in C_i | x \in C_i \cup C_j) \ \ \forall\, i = 1,\ldots,k,\ i \neq j$, mediante un modelo logístico:

$$r_{ij} \approx \frac{1}{1 + exp(A_{ij} d_{ij}(\mathbf{x}) + B_{ij})} \ \ \forall\, i,j = 1,\ldots,k,\ \ i \neq j$$

donde los parámetros A y B se estiman maximizando la log-verosimilitud de la muestra de aprendizaje y $d_{ij}(\mathbf{x}) = \text{sgn}\left(\sum_{l=1}^{N} \alpha_l y_l K(\mathbf{x}_l, \mathbf{x}) - b \right)$, i. e., *score* del modelo $SVM_{ij}:\ C_i\ vs\ C_j$ para la observación $\mathbf{x}$.

Finalmente, una vez estimadas las probabilidades r_{ij}, las estimaciones correspondientes a las probabilidades p_i pueden obtenerse a partir del siguiente problema de optimización:

$$\begin{aligned} \min_{p_1,\ldots,p_k} \quad & \frac{1}{2} \sum_{i=1}^{k} \sum_{j|j\neq i} (r_{ji} p_i - r_{ij} p_j)^2 \\ s.a. \quad & \sum_{i=1}^{k} p_i = 1 \\ & p_i \geq 0 \ \ \forall\, i = 1,\ldots,k \end{aligned}$$

el cual se basa en la igualdad:

$$\underbrace{P(\mathbf{x} \in C_j | \mathbf{x} \in C_i \cup C_j)}_{r_{ji}} \underbrace{P(\mathbf{x} \in C_i)}_{p_i} = \underbrace{P(\mathbf{x} \in C_i | \mathbf{x} \in C_i \cup C_j)}_{r_{ij}} \underbrace{P(\mathbf{x} \in C_j)}_{p_j}$$

Deep Learning

El *Aprendizaje Profundo* o *Deep Learning (DL)* se corresponde con la rama del ML centrada en el conjunto de técnicas basadas en *redes neuronales (Neural Networks, NNs)*. Inspiradas en el funcionamiento del cerebro humano y, generalmente, caracterizadas por un extenso número de capas, *layer*, las redes neuronales artificiales (*Artificial Neural Networks, ANNs*) permiten el aprendizaje de funciones matemáticas paramétricas y diferenciables altamente complejas (Sec. 1.2, [39]).

Dado cierto conjunto de datos de entrada inicial, el enfoque del DL radica en realizar sucesivas transformaciones sobre los mismos para finalmente obtener la predicción del resultado. La secuenciación o encadenamiento de dichas transformaciones, aprendidas durante el propio proceso para poder así facilitar el avance hasta el objetivo deseado, permiten establecer y dar forma a la red neuronal. Una de las ventajas del DL es que, en ocasiones, simplifica notablemente el proceso de tratamiento de variables, *feature engenieering*, permitiendo al modelo combinar la arquitectura de red neuronal con el aprendizaje de las variables de mayor interés e ir así obteniendo, para cada etapa a medida que se aumenta de nivel, variables o representaciones con mayor significado (pg. 18, [39]).

Las redes neuronales, que se corresponden en general con algoritmos supervisados (pg. 13, [39]), son una poderosa herramienta para el tratamiento y resolución de los problemas de lenguaje natural. La mayoría de las NNs para el lenguaje se caracterizan por el uso de una componente (capa o nivel) denominada *embedding layer*, asociada a la correspondencia entre los símbolos discretos del lenguaje y su representación como vectores densos y continuos, *embeddings* (fijada cierta dimensión). Es de destacar que estas representaciones son, a su vez, consideradas como parámetros del modelo y, como consecuencia, la red neuronal se centra en su determinación, junto al resto de los mismos, durante el proceso de aprendizaje y entrenamiento (pg. 3, [39]).

No obstante, además de las dificultades propias relativas a su procesamiento y tratamiento, uno de los mayores problemas del lenguaje natural es su estructuración. Esto conlleva que, a diferencia de las usuales, se requieran complejas representaciones, como son las secuencias o árboles, y, por tanto, sea necesario adaptar a las mismas los distintos modelos, ya sea bien a través de algoritmos conocidos, como los modelos lineales, o a partir de novedosas arquitecturas, como los modelos *Encoder-Decoder* y,

particularmente, *Sequence-to-Sequence (seq2seq)* (pg. 4, [39]).

De entre las estructuras basadas en NNs más utilizadas, que permiten establecer múltiples combinaciones entre sí, destacan las siguientes alternativas (Sec. 1.3, [39]):

Red Neuronal Prealimentada (*Feed-Forward NN*, *FFNN*, Sec. 10.1.1): permite trabajar con un tamaño de entrada fijo y descartar el orden de los elementos. Las más conocidas se corresponden con el *Perceptrón Multi-Capa (Multi-Layer Perceptron, MLP)* y la *Red Neuronal Convolucional* (*Convolutional NN*, *CNN*, Sec. 10.1.3).

El MLP surge como alternativa a los modelos clásicos lineales, obteniendo existosos resultados en ámbitos como el modelado del lenguaje [84], seguimiento de diálogo [85] o CCG suppertaging (asignación de categorías léxicas y sintácticas más detalladas que las estructuras gramaticales correspondientes al POS tagging) [86].

Por su parte, las CNNs son arquitecturas especializadas en extraer patrones locales a partir de los datos de entrada. Gracias a las *convolutional* y *pooling layers*, estas redes permiten aprender a través de indicadores locales sin necesidad de considerar su ubicación en el texto, resultando así de gran utilidad en tareas de identificación de idiomas o expresiones. Por ejemplo, facilitan las tareas de clasificación a través del reconocimiento de las secuencias *keyphrases* (Sec. 5.5), que ayudan a determinar la temática del documento. De entre los distintos campos de aplicación, pueden destacarse los resultados obtenidos para tareas de identificación de paráfrasis [87], análisis de sentimientos [88] o detección de eventos [89].

Red Neuronal Recurrente (*Recurrent NN*, *RNN*, Sec. 10.1.2): se corresponde con un modelo especializado para datos de carácter secuencial, generalizándose hacia una *Red Neuronal Recursiva (Recursive NN)* en el caso de presentar estructura de árbol.

Las componentes asociadas a la red toman como entrada una secuencia de elementos y producen como resultado un vector de tamaño fijo y preestablecido que resume dicha secuencia de entrada. Dada la complejidad de las representaciones, el objetivo recae en capturar las regularidades de dichas estructuras en relación a sus similitudes.

Como característica destacable, permiten abandonar el supuesto de *Markov*, prevaleciente durante décadas, y diseñar modelos capaces de condicionar sobre oraciones completas, *conditioned generation models*. Un ejemplo de ello se corresponde con las estructuras Encoder-Decoder, capaces de tomar en consideración el orden

de las palabras únicamente cuando resulte necesario y, además, sin sufrir graves dificultades en la estimación estadística (pg. 163, [39]).

En cuanto a su aplicación, uno de los campos con mayor reflejo de la contribución correspondiente a estas técnicas es el modelado del lenguaje, *language-modeling*, tarea centrada en predecir la probabilidad de la siguiente palabra dada cierta secuencia o, equivalentemente, la probabilidad de una secuencia dada cierta palabra. Entre otros, también se recogen aplicaciones destacadas en los campos de traducción [90] o generación de respuetas [91].

10.1 ARQUITECTURAS

Inspiradas en el razonamiento del cerebro humano, las redes neuronales tienen como base y unidades simples, las neuronas (Fig. 10.1, izquierda). Cada dato de entrada se recibe junto a su respectivo peso o ponderación y, tras su combinación, se aplica la función de *activación* elegida para, finalmente, obtener el resultado.

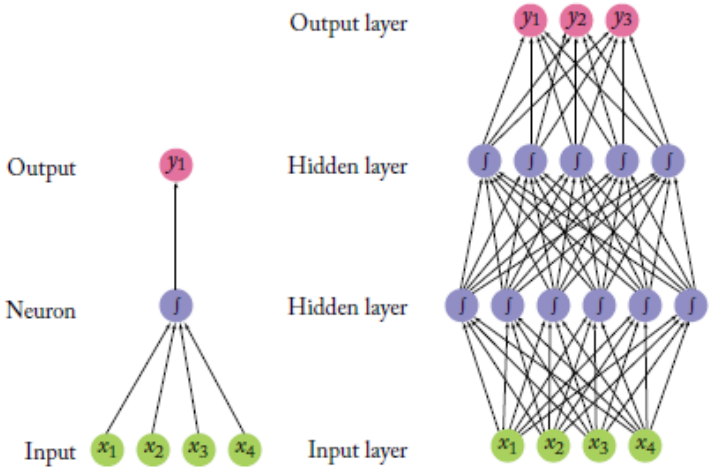


Figura 10.1: Neurona y MLP_2. Fuente: pág. 41 y 42 [39].

De forma general, la combinación de las entradas junto a sus pesos sinápticos se lleva a cabo de forma aditiva, $x = \sum_{i=1}^{n} \omega_i x_i$, de manera que, posteriormente, es la función de activación, f, según el valor de umbral o sesgo preestablecido (usualmente denotado como θ) y dicha combinación obtenida, x, la encargada de transmitir la información como la salida de la neurona, y. Análogamente, conectándose entre sí, alimentando las sucesivas capas con los resultados obtenidos en las anteriores, estas permiten construir complejas estructuras, *redes*, capaces de aproximar un amplio rango de funciones matemáticas (Sec. 4, [39]).

La organización de las neuronas en las distintas capas, con el objetivo de reflejar la importancia y flujo de la información, varía según el propósito o estructura de la red neuronal. Un ejemplo clásico de dichas combinaciones se observa en la Fig 10.1 (derecha) donde se muestra la estructura típica de red neuronal prealimentada con dos capas ocultas, *hidden layer*, y completamente conectadas, *fully connected* o *affine layer*

(Eq. 10.1). En definitiva, a medida que se va añadiendo un mayor número de capas ocultas o variando la estructura y procesamiento de los datos de entrada, la red va adquiriendo complejidad. De ahí que se denominen redes profundas, *deep* networks, y, como consecuencia, este ámbito sea conocido como *deep learning*.

Por su parte, puesto que con estos modelos se busca evitar la limitación de los métodos clásicos, generalmente, las funciones de activación o transición se corresponden con funciones de carácter no lineal, como son las funciones *Sigmoid*, *Softmax*, *Tangente hiperbólica (tanh)* y *Rectified Linear Unit (ReLU)*.

Activación	Fórmula	Propiedades
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	Generalmente referida al caso particular de función logística.
Softmax	$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_k^K e^{x_k}} \quad \forall i = 1 \dots K$	Función exponencial normalizadora. Fuerza que sus valores sumen 1 y puedan interpretarse como probabilidades.
Tanh	$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$	Entra dentro de las funciones 'S-shape' escalando los valores al rango [-1,1].
ReLU	$ReLU(x) = \max(0,x) = \begin{cases} 0 & x < 0 \\ x & c.c. \end{cases}$	A pesar de su simplicidad proporcioda muy buenos resultados, sobretodo junto a la técnica dropout.

Figura 10.2: Funciones de Activación. Fuente: Elaboración Propia.

De igual forma, puesto que el objetivo de estos algoritmos radica en obtener un buen modelo para el conjunto de entrenamiento, para aumentar su precisión, como en cualquier otro algoritmo, resulta necesario encontrar valores adecuados para sus parámetros. Para ello, el proceso de optimización se lleva a cabo según la minimización de cierta función de pérdida, *loss function*, encargada de cuantificar el error o coste de predicción, como son las funciones *Hinge*, *Log loss*, *Binary* y *Categorical cross-entropy* (Fig. 10.3), así como, a través de la consideración de un término de regularización, *regularization*, destinado a controlar el sobreajuste, como son las regularizaciones L_1, L_2 o *weight decay*, y *elastic-net* (Sec. 2, [39]).

Como última consideración, cabe precisar también que las NNs dan lugar a funciones parametrizadas diferenciables, y pese a que la función objetivo para las no lineales no es convexa y, por consiguiente, la optimización podría recaer en un mínimo local, los métodos basados en el descenso del gradiente, *gradient-based optimization*, en concreto, *Stochastic Gradient Descent*, a través del algoritmo de *backpropagation*, son los usualmente utilizados (Sec. 2.8 y pg.51, [39]).

Pérdida	Fórmula	Propiedades
Hinge (Binary)	$L_h(x,x') = \max(0, 1 - x \cdot x') \;\; \text{con } x \in \{\pm 1\}$	Conocida también como 'margin' o 'SVM loss'. Extensible al caso múltiple.
Cross-Entropy (Categorical)	$L_{ce}(x,x') = -\sum_i x_i log(x'_i)$	Analiza las diferencia entre las distribuciones de probabilidad. Se asume que se recurre a la función 'softmax' para el resultado del clasificador.

Figura 10.3: Funciones de Pérdida. Fuente: Elaboración Propia.

10.1.1 FeedForward Network (FFNN)

La red neuronal más simple, *perceptron*, se corresponde con un modelo lineal:

$$NN_{\text{perceptron}}(\mathbf{x}) = \mathbf{xW} + \mathbf{b}$$

donde $\mathbf{x} \in \mathbb{R}^{d_i}$ es el vector de entrada (*input*), $\mathbf{W} \in \mathbb{R}^{d_i \times d_o}$ la matriz de pesos o ponderaciones, y $\mathbf{b} \in \mathbb{R}^{d_o}$ término adicional (*bias*) a la capa sin conexiones de entrada.

Con el objetivo de sobrepasar esta limitación (linealidad) se introducen las capas ocultas no lineales, dando lugar a los *Perceptrones multi-capa (Multi-Layer Perceptron, MLP)*. Por ejemplo, la estructura para el correspondiente a la Fig. 10.1, con dos capas ocultas:

$$\begin{aligned} NN_{MLP2}(\mathbf{x}) &= \mathbf{y} = \mathbf{h^2 W^3} \\ &= g^2(\mathbf{h^1 W^2} + \mathbf{b^2})\mathbf{W^3} \\ &= (g^2(g^1(\mathbf{x})\mathbf{W^1} + \mathbf{b^1})\mathbf{W^2} + \mathbf{b^2})\mathbf{W^3} \end{aligned} \tag{10.1}$$

$$\mathbf{x} \in \mathbb{R}^{d_i},\ \mathbf{W^1} \in \mathbb{R}^{d_i \times d_1},\ \mathbf{b^1} \in \mathbb{R}^{d_1},\ \mathbf{W^2} \in \mathbb{R}^{d_1 \times d_2},\ \mathbf{W^3} \in \mathbb{R}^{d_2 \times d_o},\ \mathbf{b^2} \in \mathbb{R}^{d_2}$$

donde g, g^1 y g^2 corresponden a las funciones de activación asociadas a cada una de las transformaciones.

A pesar de que, teóricamente, la estructura *MLP* es considerada como un *aproximador universal*, esta no discute la capacidad de aprendizaje del modelo, i. e., se garantiza su existencia, pero no se establece la dificultad para hallar el conjunto de parámetros óptimo en base a los datos de entrenamiento y algoritmo especificado (Sec. 4.3 de [39]). Además, otra de las mayores desventajas de las FFNN es que la longitud fija, de entrada y salida, debe ser conocida a priori, existiendo numerosas aplicaciones, como el reconocimiento de voz o traducción de documentos, en las que esta determinación inicial no es factible en la práctica, suponiendo así un gran inconveniente (Sec. 2, [92]).

Por tanto, como consecuencia, es necesario recurrir a arquitecturas más complejas en comparación con este tipo de estructura inicial.

10.1.2 Recurrent Neural Network (RNN)

Las RNNs (Sec. 4, [92]) permiten relajar la imposición de las condiciones no cíclicas asociadas a las estructuras clásicas de FFNNs. Mientras que los MLPs únicamente permiten progresar hacia adelante, de entrada a resultado, las RNNs permiten que el proceso transcurrido, i. e., la historia, influya en los resultados consecutivos de la red.

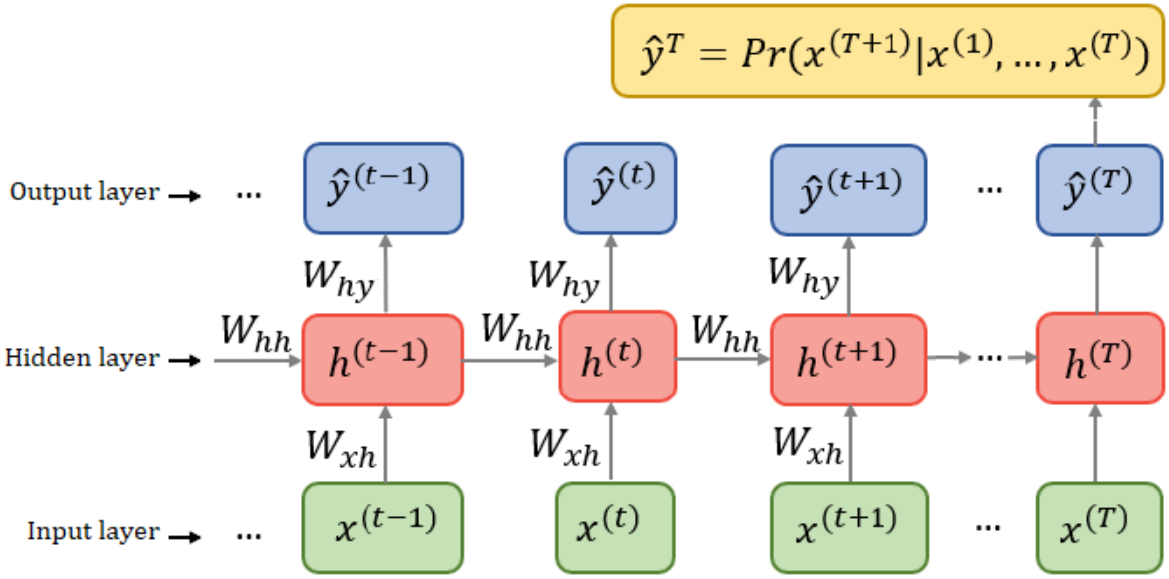


Figura 10.4: Estructura básica de una RNN. Fuente: pág. 36 [92].

Tal y como se observa en la Fig. 10.4, cada nodo está asociado con una capa de la red en cada instante de tiempo. Las entradas, $x^{(t)}$, son recurrentemente conectadas con los vectores resultantes de las capas ocultas, h, denominados *estados ocultos*. En cada instante, $h^{(t)}$ combina la información del estado oculto anterior, $h^{(t-1)}$, con la asociada a la entrada correspondiente, $x^{(t)}$. Evidentemente, puesto que no existe memoria al comienzo del proceso secuencial, es necesaria la inicialización del estado oculto, $h^{(0)}$. Finalmente, a partir de los vectores de resultado, $\hat{y}^{(t)}$, se construye la distribución predictiva $Pr\left(x^{(t+1)}|y^{(t)}\right)$ para la siguiente entrada. Por tanto, la recurrencia de la red puede formularse como:

$$h^{(t)} = g^{(t)}\left(x^{(t)}, x^{(t-1)}, \ldots, x^{(2)}, x^{(1)}\right) = f\left(h^{(t-1)}, x^{(t)}|\theta\right)$$

donde $g^{(t)}$ tiene en cuenta toda la secuencia de entradas hasta dicho instante, siendo a su vez posible, como consecuencia de su estructura cíclica, su factorización hacia repe-

tidas aplicaciones sobre f (función de activación de la capa oculta) y siempre teniendo en cuenta el efecto *parameter sharing* de los paramétros, θ, i. e., manteniendo invariantes y reutilizando las matrices de ponderaciones para las conexiones a lo largo de toda la red (para la conexiones de entrada y capa oculta, W_{xh}, conexiones recurrentes entre estados ocultos, W_{hh}, y conexiones entre los estados ocultos y salidas, W_{hy}) para así evitar el crecimiento del modelo y facilitar la generalización en diferentes posiciones simultáneamente.

Puesto que, tal y como se indicó anteriormente, el objetivo del modelo es obtener las distribuciones predictivas $Pr\left(x^{(t+1)}|y^{(t)}\right)$ para cada instante, la distribución de la red puede expressarse como:

$$Pr(x) = \prod_{t=1}^{T} Pr\left(x^{(t+1)}|y^{(t)}\right)$$

de forma que la pérdida total a minimizar durante el proceso de entrenamiento resulta simplemente la suma de pérdidas sobre todos los instantes a través de la log-verosimilitud negativa de $Pr(x)$, i. e.,

$$\mathcal{L}(x) = \sum_{t=1}^{T} \mathcal{L}^{(t)}(x) = -\sum_{t=1}^{T} log\ Pr\left(x^{(t+1)}|y^{(t)}\right)$$

En cuanto al entrenamiento del modelo, es necesario el cálculo de los gradientes asociados a las matrices de ponderaciones, W_{xh}, W_{hh} y W_{hy}. Para ello, tal y como se indicó inicialmente, se recurre al algoritmo de *backpropagation*, pero al existir, en este caso, dependencia con los resultados anteriores, es necesaria su modificación para que la obtención de los gradientes se realice teniendo en cuenta el carácter recursivo de estas estructuras, y, por tanto, en realidad, se lleva a cabo a través del procedimiento conocido como *backpropagation through time*.

No obstante, como consecuencia de dichas computaciones recursivas para el cálculo de los gradientes, estos podrían resultar muy grandes o muy pequeños, lo que da lugar a dos de los problemas para las RNNs en su forma más simple: la *explosión* y el *desvanecimiento* de gradientes, respectivamente, que conllevan la pérdida total de la información de las dependencias a largo plazo.

Con el objetivo de evitarlos, se han propuesto diferentes técnicas, como mejorar la inicialización de las matrices de ponderaciones, normalizar las componentes o utilizar penalizaciones del tipo L_1 y L_2 sobre los pesos recurrentes, pero, sin duda, las más conocidas y usuales son las *gated RNNs*. Estas se caracterizan por su capacidad de

retener la información correspondiente a los términos de la red durante largos periodos y, simultáneamente, de continuar procesando nuevos inputs de la forma más efectiva posible. Dentro de las estructuras desarrolladas, las dos más aplicadas en el ámbito del TM son las redes *Long Short-Term Memory* (*LSTM*) y *Gated Recurrent Units* (*GRU*).

Sin embargo, la exigencia de las tareas relativas a este ámbito, como la traducción de documentos o la generación de texto, provoca que, en ocasiones, estas arquitecturas resulten insuficientes y, por tanto, también sea necesario recurrir a redes más profundas, con múltiples capas ocultas, y complejas, como son las correspondientes a arquitecturas *Encoder-Decoder* (detallada posteriormente) o su extensión para tareas múltiples de los modelos *Seq2Seq*, *Multi-task learning seq2seq* [93].

Long Short-Term Memory

Desde su introdución en 1997 por Hochreiter y Schmidhuber [94], con el objetivo principal de solventar los problemas sobre las largas dependencias, diferentes autores han ido mejorando y perfeccionando estas estructuras, dando lugar a distintas variantes, como son la incorporación de conexiones *peephole* por Gers et al. [95] o los propios *GRU*.

De forma general, la clave de los *LSTMs* [96] se encuentra en la evolución de la célula de estado, *cell state*, de $C^{(t-1)}$ a $C^{(t)}$. De forma simple, esta puede identificarse con una especie de "cinta transportadora" (línea horizontal recorriendo la parte superior Fig. 10.5) de la que es posible eliminar o añadir infomación a través de la regulación de las "puertas", estructuras denomidas *gates*.

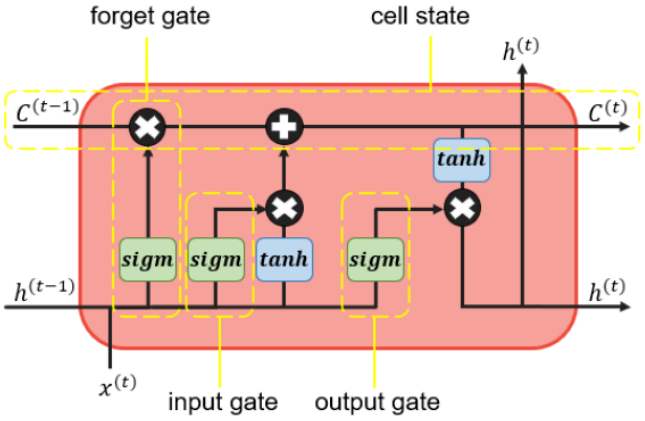


Figura 10.5: Evolución en *LSTM*. Fuente: pág. 41 [92].

Usualmente, estas se componen de una capa de red neuronal, layer, con función de activación *sigmoid*, de manera que, según el valor obtenido para la misma, se decide la cantidad de información a incorporar en la célula de estado.

El primer paso consiste en decidir qué información debe permanecer en la célula. Para ello, una primera capa, *forget gate layer*, $f^{(t)}$, tomando como entrada $h^{(t-1)}$ y $x^{(t)}$,

devuelve un valor entre 0 y 1 para cada elemento de la célula anterior, $C^{(t-1)}$, donde 0 correspondería a una eliminación total, mientras que 1 significaría que se mantiene toda la información.

A continuación, debe establecerse qué es lo que se debe incorporar. Este proceso se lleva a cabo en dos etapas. Primero se deciden los valores a actualizar según otra capa *sigmoid*, *input gate layer*, $i^{(t)}$, y, después, otra capa, a través de la función *tanh*, establece los nuevos candidatos, $\tilde{C}^{(t)}$. Por ejemplo, en este punto, el proceso de evolución de la célula en el instante t correspondería a: $C^{(t)} = f^{(t)} \cdot C^{(t-1)} + i^{(t)} \cdot \tilde{C}^{(t)}$.

Finalmente, debe decidirse qué se va a emitir como resultado. Este se basa en la actualización de la célula de estado, pero aplicándose un filtrado. Primero una capa sigmoid, *output gate*, $o^{(t)}$, y después la función *tanh* para mostrar únicamente las partes de la célula de estado seleccionadas.

$$
\begin{aligned}
f^{(t)} &= \sigma(W_{xf}x^{(t)} + W_{hf}h^{(t-1)} + b_f) \\
i^{(t)} &= \sigma(W_{xi}x^{(t)} + W_{hi}h^{(t-1)} + b_i) \\
o^{(t)} &= \sigma(W_{xo}x^{(t)} + W_{ho}h^{(t-1)} + b_o) \\
\tilde{C}^{(t)} &= tanh(W_{xc}x^{(t)} + W_{hc}h^{(t-1)} + b_c) \\
C^{(t)} &= f^{(t)} \cdot C^{(t-1)} + i^{(t)} \cdot \tilde{C}^{(t)} \\
h^{(t)} &= o^{(t)} tanh(C^{(t)})
\end{aligned}
$$

Gated Recurrent Units

Fueron introducidos por Cho et al. en 2014 [97], a través del caso práctico de captura de significado semántico y sintáctico en el aprendizaje de representaciones para frases linguísticas.

Se centran en diversas modificaciones de la estructura LSTM original, como la combinación de dos de las etapas anteriores, *forget* e *input gates*, en la denominada *update gate*, así como la fusión de la célula y estado oculto, entre otras. Puesto que finalmente consta de dos etapas: *reset* y *update gate*, resulta más simple que el modelo estándar anterior (Sec. 4.2.2, [92]).

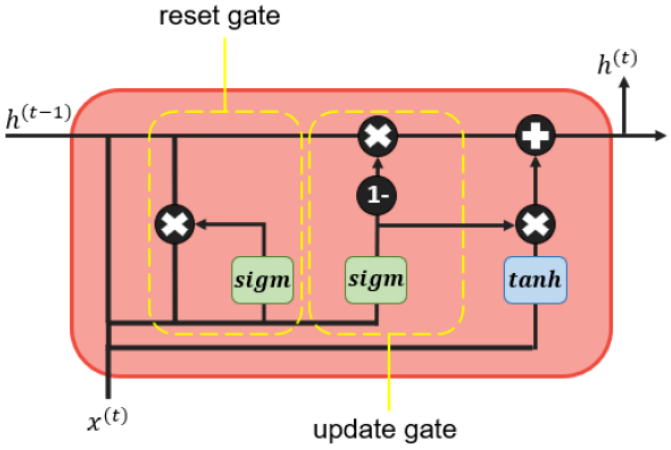


Figura 10.6: Evolución en *GRU*. Fuente: pág. 41 [92].

En la primera de ellas, *reset*, $r^{(t)}$, se decide la cantidad a ignorar asociada al estado

oculto previo, $h^{(t-1)}$. Posteriormente, se calcula el candidato a estado oculto, $\tilde{h}^{(t)}$, según la transformación *tanh* de las multiplicaciones correspondientes, $W_{xh}x^{(t)}$ y $W_{hh}(r^{(t)} \odot h^{(t-1)})$ donde $\odot$ indica producto elemento a elemento.

Por su parte, la segunda etapa, *update*, $z^{(t)}$, decide cuánta información de la almacenada en $h^{(t-1)}$ debe permanecer en $h^{(t)}$, así como la cantidad en la que el nuevo estado debe actualizarse según el candidato, $\tilde{h}^{(t)}$.

$$
\begin{aligned}
r^{(t)} &= \sigma\left(W_{xr}x^{(t)} + W_{hr}h^{(t-1)}\right) \\
\tilde{h}^{(t)} &= tanh\left(W_{xh}x^{(t)} + W_{hh}(r^{(t)} \odot h^{(t-1)})\right) \\
z^{(t)} &= \sigma\left(W_{xz}x^{(t)} + W_{hz}h^{(t-1)}\right) \\
h^{(t)} &= z^{(t)} \odot h^{(t-1)} + (1 - z^{(t)}) \odot \tilde{h}^{(t)}
\end{aligned}
$$

Encoder-Decoder

Concretamente, en los ámbitos del lenguaje, la tarea de correspondencia entre longitudes variables de entrada y salida para las secuencias de la red neuronal se conoce como aprendizaje *Sequence-to-Sequence (seq2seq)*. Aunque originalmente se introdujo como aplicación para la traducción [98], actualmente el enfoque de esta técnica se encuentra como estado del arte en muchas otras, como reconocimiento de voz [99] o captura de vídeo [100] (*Video Captioning*, intersección entre NLP y *Computer Vision* con el objetivo de obtener las declaraciones en lenguaje natural que describen el contenido visual [101]).

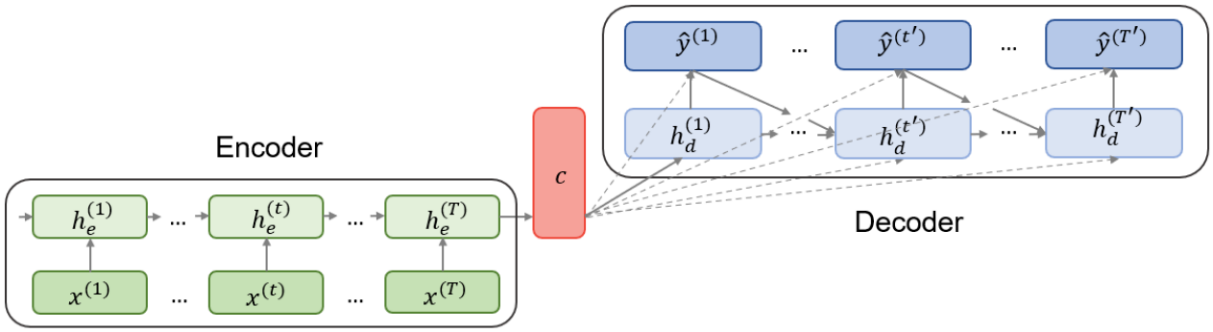


Figura 10.7: Estructura Encoder-Decoder (Seq2Seq). Fuente: pág. 43 [92].

Tal y como se observa en la Fig. 10.7, esta se divide en dos partes. La primera recae en el *Encoder*, una RNN que tiene como objetivo proporcionar el vector de contexto, c (generalmente $c = h_e^{(T)}$, i. e., información del último estado oculto), capaz de resumir, tras el entrenamiento, la información correspondiente a la secuencia de inputs, $x^{(t)}$. Por su parte, la segunda, *Decoder*, se trata de otra RNN centrada en generar predicciones, $y^{(t)}$, dados el vector de contexto, c, y todos los outputs previos, $h_e^{(t)} \rightarrow h_d^{(t)}$. De esta for-

ma, a diferencia de las estructuras simples que se indicaron anteriormente, los estados ocultos del decoder, $h_d^{(t)}$, y las predicciones, $y^{(t)}$, quedan condicionados a los resultados previos, $y^{(t-1)}$ y c, y, por tanto, según ciertas funciones de activación f y g, el cálculo del estado oculto del Decoder en el instante t resulta:

$$h_d^{(t)} = f(h_d^{(t-1)}, y^{(t-1)}, c)$$

y análogamente, para la distribución condicional de la predicción:

$$P(y^{(t)}|y^{(t-1)}, \ldots, y^{(1)}, c) = g(h_d^{(t)}, y^{(t-1)}, c)$$

De esta forma, ambas partes, Encoder-Decoder, se entrenan simultáneamente maximizando la log-verosimilitud condicional:

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} log\ Pr_\theta(y_n|x_n)$$

donde θ denota el conjunto de parámetros del modelo y (x_n, y_n) corresponden a los pares (input, output) del conjunto de entrenamiento [102].

10.1.3 Convolutional Neural Network (CNN)

Como se observa en la Fig. 10.8, la estructura básica de una CNN (Sec. 5, [92]) se basa en la iteración de múltiples capas ocultas, principalmente, *convolutional* y *pooling layers*, para finalmente aplicar una capa completamente conectada.

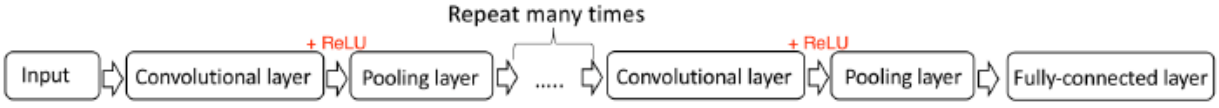


Figura 10.8: Estructura básica de una CNN. Fuente: pág. 47 [92].

Convolutional layer: se encarga de abstraer la información de entrada hacia las correspondencias denominadas *feature maps*. Para ello, recurre a ciertos filtros entrenables, *filters* o *kernels*, centrados en un enfoque algebraico, de manera que cada uno de ellos establece una correspondencia acorde a su propia dimensión. De forma general, el resultado de una capa convolucional es el conjunto completo de correspondencias, apiladas a medida que se va incrementando la profundidad.

ReLU layer: dado que las capas convolucionales se basan en multiplicaciones o sumas elemento a elemento, i. e., en definitiva, operaciones lineales, en el caso de que se presentase relación no lineal entre neuronas, esta no quedaría recogida y, por

tanto, es necesario establecer una capa de activación no lineal tras la aplicación de cada una de las mismas.

Aunque tal y como se describió en la sección introductoria, existe una gran variedad de funciones de activación no lineales para las redes neuronales, en las CNNs es especialmente frecuente el uso de la función *ReLU* (*Rectified Linear Unit*) (Fig. 10.2) puesto que evita que las correspondencias (feature maps) tomen valores negativos.

Pooling layer: tiene como objetivo reducir, de forma progresiva, el tamaño espacial de la correspondencia (generada en la capa convolucional previa) a través de la identificación de las características más importantes.

Existen múltiples operaciones de acumulación como la media (*average pooling*), la norma l_2, o *max pooling*, siendo esta última la función más utilizada.

La idea de la acumulación por máximo recae en el hecho de que la localización exacta de una característica es de menor importancia que conocer, no necesariamente con exactitud, su ubicación relativa respecto a otras de interés, i. e., localmente. Puesto que permite reducir la dimensión del resultado previo correspondiente a la capa convolucional, en definitiva, reduce el número de parámetros de la red y, por consiguiente, la complejidad computacional. De igual forma, al extraer únicamente las principales características de cada sub-región, permite controlar el sobreajuste del modelo.

Fully-Connected layer: finalmente, se aplican una o más capas completamente conectadas al resultado obtenido por la combinación de las anteriores de forma que cada neurona de esta última se encuentra totalmente conectada con todas las neuronas de la anterior (penúltima). Generalmente, al igual que ocurría en las anteriores, la función correspondiente a cada una de las neuronas suele corresponderse con la función *ReLU*.

A partir de esta estructura básica, se han ido construyendo modelos más específicos para el lenguaje, basados en arquitecturas propias de CNN y entre los que destacan (Sec. 5.2, [92]):

CNN-rand, CNN-static, CNN-non-static, CNN-multichannel: introducidos por Kim en 2014 [103], conceptualmente destacan por su simple y eficiente aplicación para tareas como el análisis de sentimientos (Sec. 5.3) o la clasificación de preguntas.

En esencia, la subestructura general se resume en las etapas de: (i) representación, (ii) capa convolucional, (iii) capa de acumulación con max-pooling, y (iv)

capa completamente conectada. No obstante, pese a seguir la misma estructura, individualmente cada uno de ellos se caracteriza por:

- *CNN-rand*: inicializa todas sus palabras aleatoriamente y lleva a cabo la modificación durante el entrenamiento.
- *CNN-static* y *CNN-non-static*: recurren a vectores pre-entrenados a través de Word2Vec (Sec. 10.2.1) y los mantienen estáticos o ajustados según cada tarea, respectivamente.
- *CNN-multichannel*: un modelo con dos canales genera dos conjuntos de vectores y cada filtro es empleado para ambos.

Character-level ConvNets: introducido por Zhang et al. en 2015 [104], este modelo aplica 6 capas convolucionales y 3 completamente conectadas, centrándose en los caracteres (*Character quantization*) en lugar de las palabras.

Deep Pyramid CNN: introducido por Johnson y Zhang en 2017 [105], a diferencia de otros modelos complejos, como las arquitecturas CNN muy profundas (*Very Deep CNN* [106]), este se construye a través de una arquitectura de baja complejidad (*low-complexity*) y basada a nivel de palabras (*word-level*). La justificación de los autores recae en que las estructuras superficiales (*shallow*), como las indicadas previamente CNN-rand o CNN-static, de menor complejidad, así como el estudio a nivel de palabras, proporcionan mayor precisión y rapidez que el resto de alternativas, como *ConvNets*, a la hora de la computación del modelo.

10.2 Words Embeddings

Cuando la entrada para la red neural contiene variables categóricas simbólicas, como un número o lista fija de palabras, y, por tanto, cada variable representa una de ellas, es común asociar cada posible característica con un vector d-dimensional. Como se indicó previamente, estos vectores son a su vez considerados parámetros del modelo y, por tanto, entrenados conjuntamente al resto de parámetros de la red. Se incluyen en la *embedding layer*, correspondiéndose simplemente con la matriz $E \in \mathbb{R}^{|V| \times d}$ donde cada fila corresponde a una palabra o unidad del vocabulario y d al número de dimensiones con el que se desea obtener la representación (Sec 4.8, [39]).

Tal y como se comentó en la sección correspondiente (Sec. 4.3), el desarrollo de este tipo de representaciones, desde el punto de vista de los modelos neuronales, se llevó a cabo bajo la idea de la mejora del modelado estadístico del lenguaje, *language modeling*.

Desde un enfoque tradicional, el *language modeling* se basaba en el supuesto de Markov de forma que la estimación para la probabilidad en cuestión se derivaba, según el orden de la secuencia, a través del método de máxima verosimilitud, que, en general, radicaba en el recuento de frecuencias en el corpus (Sec. 9.3 [39]). Sin embargo, las limitaciones de este planteamiento, como la dispersión a la hora de considerar extensas secuencias de N-gramas o la dificultad de condicionar hacia contextos más "creativos" o variantes, motivaron el uso de los modelos neuronales.

Partiendo del modelo probabilístico para el modelado del lenguaje (*Neural Network Language Model, NNLM*) propuesto por Bengio et al. en 2003 [107], fueron desarrollándose otros enfoques, como el propuesto por Collobert y Weston [108] [109].

El objetivo de este primer modelo, NNLM, es predecir la palabra sucesiva en base a cierta secuencia fijada. Utilizando una simple red neuronal prealimentada, el modelo primero se centra en aprender los embeddings y, posteriormente, en un segundo paso, la función de probabilidad para las palabras de interés. La arquitectura neuronal propuesta comienza con una capa inicial, recibiendo las palabras según una representación *one-hot*, y continúa con una capa de proyección lineal para los embeddings y una capa oculta, con la función de tangente hiperbólica como activación, para finalmente recurrir a la función *softmax* para el cálculo de las probabilidades, obteniendo así como resultado el vector de probabilidades para todas las palabras especificadas (pg. 292, [4] y Sec. 3.2.1, [92]).

Por su parte, Collobert y Weston (pg. 123, [39]), alejándose del cálculo probabilístico se centraron, a través de una estructura MLP_1, en establecer la puntuación para cada par palabra-contexto, $(w, c = (c_1 \dots c_k))$, según la concatenación de sus embeddings:

$$\begin{aligned} s(w, c_{1:k}) = &\quad g(\mathbf{x}\mathbf{U})\mathbf{v} \\ \mathbf{x} = &\quad [v_c(c_1); \dots; v_c(c_k); v_w(w)] \\ \mathbf{U} \in &\quad \mathbb{R}^{(k+1)d_e \times d_h},\ \mathbf{v} \in \mathbb{R}^{d_h} \end{aligned}$$

de forma que la red es entrenada según cierta función de pérdida (basada en margen, filosofía similar a la función de pérdida *hinge*, Fig. 10.3) para puntuar los pares correctos $(w, c_{1:k})$ sobre los incorrectos $(w', c_{1:k})$ con margen de al menos 1.

$$L(w, c, w') = max(0, 1 - s(w, c_{1:k}) - s(w', c_{1:k}))$$

En cuanto a su proceso de entrenamiento, con el objetivo de minimizar dicha pérdida, este itera sobre la selección aleatoria de w' y el cálculo de $L(w, c, w')$ según la actualización de los parámetros **U, v** y embeddings para las palabras y contexto.

No obstante, tras la introducción de estos métodos iniciales, la mejora y evolución de estos procedimientos, dieron lugar a los más recientes y conocidos: *Word2Vec* y *GloVe*.

10.2.1 Word2Vec

El algoritmo *Word2Vec* fue desarrollado en 2013 [15] y centra su base en un modelo neuronal del lenguage a través de una simple NN prealimentada, permitiendo, tal y como se comentó en la sección correspondiente (Sec. 4.3), dos posibles representaciones contextuales: *Continuous Bag-Of-Words (CBOW)* y *Skip-Gram*, según se desee predecir la palabra actual en base al contexto establecido o viceversa (Fig. 10.9). De igual forma, aunque originalmente se propuso un enfoque centrado en la función *(log-linear) Hierarchical Softmax*, en lugar de la función *softmax* estándar, con el objetivo de agilizar el cálculo de la red neuronal, posteriormente los autores propusieron un nuevo método de optimización más eficiente, *Negative Sampling (NS)* (Sec. 3.3 [92]).

Respecto a esta última variante del algoritmo, NS (pg. 124 [39]), esta entrena la red neuronal con el objetivo de distinguir las parejas palabras-contexto adecuadas, (w,c), de las erróneas o insignificantes. Sean D y $\overline{D}$ los conjuntos de pares, palabra-contexto, correctos e incorrectos, respectivamente, el algoritmo radica en estimar la probabilidad $P(D=1|w,c)$, es decir, la probabilidad de que el par provenga del conjunto correcto. Lógicamente, esta debería tomar valores elevados para los pares de D y bajos para los pares de $\overline{D}$ y, según la restricción de probabilidad, debe verificarse:

$$P(D=1|w,c) = 1 - P(D=0|w,c)$$

En cuanto a la función de probabilidad, esta se modela a través de la función *sigmoid* según la puntuación para cada par, $s(w,c)$:

$$P(D=1|w,c) = \frac{1}{1+e^{-s(w,c)}} \tag{10.2}$$

de forma que debe maximizarse la log-verosimilitud de los datos $D \cup \overline{D}$:

$$\mathcal{L}(\theta;D,\overline{D}) = \sum_{(w,c)\in D} log\ P(D=1|w,c) + \sum_{(w,c)\in \overline{D}} log\ P(D=0|w,c)$$

donde las muestras positivas, D, son generadas a partir del corpus y las negativas, $\overline{D}$, según el siguiente proceso: para cada par $(w,c) \in D$, seleccionar aleatoriamente k (parámetro del algoritmo) palabras, $w_{1:k}$, y añadir cada par (w_i,c) $(1 \leq i \leq k)$ en $\overline{D}$.

Aunque ambos enfoques de aprendizaje, CBOW y Skip-gram, siguen una filosofía parecida al enfoque anterior, NNLM, pero con la diferencia de que la capa oculta no lineal es eliminada (centrándose así propiamente en el cálculo de embeddings), y presentan una estructura similar, con capas de entrada, proyección y resultado, cada uno de los mismos deriva en un proceso final diferente (pg. 125, [39] y Sec. 3, [15]):

CBOW: en esta variante, la capa de proyección, completamente conectada, se comparte para todas las palabras del contexto, i. e., estas son proyectadas en la misma posición de forma lineal. Para conseguirlo, se lleva a cabo la agrupación de todas las componentes del contexto (predecesoras y sucesivas) a través de la suma de sus embeddings, $\mathbf{c} = \sum_{i=1}^{k} \mathbf{c_i}$, y, por tanto, al definir la puntuación como $s(w,c) = \mathbf{w} \cdot \mathbf{c}$ se pierde la información relativa al orden de los elementos (de ahí su nombre haciendo referencia a BOW). Según Eq. 10.2 resultando en:

$$P(D=1|w,c_{1:k}) = \frac{1}{1+e^{-(\mathbf{w}\cdot\mathbf{c_1}+\cdots+\mathbf{w}\cdot\mathbf{c_k})}}$$

Skip-Gram: intenta maximizar la clasificación de una palabra en base a otra de la misma oración. Se utiliza cada palabra como input para el clasificador log-lineal con capa de proyección continua y se predicen las palabras, dado cierto rango, que precederían y continuarían a la actual.

Suponiendo un contexto de k palabras, $c_{1:k}$, esta variante asume que sus elementos, c_i $(1 \leq i \leq k)$, son independientes entre sí y los trata como k contextos diferentes, i. e., representando el par $(w,c_{1:k})$ en D como k pares, (w,c_i) $(1 \leq i \leq k)$, y llevando a cabo el cálculo de la puntuación, $s(w,c)$, teniendo en cuenta dicha independencia:

$$\begin{aligned} P(D=1|w,c_i) &= \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{c_i}}} \\ P(D=1|w,c_{1:k}) &= \prod_{i=1}^{k} P(D=1|w,c_i) = \prod_{i=1}^{k} \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{c_i}}} \\ log\ P(D=1|w,c_{1:k}) &= \sum_{i=1}^{k} log \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{c_i}}} \end{aligned}$$

Los autores señalan que el aumento de dicho rango o tamaño de ventana, k, mejora la calidad de los resultados de los embeddings, pero conduce a un incremento de la complejidad computacional. Análogamente, como consecuencia de que, a mayor lejanía, menor relación, también indican que la atribución de pesos para

las palabras del contexto se realiza según su ubicación respecto a la actual, de forma que palabras más cercanas recibirán mayores contribuciones.

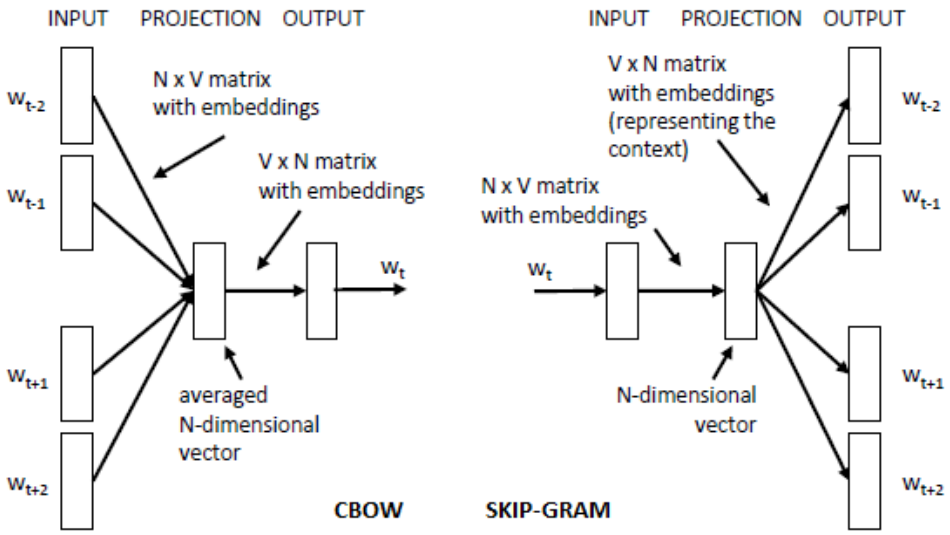


Figura 10.9: Representación CBOW y Skip-Gram. Fuente: pág. 293 [4]

10.2.2 Glove

Al igual que los métodos de factorización de matriz, como LSA (Sec. 6), los métodos *Shallow Window-Based* (NN no excesivamente profundas y basadas en ventanas de contexto), como *Word2Vec*, presentan la desventaja de no operar sobre los aspectos estadísticos relativos a las palabras del corpus. Como consecuencia, con el objetivo de mejorar los resultados a través de la incorporación de dicha información, se presentó la técnica *GloVe* [110].

A diferencia del modelo anterior, *Word2Vec*, con el que el aprendizaje de los embeddings se realiza de forma directa, este no radica únicamente en el contexto local, sino que también recurre al análisis global (de ahí su nombre, *Glo*bal *Vec*tors) sobre las co-ocurrencias de palabras. Concretamente, las probabilidades de co-ocurrencia son predichas como la relación semántica entre dos palabras examinada según el estudio de otra distinta, denomindada *probe word*.

Los autores argumentan que, en lugar de razonar sobre las propias probabilidades, calculadas a través de la matriz de co-ocurrencia término-término, X, debería considerarse el cociente de las mismas. Sean w_i y w_j las palabras de análsis y $\tilde{w}_k$ la *probe word*,

inicialmente el problema resulta en :

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \quad \text{con } P_{ik} = P(k|i) = \frac{X_{ik}}{X_i} \tag{10.3}$$

donde P_{ik} corresponde a la probabilidad de co-ocurrencia de k en el contexto de i, de forma que X_{ik} recuenta el número de veces que k co-ocurre en el contexto de i y $X_i = \sum_k X_{ik}$ representa la frecuencia de aparición de cualquier palabra en el contexto de i.

Tras ciertas consideraciones, el objetivo se transforma en:

$$F((w_i - w_j)^t \cdot \tilde{w}_k) = \frac{F(w_i^t \cdot \tilde{w}_k)}{F(w_j^t \cdot \tilde{w}_k)} \xrightarrow{(10.3)} F(w_i^t \cdot \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i}$$

y, por tanto, tomando como F la función exponencial se tiene el resultado:

$$w_i^t \cdot \tilde{w}_k = log(P_{ik}) = log(X_{ik}) - log(X_i)$$

permitiendo finalmente reducir Eq. 10.3 drásticamente a:

$$w_i^t \cdot \tilde{w}_k + \tilde{b}_i + \tilde{b}_k = log(X_{ik})$$

donde $\tilde{b}_i$ y $\tilde{b}_k$ son términos de sesgo para w_i y $\tilde{w}_k$, respectivamente.

En definitiva, la optimización resulta en la construcción de los vectores w_i y $\tilde{w}_k$ de forma que su producto proporcione un buen estimador para la transformación de sus co-ocurrencias. No obstante, para resolver el problema de optimización, los autores reformulan la ecuación considerando mínimos cuadrados e introduciendo cierta función de ponderación, f (verificando ciertas propiedades [110]), de forma que co-ocurrencias no usuales no aporten ruido ni solapen la información significativa asociada a las más frecuentes:

$$J = \sum_{i,k=1}^{V} f(X_{ik}) w_i^t \cdot \tilde{w}_k + \tilde{b}_i + \tilde{b}_k - log(X_{ik})$$

donde V hace referencia al tamaño del vocabulario.

Parte III

Casos Prácticos

11

Información Inicial

11.1 Conjunto de datos

Para llevar a cabo el caso práctico correspondiente a algunas de las aplicaciones desarrolladas a lo largo de los capítulos previos he elegido un conjunto de datos relativo a temáticas de libros [1] en lengua inglesa.

Concretamente, este recoge más de 14000 libros, disponiéndose de 6 variables para cada uno de ellos: ID, Título, Autor, Fecha de Publicación, Géneros y Resumen.

11.2 Objetivos

El propósito principal de la aplicación práctica, es la clasificación de libros (documentos) según sus géneros o temáticas (categorías) a través del estudio o aprendizaje de su reseña. Por tanto, según las distinciones que se establecieron para las posibles alternativas de clasificación (Sec. 5.1), esta se correspondería con una clasificación múltiple, basada en contenido, simple, es decir, no solapada, e independiente.

No obstante, dada la amplia extensión para algunos de los resúmenes originales de dichos libros, también he considerado, previamente a la construcción de los modelos, una reducción de la descripción asociada a cada uno de los libro seleccionados según técnicas relativas al resumen de documentos, para así evitar un excesivo coste computacional a la hora de la ejecución.

De igual forma, dado que la información elegida se basa en datos etiquetados, también consideré interesante realizar una identificación de temáticas según técnicas de topic modeling. De esta forma, sería posible comprobar si efectivamente los topics generados se corresponden con los significados y conceptos generales de los establecidos y, por tanto, si realmente estos son capaces de capturar e identificar las relaciones existentes entre los mismos, como, por ejemplo, la posible presencia o no de rendundancia, lo que ayudaría a disminuir el número considerado, de topics o, incluso, de géneros.

Por último, indicar que tanto la implementación práctica como todas las visualizaciones de resultados se han realizado utilizando la plataforma estadística *R* [111].

[1] Disponible en: http://www.cs.cmu.edu/~dbamman/booksummaries.html

11.3 Tratamiento

Como se detalló en la sección relativa al preprocesamiento (Sec. 3), son necesarias diversas tareas de tratamiento previas a la construcción y evaluación de los modelos. En este caso en particular ha sido indispensable el análisis de los géneros y resúmenes.

11.3.1 Géneros

Inicialmente, este campo venía codificado como una única cadena de caracteres en formato JSON, por lo que, para analizar las temáticas asociadas a cada uno de los libros se requería su desanidamiento en múltiples variables, una para cada género disponible (función para *R*: *fromJSON* de la librería *jsonlite*).

Tras ello, se identificaron un total de 224 géneros distintos y un número máximo de 11 temáticas para un único documento. Como consecuencia, con el objetivo de unificar los correspondientes a conceptos similares, así como los que diferían en ciertos caracteres, me centré en agrupar un número suficiente para el estudio, de manera que se representasen categorías más frecuentes y generales. En total, para llevar a cabo el análisis se han considerado los siguientes: *Fiction*, *History*, *Romance*, *Comedy*, *Fantasy*, *Adventure*, *Biographical*, *Mystery*, *Horror* y *Children's literature*.

No obstante, para conseguir libros representativos de cada una de dichas categorías, dado que gran parte de los mismos presentaba múltiples correspondencias hacia varias de ellas, fui escalando en orden creciente al número total de etiquetas (para evitar así los solapamientos) hasta conseguir un número mínimo de 200 libros para cada temática.

11.3.2 Resumen

Otra cuestión de gran importancia se corresponde con el tratamiento aplicado a las descripciones asociadas a los argumentos de cada uno de los libros. Tras la observación y análisis para una variedad de las mismas, se tuvieron en cuenta las siguientes consideraciones:

Etiquetas HTML: en ciertos casos se presentaban secuencias de caracteres correspondientes a código o etiquetas propias del lenguaje HTML, como, por ejemplo, " " o "&mdash", por lo que fue necesario su conversión a los caracteres correspondientes, así como su eliminación (función para *R*: *replace_html* del paquete *textclean*).

Secuencias: además de las anteriores, también se encontraron otras generales (ya no relativas al lenguaje anterior) a reemplazar, como fueron: """ y "'", correspondiéndose con comillas dobles y simples, respectivamente (función para *R*: *str_replace* del paquete *stringr*).

Caracteres especiales: puesto que en definitiva el objetivo se centra en el estudio de los caracteres alfabéticos, fue necesario la eliminación de ciertos caracteres especiales, como [,], \, . . . , #, y "(función para *R*: *str_remove* del paquete *stringr*).

Comillas simples: dado que en el idioma inglés existe el uso del *genitivo sajón* a través del carácter de comilla simple, el tratamiento de este fue más específico, puesto que, a priori, no tendrían por qué eliminarse los relativos a dicha referencia. Por tanto, para poder eliminar los adecuados, recurrí a una expresión regular que evitase dicho comportamiento dando preferencia a mantener el carácter en caso de posible coincidencia con el plural (i. e., -s').

11.4 Selección de la muestra

Puesto que la aplicación principal del estudio se centra en un problema de clasificación, con el objetivo de evitar posibles dificultades, a la hora de escoger las observaciones realicé una selección balanceada para las distintas categorías. En concreto, tal y como indiqué anteriormente, se eligieron 200 libros asociados a cada género y, además, de forma no solapada, i. e., procurando que cada uno de los mismos no estuviese referido a más de una temática.

Por tanto, en total se disponen de 2000 observaciones (10 categorías), de manera que, para la correspondiente partición de conjuntos, *entrenamiento* y *test*, se determinaron unos tamaños de 1500 y 500 observaciones, respectivamente, es decir, 150 y 50 observaciones para cada género en cada uno de los mismos.

11.5 Libro de referencia

Con el objetivo de ilustrar las diferentes consideraciones y procedimientos asociados a cada una de las tareas a realizar sobre el conjunto de muestra completo, he elegido una observación, es decir, un libro de referencia, de manera que el análisis de sus resultados facilite la posterior toma de decisiones para el conjunto global, y, por tanto, sea posible ir obteniendo las primeras conclusiones.

Este se corresponde con el popular libro de *Lucy Maud Montgomery* titulado *Anne*

of Green Gables, publicado por primera vez en 1908 y clasificado para todas las edades dentro de la categoría infantil, i. e., *Children's literature*.

Inspirada en un artículo periodístico real, en esta obra la autora narra la vida de Anne Shirley, una niña huérfana que, gracias a su carácter imaginativo y despierto, logra encandilar a todos los habitantes de Avonlea, el pequeño pueblo ficticio en la Isla del Príncipe Eduardo donde se desarrolla la historia a finales del siglo XIX.

Como características destacables, señalar que, desde su publicación, esta obra tuvo gran acogida llegándose a desarrollar múltiples versiones o secuelas, así como a adaptar a diversos formatos televisivos, como el original en 1985 o la serie más reciente producida por Netflix, de título *Anne with an "E"*.

Resumen de documentos

Como se indicó en los objetivos, dada la extensión de los resúmenes originales (Fig. 12.1), antes de comenzar con la exploración y análisis de la información, es necesario llevar a cabo la reducción de las descripciones asociadas a cada uno de los seleccionados en la muestra.

Para realizarlos me he centrado en un enfoque de extracción de contenido y, con el objetivo de escoger la técnica más adecuada para el conjunto global, he comparado los resultados obtenidos para el libro de referencia según 3 de las técnicas de resumen descritas en el capítulo correspondiente (Sec. 5.4): una de ellas desde el punto de vista semántico, *LSA*, y las dos restantes basadas en grafos, *TextRank* y *LexRank*.

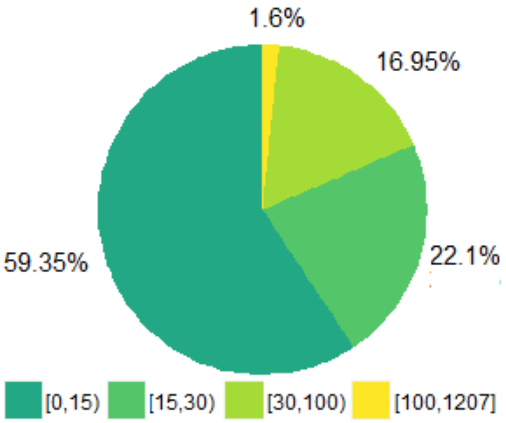


Figura 12.1: Oraciones por resumen

12.1 Comparaciones de referencia

Dado que el propósito de esta reducción radica en la extracción del contenido más relevante, esta se ha llevado a cabo tomando como unidades (de extracción) las oraciones que conformaban la reseña disponible.

Respecto a las características originales para el resumen asociado al libro escogido, según una división a través de los signos de puntuación usuales para el fin de oración (!, ?, .) y ciertas consideraciones, como continuar la frase tras abreviaturas o signos de pausa (;, : o ,), se obtuvieron un total de: 25 oraciones, 591 palabras y 306 palabras no *stopwords*, donde las de mayor frecuencia se muestran en la Fig. 12.2.

Word	Freq
Anne	17
Marilla	8
Gilbert	5
Avonlea	4
book	4
Gables	4
Green	4
Matthew	4

Figura 12.2: TopWords (Lib.Ref.)

En cuanto a los aspectos correspondientes a las distintas técnicas utilizadas, pueden destacarse los siguientes:

LSA: prefijada la extensión definitiva, a priori, $k = 8$, se obtuvo el resumen correspondiente a esta técnica de factorización de la matriz palabra-oración recurriendo, para la selección de las más relevantes, al método indicado en la sección correspondiente (Sec. 6.1.1), i. e., Gong y Liu (función para *R*: *genericSummary* del paquete *LSAfun*).

No obstante, los propios inconvenientes de dicho método, como el hecho de atribuir a cada topic el mismo nivel de importancia o seleccionar una única oración para cada uno de ellos (cuando, en realidad, los más relevantes podrían contribuir con un mayor número), hacen que no proporcione un muy buen resultado. Además, este método tampoco prevé la repetición a la hora de la selección, puesto que no se eliminan las oraciones seleccionadas para los primeros topics a medida que se avanza en el proceso hasta llegar al número preestablecido (k), lo que, en este caso en concreto, ha provocado que, de las 8 oraciones obtenidas, haya 2 coincidentes, i. e., tan solo 6 corresponden a oraciones distintas.

LexRank: tal y como se indicó en la sección de descripción (Sec. 7.2), esta técnica puede enfocarse desde dos perspectivas distintas, *LexRank Normal* y *LexRank Continuo*, por lo que también se han comparado ambas alternativas (función para *R*: *bind_lexrank* del paquete *lexRankr*, distinguiendo el argumento *continuous* según corresponda).

En esta ocasión, puesto que el algoritmo se centra en la ordenación (ranking) de las oraciones según su importancia, en lugar de tener que proporcionar el número de oraciones a seleccionar como parámetro para la función, es posible una determinación de la extensión a posteriori. En concreto, dado que los valores de relevancia del algoritmo se encuentran normalizados hacia una suma total de 1, inicialmente, como criterio de selección del número de oraciones para el resumen he recurrido al mínimo entre: el necesario para alcanzar una relevancia acumulada superior a un umbral de 0.7, y un valor de 8 oraciones.

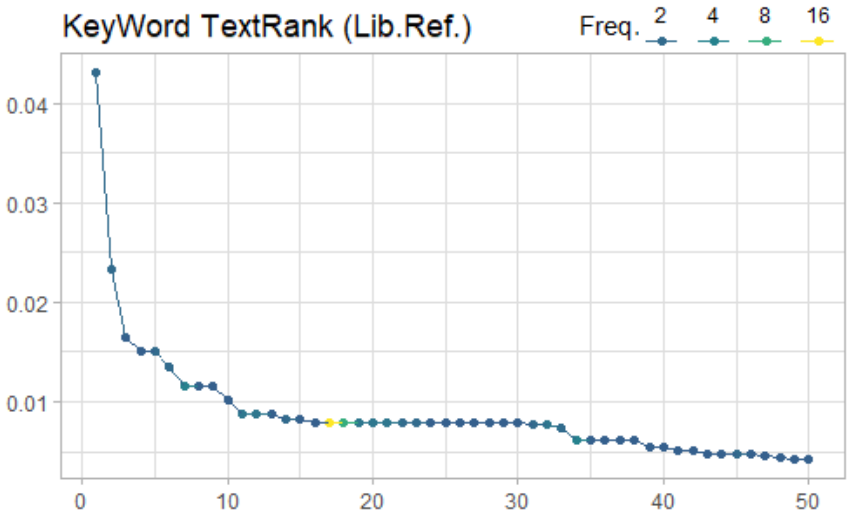
TextRank: al igual que para la técnica anterior, puesto que se basan en los mismos principios de ordenación, también he realizado la selección bajo el mismo criterio, es decir, según el nivel de relevancia acumulada. Además, en esta ocasión, a diferencia de las anteriores y como se propuso durante el desarrollo teórico de esta técnica (Sec. 7.1), he tenido en cuenta una previa etiquetación gramatical (*POS tagging*) según un modelo gramatical adicional para lengua inglesa (funciones

para *R*: *udpipe_load_model* y *udpipe_annotate* del paquete *udpipe*), para así poder identificar las categorías más influyentes para el estudio correspondiente:

(i) *KeyWords*: se consideran las palabras etiquetadas como *sustantivos*, *nombres propios*, *adjetivos* y *verbos* (función para *R*: *textrank_keywords* de la librería *textrank*).

(ii) *KeySentences*: se consideran las palabras correspondientes a *sustantivos*, *verbos* y *adjetivos* (función para *R*: *textrank_sentences* del paquete *textrank*).

Analizando en más detalle los resultados, en primer lugar, si se observa la evolución de la relevancia para la identificación de *KeyWords* (Fig. 12.3), se tiene que tras el filtrado por categorías gramaticales únicamente se proporcionan 50 de las 300 palabras obtenidas tras la eliminación de *stop-words*. De igual forma, se comprueba como dicha importancia decrece rápidamente tras las 10 primeras observaciones, así como que las más relevantes no se corresponden con las de mayor frecuencia (Fig. 12.2).



KeyWord	Freq.	TextRank
teaching	2	0.043
Nova	2	0.023
many-time	1	0.016
farming	1	0.015
School	2	0.015
run	2	0.014
Gilbert	4	0.011

Figura 12.3: Evolución de la importancia para KeyWords (Lib.Ref.)

Análogamente, pasando a analizar la evolución de la relevancia para las oraciones completas según las tres últimas técnicas consideradas (Fig. 12.4), se pueden observar distintos comportamientos. En cuanto a los asociados a *LexRank*, ambos detectan una pérdida significativa de relevancia en la oración correspondiente a la posición 15ª, distinguiendo así el conjunto de oraciones en dos grupos bastante diferenciados, sobre todo para la variante continua. Por su parte, para la alternativa normal destaca cómo existen dos subconjuntos de oraciones con mismo nivel de importancia, primero en el rango de las posiciones 7ª a 14ª, y, de nuevo, de las 18ª a 21ª, además, de que, como consecuencia de las características propias de dicho algoritmo, se produce una pérdida

de información sin obtener valor de relevancia para dos de las oraciones. Por último, respecto a los resultados obtenidos para *TextRank* cabe señalar que estos muestran la mayor diferencia de importancia con anterioridad, paso de la 5ª a la 6ª posición, así como que parece mantener una tendencia más continua hasta presentarse la igualdad para las cinco últimas observaciones.

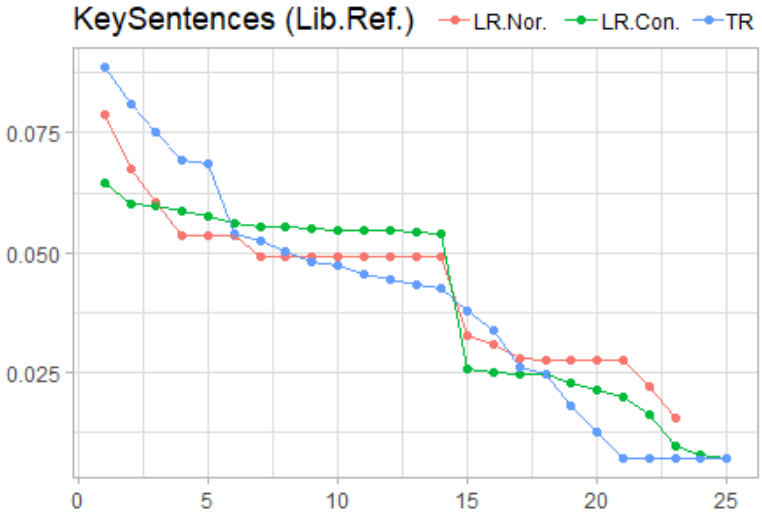


Figura 12.4: Evolución de la importancia para KeySentences (Lib.Ref.)

Finalmente, en cuanto a las oraciones extraídas para cada alternativa, se tiene que tan solo existe una oración común a los 4 resúmenes: "*Anne, a young orphan from fictional community of Bolingbroke, Nova Scotia, (based upon the real community of New London) is sent to Prince Edward Island after a childhood spent in strangers' homes and orphanages.*", destacando como esta no se corresponde con la de mayor relevancia para cada uno de ellos, sino que todos la sitúan a lo largo de la segunda mitad de las seleccionadas (Fig. 12.5).

No obstante, puesto que el resultado correspondiente a *LSA* exige una extensión preestablecida sin evitar las repeticiones y la variante normal de *LexRank* podría perder información, consideré que no se trataban de las mejores opciones para la implementación de los resúmenes y, por tanto, me centré en los obtenidos por las técnicas de *TextRank* y *LexRank Continuo*.

Técnica	Rank	Imp.
LSA	5	–
LexRank	8	0.049
LexRank Con.	6	0.056
TextRank	4	0.069

Figura 12.5: Or. Común (Lib.Ref.)

En particular, los resultados obtenidos para estas dos últimas técnicas sí mostraban una mayor similitud entre las oraciones seleccionadas, coincidiendo en 3 de las 8

escogidas, la comentada anteriormente (común a todos ellos) y las siguientes:

(i) "*Through a misunderstanding, the orphanage sends Anne Shirley.*"

(ii) "*Out of devotion to Marilla and Green Gables, Anne gives up the Avery Scholarship to stay at home and help Marilla, whose eyesight is diminishing.*"

12.2 Aplicación global

Dadas las similares características de ambas alternativas (*TexRank* y *LexRank Continuo*), así como de sus resultados para el libro de referencia, para decidir la técnica final a aplicar sobre el conjunto de muestra completo analicé su exigencia computacional a través del tiempo empleado hasta obtener el resultado.

Para el primero de ellos, *TextRank*, el modelo necesitó 2.69 s para obtener el resumen del libro analizado y un total de 31.96 s para una muestra de 5 libros. En cambio, los resultados correspondientes a la segunda alternativa, *LexRank Continuo*, tan solo requirieron 0.40 s y 1.44 s, respectivamente. Por consiguiente, pese a que con *TextRank* se tenía en cuenta un etiquetado gramatical, al no diferir significativamente los resultados y dadas las notables discrepancias computacionales, elegí esta última como técnica final.

Sin embargo, existían algunos libros de la muestra para los que el procedimiento de *LexRank* no era capaz de obtener el resultado como consecuencia del cálculo de las frecuencias correspondientes a este algoritmo, por lo que en dichos casos excepcionales sí realicé el resumen a través de la técnica de *TextRank*.

Respecto a las consideraciones destacables para su ejecución, cabe señalar que, al igual que existían resúmenes extensos (longitud máxima de 1207 oraciones), también se encontraban reseñas bastante breves, con escasamente un par de oraciones. Por ello, con el objetivo de obtener un resultado representativo para cada observación, decidí no alterar las descripciones que originalmente dispusiesen de una longitud inferior a 3 oraciones.

No obstante, para ilustrar la importancia de esta aplicación, es preciso añadir que, finalmente, tan solo 205 libros de los 2000 seleccionados para la muestra no disminuyeron la extensión de su resumen, destacando que, de entre los mismos, únicamente 35 no se correspondían a dicha imposición inicial, i. e., contaban originalmente con más de 3 oraciones. De igual forma, señalar que de entre aquellos resumidos (1795) más del 55 % (999) obtuvieron una longitud inferior a la máxima considerada, i. e., 8 oraciones.

Por último, también llama la atención como para los 123 que se encontraban al límite de dicha extensión mínima, i. e., contando exactamente con 3 oraciones iniciales, 88 redujeron dicho número a una única oración y, respectivamente, 1 observación redujo a longitud 2, lo que muestra la capacidad del algoritmo, en estas situaciones, de captar la gran representatividad de dichas oraciones para los libros asociados.

Exploración y Análisis

Puesto que las representaciones vectoriales asociadas a cada uno de los libros elegidos para el estudio se establecieron en base al contenido global tras la reducción de los resúmenes originales, una vez obtenidos fue posible el análisis y exploración inicial de la información resultante.

En primer lugar, analizando las palabras de mayor frecuencia a lo largo de toda la colección de documentos (Fig. 13.2) se observa como, salvo algunas excepciones de ciertos verbos y adverbios, la mayoría se corresponden con sustantivos característicos o coherentes a historias o relatos. No obstante, si se observan las palabras con mayor frecuencia según cada uno de los géneros (Fig. 13.3) se comprueba como pese a algunas coincidencias, i. e., términos más comunes o generales, en la mayoría de ellos es posible distinguir palabras representativas a sus significados.

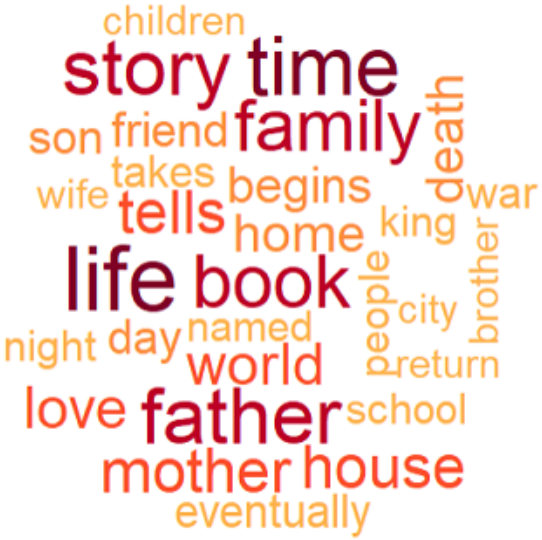


Figura 13.1: WordCloud (Corpus)

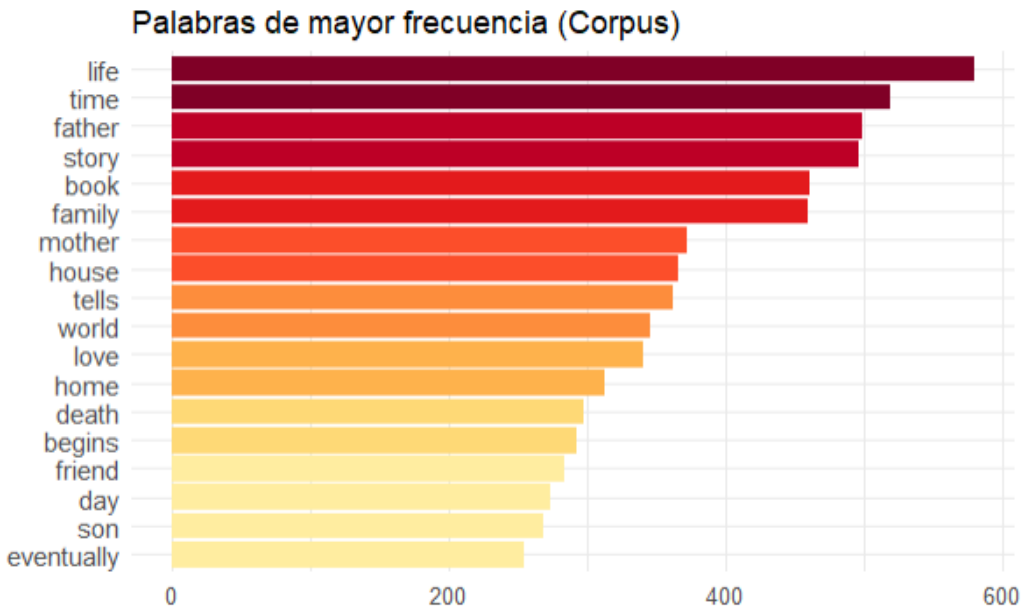


Figura 13.2: Palabras de mayor frecuencia global (Corpus)

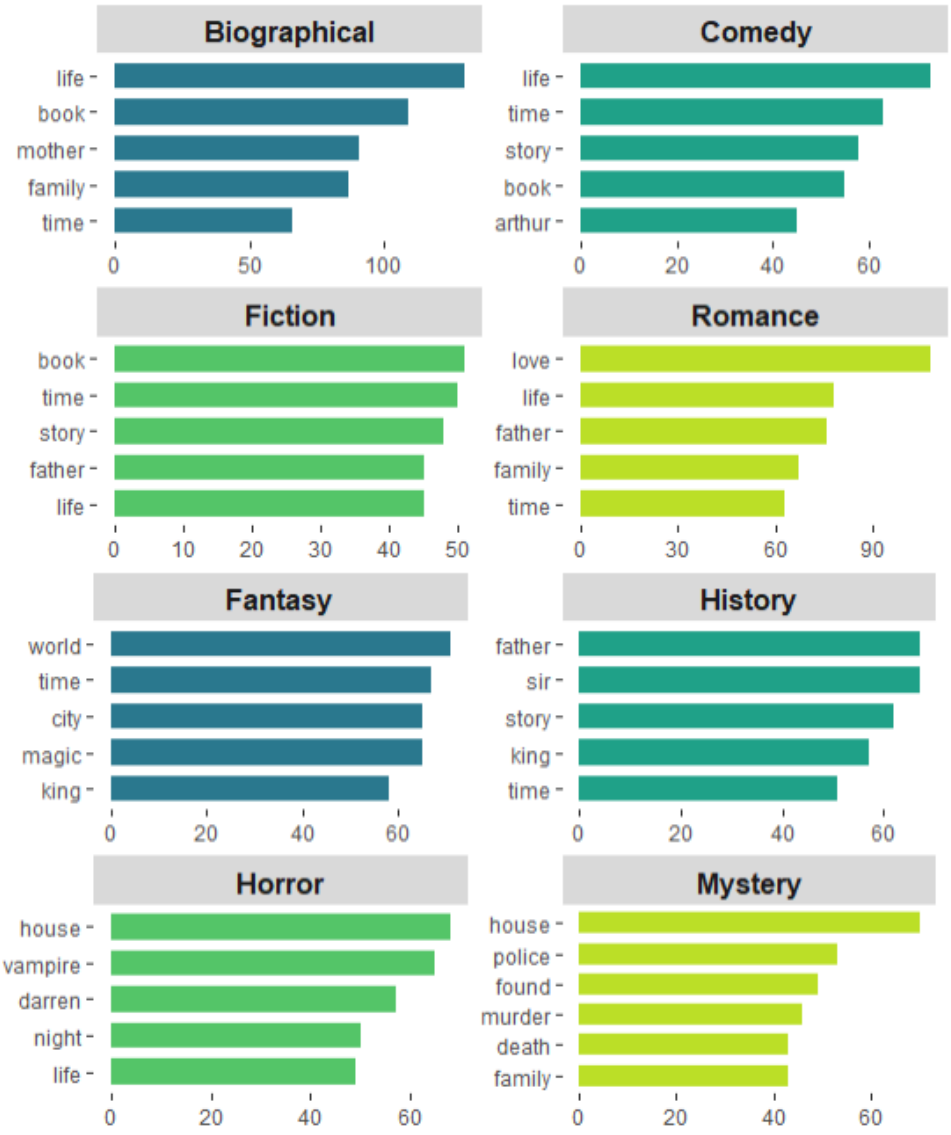


Figura 13.3: Palabras de mayor frecuencia para 8 de los géneros (Corpus)

Para las 4 temáticas superiores se obtienen los términos de mayor coincidencia común, observándose como estos se corresponden con las palabras de mayor frecuencia global (representación *WordCloud*, Fig. 13.1). Concretamente, se tiene que los términos *life* y *time* tienen alto número de ocurrencias en los 4 géneros, seguidos de *book* con presencia en 3 y, finalmente, *story*, *family* y *father* en 2.

Por su parte, para los 4 géneros inferiores es posible observar como, a diferencia de los anteriores, estos sí presentan términos más característicos y específicos para los conceptos usuales asociados a los mismos, como son las palabras *magic* y *world* para *Fantasy*, *king* y *sir* para *History*, *vampire* y *night* para *Horror*, y *murder*, *death* y *police* para *Mystery*.

REPRESENTACIONES

Tras todas las consideraciones previas de procesamiento y exploración, finalmente, para poder llevar a cabo la construcción de los modelos de clasificación, tan solo es necesario establecer la representación de las características para cada uno de los documentos.

Con el objetivo de ilustrar las distintas alternativas expuestas a lo largo del desarrollo teórico (Sec. 4), he considerado tres representaciones distintas: una para *BOW* y dos para *Words Embeddings*.

14.1 BOW (TF-IDF)

Partiendo del conjunto de resúmenes obtenido tokenizado a nivel de palabras, es decir, de la división del corpus hacia su lista de palabras, *BOW* (función para *R*: *unnest_tokens* del paquete *tidytext*), se lleva a cabo el cálculo de la medida *TF-IDF* para cada una de ellas (función para *R*: *bind_tf_idf* de la librería *tidytext*) dando lugar a la matriz documento-término. Para esta, cada fila hace referencia a una observación, cada columna a un token o palabra y cada componente al valor correspondiente de la medida TF-IDF para dicha correspondencia (doc, token), es decir, en definitiva, se corresponde con la matriz traspuesta de la denominada término-documento (función para *R*: *cast_dfm* de la librería *quanteda*).

Finalmente, como se esperaba, la matriz correspondiente a este enfoque presenta 2000 filas (documentos), así como una gran dimensionalidad y dispersión, con un total de 26668 columnas (características), y un 99.78 % de ausencia o huecos (*sparsity*).

14.2 WORD2VEC (CONSTRUIDOS)

Tal y como se indicó en la sección correspondiente (Sec. 10.2.1), para llevar a cabo esta técnica son posibles distintas alternativas, según: (i) la representación contextual escogida, *CBOW* y *Skip-Gram* y, (ii) el método o función de optimización, *Hierarchical Softmax* y *Negative Sampling*.

Coincidiendo con el desarrollo expuesto, para obtener las representaciones vectoriales densas asociadas al corpus he considerado la combinación de la última opción,

procedimiento *Negative Sampling*, junto al enfoque *Skip-Gram* (función para *R*: *word2vec* de la librería *word2vec*), con un tamaño de ventana de 10 palabras y una dimensión final de 100.

Respecto a los resultados, a diferencia de la representación anterior, donde se consideraban más de 26000 palabras, en esta ocasión, la propia función, al recibir como entrada el resumen completo (sin tokenizar), lleva a cabo su propio filtrado de palabras, eliminando las menos frecuentes y devolviendo aquellas con más de 22 ocurrencias a lo largo del corpus, en total 4134 tokens y manteniendo un número mínimo de 42 palabras para la representación de cada documento.

Por último, puesto que el objetivo se centra en obtener representaciones densas para cada documento, es necesario llevar a cabo, para cada uno de los mismos, la agregación de las asociadas a los tokens que los identifican. Para ello, tal y como se indicó en el desarrollo teórico, he tomado como representación global el valor medio de los correspondientes.

14.3 GloVe (Preentrenados)

Finalmente, como última alternativa he considerado la elección de representaciones densas ya construidas, i. e., *preentrenadas*, con el objetivo de que, gracias un entrenamiento más complejo, los *words embeddings* asociados a cada palabra recojan una mayor capacidad semántica que los contruidos bajo el corpus disponible para esta aplicación práctica y, por tanto, mejoren los resultados a la hora de capturar las relaciones entre las mismas.

Para ello, he recurrido a las representaciones obtenidas según la técnica *GloVe* (Sec. 10.2.2) para un total de 400000 tokens a partir de un extenso corpus extraído de Wikipedia y disponibles para múltiples dimensiones (50, 100, 200 y 300) [1]. En concreto, para poder disponer de los valores correspondientes he utilizado la función *embedding_glove6b* de la librería *textdata*, tomando como dimensión un valor de 100.

En cuanto a los resultados, cabe destacar que, pese al alto número de tokens disponibles, existían 4327 de los que no se recogía información, lo que redujo el número inicial a un total de 22341, permitiendo así aumentar, respecto a la técnica anterior, el número de tokens de representación para cada documento a un mínimo de 66. Por último, al igual que en el caso anterior, las representaciones asociadas a cada documento se han obtenido según el valor medio de las asocidas a los tokens correspondientes.

[1] Para más información https://nlp.stanford.edu/projects/glove/

Modelos de Clasificación

Una vez establecidas las representaciones para cada documento, según las distintas alternativas comentadas en la sección anterior, es posible establecer los modelos de clasificación.

Con el objetivo de obtener resultados comparables respecto a la adecuación de cada una de dichas técnicas (de representación), inicialmente, recurrí a una misma clase para el modelo de evaluación de resultados. Concretamente, un modelo *SVM de base radial* (función para *R*: *svm_rbf* de la librería *tidymodels*, que hace uso del paquete *kernlab*).

En cuanto al tratamiento previo para cada uno de ellos, cabe destacar que para la representación BOW se llevó a cabo, tras la etapa de *tokenization* y previamente a la asignación numérica según la medida *TF-IDF* y, por consiguiente, previamente a la construcción del modelo, una selección de características según la frecuencia de aparación en el corpus, en concreto, se exigió que esta fuese mayor a 3 ocurrencias (librería para *R*: *textrecipes*). Esto provocó, por ejemplo, que el espacio de dimensión asociado al conjunto de entrenamiento se redujese a tan solo 6104 variables (de las 26668 características totales indicadas anteriormente para el corpus completo, 22761 se correspondían a las presentes para el conjunto de entrenamiento).

Del mismo modo, es preciso señalar también el procedimiento llevado a cabo para la optimización de los hiperparámetros asociados a los modelos. Este se realizó según *validación cruzada*, considerando 10 pliegues para el conjunto de entrenamiento, junto a las combinaciones de los valores para los parámetros en los rangos: $C \in \{2^{-5},\ldots,2^{15}\}$ y $\sigma \in \{2^{-8},\ldots,2^{8}\}$, para el *coste* y el parámetro de la función núcleo, resprectivamente. Finalmente, una vez obtenidos los valores correspondienes a tal optimización (Fig. 15.6), fue posible llevar a cabo el entrenamiento de los modelos y, por último, su evaluación según el conjunto test a través de las distintas métricas consideradas.

Como se observa (Fig. 15.1), en general, el acierto medio obtenido por los modelos no es muy elevado, aunque sí se presentan notables diferencias entre los mismos. Como era de esperar, el asociado a la representación *BOW* muestra los peores resultados, mientras que los construidos a través de los *words embeddings* consiguen niveles más aceptables, teniendo en cuenta que se están considerando 10 categorías para la clasificación. Igualmente, cabe señalar que los resultados obtenidos para estas dos últimas

representaciones, *embeddings*, también difieren notablemente entre sí, concluyendo que los correspondientes a los preentrenados, *GloVe*, son capaces de captar de mejor forma las relaciones presentes en la información disponible, a diferencia de los establecidos a partir de la misma, *Word2Vec*. Por último, como consecuencia del amplio número de temáticas consideradas, pueden destacarse también los altos valores obtenidos para la especificidad, es decir, cómo los modelos son capaces de identificar correctamente las observaciones no asociadas al género respectivo ("negativas"), observándose además que esta medida es la que presenta menor variación entre los modelos, superando el valor de 0.9 en todos ellos.

Model	accuracy	spec	sens	precision	f_meas	roc_auc
BOW (TF-IDF)	0.216	0.913	0.216	0.391	0.231	0.594
Word2Vec	0.374	0.930	0.374	0.363	0.364	0.706
GloVe	0.424	0.936	0.424	0.415	0.416	0.733

Figura 15.1: Valores para las métricas consideradas I (Test).

No obstante, tras la observación de dichos resultados, me cuestioné si sería posible obtener alguna mejora en los mismos y crear así modelos más precisos y adecuados. Dado que ya se había realizado la correspondiente optimización de hiperparámetros y, además, con un amplio número de combinaciones para los valores de los mismos, consideré como posible opción, aunque supusiese mayor exigencia computacional, ampliar la información disponible con la que llevar a cabo la construcción de los modelos.

Para ello, modifiqué la extensión máxima establecida a la hora de obtener los resúmenes de cada documento, pasando de la considerada inicialmente (8 oraciones) a 20 y 60 oraciones, y llevé a cabo la construcción de los modelos, respectivamente a cada una de dichas extensiones, según la última alternativa de representación (*GloVe preentrenados*), puesto que, de entre las tres iniciales, fue la que presentó mejores resultados.

Del mismo modo, con el objetivo de disponer de alguna otra alternativa adicional, también consideré según esta última combinación (representación *GloVe* y longitud 60) un modelo centrado en la construcción de una red neuronal convolucional (librería para *R*: *keras*), considerando como función de pérdida *Categorical Cross-Entropy* (Fig. 10.3). Concretamente, la red consta, además de la capa inicial referida a los embeddings (función para *R*: *layer_embeddings*), de una capa convolucional de una dimensión junto a la función de activación *ReLU* (función para *R*: *layer_conv_1d*), una capa de re-

ducción global *max pooling* (función para *R*: *layer_global_max_pooling_1d*), así como dos capas *dropout* para evitar el sobreajuste (función para *R*: *layer_dropout*), otra capa de activación *ReLU* y, por último, la capa densa final con función de activación *softmax* para establecer la normalización de los resultados hacia las probabilidades de clasificación. En cuanto a los valores para los parámetros principales de estos modelos, análogamente a los casos anteriores, pueden observarse en la Fig. 15.6 para los modelos SVM y en la Fig. 15.7 en el caso del modelo CNN.

Model	accuracy	spec	sens	precision	f_meas	roc_auc
GloVe (Len. 20)	0.452	0.939	0.452	0.451	0.448	0.743
GloVe (Len. 60)	0.448	0.939	0.448	0.448	0.446	0.744
CNN	0.422	0.936	0.422	0.445	0.391	0.820

Figura 15.2: Valores para las métricas consideradas II (Test).

Respecto a los resultados (Fig. 15.2), cabe destacar que para los correspondientes a los modelos *SVM* se observa que el aumento de la extensión para los resúmenes consigue mejorar levemente los valores respecto a los obtenidos para el modelo de *GloVe* inicial. No obstante, una diferencia que, sin duda, sí se incrementa significativamente se encuentra en la correspondiente al acierto obtenido para el modelo base (*BOW TF-IDF*) y el último considerado (GloVe Len. 60), pasando de un valor de 0.21 a 0.45.

Por su parte, respecto a los resultados correspondientes al modelo *CNN*, señalar que, a pesar de presentar un acierto menor que los asociados a los últimos modelos *SVM*, este sí consigue alcanzar valores más adecuados para la métrica *AUC* (Área bajo la curva ROC). Esta refleja un notable incremento puesto que consigue pasar de los valores 0.59 y 0.74 obtenidos para el modelo base *BOW (TF-IDF)* y la combinación *GloVe (Len. 60)*, respectivamente, hasta superar el rango del 0.8. Por último, es preciso indicar también que, para esta última métrica, *AUC*, al encontrarnos bajo un problema multiclase, el cálculo global se ha llevado a cabo bajo la consideración usual propuesta por Hand y Till [112] según el promedio de las comparaciones dos a dos.

Análogamente, continuando con el análisis de resultados, también resulta de interés llevar a cabo el estudio individual para cada una de las temáticas consideradas. Para ello, observando la representación de la matriz de confusión obtenida para los resultados del conjunto test (Fig. 15.5) y el acierto alcanzado (Fig. 15.3) o las representaciones para las curvas ROC individuales (Fig. 15.4), correspondientes a cada uno

de los géneros a través del modelo *CNN*, rápidamente se comprueba la existencia de claras diferencias entre los mismos.

En primer lugar, a través del valor obtenido para dicho acierto (Fig. 15.3) es posible diferenciar las temáticas en dos subgrupos, según este sea superior o inferior al 40 %, donde destacarían las asociadas a *Fantasy*, *Children* y *Mystery*, superando el 70 %, y, por el contrario, las cuatro últimas, sin sobrepasar el 25 %.

Mystery	Children	Fantasy	Horror	Biographical
0.78	0.72	0.72	0.46	0.44

History	Adventure	Fiction	Romance	Comedy
0.36	0.24	0.22	0.2	0.08

Figura 15.3: Acierto por géneros. Modelo CNN (Test).

Del mismo modo, las representaciones para las curvas ROC (Fig. 15.4) también reflejan el comportamiento anterior, aunque evidenciando, en mayor medida, las diferencias entre los géneros *Fiction* y *Comedy*, respecto al resto.

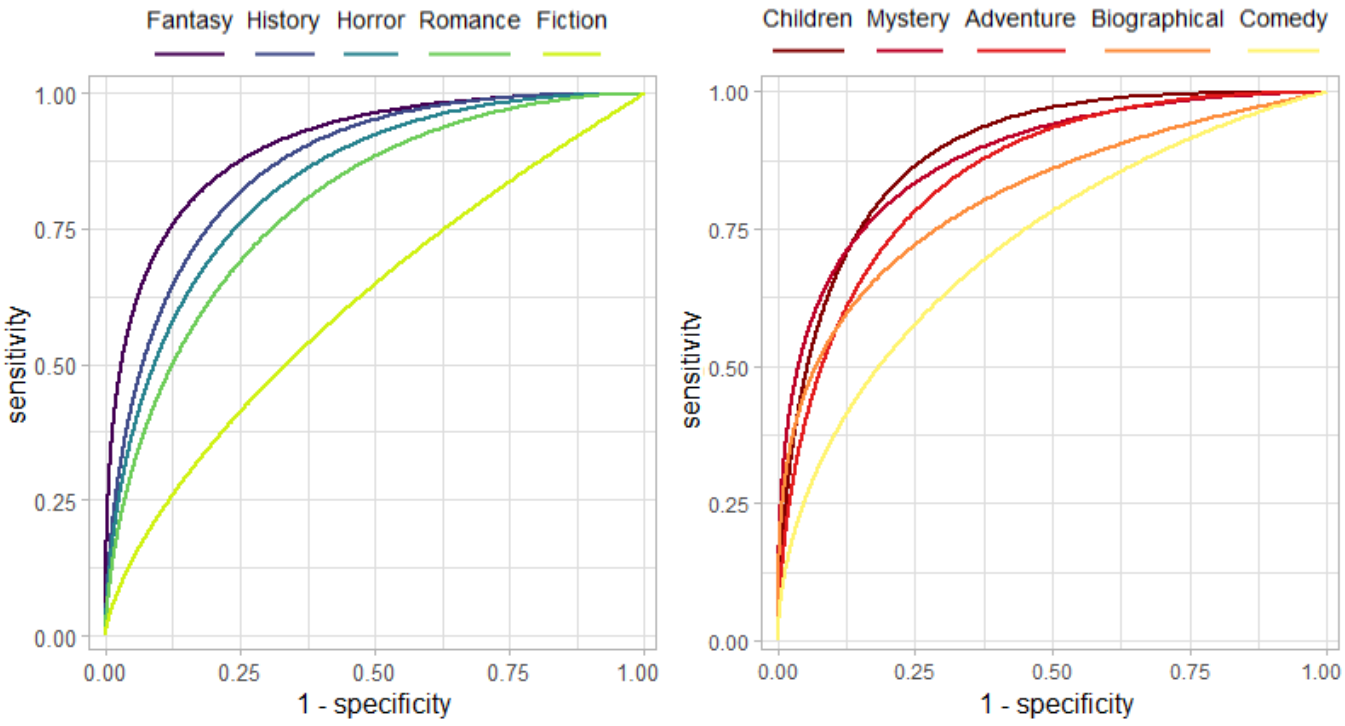


Figura 15.4: Curva ROC para cada género. Modelo CNN (Test).

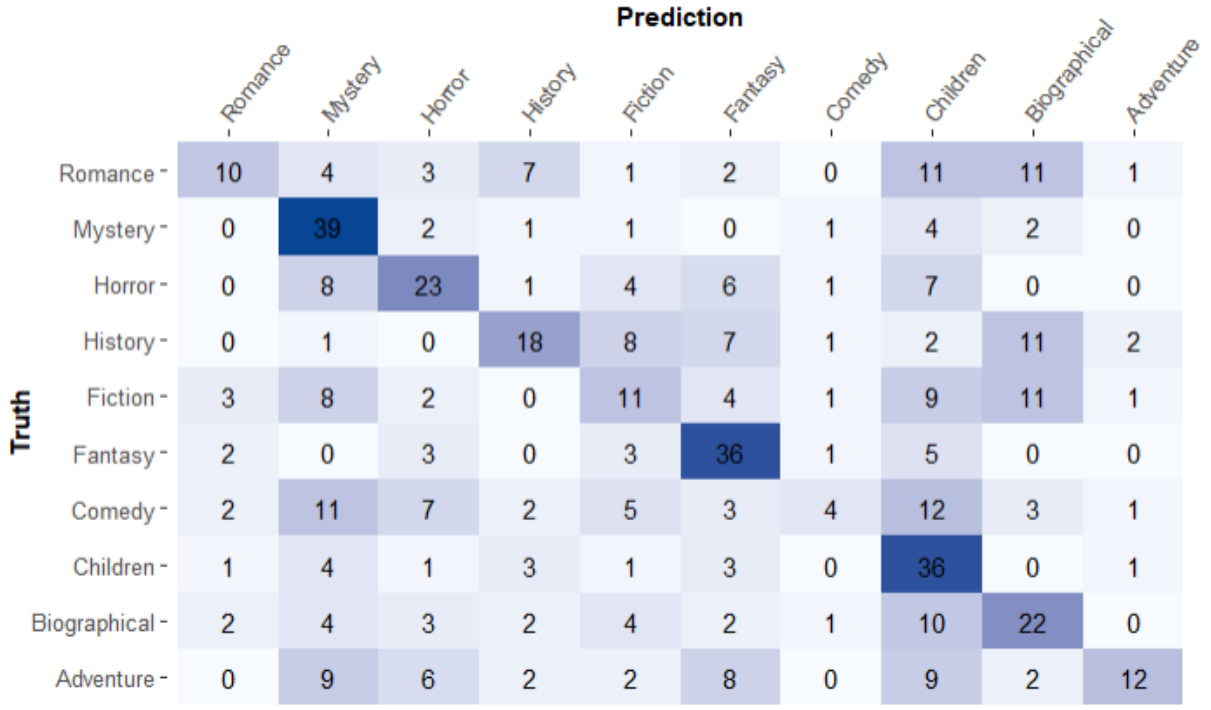


Figura 15.5: Matriz de confusión. Modelo CNN (Test)

De igual forma, dado que la selección inicial de la muestra se realizó de forma balanceada (concretamente con 50 observaciones por categoría para el conjunto test), pueden indicarse como adecuados los resultados obtenidos para los tres géneros mencionados anteriormente (*Fantasy*, *Children* y *Mystery*), mientras que para los correspondientes a los más imprecisos, la representación correspondiente a la matriz de confusión resultante (Fig. 15.5) permite observar los géneros sobre los que se produce un mayor número de errores.

A pesar de que esta muestra que para un gran número de las correspondencias entre asociaciones incorrectas, es decir, fallos, se obtienen valores bajos (0,1,2 o 3), sí es cierto que el resto toma valores más elevados. Respecto a estos, destaca que, en su mayoría, se corresponden con los géneros que a priori presentaban una mayor coincidencia entre términos comunes o generales (detectados en la exploración inicial de los datos y, en concreto, estos hacían referencia a *Fiction*, *Comedy*, *Romance*, *Biographical* y *Children*). Por tanto, dichos fallos podrían resultar esperables, de manera que, en parte, los resultados obtenidos permiten confirmar las intuiciones ya planteadas sobre las relaciones exisentes entre los géneros considerados.

Model	cost	rbf_sigma
BOW (TF-IDF)	1	0.0000851
Word2Vec	2	0.0078125
GloVe (Len. 8)	2	0.0078125
GloVe (Len. 20)	2	0.0156250
GloVe (Len. 60)	2	0.0156250

Figura 15.6: Valores para los parámetros. Modelos SVM.

Model	CNN
batch_size	300
epochs	25
layers	8
input_dim (Embeddings)	5001
input_length (Embeddings)	300
output_dim (Embeddings)	100
dropout	0.3
filters (Conv_1d)	250
kernel_size (Conv_1d)	3
dropout	0.4
units (Dense)	250
units (Dense Fin.)	10

Figura 15.7: Valores para los parámetros. Modelo CNN.

Identificación de temáticas

Tal y como se indicó en los objetivos del caso práctico, dado que nos encontramos ante datos de carácter etiquetado, también resulta conveniente realizar una identificación de temáticas, a priori, desconocidas, para comprobar si efectivamente es posible capturar las distintas relaciones existentes entre los documentos e identificarlas con los géneros preestablecidos para la clasificación.

Por tanto, para recurrir a esta aplicación de *topic modeling*, como se comentó en la sección correspondiente (Sec. 5.6), he recurrido a la técnica usual asociada al *Análisis Latente de Dirichlet (LDA)* (Sec. 6.3) disponible en la función *FitLdaModel* de la librería *textmineR*. En cuanto a su implementación, esta se lleva a cabo según el método de remuestreo de *Gibb sampling* [67] y para los parámetros asociados a las distribuciones de probabilidad, al establecer $k = 10$ topics, he tomado los valores $\eta = 0{,}02$ (aunque también es usual recurrir a un valor de 0.1) y $\alpha = 50/k = 5$ según la consideración de Griffiths y Steyvers [113].

Respecto a los resultados, observando inicialmente las 10 palabras más relevantes para cada uno de los topics (Fig. 16.1), es decir, las de mayor peso según la mixtura asociada a cada uno de los mismos, se pueden obtener algunas conclusiones. Comenzando con los que presentan una mayor claridad, en primer lugar, cabe destacar el segundo de ellos puesto que las palabras *house*, *found*, *police* y *killed* pueden indicar que este se corresponde con el género asociado a *Mystery*. Análogamente, para el siguente, 3º, los términos *king*, *war*, *army* y *attack* se correspoden adecuadamente con la temática *History*, y respecto al 9º las palabras *ship*, *island*, *sea* y *rescue* muestran como este refleja el género *Adventure*. De igual forma, para el último, 10º, los términos *world*, *power* y *magic*, podrían relacionarse con la temática *Fantasy*, así como *vampire* a la asociada a *Horror*. Sin embargo, algunos de ellos también presentan cierta ambigüedad; por ejemplo, para los topics 6º y 7º la mayoría de las palabras se corresponden con las identificadas al inicio de la exploración del conjunto de datos como las más generales o comunes (Fig. 13.3), por lo que cabe esperar que los géneros correspondientes (*Romance*, *Fiction*, *Comedy*, *Biographical* y *Children's literature*) presenten alta correspondencia hacia los mismos. Finalmente, en cuanto al resto de topics, los términos presentados tampoco caracterizan de forma única a cierto género, sino que parecen capturar las relaciones entre algunos de ellos, lo que impide que la identificación sea tan evidente.

Top_1	Top_2	Top_3	Top_4	Top_5	Top_6	Top_7	Top_8	Top_9	Top_10
night	house	king	tells	tom	father	story	time	ship	world
tells	john	war	de	city	family	book	death	island	lord
time	found	army	george	return	mother	life	city	captain	power
henry	sir	arthur	paul	women	love	events	jack	crew	magic
life	police	american	narrator	leaves	life	character	attempt	sea	return
people	lady	people	friends	party	school	characters	day	richard	human
returns	killed	attack	day	named	children	york	god	simon	earth
woman	friend	united	job	leave	home	woman	past	rescue	vampire

Figura 16.1: Palabras de mayor influencia en cada topic

No obstante, puesto que sí se han podido establecer ciertas correspondencias entre algunos de ellos, con el objetivo de comprobar realmente dicha adecuación, consideré para cada libro el topic para el que presentaba una mayor probabilidad, i. e., un mayor peso de mixtura, de manera que fuese posible establecer una "matriz de confusión" entre los géneros originales y los topics generados (Fig. 16.2: con el objetivo de esclarecer la interpretación se muestran las 4 correspondencias de mayor frecuencia para cada género).

Efectivamente, los valores de mayor tonalidad destacan los géneros de *Mystery*, *Hystory*, *Adventure* y, *Fantasy* y *Horror* para los topics 2º, 3º, 9º y 10º, respectivamente, mientras que, tal y como se intuía, los topics 6º y 7º recogen las relaciones generales entre los de mayor similitud, *Romance*, *Children* y *Biographical*, para el primero, y *Biographical*, *Fiction*, *Comedy* e *History*, para el segundo.

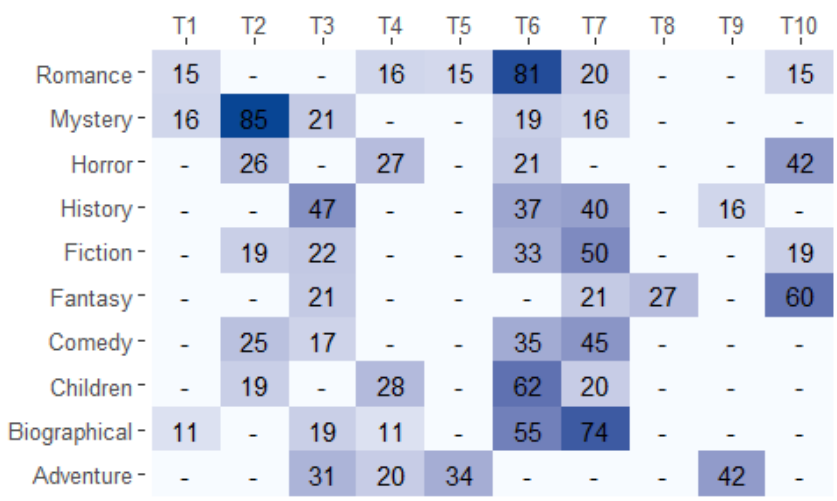


Figura 16.2: Correspondencia entre géneros y topics

Conclusiones

Tras llevar a cabo las diferentes etapas expuestas en los capítulos anteriores, puede concluirse que, a pesar de que las tasas de acierto y, en general, los valores para la mayoría de las métricas consideradas para la evalucación de los modelos, se encuentran lejos de tomar los valores máximos, el análisis realizado resulta aceptable.

En cuanto al objetivo principal, evidentemente, se ha comprobado que el hecho de ampliar la extensión máxima considerada para el resumen de cada documento mejora los resultados obtenidos para la clasificación, puesto que, en definitiva, se incrementa la información disponible para el aprendizaje. No obstante, cabe destacar que, en ciertas situaciones, tales mejoras podrían resultar insuficientes si se tienen en cuenta los requisitos computacionales exigidos. Del mimo modo, con el propósito de poder establecer una comparación lo más enfocada posible hacia los aspectos propios del *Text Mining* (como las distintas representaciones: *BOW (TF-IDF)*, *Word2Vec* y *GloVe*), únicamente se han considerado dos técnicas para los modelos de clasificación, *SVM* y *CNN*, lo que, claramente, supone una limitación respecto a la inmensa variedad existente y, por tanto, podrían encontrarse mejoras significativas a través de otras posibilidades que resultasen más acertadas.

Por su parte, respecto a las aplicaciones auxiliares para llevar a cabo la clasificación, las etapas de preprocesamiento: adecuación de géneros y "limpieza" de resúmenes, resultaron claves para poder evaluar la idoneidad de las distintas alternativas asociadas a la reducción de su extensión (*LSA, TextRank* y *LexRank*), siendo, por consiguiente, esta elección, a su vez, determinante para posibilitar la implementación de los modelos. De igual forma, el estudio detallado realizado para el libro escogido de referencia permitió, además de la selección del procedimiento más conveniente (*LexRank Continuo*), un análisis de mayor profundidad para cada una de dichas técnicas, siendo posible así establecer sus ventajas e inconvenientes (como la redundancia y repetición, pérdida de información, posibilidad de elección bajo umbral, etc.) y, por tanto, mejorar en su comprensión y funcionamiento.

Por último, en relación a la tarea asociada a la identificación de temáticas y captura de relaciones entre documentos (*LDA*), es preciso destacar especialmente que, al establecer el mismo número de topics para su detección, que de géneros preestablecidos, sí ha sido posible determinar las correspondencias entre algunos de los mismos (coin-

cidiendo, los más evidentes, con los de mejor evaluación para los modelos de clasificación), mientras que, coherentemente, en el resto de los casos dichos topics recogían las conexiones existentes entre varias de las temáticas originales, las cuales ya habían sido observadas e intuídas a través de la exploración inicial de la información.

Finalmente, concluir que estos resultados, los correspondientes a la identificación de temáticas junto a los obtenidos de forma individual para las métricas de los modelos, confirman que, efectivamente, no todos los géneros establecidos presentan el mismo grado de distinción y, por tanto, de ahí que sea adecuado plantearse, además de, tal y como se ha comentado anteriormente, las posibles alternativas hacia distintos modelos y técnicas de clasificación, otras consideraciones como establecer una agrupación adicional entre los mismos, permitir una clasificación solapada o, incluso, buscar un mayor grado de singularidad o especificidad en los propios documentos.

Parte IV

Anexos

A

Código R

A.1 Importación

```
library(tidyverse)
dat <- read_tsv("C:/Users/Propietario/Desktop/TFG/Casos_Practicos/Datos/
        booksummaries/booksummaries2.txt",col_names = c("ID", "ID_base",
        "Title", "Author", "Publication_Date","Genres","Summary"),
        col_types = cols(.default=col_character(),ID_base = col_skip()))
# Filtar las filas que no tienen resumen en blanco
dat %>% filter(!str_detect(Summary,"&#6(2|0)")) -> dat_sum_not_blank
```

A.2 Tratamiento

A.2.1 Géneros

```
# Función para obtener los géneros de cada observación
library(jsonlite)
fun_json <- function (x) {if (!is.na(x)) {
  x[[1]] %>% fromJSON %>% unlist %>% unname} else {NA}}
# Extracción de los géneros para cada observación
dat_sum_not_blank %>% mutate(Genres = map(Genres,fun_json)) -> dat_t1
# Expansión de cada género a una columna: Genres_X
dat_t1 %>% unnest_wider(col = Genres,names_sep = "_") %>%
  # filtrado de las que al menos tienen un género
  filter(!if_all(.cols=starts_with("Genres_"),.fns = is.na)) -> dat_t2
# Unificación de Géneros para la posterior clasificación
dat_t2 %>% select(ID, Title, starts_with("Genres_")) %>%
  pivot_longer(-(ID:Title), names_to = "GenresX",values_to = "Genres",
  values_drop_na = T) %>% mutate(Genres=parse_factor(Genres))-> dat_t3
dat_t3 %>%
 mutate(Genres = Genres %>% fct_collapse(
  "Fiction" = str_extract(levels(.), ".*((F|f)iction).*"),
```

```
  "History" = str_extract(levels(.), ".*((H|h)istorical|History).*"),
  "Romance" = str_extract(levels(.), ".*((R|r)omance|Roman).*"),
  "Comedy" = str_extract(levels(.), ".*((c|C)omedy|Humour).*"),
  "Fantasy" = str_extract(levels(.), ".*((F|f)fantasy).*"),
  "Adventure" = str_extract(levels(.), ".*((a|A)dventure).*"),
  "Biographical" = str_extract(levels(.),
                    ".*((b|B)iographical|(b|B)iogra).*"))) -> dat_t4
# Selección de las observaciones que presentan algún género de interés
dat_t4 %>% filter(Genres %>% str_detect("^Fiction$|History|Romance|
                  ^Fantasy$|Adventure|Comedy|Biographical|Mystery|
                  Children's literature|Horror")) -> dat_t5
dat_t5 %>% pivot_wider(names_from = "GenresX",
                       values_from = "Genres") -> dat_t6
# Comedia
dat_t6 %>% filter(if_any(.cols = starts_with("Genres_"),
                         .fns = ~ .x == "Comedy")) -> dat_comedy_ini
dat_comedy_ini %>% select(-starts_with("Genres_")) %>%
                     mutate(Genres = "Comedy") -> dat_comedy
# Biografía
setdiff(dat_t6 %>% filter(if_any(.cols = starts_with("Genres_"),
      .fns = ~ .x == "Biographical")), dat_comedy_ini) -> dat_bio_ini
dat_bio_ini %>% select(-starts_with("Genres_")) %>%
                    mutate(Genres = "Biographical") -> dat_bio
# Aventuras
setdiff(setdiff(dat_t6, dat_comedy_ini) %>%
            filter(if_any(.cols=starts_with("Genres_"),
              .fns = ~ .x=="Adventure")), dat_bio_ini) -> dat_adv_ini
dat_adv_ini %>% select(-starts_with("Genres_")) %>%
                     mutate(Genres="Adventure") -> dat_adventure
# Observaciones restantes
setdiff(dat_t6,dat_comedy_ini) %>%
  setdiff(dat_bio_ini) %>% setdiff(dat_adv_ini)-> dat_t7
# Selección del resto
## Observaciones con 1 único género definido
dat_t7 %>% rowwise() %>%
  mutate(Tot_NA = c_across(cols = starts_with("Genres_")) %>%
```

```
                                        is.na %>% sum) -> dat_t7_na
dat_t7_na %>% filter(Tot_NA == 9) %>% select(-Tot_NA) %>%
  pivot_longer(-(ID:Title), names_to = "GenresX", values_to = "Genres",
               values_drop_na = T) -> dat_t7_1gen
dat_t7_1gen %>% select(-GenresX) -> dat_est_1gen
## Observaciones con 2 géneros definidos, uno de ellos Fiction
dat_t7_na %>% filter(Tot_NA == 8,if_any(.cols = starts_with("Genres_"),
                    .fns = ~ .x == "Fiction")) %>% select(-Tot_NA) %>%
  pivot_longer(-(ID:Title), names_to = "GenresX", values_to = "Genres",
               values_drop_na = T) -> dat_t7_2gen_fic
dat_t7_2gen_fic %>% filter(str_detect(Genres,
    "Romance|Horror|Mystery")) %>% select(-GenresX) -> dat_est_2gen_fic
# Observaciones con 2 géneros definidos, uno de ellos Fantasy
dat_t7_na %>% filter(Tot_NA == 8,if_any(.cols = starts_with("Genres_"),
                    .fns = ~ .x == "Fantasy")) %>% select(-Tot_NA) %>%
  pivot_longer(-(ID:Title), names_to = "GenresX", values_to = "Genres",
               values_drop_na = T) -> dat_t7_2gen_fan
dat_t7_2gen_fan %>% filter(str_detect(Genres, "Romance|Horror")) %>%
                      select(-GenresX) -> dat_est_2gen_fan
## Observaciones con 2 géneros definidos, uno de ellos History
dat_t7_na %>% filter(Tot_NA == 8,if_any(.cols = starts_with("Genres_"),
                                        .fns = ~ .x == "History")) %>%
  select(-Tot_NA) %>%
  pivot_longer(-(ID:Title), names_to = "GenresX", values_to = "Genres",
               values_drop_na = T) -> dat_t7_2gen_his
dat_t7_2gen_his %>% filter(str_detect(Genres, "Romance|Horror"))%>%
                      select(-GenresX) -> dat_est_2gen_his
## Observaciones con 2 géneros, uno de ellos Children's literarure
dat_t7_na %>% filter(Tot_NA == 8, if_any(.cols = starts_with("Genres_"),
       .fns = ~ .x == "Children's literature")) %>% select(-Tot_NA) %>%
  pivot_longer(-(ID:Title), names_to = "GenresX", values_to = "Genres",
               values_drop_na = T) -> dat_t7_2gen_chil
dat_t7_2gen_chil %>% filter(str_detect(Genres, "Romance|Horror"))%>%
                      select(-GenresX) -> dat_est_2gen_chil
## Observaciones con 3 géneros definidos, dos de ellos Fiction y Fantasy
dat_t7_na %>% filter(Tot_NA == 7,if_any(.cols = starts_with("Genres_"),
```

```
   .fns = ~ .x == "Fantasy"), if_any(.cols = starts_with("Genres_"),
   .fns = ~ .x == "Fiction")) %>% select(-Tot_NA) %>%
  pivot_longer(-(ID:Title), names_to = "GenresX", values_to = "Genres",
               values_drop_na = T) -> dat_t7_3gen_fic_fan
dat_t7_3gen_fic_fan %>% filter(str_detect(Genres, "Romance|Horror"))%>%
                        select(-GenresX) -> dat_est_3gen_fic_fan
## Observaciones con 3 géneros definidos, dos de ellos Romance y Fantasy
dat_t7_na %>% filter(Tot_NA == 7,if_any(.cols = starts_with("Genres_"),
   .fns = ~ .x == "Romance"), if_any(.cols = starts_with("Genres_"),
   .fns = ~ .x == "Fantasy")) %>% select(-Tot_NA) %>%
  pivot_longer(-(ID:Title), names_to = "GenresX", values_to = "Genres",
               values_drop_na = T) -> dat_t7_3gen_rom_fan
dat_t7_3gen_rom_fan %>% filter(str_detect(Genres,
 "Horror|Mystery|Romance")) %>% select(-GenresX) -> dat_est_3gen_rom_fan
## Observaciones con 3 géneros definidos, dos de ellos Mystery y Fantasy
dat_t7_na %>% filter(Tot_NA == 7,if_any(.cols = starts_with("Genres_"),
   .fns = ~ .x == "Mystery"), if_any(.cols = starts_with("Genres_"),
   .fns = ~ .x == "Fantasy")) %>% select(-Tot_NA) %>%
  pivot_longer(-(ID:Title), names_to = "GenresX", values_to = "Genres",
               values_drop_na = T) -> dat_t7_3gen_mis_fan
dat_t7_3gen_mis_fan %>% filter(str_detect(Genres, "Romance|Horror"))%>%
                        select(-GenresX) -> dat_est_3gen_mis_fan
## Observaciones con 4 géneros, tres de ellos Fiction, Fantasy y Mystery
dat_t7_na %>% filter(Tot_NA == 6,if_any(.cols = starts_with("Genres_"),
  .fns = ~ .x == "Fiction"), if_any(.cols = starts_with("Genres_"),
  .fns = ~ .x == "Fantasy"), if_any(.cols = starts_with("Genres_"),
  .fns = ~ .x == "Mistery")) %>% select(-Tot_NA) %>%
  pivot_longer(-(ID:Title), names_to = "GenresX", values_to = "Genres",
               values_drop_na = T) -> dat_t7_4gen_fic_fan_mis
dat_t7_4gen_fic_fan_mis %>%
  filter(str_detect(Genres, "Romance|Horror|Adventure")) %>%
  select(-GenresX) -> dat_est_4gen_fic_fan_mis
# Unión de todos los conjuntos
union(dat_comedy,dat_bio) %>% union(dat_adventure) %>%
  union(dat_est_1gen) %>% union(dat_est_2gen_fan) %>%
  union(dat_est_2gen_fic) %>% union(dat_est_2gen_his) %>%
```

```
  union(dat_est_2gen_chil) %>% union(dat_est_3gen_fic_fan) %>%
  union(dat_est_3gen_rom_fan) %>% union(dat_est_3gen_mis_fan) %>%
  union(dat_est_4gen_fic_fan_mis) -> dat_est_ini
# ID repetidos (19)
id_rep <- dat_est_ini%>%count(ID)%>%filter(n >1)%>%select(ID)%>%.[[1]]
dat_est <- dat_est_ini %>% filter(!(ID %in% c(id_rep,"9388867")))
# Recuento (+ de 200 para cada género)
dat_est %>% count(Genres,sort=T)
```

A.2.2 Resúmenes

```
# Unión a la selección para el estudio
dat_est %>%
 left_join(dat %>% select(ID,Author,Summary), by = "ID")-> dat_est_sum
# Reemplazo y eliminación de caracteres especiales
library(textclean)
dat_est_sum %>% # eliminación de todas las ocurrencias \"
 mutate(Summary = Summary %>% replace_html() %>%
  str_replace_all("&#34;",'\\"') %>% str_remove_all("\\*") %>%
  str_replace_all("&#39;","'") %>% str_remove_all("#") %>%
  str_remove_all("\"") %>% str_remove_all("\\[") %>%
  str_remove_all(pattern = "\\]") %>%
  str_remove_all("\\.\\.\\.")) -> dat_est_sum_t2
# Eliminación de comillas simples que no son genitivo sajón
dat_est_sum_t2 %>% mutate(Summary = map_chr(.x = Summary,
         .f = ~gsub(pattern = "([^s])(')([^s|\\s])",
                    replacement = "\\1\\3", .x))) -> dat_est_sum_t3
```

A.3 Selección de la Muestra

```
# Fijación de la semilla
set.seed(12345)
# Definición de la función para la selección de cada muestra
Mues_Gen <- function(tb,n_train=100, n_test=50) {
  index <- 1:nrow(tb)
```

```
  index_train <- sample(index, size = n_train, replace = F)
  index2 <- setdiff(index, index_train)
  index_test <- sample(index2, size = n_test, replace = F)
  return(list(Train = tb[index_train,], Test = tb[index_test,]))
}
# Selección de la muestra para cada género, para que sea balanceado
dat_est_sum_t3 %>% nest(-Genres) %>%
  mutate(Sample = data %>% map(~ Mues_Gen(.))) %>%
  unnest_wider(Sample) -> dat_samples
# Extracciones de cada conjunto: Train, Test
dat_samples %>% select(-data) %>%
  pivot_longer(-Genres, names_to="Set", values_to = "data_cols") %>%
  unnest(data_cols) -> dat_samples_sets
# Incorporación de la columna Longitud (Número de unidades)
Count_split_sent <- function(Summary) {
  return(Summary %>%  as_tibble %>%
  unnest_tokens(token = "regex", input = value, output = Sentences,
     pattern = "(?<!([[:punct:]]|\\s|^)[[:alpha:]]{1,3})\\.
      (?!\\s[a-z])|\\!|\\?|:", to_lower = F) %>% nrow)
  }
dat_samples_sets %>% rowwise() %>%
 mutate(Summary_Len=Count_split_sent(Summary))->dat_samples_sets_SumLen
# Extracción de la muestra de entrenamiento
dat_samples_sets_SumLen %>% filter(Set == "Train") -> dat_train
# Extracción de la muesta de test
dat_samples_sets_SumLen %>% filter(Set == "Test")-> dat_test
```

A.4 Libro de referencia

```
dat_est_sum_t3 %>% filter(ID == "58406") -> dat_agg_est
# Número de oraciones del resumen original
dat_agg %>% select(Summary) %>%
 tidytext::unnest_tokens(output = sentences, input = Summary,
   token="regex",to_lower = F, pattern = "(?<!([[:punct:]]|\\s)
                [[:alpha:]]{1,3})\\.(?!\\s[a-z])|\\!|\\?|:") %>% nrow
# Número de palabras
```

```
dat_agg %>% select(Summary) %>% tidytext::unnest_tokens(output=word,
                      input = Summary, token="words") -> dat_agg_sum_w
dat_agg_sum_w %>% nrow
# Número de palabras No Stop-words
dat_agg_sum_w %>% anti_join(stop_words) -> dat_agg_nsw
dat_agg_nsw %>% nrow
# Palabras de mayor frecuencia
dat_agg_nsw %>% mutate(word = str_remove(word, "'.*")) %>%
 count(word, sort=T) %>% head(8) %>%
  rename(Word = word, Freq = n) -> dat_agg_nsw_freq
```

A.5 Resumen del libro de Referencia

A.5.1 LSA

```
library(LSAfun) # Resumen por extracción de oraciones
sum_agg_lsa <- genericSummary(text = dat_agg_est$Summary[1],
                      k = 8, split = c(".","!","?"))
sum_agg_lsa %>%  str_trim # Resultado por importancia
library(tidytext) # Buscamos la ordenación según el resumen original
dat_agg_est %>% select(Summary) %>%
  unnest_tokens(input = Summary, output = sentences, to_lower = F,
    token = "regex", pattern = "\\.|!|\\?") -> dat_agg_sum_split_lsa
# Intersección entre partición original y selección obtenida
intersect(dat_agg_sum_split_lsa$sentences %>% str_trim,
          sum_agg_lsa %>% str_trim) -> sum_agg_lsa_nsent
sum_agg_lsa_nsent # Resultado ordenado
```

A.5.2 LexRank

```
library(lexRankr)
# Split: extracción de las oraciones según un patrón más elaborado
dat_agg_est %>% select(Summary)  %>% mutate(ID_doc = 1) %>%
 tidytext::unnest_tokens(output = sentences, input = Summary,
 pattern = "(?<!([[:punct:]]|\\s)[[:alpha:]]{1,3})\\.(?!\\s[a-z])|
```

```
  \\!|\\?|:", token="regex", to_lower = F) %>%
 mutate(ID_sent = 1:nrow(.),
  sentences = map_chr(sentences,str_trim)) -> dat_agg_sum_split_lr
# LexRank Normal
dat_agg_sum_split_lr %>% bind_lexrank(sentences, ID_doc,
             level = 'sentences',continuous = F) -> sum_agg_lr
# LexRank Contiuo
dat_agg_sum_split_lr %>% bind_lexrank(sentences, ID_doc,
         level = 'sentences',continuous = T) -> sum_agg_lr_con
# Acumulación de la importancia, para selección posterior
sum_agg_lr %>% arrange(desc(lexrank)) %>%
           mutate(Cum_Sum = cumsum(lexrank)) -> sum_agg_lr_ord
sum_agg_lr_con %>% arrange(desc(lexrank)) %>%
       mutate(Cum_Sum = cumsum(lexrank)) -> sum_agg_lr_ord_con
# Número de oraciones a seleccionar para que la acumulada > umbral
which(sum_agg_lr_ord$Cum_Sum>0.7)[1] -> n_sent_lr
which(sum_agg_lr_ord_con$Cum_Sum>0.7)[1] -> n_sent_lr_con
sum_agg_lr_ord %>% head(n_sent_lr) -> sum_agg_lr_nsent
sum_agg_lr_ord_con[1:n_sent_lr_con,]->sum_agg_lr_nsent_con
# Ordenación de la selección por aparición original (coherencia)
sum_agg_lr_nsent %>%  arrange(ID_sent) %>%
        select(sentences) %>% pluck(1) -> sum_agg_lr_nsent_str
sum_agg_lr_nsent_con %>%  arrange(ID_sent) %>%
      select(sentences) %>% .[[1]] -> sum_agg_lr_nsent_str_con
# Evolución de la importancia para cada unidad
## Normal
sum_agg_lr_ord %>%  mutate(Index = 1:nrow(.)) %>%
 ggplot(aes(x=Index,y=lexrank)) +
  geom_point(color="#a8003b",na.rm = T) +
  geom_line(color="#a8003b", nar.rm = T) + ylab("") +
  labs(title = "Evolución de la importancia para cada oración",
       subtitle = "LexRank (Lib. agg)") + theme_light()
## Continuo
sum_agg_lr_ord_con %>%  mutate(Index = 1:nrow(.)) %>%
ggplot(aes(x=Index,y=lexrank)) +
  geom_point(color="#a8003b",na.rm = T) + ylab("") +
```

```
  geom_line(color="#a8003b", nar.rm = T) + theme_light() +
  labs(title = "Evolución de la importancia para cada oración",
       subtitle = "LexRank Continuo (Lib. agg)")
## Ambas
union(sum_agg_lr_ord %>% select(ID_doc,lexrank) %>%
  mutate(index = 1:nrow(.)), sum_agg_lr_ord_con %>%
   select(lexrank) %>% mutate(ID_doc = 2, index = 1:nrow(.))) %>%
  mutate(ID_doc = factor(ID_doc,levels=c(1,2),
                         labels=c("Normal", "Continuo"))) %>%
  ggplot(aes(x = index, y = lexrank, colour=ID_doc)) + ylab("") +
  geom_point(na.rm = T) + geom_line(na.rm = T,lwd=0.7) + xlab("") +
  labs(title = "Evolución de la importancia para cada oración",
       subtitle = "Lib. agg", color = "LexRank") +   theme_light() +
  theme(legend.box.margin = margin(-30,0,250,-190),
        legend.direction = "horizontal")
```

A.5.3 TextRank

```
library(textrank)
library(udpipe) # Añadir POS y tokenizar
tagger <- udpipe_download_model("english")
tagger <- udpipe_load_model(tagger$file_model)
dat_agg_ann_tr <- udpipe_annotate(tagger,
                    dat_agg_est$Summary[1]) %>% as_tibble
# Keywords
kw_agg_tr <- textrank_keywords(dat_agg_ann_tr$lemma,
  relevant = dat_agg_ann_tr$upos %in% c("NOUN", "VERB",
                                        "ADJ","PROPN"))
kw_agg_tr$keywords %>%
 mutate(TextRank = kw_agg_tr$pagerank$vector[1:22]) %>%
 arrange(desc(TextRank)) %>%
 rename(Frecuencia=freq,KeyWord=keyword) -> kw_agg_tr_ord
# KeyWords de mayor relevancia
kw_agg_tr_ord %>% head(7) %>% select(-ngram)
# Representación: evolución de la puntuación según su frecuencia
kw_agg_tr_ord %>% mutate(Index = 1:nrow(.))  %>%
```

```
ggplot(aes(x=Index, y=TextRank, color=Frecuencia)) +
 geom_point() + geom_line() +  ylab("") + xlab("") +
 scale_color_continuous(type = "viridis",begin = 0.3,
    breaks=c(2,4,8,16))+ theme_light() +
 labs(title = "KeyWord TextRank (Lib.Ref.)", colour = "Freq") +
 theme(legend.box.margin = margin(0,0,20,-130),
 legend.direction="horizontal",legend.key.size=unit(0.6,"cm"))+
 guides(color = guide_legend(label.position = "top"))
# KeySentences
dat_agg_ann_tr %>% # Creación de un identificador
  mutate(textrank_id  = unique_identifier(.,
    c("doc_id", "paragraph_id", "sentence_id"))) -> dat_agg_tr
sentences_agg_tr <- unique(dat_agg_tr[,
                                c("textrank_id", "sentence")])
dat_agg_tr%>% # Selección de las categorías relevantes
  filter(upos %in% c("NOUN","ADJ","VERB")) %>%
  select(textrank_id,lemma) -> terminology_agg_tr
# Textrank
sum_agg_tr <- textrank_sentences(data = sentences_agg_tr,
                              terminology = terminology_agg_tr)
# Acumulación de la importancia, para selección posterior
sum_agg_tr$sentences %>% arrange(desc(textrank)) %>%
  mutate(Cum_Sum = cumsum(textrank)) -> sum_agg_tr_ord
# Evolución de la puntuación para cada unidad
sum_agg_tr_ord %>% mutate( Index = 1:nrow(.)) %>%
 ggplot(aes(x=Index,y=textrank)) + geom_point(color="#a8003b") +
  geom_line(color="#a8003b") +
  labs(title = "Evolución de la importancia para cada oración",
       subtitle = "TextRank (Lib. 1)") + ylab("") + theme_light()
# Número de oraciones a seleccionar para alcanzar el umbral
min(which(sum_agg_tr_ord$Cum_Sum > 0.7)[1],8) -> n_sent_tr
# Ordenación de la selección por aparición original (coherencia)
sum_agg_tr_ord %>% head(n_sent_tr) ->  sum_agg_tr_nsent
sum_agg_tr_nsent %>%
 arrange(textrank_id) %>% select(sentence) %>% .[[1]] %>%
 str_replace(pattern = "\\.$",replacement = "")->sum_agg_tr_nsent_str
```

A.5.4 Comparaciones

```
# Evolución conjunta de la puntuación para cada oración
union(sum_agg_lr_ord %>% select(ID_doc, lexrank) %>%
  mutate(Index = 1:nrow(.)), sum_agg_lr_ord_con %>% select(lexrank) %>%
    mutate(ID_doc=2,Index=1:nrow(.))) %>% rename(Imp = lexrank) %>%
 union(dat_tr %>% select(textrank, Index) %>% rename(Imp=textrank) %>%
  mutate( ID_doc = 3)) %>% mutate(ID_doc = factor(ID_doc,
    levels=c(1,2,3), labels=c("LR.Nor.", "LR.Con.", "TR"))) %>%
 ggplot(aes(x = Index, y = Imp, colour=ID_doc)) + xlab("") +
  geom_point(na.rm = T) + geom_line(na.rm = T,lwd=0.7) + ylab("") +
  labs(title = "KeySentences (Lib.Ref.)", color = "") + theme_light() +
  theme(plot.subtitle = element_text(margin=margin(0,0,10,0)),
  legend.box.margin=margin(0,0,200,-140),legend.direction="horizontal")
# Intersección entre las oraciones extraídas
## Todos
intersect(sum_agg_lr_nsent_str_con, sum_agg_tr_nsent_str) %>%
 intersect(sum_agg_lr_nsent_str) %>% intersect(sum_agg_lsa_nsent)
## LexRank Continuo y TextRank
intersect(sum_agg_lr_nsent_str_con, sum_agg_tr_nsent_str)
# Puntuación y Ranking para la oración común para cada técnica
## LSA
which(sum_agg_lsa %>% str_detect("Anne, a young"))
## LexRank Normal
sum_agg_lr_nsent %>%
 filter(sentences %>% str_detect("Anne, a young orphan")) %>%
  transmute(Rank = which(sum_agg_lr_nsent$sentences %>%
    str_detect("Anne, a young")), Tec. = "LexRank", Imp. = lexrank)
## LexRank Continuo
sum_agg_lr_nsent_con %>%
  filter(sentences %>% str_detect("Anne, a young orphan")) %>%
  transmute(Rank = which(sum_agg_lr_nsent_con$sentences %>%
    str_detect("Anne, a young")), Tec. = "LexRank Con.", Imp. = lexrank)
## TextRank
sum_agg_tr_nsent %>%
  filter(sentence %>% str_detect("Anne, a young orphan")) %>%
```

```
  transmute(Rank = which(sum_agg_tr_nsent$sentence %>%
    str_detect("Anne, a young")), Tec. = "TextRank", Imp. = textrank)
# Diferencia del tiempo computacional
## TexRank para agg:
start_time_tr_fl <- Sys.time()
dat_agg_est$Summary %>% Sum_TextRank()
end_time_tr_fl <- Sys.time()
diff_time_tr_fl <- end_time_tr_fl - start_time_tr_fl
## TexRank para los 5 primeros del conjunto
start_time_tr_h5 <- Sys.time()
dat_est_sum_t3 %>% head(5) %>%
  rowwise() %>% mutate(Sum_pr = Sum_TextRank(Summary))
end_time_tr_h5 <- Sys.time()
diff_time_tr_h5 <- end_time_tr_h5 - start_time_tr_h5
## LexRank Continuo para agg
start_time_lr_fl <- Sys.time()
dat_agg_est$Summary %>% Sum_LexRank()
end_time_lr_fl <- Sys.time()
diff_time_lr_fl <- end_time_lr_fl - start_time_lr_fl
diff_time_lr_fl
## LexRank Continuo para los 5 primeros del conjunto
start_time_lr_h5 <- Sys.time()
dat_est_sum_t3 %>% head(5) %>%
  rowwise() %>% mutate(Sum_pr = Sum_LexRank(Summary))
end_time_lr_h5 <- Sys.time()
diff_time_lr_h5 <- end_time_lr_h5 - start_time_lr_h5
diff_time_lr_h5
```

A.5.5 Funciones genéricas

```
Sum_TextRank <- function(Summary,threshold = 0.7,sent_min = 8) {
 # anotación de POS-tagging
 dat_ann_tr <- udpipe_annotate(tagger, Summary) %>% as_tibble %>%
          mutate(textrank_id = unique_identifier(.,
          c("doc_id", "paragraph_id", "sentence_id"))) -> dat_tr
 # oraciones
```

```
 sentences_tr <- unique(dat_tr[, c("textrank_id", "sentence")])
 # palabras relevantes de cada oración: nombres, adjetivos y verbos
 dat_tr%>% filter(upos %in% c("NOUN","ADJ","VERB")) %>%
    select(textrank_id,lemma) -> terminology_tr
 # Ranking de oraciones
 sum_tr <- textrank_sentences(data = sentences_tr,
                              terminology = terminology_tr)
 # Ordenacion según la relevancia y cálculo de la acumulada
 sum_tr$sentences %>% arrange(desc(textrank)) %>%
    mutate(Cum_Sum = cumsum(textrank)) -> sum_tr_ord
 # Número de oraciones a seleccionar según el umbral (Máximo 8)
 min(which(sum_tr_ord$Cum_Sum > threshold)[1],sent_min) -> n_sent_tr
 # Selección de las oraciones según el número establecido
 sum_tr_ord %>% head(n_sent_tr) %>% arrange(textrank_id) %>%
  select(sentence) %>% .[[1]] %>% # seleccion de las oraciones
  str_c(collapse = " ") -> sum_tr_nsent_str
  return(sum_tr_nsent_str)}

Sum_LexRank <- function(Summary,threshold = 0.7,sent_min = 8) {
  # Extracción de las oraciones
  Summary %>% as_tibble %>% mutate(ID_doc = 1) %>%
    tidytext::unnest_tokens(output = sentences, input = value,
      pattern = "(?<!([[:punct:]]|\\s|^)[[:alpha:]]{1,3})\\.
       (?!\\s[a-z])|\\!|\\?|:", token="regex", to_lower = F) %>%
    mutate(ID_sent = 1:nrow(.),
       sentences = map_chr(sentences,str_trim)) -> dat_sum_split_lr
  # Ranking por LexRank Continuo
  dat_sum_split_lr %>% bind_lexrank(sentences, ID_doc,
                    level = 'sentences', continuous = T) -> sum_lr
  # Acumulación de la relevancia
  sum_lr%>% arrange(desc(lexrank)) %>%
    mutate(Cum_Sum = cumsum(lexrank)) -> sum_lr_ord
  # Número de oraciones a seleccionar (Máximo 8)
  min(which(sum_lr_ord$Cum_Sum>threshold)[1],sent_min) -> n_sent_lr
  # Ordenación de la selección por aparición original (coherencia)
  sum_lr_ord %>% head(n_sent_lr) %>%  arrange(ID_sent) %>%
```

```
  select(sentences) %>% .[[1]] %>%
    str_c(collapse = ". ") -> sum_lr_nsent_str
  return(sum_lr_nsent_str)}

Sum_Cases <- function(Summary, thres, smin) {
 tryCatch(expr=Summary %>% Sum_LexRank(threshold=thres,sent_min=smin),
  error = function(e) {
   tryCatch(expr=Summary%>%Sum_TextRank(threshold=thres,sent_min=smin),
            error = function(e) Summary)})}
```

A.6 Resúmenes para cada observación

```
# Incorporación de la columna Resumen según la técnica asociada
dat_test %>% rowwise() %>% mutate(Summary_Out =
  ifelse(test = Summary_Len <3, yes = Summary,
              no = Sum_Cases(Summary))) -> dat_test_summaries
dat_train %>% rowwise() %>% mutate(Summary_Out =
  ifelse(test = Summary_Len <3, yes = Summary,
              no = Sum_Cases(Summary))) -> dat_train_summaries
# Incorporación de la columna de longitud respectiva al obtenido
dat_test_summaries_len <- dat_test_summaries %>%
   mutate(Summary_Out_Len = Count_split_sent(Summary_Out))
dat_train_summaries_len <- dat_train_summaries %>%
  mutate(Summary_Out_Len = Count_split_sent(Summary_Out))
# Union de los tres conjuntos
union(dat_train_summaries_len, dat_val_summaries_len) %>%
  union(dat_test_summaries_len) -> dat_samples_sets_Summaries_Out
library(magrittr) # Incorporación de la columna de diferncia
dat_samples_sets_Summaries_Out %<>% rowwise() %>%
  mutate(Summary_Diff_Len = Summary_Len - Summary_Out_Len)
# Análisis de la diferencia de longitudes
## Distribución de la variable diferencia
dat_samples_sets_Summaries_Out$Summary_Out_Len %>%  summary
## Número de libros para cada valor de la diferencia de longitud
dat_samples_sets_Summaries_Out %>% count(Summary_Diff_Len)
## Libros no resumidos según sus longitudes iniciales
```

```
dat_samples_sets_Summaries_Out %>%
  filter(Summary_Diff_Len == 0) %>% count(Summary_Len)
## Número de libros según longitudes originales
dat_samples_sets_Summaries_Out %>% count(Summary_Len)
## Extensión resultante de los resumidos de longitud incial 3
dat_samples_sets_Summaries_Out %>% filter(Summary_Len==3,
                     Summary_Diff_Len>0)%>%count(Summary_Diff_Len)
```

A.7 Exploración y Análisis

```
# Selección de las variables necesarias
dat_samples_sets_Summaries_Out %>%
 select(ID, Set, Title, Summary_Out, Genres)->dat_samples_rep_car
# Tokenización del corpues por palabras
dat_rep_car_tok <- dat_samples_rep_car %>% ungroup() %>%
  unnest_tokens(output="word",input=Summary_Out,to_lower=T) %>%
  anti_join(stop_words) %>% filter(!str_detect(word, "\\d"))
# Representación de las palabras más frecuentes (>250 ocurrencias)
library(RColorBrewer)
dat_bow_w_freq %>% filter(n > 250) %>% add_column(
  breaks = c(rep(seq(2,16,2),each=2),16,16) %>% as.factor()) %>%
ggplot(aes(x=fct_reorder(word,n), y=n, fill=breaks)) + ylab("") +
  geom_col(show.legend=F)+coord_flip()+xlab("")+theme_minimal() +
  scale_fill_manual(values = rev(brewer.pal(name = "YlOrRd",
   n = 9)[2:9]))+ ggtitle("Palabras de mayor frecuencia (Corpus)")
# WordCloud
library(wordcloud)
wordcloud(words=dat_bow_w_freq$word,colors=brewer.pal(9,"YlOrRd"),
          freq = dat_bow_w_freq$n,max.words = 30)
# Frecuencias por Géneros
## 4 Superiores: Romance, Biographical, Comedy y Fiction
dat_rep_car_tok %>% filter(str_detect(Genres,
 "Romance|Biographical|Comedy|Fiction")) %>% group_by(Genres) %>%
 count(word) %>% top_n(5) %>% mutate(word=fct_reorder(word,n))%>%
 group_by(Genres, word) %>% arrange(desc(n)) %>% ungroup() %>%
 mutate(word = factor(paste(word, Genres, sep = "__"),
```

```
              levels = rev(paste(word, Genres, sep = "__")))) %>%
ggplot(aes(x = word, y = n, fill = Genres)) +
  geom_col(show.legend = F,width = 0.7)  + xlab("") + ylab("") +
  theme(panel.background = element_blank(),
        strip.text.x = element_text(size =12, face="bold")) +
  facet_wrap(~ Genres, nrow = 2, scales="free") +
  scale_fill_viridis_d(begin = 0.4,end = 0.9) + coord_flip() +
  scale_x_discrete(labels = function(x) gsub("__.+$", "", x))
ggsave("Freq_Genres_Com.png")
## 4 Inferiores: History, Fantasy, Horror y Mystery
dat_rep_car_tok %>% filter(str_detect(Genres,
 "History|Fantasy|Horror|Mystery|")) %>% group_by(Genres) %>%
 count(word) %>% top_n(5) %>% mutate(word=fct_reorder(word,n))%>%
 group_by(Genres, word) %>% arrange(desc(n)) %>% ungroup() %>%
 mutate(word = factor(paste(word, Genres, sep = "__"),
              levels = rev(paste(word, Genres, sep = "__")))) %>%
ggplot(aes(x = word, y = n, fill = Genres)) +
  geom_col(show.legend = F,width = 0.7)   + ylab("")+
  facet_wrap(~ Genres, ncol = 2, scales="free") +  xlab("") +
  scale_fill_viridis_d(begin = 0.4,end = 0.9) + coord_flip() +
  theme(panel.background = element_blank(),
        strip.text.x = element_text(size =12, face="bold"))+
  scale_x_discrete(labels = function(x) gsub("__.+$", "", x))
ggsave("Freq_Genres_Esp.png",width = 8)
```

A.8 Representaciones

A.8.1 Representación BOW (TF-IDF)

```
library(quanteda)
# Cálculo de los valores TF-IDF y matriz documento-término
dat_rep_car_tok %>% count(ID,word) %>% bind_tf_idf(word,ID,n) %>%
  cast_dfm(ID, word, tf_idf) -> doc_term_matrix_tf_idf
```

A.8.2 Representación Word2Vec

```
library(word2vec)
# Word2Vec mediante skip-gram y negative sampling
word2vec(x = dat_samples_rep_car,"Summary_Out"][[1]],
 type = "skip-gram", dim = 100,window = 10, hs = F) -> model_sg_ns
word2vec_sg_ns <- as.matrix(model_sg_ns) %>% as.data.frame() %>%
  rownames_to_column(var="word") %>%
  rename_with(.cols = starts_with("V"), ~ str_replace(.x, "V","D"))
# Agrupación de términos por libro y agregación de words embeddings
dat_word2vec_sg_ns %>% select(-word) %>% nest(-(ID:Genres)) %>%
  mutate(W2V_Doc = map(data,colMeans)) %>% unnest_wider(W2V_Doc) %>%
  select(Set, Genres, starts_with("D",ignore.case = F))  %>%
  mutate(Genres = as.factor(Genres)) -> dat_w2v_t3
# Selección de las observaciones correspondientes a Train y Test
dat_w2v_t3 %>% filter(Set!="Test") %>% select(-Set) -> dat_w2v_train
dat_w2v_t3 %>% filter(Set=="Test") %>% select(-Set) -> dat_w2v_test
```

A.8.3 Representación GloVe

```
library(textdata)
glove6b <- embedding_glove6b(dimensions = 100)
glove6b %>% rename(word = token) %>%
  rename_with(.cols = starts_with("d"), toupper) -> glove
```

A.9 MODELOS DE CLASIFICACIÓN

```
library(tidymodels)
# Modelo para optimización de hiperparámetros
mod_svm_tune <- svm_rbf(cost=tune(), rbf_sigma=tune()) %>%
  set_engine("kernlab") %>% set_mode("classification")
# Combinaciones de parámetros para optimizar
grid_params <- expand_grid(cost = 2^(-5:15), rbf_sigma =  2^(-8:8))
# Métricas para evaluar la optimización
```

```
metrics_class <- metric_set(accuracy, roc_auc)
# Métricas para la evaluación final de resultados
custom_metric <- metric_set(accuracy, specificity, sens,
                            precision, f_meas, roc_auc)
```

A.9.1 Modelo BOW (TF-IDF)

```
# Selección de los datos resultantes tras el resumen
dat_samples_sets_Summaries_Out %>%
  select(ID, Set, Title, Summary_Out, Genres) %>%
  mutate(Genres = as.factor(Genres)) -> dat_samples_rep_car_tf
# Conjuntos Train y Test
dat_bow_train <- dat_samples_rep_car_tf %>% filter(Set == "Train")
dat_bow_test <- dat_samples_rep_car_tf %>% filter(Set == "Test")
# Tratamiento para obtener la representación
library(textrecipes)
tfidf_summ_rec <- recipe(Genres ~ Summary_Out, data = dat_bow_train)%>%
  step_tokenize(Summary_Out) %>% step_stopwords(Summary_Out) %>%
  step_tokenfilter(Summary_Out, min_times = 3, max_tokens = 6104) %>%
  step_tfidf(Summary_Out)
# Conjuntos de datos procesados
dat_tfidf_train_prep <- tfidf_summ_rec%>%prep(training=dat_bow_train)
dat_tfidf_train_bake <- dat_tfidf_train_prep %>% bake(new_data=NULL)
dat_tfidf_test_bake<-dat_tfidf_train_prep %>%bake(new_data=dat_bow_test)
# Entrenamiento del modelo
svm_rbf() %>% set_mode("classification") %>% set_engine("kernlab") %>%
  fit(Genres ~., dat_tfidf_train_bake) -> svm_tfidf_fit
# Predicciones
pred_tfidf_class <- predict(svm_tfidf_fit, dat_tfidf_test_bake)
pred_tfidf_probs<-predict(svm_tfidf_fit,dat_tfidf_test_bake,type="prob")
# Resultados
dat_tfidf_test_bake %>% select(Genres) %>%
  bind_cols(pred_tfidf_class, pred_tfidf_probs) -> dat_tfidf_res
# Métricas de evaluación
custom_metric(dat_tfidf_res, truth = Genres,
              estimate = .pred_class, .pred_Comedy:.pred_Horror) %>%
```

```
  mutate(.model = "BOW (TF-IDF)") -> met_tfidf
```

A.9.2 Modelo Word2Vec

```
# Función para obtener los resultados del entrenamiento y evaluación
# para el mejor modelo tras la optimización
res_met <- function(wf, wf_tune_fit, dat_train, dat_test, model_name) {
  # Selección del mejor modelo
  best_model <- wf_tune_fit %>% select_best()
  # Finalización del workflow
  wf_fin <- wf %>% finalize_workflow(best_model)
  # Entrenamiento del mejor modelo. Conjunto Train
  wf_fin_fit <- wf_fin %>% fit(dat_train)
  # Obtención de predicciones para el conjunto Test
  pred_class <- predict(wf_fin_fit, dat_test)
  pred_probs <- predict(wf_fin_fit, dat_test, type="prob")
  dat_test %>% select(Genres) %>%
    bind_cols(pred_class, pred_probs) -> dat_res
  # Métricas de evaluación
  met <- custom_metric(dat_res,truth = Genres, estimate = .pred_class,
                       .pred_Adventure:.pred_Romance) %>%
          mutate(.model = model_name)
  return(list(Res = dat_res, Met = met, Best_Mod=best_model))
}

# Workflow
svm_w2v_wf <- workflow() %>%
                add_model(mod_svm_tune) %>% add_formula(Genres~.)
# Pliegues para optimización por CV
w2v_train_folds <- vfold_cv(dat_w2v_train, v=10, strata=Genres)
# Optimización de hiperparámetros
svm_w2v_tune_fit2 <- tune_grid(svm_w2v_wf, resamples=w2v_train_folds,
                          grid = grid_params, metrics = metrics_class)
# Resultados y métricas
svm_w2v_res <- res_met(svm_w2v_wf, svm_w2v_tune_fit, dat_w2v_train,
                    dat_w2v_test, "Word2Vec")
```

A.9.3 Modelo GloVe

```
# Funcion para obtener los conjuntos Train y Test según la longitud
dat_glv <- function(dat_samples_Summary_LenX) {
  dat_samples_Summary_LenX %>%
    select(ID, Set, Title, Summary_Out, Genres) -> dat_samples_rep_car
    # Tokenizar cada resumen obtenido
    dat_rep_car_tok <- dat_samples_rep_car %>% ungroup() %>%
      unnest_tokens(output="word", input=Summary_Out, to_lower=T) %>%
      anti_join(stop_words) %>%
      filter(!str_detect(word, "\\d"))
    dat_rep_car_tok %>%
      left_join(glove, by = "word") -> dat_glove6b
    # hay muchas palabras que no se encuentran entre las disponibles
    # por lo que se pierde mucha informacion
    dat_glove6b %>%
      filter(!if_any(.fns = is.na)) -> dat_glove6b_nna
    dat_glove6b_nna %>%
      select(-word) %>%
      nest(-(ID:Genres)) -> dat_glove6b_t1
    dat_glove6b_t1 %>%
      mutate(GV_Doc = map(data,colMeans)) -> dat_glove6b_t2
    dat_glove6b_t2 %>%
      unnest_wider(GV_Doc) %>%
      select(Set, Genres, starts_with("D",ignore.case = F))  %>%
      mutate(Genres = as.factor(Genres)) -> dat_glove6b_t3
    dat_glove6b_t3 %>%
      filter(Set != "Test") %>% select(-Set) -> dat_train
    dat_glove6b_t3 %>%
      filter(Set == "Test") %>% select(-Set) -> dat_test
    return (list(Train = dat_train, Test = dat_test))
}

# Longitud 8 (Inicial)
## Conjuntos Train y Test
dat_glv8 <- dat_glv(dat_samples_sets_Summaries_Out)
```

```
## Pliegues para CV
glv8_train_folds <- vfold_cv(dat_glv8_train, v=10, strata=Genres)
## Workflow
svm_glv8_wf <- workflow() %>%
                add_model(mod_svm_tune) %>% add_formula(Genres~.)
## Optimización de hiperparámetros
svm_glv8_tune_fit <- tune_grid(svm_glv8_wf, resamples=glv8_train_folds,
                          grid = grid_params, metrics = metrics_class)
## Resultados y métricas
svm_glv8_res <- res_met(svm_glv8_wf, svm_glv8_tune_fit, dat_glv8$Train,
                        dat_glv8$Test, "GloVe (Len. 8)")

# Longitud 20
## Conjuntos Train y Test
dat_glv20 <- dat_glv(dat_samples_sets_Summaries_Out20)
## Pliegues para CV
glv20_train_folds <- vfold_cv(dat_glv20_train, v=10, strata=Genres)
## Workflow
svm_glv20_wf <- workflow() %>%
                    add_model(mod_svm_tune) %>% add_formula(Genres~.)
## Optimización de hiperparámetros
svm_glv20_tune_fit<-tune_grid(svm_glv20_wf,resamples=glv60_train_folds,
                           grid = grid_params, metrics = metrics_class)
## Resultados y métricas
svm_glv20_res <- res_met(svm_glv20_wf, svm_glv20_tune_fit,
                   dat_glv20$Train, dat_glv20$Test, "GloVe (Len. 20)")
# Longitud 60
## Conjuntos Train y Test
dat_glv60 <- dat_glv(dat_samples_sets_Summaries_Out60)
## Pliegues para CV
glv60_train_folds <- vfold_cv(dat_glv60_train, v=10, strata=Genres)
## Workflow
svm_glv60_wf <- workflow() %>%
                    add_model(mod_svm_tune) %>% add_formula(Genres~.)
## Optimización de hiperparámetros
svm_glv60_tune_fit<-tune_grid(svm_glv60_wf,resamples=glv60_train_folds,
```

```
                            grid = grid_params, metrics = metrics_class)
## Resultados y métricas
svm_glv60_res <- res_met(svm_glv60_wf, svm_glv60_tune_fit,
                  dat_glv60$Train, dat_glv60$Test, "GloVe (Len. 60)")
```

A.9.4 Modelo CNN

```
# Resúmenes de extensión 60
dat_samples_sets_Summaries_Out60 %>%
  select(ID, Set, Title, Summary_Out, Genres) %>% ungroup() %>%
  mutate(Genres = as.factor(Genres), # tratamiento numérico de 0 a 10
         Genres_Num = as.numeric(Genres)-1) -> dat_rep60
# Conjuntos Train y Test
dat_cnn_train <- dat_rep60 %>% filter(Set != "Test")
dat_cnn_test <- dat_rep60 %>% filter(Set == "Test")
# Tratamiento del conjunto de datos
library(textrecipes)
max_words <-  5000 # Número máximo de características
max_length <- 300 # número máximo para la representación de 1 doc
dat_cnn_rec <- recipe(~ Summary_Out, data = dat_cnn_train) %>%
 step_tokenize(Summary_Out) %>% # tokenization, filtrado y onehot-encod.
 step_tokenfilter(Summary_Out,max_tokens = max_words) %>%
 step_sequence_onehot(Summary_Out, sequence_length = max_length)
# Conjuntos de datos procesados
dat_cnn_train_prep <- prep(dat_cnn_rec)
dat_cnn_train_bake <- bake(dat_cnn_train_prep, new_data = NULL,
                           composition = "matrix")
dat_cnn_test_bake <- bake(dat_cnn_train_prep, new_data = dat_cnn_test,
                                composition = "matrix")
# Embeddings
glove6bEmb <- tidy(dat_cnn_trainprep, 3) %>% select(token) %>%
  left_join(glove %>% rename("token" = word), by ="token") %>%
  mutate_all(replace_na, 0) %>% select(-token) %>%
  as.matrix() %>% rbind(0, .)
# Genres One-Hot Encoding
dat_cnn_train$Genres_Num %>% matrix(nrow =1500, ncol=1) %>%
```

```
  to_categorical(num_classes = 10) -> genres_train_one_hot
dat_cnn_test$Genres_Num %>% matrix(nrow =500, ncol=1) %>%
  to_categorical(num_classes = 10) -> genres_test_one_hot
# Modelo
## Parámetros
batch_size <- 300; embedding_dims <- 100; filters <- 250
kernel_size <- 3; hidden_dims <- 250;epochs <- 25
## Construcción
model <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_words+1,
          output_dim = embedding_dims, input_length = max_length) %>%
  layer_dropout(0.3) %>%
  layer_conv_1d(filters, kernel_size, padding = "valid",
                activation = "relu", strides = 1) %>%
  layer_global_max_pooling_1d() %>% layer_dense(hidden_dims) %>%
  layer_dropout(0.4) %>% layer_activation("relu") %>%
  layer_dense(units = 10, activation = "softmax")
## Asignación de embeddings iniciales
model %>%
  get_layer(index = 1) %>%
  set_weights(list(glove6bEmb))
## Asignación función de pérdida, métrica y optimizador
model %>%
  compile(optimizer = "adam",
    loss = "categorical_crossentropy",
    metrics = c("accuracy"))
## Entrenamiento
model_cnn_history <- model %>% fit(dat_cnn_train_bake,
      genres_train_one_hot, batch_size = batch_size, epochs = epochs,
      validation_data = list(dat_cnn_test_bake, genres_test_one_hot))
# Predicciones
pred_cnn <-predict(model2, dat_cnn_test_bake) %>% as_tibble() %>%
  bind_cols(dat_cnn_test_bake %>% select(Genres_Num)) -> pred_cnn_res
# Tratamiento para volver a establecer los géneros
pred_cnn_res_t1 <- pred_cnn_res %>% rowwise() %>%
  mutate(.pred_class = which.max(c_across(V1:V10)),
```

```
          Genres_Num = Genres_Num+1,
      across(c(.pred_class,Genres_Num),.fns=~factor(.,levels=1:10))) %>%
  rename("Genres"="Genres_Num", "Adventure"="V1", "Biographical"="V2",
    "Children"="V3", "Comedy"="V4", "Fantasy"="V5", "Fiction"="V6",
    "History"="V7", "Horror"="V8", "Mystery"="V9", "Romance"="V10")%>%
  rename_with(.cols = -(Genres:.pred_class), .fn = ~ str_c(".pred_",.))
levGen <- colnames(pred_cnn_res_t1) %>%head(10) %>% str_remove(".pred_")
pred_cnn_res_t2 <- pred_cnn_res_t1 %>% rowwise() %>%
  mutate(across(.cols = c(Genres,.pred_class), .fns = ~levGen[.])) %>%
  ungroup() %>% mutate(across(.cols = c(Genres,.pred_class),
                              .fns = ~factor(.,levels=levGen)))
# Métricas
pred_cnn_res_t2 %>% ungroup() %>%
  custom_metric(truth = Genres, estimate = .pred_class,
                .pred_Adventure:.pred_Romance) %>%
  mutate(.model = "CNN") -> met_cnn
# Matriz de Confusión
conf_mat_cnn <- pred_cnn_res_t2  %>%
  conf_mat(truth = Genres, estimate = .pred_class)
conf_mat_cnn <- ggplot2::autoplot(type ="heatmap") +
 scale_fill_gradient(low = "#F7FBFF", high = "#084594") +
 theme(axis.text.x = element_text(angle = 50, hjust =-0.02, size=10),
       axis.text.y = element_text(size = 10),
       axis.title.x = element_text(face = "bold"),
       axis.title.y = element_text(face = "bold")) + coord_flip() +
 scale_y_discrete(position = "right")
# Acierto por géneros
conf_mat_cnn %>% tidy() %>% select(value) %>% pluck(1) %>%
  matrix(byrow = T,nrow=10,ncol=10) %>%
  prop.table(margin = 1) %>% diag() -> cnn_acc_gen
names(cnn_acc_gen) <- levGen
# Valores curva ROC
roc_curve(pred_cnn_res_t2,truth=Genres,.pred_Adventure:.pred_Romance,
          options = list(smooth = T,smooth.n = 700)) -> roc_cnn_60
## Función para generar el gráfico
fun_ROC <- function (df,sub_grup = 1) {
```

```
  if (sub_grup == 1) {col = c("#440154FF","#3E4A89FF","#26828EFF",
    "#6DCD59FF","#DOF212")} else {col = rev(c("#FFF46F","#FD8D3C",
                                  "#E31A1C","#BD0026", "#810000"))}
  df %>% ggplot(aes(x = ejeX, y = sensitivity, color = .level)) +
  geom_line(lwd=1) + xlab("1 - specificity") + theme_light() +
  labs(title = "", color = "") + scale_color_manual(values=col) +
  theme(legend.direction="horizontal",legend.key.size=unit(0.6,"cm"),
        legend.position = "top")+
  guides(color = guide_legend(label.position = "top")) + coord_fixed()}
### Primer subgrupo  de géneros
roc_cnn_60 %>%  filter(str_detect(.level,
         "Romance|Fantasy|History|Fiction|Bipgraphical|Horror")) %>%
 mutate(ejeX = 1-specificity, .level = factor(.level,
  levels = c("Fantasy","History","Horror","Romance","Fiction"))) %>%
  fun_ROC
### Segundo subgrupo de géneros
roc_cnn_60 %>% filter(str_detect(.level,
         "Children|Biographical|Mystery|Adventure|Comedy")) %>%
  mutate(ejeX = 1-specificity, .level = factor(.level, levels =
   c("Children's literature", "Mystery", "Adventure", "Biographical",
     "Comedy"))) %>% fun_ROC(sub_grup=2)
```

A.10 IDENTIFICACIÓN DE TEMÁTICAS

```
library(textmineR)
# Creación de la matriz documento-término con el tipo adecuado
CreateDtm(doc_vec = dat_samples_rep_car$Summary_Out,
  doc_names = dat_samples_rep_car$ID, ngram_window = c(1,2),
  stopword_vec = stop_words, verbose=F) -> text_mine_dtm
# Filtrando las palabras de mayor de 2 frecuencia
tm_dtm_fi <- text_mine_dtm[,colSums(text_mine_dtm) > 2]
# Modelo LDA
lda_tm10 <- FitLdaModel(dtm = tm_dtm_fi, k = 10,iterations = 500,
               burnin = 100, alpha = 5, beta = 0.02)
# Palabras de mayor importancia para cada topic
GetTopTerms(phi = lda_tm10$phi, M = 10) %>% as_tibble() %>%
```

```
  rename_with(.fn = str_to_title) -> tab_top_10
# Correspondencias entre observaciones topic de mayor peso por género
lda_tm10$theta %>% as_tibble %>% rename_with(str_to_title) %>%
  rowwise %>% mutate(Top_Max = which.max(c_across(T_1:T_10))) %>%
  bind_cols(Genres = dat_samples_rep_car$Genres %>% as.factor()) %>%
  group_by(Genres) %>% count(Top_Max) -> dat_top_gen_tm10
# Selección de los 4 topics más frecuentes para cada género
dat_top_gen_tm10 %>% top_n(4) %>%
  pivot_wider(names_from = Top_Max, values_from = n) %>%
  select(Genres,as.character(c(1:10))) -> dat_top_gen_tm10_4top
# Representación
dat_top_gen_tm10_4top %>%
 mutate(Genres = str_remove(Genres, "'s literature")) %>%
 pivot_longer(-Genres, names_to = "Top_Max", values_to = "n") %>%
 mutate(Top_Max=factor(Top_Max,levels=1:10,labels=str_c("T", 1:10)),
  n_na = n %>% replace_na("-"), n_0 = n %>% replace_na(0)) %>%
  ggplot(aes(y = Genres, x = Top_Max, fill = n_0)) +
  geom_tile() + geom_text(aes(label = n_na))  +
  scale_fill_gradient(low = "#F7FBFF", high = "#084594") +
  scale_x_discrete(position = "top") + xlab("") + ylab("") +
  theme(panel.background = element_blank(),legend.position = "none",
   axis.text.x=element_text(size=10),axis.text.y=element_text(size=10))
```

A.11 Tablas

```
con_est <- c("striped","hover","responsive")
# Funciones de Activación
t1 <- c("$\\mathbf{Sigmoid}$", "$\\mathbf{Softmax}$",
        "$\\mathbf{Tanh}$", "$\\mathbf{ReLU}$")
t2 <- c("$\\displaystyle \\sigma(x) = \\frac{1}{1+e^{-x}}$",
        "$\\displaystyle \\text{softmax}(x)_i = \\frac{e^{x_i}}
         {\\sum_k^{K} e^{x_k}} \\ \\ \\forall i = 1\\dots K$",
        "$\\displaystyle \\text{tanh}(x)=\\frac{e^{2x-1}}{e^{2x+1}}$",
        "$ReLU(x) = \\max(0,x) = \\left \\lbrace \\begin{array}{ll} 0
         & x \\lt 0 \\\\ x & c.c. \\end{array} \\right.$")
t3 <- c("Generalmente referida al caso particular de función logística",
```

```
        "Función exponencial normalizadora. Fuerza que sus valores
         sumen 1 y puedan interpretarse como probabilidades.",
        "Entra dentro de las funciones 'S-shape' escalando los valores
         al rango [-1,1].", "A pesar de su simplicidad proporcioda muy
         buenos resultados, sobretodo junto a la técnica dropout.")
tb1 <- data.frame(C1 = t1,C2 = t2, C3= t3)
tb1 %>% knitr::kable(align=c('c','c','l'), format="html", escape=F,
  col.names = c("$\\mathbf{Activaci\\acute{o}n}$",
  "$\\mathbf{F\\acute{o}rmula}$", "$\\mathbf{Propiedades}$")) %>%
  kable_styling(bootstrap_options = con_est, font_size = 14)
# Funciones de Pérdida
tb21 <- c("$\\mathbf{Hinge}$ $\\mathbf{(Binary)}$",
  "$\\textbf{Cross}$-$\\textbf{Entropy}$  $\\textbf{(Categorical)}$")
tb22 <- c("$L_h(x,x') = \\max(0, 1- x \\cdot x') \\ \\ \\text{con}
           \\ x \\in \\{\\pm 1\\}$",
          "$L_{ce}(x,x') = - \\displaystyle \\sum_{i}x_i log(x'_i)$")
tb23 <- c("Conocida también como 'margin' o 'SVM loss'. Extensible al
          caso múltiple", "Analiza las diferencia entre las distribu-
          ciones de probabilidad. Se asume que se recurre a la función
          'softmax' para el resultado del clasificador.")
tb2 <- data.frame(C1 = tb21,C2 = tb22, C3= tb23)
tb2 %>% knitr::kable(align=c('c','c','l'), format="html", escape = F,
  col.names = c("$\\mathbf{P\\acute{e}rdida}$",
      "$\\mathbf{F\\acute{o}rmula}$", "$\\mathbf{Propiedades}$")) %>%
  kable_styling(bootstrap_options = con_est, font_size = 14)
# Funciones Kernel (Núcleo)
t1 <-c("$\\mathbf{Polin\\acute{o}mico}$",
       "$\\mathbf{Gaussiano}$",
       "$\\mathbf{BOW}$",
       "$\\mathbf{Word}$-$\\mathbf{Pos.}$",
       "$\\mathbf{String}$",
       "$\\mathbf{Word}$-$\\mathbf{Seq.}$")
t2 <- c("$K_{pol}(x,x') = (x^{t}x' + c)^{p}$",
 "$K_{gau}(x,x') = exp\\bigg(\\frac{-||x - x'||^2}{2\\sigma^2}\\bigg)$",
 "$\\displaystyle K_{bow}(x,x')=\\sum_{t\\in V}tf(t,x)\\cdot tf(t,x')$",
 "$\\displaystyle K_{wpos}(x,x') = \\sum_{t \\in V} \\sum_{p = -c} ^{c}
```

```
  \\sum_{q=-c}^{c}tf(t,p,x) \\cdot tf(t,q,x') \\cdot g(p,q)$<br>$g(p,q)
  = exp\\big(-\\theta_1(p^2+q^2) - \\theta_2(p-q)^2\\big) + \\theta_3$",
 "$\\displaystyle K_{str}(x,x')=\\sum_{u\\in\\Sigma^{n}}\\sum_{i:u=x[i]}
  \\sum_{j:u=x'[j]} \\lambda^{l(i)+ l(j)}$",
 "$\\displaystyle K_{str}(x,x')=\\sum_{u\\in\\Sigma^{n}}\\sum_{i:u=x[i]}
  \\sum_{j:u=x'[j]} \\prod_{i_1\\leq k \\leq i_n} \\prod_{j_1 \\leq l
  \\leq j_n} \\lambda_{x_k} \\lambda_{x'_l}$")
t3 <- c("$p \\in \\mathbb{Z}^{+}, \\ c \\gt 0$. En NLP suele elegirse
 $p = 2$ (cuadrático) para no sobreajustar el resultado.",
 "$\\sigma^2 \\gt 0$. Define un espacio de características de dimensión
 infinita.", "$x=(t_{-n},\\dots,t_{-1},t_{1},\\dots,t_m)$<br>contexto
 de $t_0 \\in V$, <br> $tf(t,x)$: frecuencia de $t$ en $x$",
 "$c$: longitud del contexto, <br> $g(p,q)$: distancia entre posiciones,
 $\\theta_1$, $\\theta_2$: control del efecto entre posiciones,
 $\\theta_3$: bias.", "$u$: subsecuencia de $x$, $\\lambda \\leq 1$,
 $\\Sigma^{n}$: conjunto de secuencias de longitud n", "$\\lambda_{x_k},
 \\lambda_{x_l} \\leq 1 \\ \\ \\forall k,l$ (distintos factores de
 descenso para las sub-secuencias)")
t <- data.frame(C1 =t1,C2 =t2, C3=t3)
t %>% knitr::kable(align = c('c','c','l'), format = "html", escape = F,
 col.names = c("$\\textbf{Kernel}$","$\\mathbf{F\\acute{o}rmula}$",
 "$\\mathbf{Par\\acute{a}metros}$")) %>%
  kable_styling(bootstrap_options = con_est, font_size = 14)
# Funciones relativas a la Entropía
t1 <- c("$\\mathbf{Entropy}$",
        "$\\mathbf{Conditional}$ $\\mathbf{Entropy}$",
        "$\\mathbf{Information}$ $\\mathbf{Gain}$",
        "$\\mathbf{Symmetrical}$ $\\mathbf{Uncertainty}$")
t2 <- c("$\\displaystyle H(X)=- \\sum_{i}P(x_i)log\\big(P(x_i)\\big)$",
  "$\\displaystyle H(X|Y)=-\\sum_{j}P(y_j)\\sum_iP(x_i|y_j)log
   \\big(P(x_i|y_j)\\big)$", "$\\displaystyle IG(X|Y)=H(X)-H(X|Y)$",
  "$\\displaystyleSU(X,Y)=2\\Bigg[\\frac{IG(X|Y)}{H(X) + H(Y)}\\Bigg]$")
tb4 <- data.frame(C1 = t1,C2 = t2)
tb4 %>% knitr::kable(align = c('c','c','l'), format = "html", escape=F,
 col.names=c("$\\mathbf{Funci\\acute{o}n}$",
  "$\\mathbf{F\\acute{o}rmula}$")) %>%
```

```
  kable_styling(bootstrap_options = con_est, font_size = 14)
# Frecuencia de palabras AGG
dat_agg_nsw_freq %>% knitr::kable(align = c('c','c'), format = "html",
  escape = F) %>% kable_styling(bootstrap_options=con_est,font_size=14)
# Top KeyWords
kw_agg_tr_ord %>% head(7) %>% select(-ngram) %>%
  mutate(TextRank = round(TextRank,3)) %>%
  knitr::kable(align = c('c','c'),format = "html",escape = F) %>%
  kable_styling(bootstrap_options = con_est, font_size = 14)
# Oración Común a las 4 técnicas
t1 <- c("LSA", "LexRank", "LexRank Con.", "TextRank")
t2 <- c(5, 8, 6, 4)
t3 <- c("--", 0.049, 0.056, 0.069)
tb7 <- data.frame(C1 = t1,C2 = t2, C3 = t3)
tb7 %>% knitr::kable(align=c('l','c','c'), format="html",escape=F,
 col.names = c("Técnica", "Rank", "Imp.")) %>%
 kable_styling(bootstrap_options = con_est, font_size = 14)
# Métricas
met_mod_def <- bind_rows(met_tfidf, svm_w2v_res$Met,
                svm_glv8_res$Met,svm_glv20_res$Met,svm_glv60_res$Met)
## Métricas modelos SVM iniciales
met_mod_def %>% head(3) %>%
 knitr::kable(align=c('l',rep('c',6)),format="html",escape=F) %>%
 kable_styling(bootstrap_options = con_est, font_size = 14)
## Métricas modelos GloVe SVM y CNN
met_mod_def[4:6,] %>%
 knitr::kable(align=c('l',rep('c',6)),format="html",escape=F) %>%
 kable_styling(bootstrap_options = con_est, font_size = 14)
# Parámetros de los modelos
## SVM
svm_w2v_res$Best_Mod %>%
  bind_rows(svm_glv8_res$Best_Mod, svm_glv20_res$Best_Mod,
            svm_glv60_res$Best_Mod) %>% select(-.config) %>%
  bind_cols(Model=c("Word2Vec",
                    str_c("GloVe (Len. ", c(8,20,60),")"))) %>%
  bind_rows(tibble(cost = 1, rbf_sigma = 8.50694324931448e-05,
```

```
                    Model = "BOW (TF-IDF)"))-> params_opt
params_opt[c(5,1,2:4),c(3,1,2)] %>%
  knitr::kable(align=c('l',rep('c',2)),format="html",escape=F)%>%
  kable_styling(bootstrap_options = con_est, font_size = 14)
## CNN
params_cnn <- cbind.data.frame(Model= c("batch_size",
                          "epochs","layers"), CNN = c(300, 25,8))
df1_cnn <- rbind.data.frame(c("input_dim (Embeddings)", 5001),
 c("input_length (Embeddings)",300),c("output_dim (Embeddings)",100),
 c("dropout",0.3),c("filters(Conv_1d)",250),c("kernel_size(Conv_1d)",3),
 c("dropout", 0.4),c("units (Dense)", 250),c("units (Dense Fin.)",10))
colnames(df1) <- c("Model", "CNN")
params_cnn %>% mutate(CNN = CNN%>%as.character) %>% bind_rows(df1)%>%
  knitr::kable(align = c('l','c'),format="html",escape = F) %>%
  kable_styling(bootstrap_options = con_est, font_size = 14)
# Acierto por género. Modelo CNN
cnn_acc_gen %>% as_tibble %>%
  bind_cols(Genres = cnn_acc_gen %>% names) %>%
  arrange(desc(value)) %>%
  pivot_wider(names_from = Genres, values_from = value) %>%
  knitr::kable(align=c(rep('c',10)),format="html",escape=F) %>%
  kable_styling(bootstrap_options = con_est, font_size = 14)
# Palabras de mayor relevancia para cada topic
tab_top_10 %>% head(8) %>%
  rename_with(.fn = ~str_replace(string = ., "T","Top")) %>%
  knitr::kable(align=c(rep('c',10)),format="html",escape=F) %>%
  kable_styling(bootstrap_options = con_est, font_size = 14)
```

Bibliografía

[1] G. Miner, J. Elder, R. Nisbet, D. Delen, A. Fast, & T. Hill. *Practical Text Mining and Statistical Analysis for non-structured text data applications*. Academic Press, 2012.

[2] "10 Text Mining Examples." 2020. URL: https://www.expert.ai/blog/10-text-mining-examples/

[3] D. Sarkar. *Text Analytics with Python: A Practitioner's Guide to Natural Language Processing*, 2nd ed. Apress, 2019.

[4] J. Žižka, F. Dařena, & A. Svoboda. *Text Mining with Machine Learning: Principles and Techniques*. CRC Press, 2019.

[5] M. A. Hearst. "Untangling Text Data Mining." In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. pp. 3–10. College Park, Maryland, USA: Association for Computational Linguistics, Jun. 1999. URL: https://www.aclweb.org/anthology/P99-1001

[6] J. Dörre, P. Gerstl, & R. Seiffert. "Text Mining: Finding Nuggets in Mountains of Textual Data." In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp. 398–401. New York, NY, USA: Association for Computing Machinery, 1999.

[7] W. Li, B. Lee, F. Krausz, & K. Sahin. "Text Classification by a Neural Network." In *Proceedings of the 23rd Annual Summer Computer Simulation Conference*. pp. 313–318. Jul. 1991.

[8] C. Apté, F. Damerau, & S. M. Weiss. "Automated Learning of Decision Rules for Text Categorization." *ACM Transactions On Information Systems (TOIS)*. Vol. 12. No. 3. pp. 233–251. Association for Computing Machinery, 1994.

[9] P. J. Bickel, F. Götze, & W. R. van Zwet. "Resampling fewer than n Observatioon: gains, losses, and remedies for looses." *Statistica Sinica*. Vol. 7. pp. 1–31. Institute of Statistical Science, Academia Sinica, 1997.

[10] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, & R. Harshman. "Using Latent Semantic Analysis To Improve Access To Textual Information."

In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. pp. 281–285. Association for Computing Machinery, May 1988.

[11] T. K. Landauer, P. W. Foltz, & D. Laham. "An introduction to Latent Semantic Analysis." *Discourse Process*. Vol. 25. No. 2-3. pp. 259–284. 1998.

[12] T. Hofmann. "Probabilistic Latent Semantic Indexing." In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and development in information retrieval*. pp. 50–57. Aug. 1999.

[13] D. M. Blei, A. Y. Ng, & M. I. Jordan. "Latent Dirichlet Allocation." *Journal of Machine Learning Research*. Vol. 3. pp. 993–1022. 2003.

[14] S. Kim, H. Park, & J. Lee. "Word2vec-based latent semantic analysis (W2V-LSA) for topic modeling: A study on blockchain technology trend analysis." *Expert Systems With Applications*. Vol. 152. Elsevier, 2020.

[15] T. Mikolov, K. Chen, G. Corrado, & J. Dean. "Efficient Estimation of Word Representations in Vector Space." *International Conference on Learning Representations*. 2013. URL: https://arxiv.org/pdf/1301.3781.pdf

[16] T. Mikolov, K. Chen, G. Corrado, J. Dean, & I. Sutskever. "Distributed Representations of Words and Phrases and their Compositionality." *Neural Information Processing Systems Conference*. 2013. URL: https://arxiv.org/pdf/1310.4546.pdf

[17] K. Hornik, I. Feinerer, M. Kober, & C. Buchta. "Spherical k-Means Clustering." *Journal of Statistical Software*. Vol. 50. No. 10. pp. 1–22. 2012. URL: https://www.jstatsoft.org/index.php/jss/article/view/v050i10/v50i10.pdf

[18] T. Jo. *Text Mining: Concepts, Implementation and Big Data Challenge*. Springer, 2019.

[19] P. Jackson & I. Moulinier. *Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization*. Vol. 5. John Benjamins Publishing Company, 2007.

[20] "Natural Language Understanding: What is it and How is it Different from NLP?" 2020. URL: https://www.expert.ai/blog/natural-language-understanding-different-nlp/

[21] N. Chomsky. *Syntactic Structures*. The Hauge: Mounton & Co. Repreinted 1978, Peter Lang Publishing, 1957.

[22] B. H. Juang & L. R. Rabiner. "Hidden Markov Models for Speech Recognition." *Technometrics*. Vol. 33. No. 3. pp. 251–272. 1991.

[23] H. Zhao & J. Yu. "Part-of-Speech Tagging Based on Maximum Entropy." *International Journal of Circuits, Systems and Signal processing*. Vol. 12. pp. 483–487. 2018. URL: https://www.naun.org/main/NAUN/circuitssystemssignal/2018/b362005-aft.pdf

[24] H. Wang, Q. Luo, Z. Shang, G. Li, & X. Shi. "Network Traffic Text Classification Based on Multi-instance Learning and Principal Component Analysis." In *International Conference in Communications, Signal Processing, and Systems*. pp. 2519–2524. Springer, Singapore, 2019.

[25] P. C. Barman, N. Iqbal, & S. Y. Lee. "Non-negative Matrix Factorization Based Text Mining: Feature Extraction and Classification." In *International Conference on Neural Information Processing. Neural Information Processing*. pp. 703–712. Springer Berlin Heidelberg, 2006.

[26] G. Chandrashekar & F. Sahin. "A survey on feature selection methods." *Computers & Electrical Engineering*. Vol. 40. No. 1. pp. 16–28. 2014.

[27] M. Dash & H. Liu. "Feature selection for classification." *Intelligent data analysis*. Vol. 1. No. 1-4. pp. 131–156. 1997.

[28] G. Forman. "An extensive empirical study of feature selection metrics for text classification." *Journal of Machine Learning Research*. Vol. 3. pp. 1289–1305. 2003. URL: https://www.jmlr.org/papers/volume3/forman03a/forman03a.pdf

[29] T. Liu, S. Liu, & Z. Chen. "An evaluation on feature selection for text clustering." In *Proceedings 20th International Conference on Machine Learning*. pp. 488–495. 2003. URL: https://www.aaai.org/Papers/ICML/2003/ICML03-065.pdf

[30] K. Michalak & H. Kwasnicka. "Correlation based feature selection method." *International Journal of Bio-Inspired Computation*. Vol. 2. No. 5. pp. 319–332. 2010.

[31] Y. Lei & H. Liu. "Feature selection for high-dimensional data: a fast correlation-based filter solution." In *Proceedings of the 20th International Conference on International Conference on Machine Learning*. pp. 856–863. 2003. URL: https://www.aaai.org/Papers/ICML/2003/ICML03-111.pdf

[32] S. Sharma & A. Jain. "An Empirical Evaluation of Correlation Based Feature Selection for Tweet Sentiment Classification." In *Advances in Cybernetics, Cognition, and Machine Learning for Communication Technologies*. pp. 199–208. Springer, Singapore, 2020.

[33] B. Hawashin, A. M. Mansour, & S. Aljawarneh. "An Efficient Feature Selection Method for Arabic Text Classification." *International Journal of Computer Applications*. Vol. 83. No. 17. 2013.

[34] I. Guyon, J. Weston, S. Barnhill, & V. Vapnik. "Gene Selection for Cancer Classification using Support Vector Machines." *Machine Learning*. Vol. 46. No. 1. pp. 389–422. 2002.

[35] J. Bedo, C. Sanderson, & A. Kowalczyk. "An Efficient Alternative to SVM Based Recursive Feature Elimination with Applications in Natural Language Processing and Bioinformatics." In *Australasian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*. pp. 170–180. Springer, Berlin, Heidelberg, 2006.

[36] M. Labani, P. Moradi, & M. Jalili. "A multi-objective genetic algorithm for text feature selection using the relative discriminative criterion." *Expert Systems With Applications*. Vol. 149. pp. 113276. Elsevier, 2020.

[37] M. E. Basiri & S. Nemati. "A novel hybrid ACO-GA algorithm for text feature selection." In *IEEE Congress on Evolutionary Computation*. pp. 2561–2568. 2009. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4983263

[38] "TF-IDF." *Wikipedia. The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[39] Y. Goldberg. *Neural Network Methods for Natural Language Processing*. Morgan & Claypool, 2017. Synthesis Lectures on Human Language Technologies.

[40] K. W. Church & P. Hanks. "Word Association Norms, Mutual Information, and Lexicography." In *27th Annual Meeting of the Association for Computational Linguistics*. pp. 76–83. Association for Computational Linguistics, Jun. 1989. URL: https://www.aclweb.org/anthology/P89-1010

[41] M. J. Kusner, Y. Sun, N. I. Kolkin, & K. Q. Weinberger. "From Word Embeddings To Document Distances." In *Proceedings of the 32nd International Conference on Machine Learning*. pp. 957–966. Jun. 2015. URL: http://proceedings.mlr.press/v37/kusnerb15.pdf

[42] A. Neelakantan, J. Shankar, A. Passos, & A. McCallum. "Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space." 2015. URL: https://arxiv.org/pdf/1504.06654.pdf

[43] T. Ruas, W. Grosky, & A. Aizawa. "Multi-sense embeddings through a word sense disambiguation process." *Expert Systems with Applications*. Vol. 136. pp. 288–303. Elsevier, 2019. URL: https://arxiv.org/pdf/2101.08700.pdf

[44] M. M. Mirończuk & J. Protasiewicz. "A recent overview of the state-of-the-art elements of text classification." *Expert Systems with Applications*. Vol. 106. pp. 36–54. Elsevier, 2018.

[45] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, & D. Brown. "Text Classification Algorithms: A Survey." *Information*. Vol. 10. No. 4. pp. 150. 2019. URL: https://doi.org/10.3390/info10040150

[46] C. Zhang, X. Wu, Z. Niu, & W. Ding. "Authorship identification from unstructured texts." *Knowledge-Based Systems*. Vol. 66. pp. 99–111. Elsevier, 2014.

[47] T. C. Jo, J. H. Seo, & H. Kim. "Topic Spotting on News Articles with Topic Repository by Controlled Indexing." In *Intelligent Data Engineering and Automated Learning. Data Mining, Financial Engineering, and Intelligent Agents*. pp. 386–391. Springer, Berlin, Heidelberg, 2000.

[48] T. Kwartler. *Text Mining in Practice with R*. John Wiley & Sons, 2017.

[49] N. Jindal & B. Liu. "Identifying comparative sentences in text documents." In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. pp. 244–251. Aug. 2006. URL: https://www.cs.uic.edu/~liub/publications/sigir06-comp.pdf

[50] T. Wilson, J. Wiebe, & P. Hoffmann. "Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis." pp. 347–354. Vancouver, British Columbia, Canada: Association for Computational Linguistics, Oct. 2005. URL: https://www.aclweb.org/anthology/H05-1044.pdf

[51] F. A. Nielsen. "A new ANEW: Evaluation of a word list for sentiment analysis in microblogs." In *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages*. pp. 93–98. 2011. URL: https://arxiv.org/pdf/1103.2903.pdf

[52] W. S. El-Kassas, C. R. Salama, A. A. Rafea, & H. K. Mohamed. "Automatic text summarization: A comprehensive survey." *Expert Systems With Applications*. Vol. 165. Elsevier, 2020.

[53] D. Jain, M. D. Borah, & A. Biswas. "Summarization of legal documents: Where are we now and the way forward." *Computer Science Review*. Vol. 40. Elsevier, 2021.

[54] J. M. Sanchez-Gomez, M. A. Vega-Rodríguez, & C. J. Pérez. "Extractive multi-document text summarization using a multi-objective artificial bee colony optimization approach." *Knowledge-Based Systems*. Vol. 159. pp. 1–8. 2017.

[55] V. Priya & K. Umamaheswari. "Enhanced continuous and discrete multi objective particle swarm optimization for text summarization." *Cluster Computing*. Vol. 22. No. 1. pp. 229–240. Springer, 2019.

[56] P. Fung, G. Ngai, & C. S. Cheung. "Combining optimal clustering and hidden Markov models for extractive summarization." In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering*. pp. 21–28. Sapporo, Japan: Association for Computational Linguistics, Jul. 2003. URL: https://www.aclweb.org/anthology/W03-1203

[57] G. Erkan & D. R. Radev. "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization." *Journal of Artificial Intelligence Research*. Vol. 22. pp. 457–479. 2004. URL: https://www.jair.org/index.php/jair/article/view/10396/24901

[58] R. Mihalcea & P. Tarau. "TextRank: Bringing Order into Texts." In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. pp. 404–411. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004. URL: https://www.aclweb.org/anthology/W04-3252

[59] K. Ganesan, C. Zhai, & J. Han. "Opinosis: A Graph Based Approach to Abstractive Summarization of Highly Redundant Opinions." In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. pp. 340–348. Beijing, China: Coling 2010 Organizing Committee, Aug. 2010. URL: https://www.aclweb.org/anthology/C10-1039

[60] S. Gupta & S. K. Gupta. "Abstractive summarization: An overview of the state of the art." *Expert Systems With Applications*. Vol. 121. pp. 49–65. Elsevier, 2019.

[61] S. Siddiqi & A. Sharan. "Keyword and Keyphrase Extraction Techniques: A Literature Review." *International Journal of Computer Applications*. Vol. 109. pp. 18–23. 2015.

[62] I. Vayansky & S. A. Kumar. "A review of topic modeling methods." *Information Systems*. Vol. 94. pp. 101582. Elsevier, 2020.

[63] C. H. Papadimitriou, P. Raghavan, H. Tamaki, & S. Vempala. "Latent Semantic Indexing: A Probabilistic Analysis." *Journal of Computer and System Sciences*. Vol. 61. pp. 217–235. Elsevier, 2000.

[64] C. Eckart & G. Young. "The approximation of one matrix by another of lower rank." *Psychometrika*. Vol. 1. No. 3. pp. 211–218. 1936.

[65] Y. Gong & X. Liu. "Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis." In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. pp. 19–25. Sep. 2001. URL: https://www.cs.bham.ac.uk/~pxt/IDA/text_summary.pdf

[66] S. Tu. "The Dirichlet-Multinomial and Dirichlet-Categorical models for Bayesian inference." *Computer Science Division, UC Berkeley*. 2014. URL: https://stephentu.github.io/writeups/dirichlet-conjugate-prior.pdf

[67] W. M. Darling. "A Theoretical and Practical Implementation Tutorial on Topic Modeling and Gibbs Sampling." In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*. pp. 642–647. Dec. 2011. URL: https://u.cs.biu.ac.il/~89-680/darling-lda.pdf

[68] L. Page, S. Brin, R. Motwani, & T. Winograd. "The PageRank Citation Ranking: Bringing Order to the Web." *Technical report, Stanford University, Stanford InfoLab, CA*. 1998.

[69] T. M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[70] "Kernel method." *Wikipedia. The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Kernel_method

[71] T. Joachims. "Text Categorization with Support Vector Machines: Learning with Many Relevant Features." In *Proceedings of the European Conference on Machine Learning,*. pp. 137–142. Springer, Berlin, Heidelberg, Apr. 1998. URL: https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf

[72] X. Li, S. Qing, H. Zhang, T. Wang, & H. Yang. "Kernel methods for word sense disambiguation." *Artificial Intelligence Review*. Vol. 46. No. 1. pp. 41–58. 2016.

[73] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, & C. Watkins. "Text Classification using String Kernels." *Journal of Machine Learning Research*. Vol. 2. pp. 419–444. Feb. 2002. URL: https://www.jmlr.org/papers/volume2/lodhi02a/lodhi02a.pdf

[74] D. Haussler. "Convolution kernels on discrete structures." *UCSC-CRL-99-10. Technical report, Department of Computer Science, University of California at Santa Cruz*. 1999. URL: https://www.soe.ucsc.edu/sites/default/files/technical-reports/UCSC-CRL-99-10.pdf

[75] C. Watkins. "Dynamic Alignment Kernels." *Advances in Neural Information Processing Systems*. pp. 39–50. MIT, 1999. URL: https://www.researchgate.net/publication/2271137_Dynamic_Alignment_Kernels

[76] N. Cancedda, E. Gaussier, C. Goutte, & J. M. Renders. "Word-Sequence Kernels." *Journal of Machine Learning Research*. Vol. 3. pp. 1059–1082. 2003. URL: https://www.jmlr.org/papers/volume3/cancedda03a/cancedda03a.pdf

[77] B. E. Boser, I. M. Guyon, & V. N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers." In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. D. Haussler, ED. pp. 144–152. ACM Press, Jul. 1992.

[78] C. Cortes & V. Vapnik. "Support Vector Networks." *Machine Learning*. Vol. 20. No. 3. pp. 273–297. 1995.

[79] "Support Vector Machine." Wikipedia. The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Support-vector_machine

[80] A. Ben-Hur, D. Horn, H. T. Siegelmann, & V. Vapnik. "Support Vector Clustering." *Journal of Machine Learning Research*. Vol. 2. pp. 125–137. Dec. 2001. URL: https://www.jmlr.org/papers/volume2/horn01a/horn01a.pdf

[81] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, & V. Vapoik. "Support Vector Regression Machines." *Advances in Neural Information Processing Systems*. Vol. 9. pp. 155–161. MIT Press, 1997. URL: https://proceedings.neurips.cc/paper/1996/file/d38901788c533e8286cb6400b40b386d-Paper.pdf

[82] C. W. Hsu & C. J. Lin. "A Comparison of Methods for Multiclass Support Vector Machines." *IEEE Transactions on Neural Networks*. Vol. 13. No. 2. pp. 415–425. 2002.

[83] J. C. Platt. "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods." *Advances in large margin classifiers*. Vol. 10. No. 3. pp. 61–74. 1999. URL: https://home.cs.colorado.edu/~mozer/Teaching/syllabi/6622/papers/Platt1999.pdf

[84] A. Vaswani, Y. Zhao, V. Fossum, & D. Chiang. "Decoding with Large-Scale Neural Language Models Improves Translation." In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pp. 1387–1392. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013. URL: https://www.aclweb.org/anthology/D13-1140

[85] M. Henderson, B. Thomson, & S. Young. "Deep Neural Network Approach for the Dialog State Tracking Challenge." In *Proceedings of the SIGDIAL 2013 Conference*. pp. 467–471. Metz, France: Association for Computational Linguistics, Aug. 2013. URL: https://www.aclweb.org/anthology/W13-4073

[86] M. Lewis & M. Steedman. "Improved CCG Parsing with Semi-supervised Supertagging." *Transactions of the Association for Computational Linguistics*. Vol. 2. pp. 327–338. 2014. URL: https://www.aclweb.org/anthology/Q14-1026

[87] W. Yin & H. Schütze. "Convolutional Neural Network for Paraphrase Identification." In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pp. 901–911. Denver, Colorado: Association for Computational Linguistics, May 2015. URL: https://www.aclweb.org/anthology/N15-1091

[88] N. Kalchbrenner, E. Grefenstette, & P. Blunsom. "A Convolutional Neural Network for Modelling Sentences." In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pp. 655–665. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014. URL: https://www.aclweb.org/anthology/P14-1062

[89] T. H. Nguyen & R. Grishman. "Event Detection and Domain Adaptation with Convolutional Neural Networks." In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Vol. 2. pp. 365–371. Beijing,

China: Association for Computational Linguistics, Jul. 2015. URL: https://www.aclweb.org/anthology/P15-2060

[90] M. Sundermeyer, T. Alkhouli, J. Wuebker, & H. Ney. "Translation modeling with bidirectional recurrent neural networks." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pp. 14–25. Doha, Qatar: Association for Computational Linguistics, Oct. 2014. URL: https://www.aclweb.org/anthology/D14-1003

[91] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J. Y. Nie, J. Gao, & B. Dolan. "A Neural Network Approach to Context-Sensitive Generation of Conversational Responses." In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. pp. 196–205. Denver, Colorado: Association for Computational Linguistics, May–Jun. 2015. URL: https://www.aclweb.org/anthology/N15-1020

[92] C. Becker, N. Hahn, B. He, H. Jabbar, M. Plesiak, V. Szabo, X.-Y. To, R. Yang, & J. Wagner. "Modern Approaches in Natural Language Processing." 2020. URL: https://compstat-lmu.github.io/seminar_nlp_ss20/

[93] M. T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, & L. Kaiser. "Multi-task Sequence to Sequence Learning." In *Proceeding of the 4th International Conference on Learning Representations (ICLR)*. San Juan, Puerto Rico, May 2016. URL: https://arxiv.org/pdf/1511.06114.pdf

[94] S. Hochreiter & J. Schmidhuber. "Long Short-Term Memory." *Neural Computation*. Vol. 9. No. 8. pp. 1735–1780. 1997.

[95] F. A. Gers, J. Schmidhuber, & F. A. Cummins. "Learning to Forget: Continual Prediction with LSTM." *Neural Computation*. Vol. 12. No. 10. pp. 2451–2471. 2000.

[96] C. Olah. "Understanding LSTM Networks." URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[97] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, Y. Bengio, & D. Bahdanau. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pp. 1724–1734. Doha, Qatar: Association for Computational Linguistics, Oct. 2014. URL: https://www.aclweb.org/anthology/D14-1179

[98] I. Sutskever, O. Vinyals, & Q. V. Le. "Sequence to Sequence Learning with Neural Networks." Vol. 27. pp. 3104–3112. Curran Associates, Inc., Dec. 2014. URL: https://papers.nips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

[99] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, & N. Jaitly. "A Comparison of Sequence-to-Sequence Models for Speech Recognition." pp. 939–943. 2017. URL: https://www.isca-speech.org/archive/Interspeech_2017/pdfs/0233.PDF

[100] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, & K. Saenko. "Sequence to Sequence-Video to Text." In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Vol. 1. pp. 4534–4542. 2015. URL: https://openaccess.thecvf.com/content_iccv_2015/papers/Venugopalan_Sequence_to_Sequence_ICCV_2015_paper.pdf

[101] S. Chen, T. Yao, & Y.-G. Jiang. "Deep Learning for Video Captioning: A Review." In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-19)*. pp. 6283–6290. International Joint Conferences on Artificial Intelligence Organization, Jul. 2019. URL: https://www.ijcai.org/proceedings/2019/0877.pdf

[102] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, & Y. Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pp. 1724–1734. Doha, Qatar: Association for Computational Linguistics, Oct. 2014. URL: https://www.aclweb.org/anthology/D14-1179

[103] Y. Kim. "Convolutional Neural Networks for Sentence Classification." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pp. 1746–1751. Doha, Qatar: Association for Computational Linguistics, Oct. 2014. URL: https://www.aclweb.org/anthology/D14-1181.pdf

[104] X. Zhang, J. Zhao, & Y. LeCun. "Character-level Convolutional Networks for Text Classification." In *Proceedings of the 29th Conferece on Advances in Neural Information Processing Systems (NIPS)*. Vol. 28. pp. 649–657. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf

[105] R. Johnson & T. Zhang. "Deep Pyramid Convolutional Neural Networks for Text Categorization." In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vol. 1. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017. URL: https://www.aclweb.org/anthology/P17-1052

[106] A. Conneau, H. Schwenk, L. Barrault, & Y. Lecun. "Very Deep Convolutional Networks for Text Classification." In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Vol. 1. pp. 1107–1116. Valencia, Spain: Association for Computational Linguistics, Apr. 2017. URL: https://www.aclweb.org/anthology/E17-1104

[107] Y. Bengio, R. Ducharme, C. Jauvin, & P. Vincent. "A Neural Probabilistic Language Model." *Journal of Machine Learning Research*. Vol. 3. pp. 1137–1155. 2003. URL: https://jmlr.org/papers/volume3/bengio03a/bengio03a.pdf

[108] R. Collobert & J. Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning." In *Proceedings of the 25th International Conference on Machine Learning*. pp. 160–167. 2008.

[109] Y. Bengio, J. Louradour, R. Collobert, & J. Weston. "Curriculum learning." In *Proceeding of the 26th Annual International Conference on Machine Learning*. pp. 41–48. 2009.

[110] J. Pennington, R. Socher, & C. Manning. "GloVe: Global Vectors for Word Representation." pp. 1532–1543. Doha, Qatar: Association for Computational Linguistics, Oct. 2014. URL: https://www.aclweb.org/anthology/D14-1162

[111] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2021. URL: https://www.R-project.org/

[112] D. J. Hand & R. J. Till. "A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems." *Machine Learning*. Vol. 45. No. 2. pp. 171–186. 2001. URL: https://link.springer.com/content/pdf/10.1023/A:1010920819831.pdf

[113] T. L. Griffiths & M. Steyvers. "Finding Scientific Topics." In *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*. Vol. 101. No. suppl 1. pp. 5228–5235. National Academy of Sciences, Apr. 2004. URL: https://www.pnas.org/content/101/suppl_1/5228