UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Clasificación de textos basada en redes neuronales

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Mario Campos Mocholí

Tutores: Encarnación Segarra Soriano
Lluís Felip Hurtado Oliver
Emilio Sanchis Arnal

Curso 2020-2021

Resum

Cada vegada són més les empreses que opten per utilitzar la intel·ligència artificial per a automatitzar tasques dins de les organitzacions, permetent així reassignar al personal dedicat a aquestes i aconseguint una major productivitat. És el cas de la Corporació Valenciana de Mitjans de Comunicació, la qual ha plantejat la necessitat d'aconseguir un sistema d'ajuda a l'equip de documentació i catalogació.

La finalitat d'aquest projecte és determinar si és possible el desenvolupament d'un sistema d'aprenentatge automàtic el qual siga capaç de fer aquesta tasca de manera automàtica o bé proporcionar ajuda a l'equip de documentació i catalogació en la seua tasca, suggerint-li un llistat de categories ordenat segons la seua probabilitat. Per abordar el problema d'automatitzar el més possible la tasca de classificació en el context d'Apunt, s'ha triat treballar en l'àrea dels noticiaris. La corporació ens ha proporcionat les dades d'aquest sistema, el qual han estat elaborant des de fa quasi dues dècades.

L'objectiu principal d'aquest treball fi de grau és abordar un problema de classificació multiclasse, emmarcat dins del camp de la intel·ligència artificial del processament de llenguatge natural, aplicat a la tasca de la catalogació de noticiaris en català. En el projecte es realitza i analitza un *corpus* únic, utilitzat per a entrenar diferents classificadors basats en aprenentatge automàtic, entre ells un basat en xarxes neuronals, per a realitzar una solució software d'ajuda a la classificació per a la Corporació.

Paraules clau: classificació automàtica, processament de llenguatge natural, xarxes neuronals, textos periodístics en català

Resumen

Cada vez son más las empresas que optan para utilizar la inteligencia artificial para automatizar tareas dentro de las organizaciones, permitiendo así reasignar al personal dedicado a estas y consiguiendo una mayor productividad. Es el caso de la Corporació Valenciana de Mitjans de Comunicació, la cual ha planteado la necesidad de conseguir un sistema de ayuda al equipo de documentación y catalogación.

La finalidad de este proyecto es determinar si es posible el desarrollo de un sistema de aprendizaje automático el cual sea capaz de hacer esta tarea de manera automática o bien proporcionar ayuda al equipo de documentación y catalogación en su tarea, sugiriéndole un listado de categorías ordenado según su probabilidad. Para abordar el problema de automatizar el más posible la tarea de clasificación en el contexto de Apunte, se ha elegido trabajar en el área de los noticieros. La corporación nos ha proporcionado los datos de este sistema, el cual han estado elaborando desde hace casi dos décadas.

El objetivo principal de este trabajo final de grado es abordar un problema de clasificación multiclase, enmarcado dentro del campo de la inteligencia artificial del procesamiento de lenguaje natural, aplicado a la tarea de la catalogación de noticieros en catalán. En el proyecto se realiza y analiza un *corpus* único, utilizado para entrenar diferentes clasificadores basados en aprendizaje automático, entre ellos un basado en redes neuronales, para realizar una solución software de ayuda a la clasificación para la Corporación.

Palabras clave: clasificación automática, procesamiento de lenguaje natural, redes neuronales, textos periodísticos en catalán

Abstract

More and more companies are choosing to use artificial intelligence to automate tasks within organizations, allowing the reassignment of dedicated staff and achieving greater productivity. This is the case of the Corporació Valenciana de Mitjans de Comunicació, which has raised the need to get a system of help to the documentation and cataloguing team.

The purpose of this project is to determine whether it is possible to develop a machine learning system which is capable of doing this task automatically or to provide assistance to the documentation and cataloguing team in their task, suggesting a list of categories ordered according to their probability. To address the problem of automating the classification task in the context of Àpunt, we have chosen to work in the news area. The corporation has provided us with the data from this system, which they have been developing for almost two decades.

The main objective of this final degree work is to address a problem of multiclass classification, framed within the field of artificial intelligence of natural language processing, applied to the task of cataloguing news in Catalan. The project performs and analyses a unique *corpus* used to train different classifiers based on machine learning, including a neural network-based, to perform a software solution to help the classification for the Corporation.

Key words: automatic classification, natural language processing, neural networks, journalistic texts in Catalan

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En el primer capítulo de la memoria se introduce a la temática del proyecto que consiste en el análisis de un conjunto de datos para averiguar si es posible la realización de un sistema de etiquetado automático para el sistema interno de catalogación de la Corporació Valenciana de Mitjans de Comunicació mediante técnicas de *machine learning* orientadas la clasificación.

Así mismo, se expone la motivación para desarrollar el proyecto, así como los objetivos y el impacto que se espera lograr. Además, se relaciona el proyecto con los objetivos y metas de desarrollo sostenible de la ONU, así como con las asignaturas cursadas durante la carrera y que han sido de vital importancia para el desarrollo de toda la experimentación. Finalmente, se detalla la estructura del proyecto y el alcance de cada capítulo de la memoria.

1.1 Motivación

Cada día es más común que las empresas opten por la digitalización y utilización de las nuevas tecnologías para llevar a cabo tareas rutinarias en la organización de forma automática o más amena para los trabajadores. Es el caso de la Corporació Valenciana de Mitjans de Comunicació la cual ha solicitado un estudio para ver si es posible realizar de forma automática la clasificación de sus noticiarios en su sistema interno de documentación y catalogación.

El procesamiento de lenguaje natural [1], o *natural language processing*, es la rama de la computación y de la inteligencia artificial encargada de dotar a los ordenadores de la capacidad de entender el lenguaje humano tal como lo haría una persona. Algunas de las tareas más comunes de este campo comprenden el procesamiento del habla, el análisis de sentimientos, la generación automática de texto o el reconocimiento de entidades nombradas.

Así pues, el proyecto a desarrollar se embarca en este ámbito ya que su finalidad es la experimentación con un conjunto de textos en lengua catalana para determinar si es posible la realización de un producto final de *machine learning* para llevar a cabo la tarea descrita de forma autónoma o, en su defecto, un sistema de apoyo para el personal encargado de esta tarea.

Durante el proyecto se llevarán a cabo diversos procesamientos sobre los datos para construir un *corpus* ya que la información proporcionada se encuentra de forma desestructurada e imposible de tratar por un modelo de aprendizaje automático. Posteriormente se realizan análisis y extraen estadísticas de este conjunto resultante para el pos-

terior entrenamiento de diversos modelos. Finalmente, se presentan los análisis de los resultados obtenidos y la conclusión final.

1.2 Objetivos

El objetivo de este trabajo final de grado es tratar un conjunto desestructurado de datos procedentes de noticias en catalán para elaborar un *corpus* con el que entrenar diversos modelos de clasificación con diferentes representaciones del lenguaje para finalmente realizar una comparativa entre ellos.

La dificultad de este problema recae en que las técnicas convencionales de procesamiento de lenguaje natural están orientadas a textos en lenguas mayoritarias como el inglés o el español y no se cuenta con librerías para la eliminación de *stopwords* ni la realización de *stemming* ni *lemmatization* para el catalán.

El objetivo global esperado de la realización de este proyecto es el desarrollo de una herramienta que sea capaz de ayudar al equipo de catalogación de la Corporació Valenciana de Mitjans de Comunicació en su labor de etiquetado y clasificación de noticias en catalán, así como aportar más conocimientos a la corporación sobre la naturaleza y distribución de sus datos.

1.3 Objetivos ODS

Los objetivos y metas de desarrollo sostenible [2], o ODS, son una total de 17 metas globales, reflejados en la Figura 1.1, adoptadas en septiembre de 2015 por los líderes mundiales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todo el mundo como parte de una agenda de desarrollo sostenible. Cada objetivo tiene unas metas específicas que deben alcanzarse en el año 2030. El alcance de estas metas depende tanto de los gobiernos y de las empresas, así como de la responsabilidad individual de cada uno.

Figura 1.1: Objetivos ODS de la ONU

La realización de este proyecto está estrechamente relacionada con el objetivo número 4 titulado "Educación de calidad" ya que la Corporació Valenciana de Mitjans de Comunicació tiene una importante función didáctica en la sociedad valenciana y no es posible una educación de calidad si no se abarcan todos los aspectos y dimensiones de la cultura autóctona de la región donde se realiza.

Otro objetivo ODS con el cual el proyecto está relacionado es el 10 titulado "Reducción de las desigualdades". La lengua catalana se encuentra actualmente en situación de diglosia con la lengua castellana, siendo la primera la perjudicada en esta situación. La elaboración de cualquier proyecto relacionado con la lengua, por pequeño que sea, ayudan a fortalecerla. Además, cabe recordar la gran función de promoción y normalización que realiza la corporación y que, gracias a un sistema como el que se pretende elaborar en este proyecto, se está ayudando de forma indirecta a esta tarea.

1.4 Asignaturas relacionadas

Durante el desarrollo de la memoria se han documentado varios conceptos teóricos basándose en los temarios y conocimientos impartidos por el profesorado durante el grado, sobre todo de las asignaturas pertenecientes a la mención en Computación. Además, de forma indirecta o directa, la gran mayoría de asignaturas han influido en la realización del proyecto, bien por la utilización de alguna tecnología ya impartida, o bien como teoría base para entender y desarrollar conceptos más complejos.

En primer lugar se encuentran las asignaturas relacionadas con el *machine learning* las cuales han aportado tanto las bases teóricas como prácticas para la comprensión del campo de la inteligencia artificial, así como de la realización de modelos. Estás asignaturas son "Sistemas Inteligentes" y "Percepción", pertenecientes a tercer curso, y "Aprendizaje Automático", correspondiente a cuarto curso. En ellas se ha obtenido los conocimientos necesarios para comprender el mundo de la inteligencia artificial partiendo de bases teóricas y matemáticas.

Otra asignatura directamente relacionada con el proyecto e impartida en tercero es "Sistema de Almacenamiento y Recuperación de la Información" materia que sirve como introducción al procesamiento de lenguaje natural, a la representación de la información y al tratamiento de datos. En la parte de prácticas de la asignatura se han afianzado dichos conceptos desarrollándolos en el lenguaje de programación Python, el cual es el utilizado para el desarrollo de la experimentación del proyecto.

Como mención especial cabe nombrar a las asignaturas de "Agentes Inteligentes" y "Técnicas, Aplicaciones y Entornos de Inteligencia Artificial", las cuales han aportado una visión más general del campo y a su vez conceptos útiles para la realización del proyecto.

1.5 Estructura de la memoria

La memoria del proyecto se divide en un total de siete capítulos, incluido el capítulo actual. En esta sección se realizará una introducción de cada uno para tener una visión global del proyecto, así como los puntos tratados en cada uno.

- **Capítulo 1, Introducción:** en este primer capítulo se introduce la temática general del proyecto focalizada en la tarea a desarrollar para la organización. Así mismo, se presenta la motivación para la realización del proyecto, así como los objetivos a

tratar y el objetivo esperado que tenga. Adicionalmente, se comentan los objetivos ODS relacionados, así como las asignaturas relacionadas. Finalmente, se describe la estructura propia del proyecto.

- **Capítulo 2, Modelos de aprendizaje automático para la clasificación y su evaluación:** en el segundo capítulo se introduce al *machine learning* y algunos conceptos clave de esta rama de la computación. Posteriormente, se describen teóricamente diversos modelos de aprendizaje automático utilizados en el proyecto, así como las implementaciones utilizadas de cada uno. Finalmente, se presentan diversas métricas utilizadas en el aprendizaje automático y diferentes técnicas para estimar el error producido por los modelos.

- **Capítulo 3, Representación numérica del lenguaje:** en el capítulo se tratan las diferentes representaciones numéricas de los textos producidas por diferentes modelos de representación del lenguaje. Se incluyen representaciones básicas que introducen técnicas utilizadas de forma común por diversos modelos y, posteriormente, se desarrollan modelos capaces de capturar el contexto y semántica de las oraciones. Por último, se introducen diversos modelos basados en redes neuronales que representan mediante *embeddings* las relaciones del lenguaje así como la implementación *FastText* utilizada en el proyecto.

- **Capítulo 4, *Corpus* CVMC y preprocesamiento:** en el cuarto capítulo se expone la procedencia del conjunto de datos utilizado para elaborar el *corpus* y la justificación de cuáles de estos han sido utilizados o no. Además, se describen las diversas técnicas utilizadas y filtros realizados para procesar los datos, obteniendo así el *corpus* final que será utilizado por los modelos de aprendizaje automático. Finalmente, se presente diversas métricas extraídas del conjunto de datos que aportan información sobre la estructura de estos.

- **Capítulo 5, Dominio del problema, análisis de resultados y aplicaciones:** en el penúltimo capítulo se comienza mostrando el entorno de trabajo en el cual se ha realizado la experimentación y las librerías utilizadas a lo largo de todo el proyecto, así como las que se tenía esperado utilizar y el porqué de su no utilización. Posteriormente, se da una visión global de los problemas de clasificación multiclase para seguidamente mostrar las estadísticas obtenidas para cada modelo y configuración, así como un análisis de estos resultados. Finalmente, se propone una aplicación de los modelos desarrollados para la CVMC fruto del anterior análisis.

- **Capítulo 6, Conclusiones y trabajo futuro:** en el capítulo final se hace una recapitulación del proyecto completo aportando las conclusiones finales sobre las tareas realizadas. Finalmente, se presentan varias directrices para seguir desarrollando el proyecto, así como ideas propias.

CAPÍTULO 2

Modelos de aprendizaje automático para la clasificación y su evaluación

En el capítulo actual se realiza una revisión general al área del aprendizaje automático donde se definen los conceptos básicos, así como las distintos categorías y tareas para el que está orientado.

Así mismo, se detalla el funcionamiento y los fundamentos de diversos modelos utilizados en el desarrollo del proyecto, con especial énfasis en sus variantes dedicadas a la clasificación automática, que es el problema que se aborda.

Finalmente, se realiza una definición, así como su justificación, de las métricas utilizadas en estos problemas y cuáles han sido seleccionadas para evaluar los modelos realizados.

2.1 Introducción al *machine learning*

El *machine learning*, o aprendizaje automático, es la rama de la computación e inteligencia artificial encargada de desarrollar técnicas y algoritmos capaces de aprender a realizar una o más tareas concretas, a partir de un conjunto de datos, sin haber sido explícitamente programados para ello. El objetivo final es tener un modelo, o función, capaz de generalizar correctamente estos datos, es decir, predecir la salida de nuevos valores que no haya visto.

Formalmente, se dispone de un conjunto de datos, llamados de aprendizaje o entrenamiento, de entrada $x \in \mathcal{X}$ y de salida $y \in \mathcal{Y}$. El objetivo es obtener un modelo, o función $f : \mathcal{X} \to \mathcal{Y}$, capaz de generalizar estos de forma correcta. Estos modelos deben ser capaces de generalizar el problema, es decir, no especializarse en el subconjunto de datos visto, sino encontrar el patrón general para el dominio completo del problema.

Según la información disponible de los datos se pueden discernir dos tipos de aprendizaje:

- **Aprendizaje supervisado**: Se cuenta tanto con los datos de entrada $x \in \mathcal{X}$ como los de salida $y \in \mathcal{Y}$ y se busca el modelo matemático que generalice f.
- **Aprendizaje no supervisado**: Únicamente se cuenta con los datos de entrada $x \in \mathcal{X}$ y la finalidad es encontrar información sobre el modelo de salida $\mathcal{Y}$. En la Figura 2.1 se observa una aproximación visual de esta tarea mediante el llamado *clustering*.

Figura 2.1: Representación gráfica de aprendizaje no supervisado [3]

Los modelos de aprendizaje automático se utilizan principalmente para realizar dos tareas concretas:

- **Clasificación**: a partir de un conjunto arbitrario de entrada x, el modelo clasifica los datos en un conjunto finito de salida $\mathcal{Y}$ acotado en un número determinado de clases $\mathcal{C}$.
- **Regresión**: tanto el conjunto de entrada x como el de salida $\mathcal{Y}$ son arbitrarios y el modelo infiere una función para predecir en el dominio de valores de la salida. En la Figura 2.2 se aprecia un ejemplo concreto de regresión lineal.

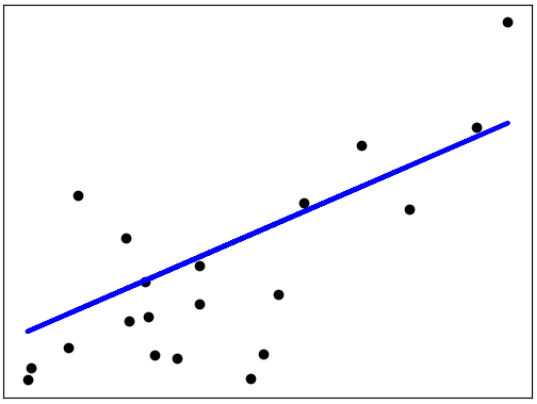

Figura 2.2: Representación gráfica de regresión lineal [4]

2.1.1. El balance *bias-variance*

El objetivo de cualquier método de aprendizaje supervisado es encontrar el mejor estimador de la función de predicción f para el dominio de salida $\mathcal{Y}$ dado el conjunto de entrada x. El error de predicción producido por esta función se puede dividir en tres tipos [5]:

- El error irreducible.
- El error de *bias*, o sesgo.
- El error de *variance*, o varianza.

El primero de todos, el error irreducible, es un error intrínseco del problema y no puede ser reducido de ninguna manera, ni con diferentes algoritmos ni con diversas configuraciones de estos. Puede ser producido tanto por la representación de los datos elegida como la influencia de variables ocultas que no se han conseguido modelar o se desconocen.

Por otra parte, el error de *bias*, es producido por el modelo durante el entrenamiento, el cual ha realizado asunciones erróneas con el fin de aprender más rápidamente la función de predicción. Los modelos con un bajo sesgo realizan menos asunciones a priori sobre la función mientras que los modelos con alto sesgo las realizan con más frecuencia. Normalmente, los algoritmos lineales cuentan con un gran *bias*, haciéndolos más rápidos en entrenamiento y de interpretar, pero a su vez menos flexibles a nuevos datos, por lo que su rendimiento en problemas complejos se reduce considerablemente.

Por último, el error de *variance*. La varianza refleja en qué medida cambia la función de un modelo si se utilizan distintos conjuntos de entrenamiento. Idealmente, un modelo debería tener algo de varianza, ya que debe ser capaz de adaptarse a los cambios de datos, pero a su vez no debe tener demasiado ya que, en tal caso, la función de predicción cambiaría totalmente al entrenar al modelo con diferentes subconjuntos de datos. Generalmente, los modelos no lineales, tienen una gran varianza.

El objetivo de un bueno modelo de aprendizaje automático es conseguir un bajo *bias* y a su vez una baja *variance*, pero reducir la varianza aumenta el sesgo y por su parte, reducir el sesgo aumentará la varianza, por lo que se debe buscar un balance [5] entre ambos errores para conseguir desarrollar un modelo capaz de aproximar lo máximo posible la función al dominio real de los datos. Esto se logrará eligiendo el algoritmo correcto para el problema concreto, así como su parametrización.

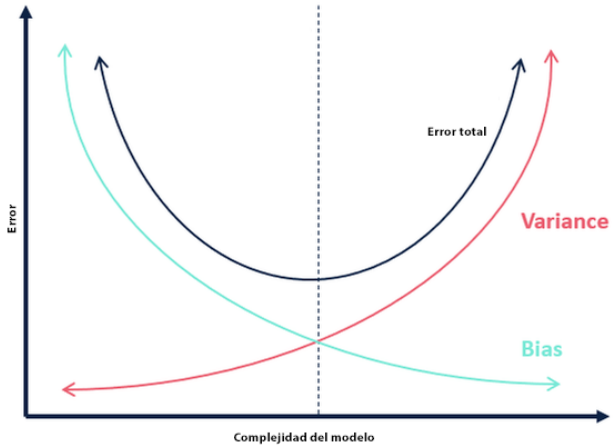

Figura 2.3: Curvas de error *bias-variance*

Un mal ajuste de este balance puede llegar a producir dos sucesos:

- *Over-fitting*: ocurre cuando un modelo ha aprendido con demasiado detalle el patrón del conjunto de entrenamiento, modelando incluso el ruido, por lo que su rendimiento bajara al ver nuevos datos.
- *Under-fitting*: es el suceso contrario. Cuando el modelo no es capaz de aprender el patrón del conjunto de entrenamiento ni generalizarlo para nuevos datos.

En conclusión, encontrar este balance es vital para poder desarrollar un modelo de *machine learning* capaz de llevar a cabo una tarea real con la mayor eficacia pero que a su

vez no esté aportando unos resultados reales sobre el dominio completo del sistema. En secciones posteriores del capítulo se utilizarán los términos descritos para definir correctamente los modelos, así como sus puntos fuertes y débiles.

2.2 *Naive Bayes*

Los modelos de aprendizaje automático del tipo *Naive Bayes* son un grupo de algoritmos extremadamente rápidos, pero a la vez simples, orientados en la tarea de clasificación, sobre todo para conjunto de datos con una alta dimensionalidad. Por su simplicidad y bajo número de parámetros, son modelos utilizados principalmente para prototipado o como una solución básica al problema de clasificación.

Estos clasificadores se basan en el teorema de Bayes cuya ecuación describe la relación de probabilidad condicional de un evento aleatoria A dado B relacionado con la distribución de probabilidad condicional del evento B dado A y la distribución de probabilidad de A. En la clasificación Bayesiana estamos interesados en encontrar la probabilidad de una clase a partir de la observación de sus características. Podemos definir está relación como la Ecuación 2.1.

$$P(A \mid \text{características}) = \frac{P(\text{características} \mid A) \cdot P(A)}{P(\text{características})} \tag{2.1}$$

En el caso particular de la clasificación binaria con dos clasesC_1 y C_2, una posible aproximación sería calcular el ratio de la probabilidad a *posteriori* de cada muestra, tal como se define en la Ecuación 2.2.

$$\frac{P(C_1 \mid \text{características})}{P(C_2 \mid \text{características})} = \frac{P(\text{características} \mid C_1) \cdot P(C_1)}{P(\text{características} \mid C_2) \cdot P(C_2)} \tag{2.2}$$

Para conseguir resolver la ecuación es necesario contar con un modelo que se capaz de calcular $P(caracteristicas \mid C_x)$ para cada muestra. En estadística, este modelo se llama modelo generador ya que especifica el suceso aleatorio por el cual se generan los datos. Definir un modelo de este tipo es una tarea complicada por lo que, para simplificar la tarea, se realizan asunciones previas al entrenamiento sobre la distribución generadora del modelo. Según la asunción que se haga, el modelo se llamara de una forma u otra.

En el proyecto se ha utilizado la implementación de *Scikit-learn*, concretamente el modelo *GaussianNB* [6] cuya asunción *naive* es que los datos de cada clase han sido generados por una distribución gaussiana, o normal.

2.3 Máquinas de vectores soporte

Las máquinas de vectores soporte, o *Support Vector Machine*, son un conjunto de algoritmos de aprendizaje supervisado utilizados en clasificación, regresión y detección de valores atípicos.

El fundamento de la SVM reside en seleccionar un hiperplano de separación que equidiste de las muestras más próximas de cada clase 2.4, logrando así el llamado margen máximo respecto a los hiperplanos. Para definir dicho hiperplano únicamente se tiene en cuenta las muestras de cada clase que residen justamente en la frontera de los márgenes, obteniendo así los llamados vectores soporte. Así pues, el clasificador de máximo

margen queda definido por la función discriminante lineal 2.3 donde θ^* y θ_0^* son los parámetros a optimizar. Por tanto, este tipo problema se enmarca en los llamados problemas de optimización.

$$\phi(x;\Theta) = \theta^{*^t} \cdot x + \theta_0^* \tag{2.3}$$

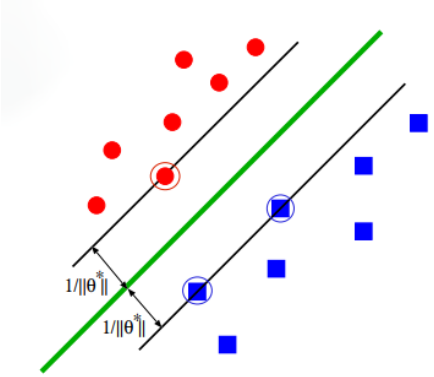

Figura 2.4: Representación gráfica del hiperplano separador de una SVM con $\mathcal{C} = 2$ [7]

Ahora bien, según el problema, no siempre es posible encontrar dicha función discriminante lineal por lo que hay que recurrir a implementaciones alternativas. Una de ellas es los *kernels* que permiten escalar el problema en un espacio de representación alternativo permitiendo así definir funciones discriminantes más complejas sin tener que modificar la dimensionalidad de los datos de entrada.

En *machine learning*, los *kernels* son algoritmos que permiten operar en espacios de representación de grandes dimensiones sin tener que calcular las coordenadas de todas las muestras en dicho espacio, sino calculado el espacio de producto interior, o espacio prehilbertiano, entre el rango de cada par de datos, operación considerablemente más rápida.

Las principales ventajas de las SVM son su alta efectividad en espacios de representación grandes, su robustez cuando la dimensionalidad es mayor que el número de muestras, su eficiencia en memoria gracias a los vectores soporte y su versatilidad, ya que aceptan distintos tipos de *kernels*.

Las SVM están concebidas para resolver problemas de clasificación binaria. La solución más utilizada para extender su funcionamiento al dominio de la clasificación multiclase es dividir el problema en múltiples problemas de clasificación binaria y posteriormente abordarlo mediante clasificación *one-versus-all*, o uno contra todos, o mediante *one-versus-one*, o uno contra uno. En el primer caso se construye un clasificador por clase que distinga entre esta y todo el resto, eligiendo la clase del clasificador cuyo valor de la salida sea mayor. Para el segundo caso se construyen $N * (N-1)/2$ clasificadores, uno por cada par de clases, y se selecciona la clase a la que más clasificadores hayan votado. En la Figura 2.5 se puede observar una representación visual de este último caso.

La implementación de la SVM utilizada en el proyecto es la ofrecida por *Scikit-learn*. Dentro de esta librería existen varias implementaciones de la cuales se ha optado por utilizar *SGDClassifier* [8] ya que es indicada para conjuntos de datos elevados. Su implementación se basa en aprendizaje por descenso de gradiente estocástica, o SGD, de ahí su nombre y, además, aborda el problema de clasificación multiclase con la técnica *one-versus-all* descrita anteriormente.

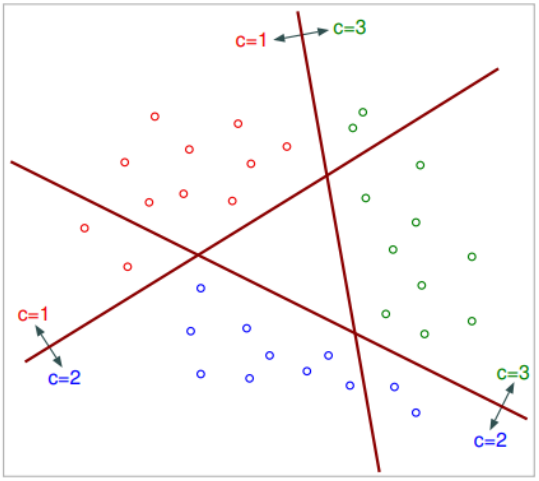

Figura 2.5: Representación gráfica de una SVM en clasificación multiclase con $\mathcal{C} = 3$ [7]

2.4 *Decision Trees* y *Random Forests*

Los árboles de decisión, o Decision Trees, son un modelo de árbol, normalmente binario, donde cada nodo representa una característica del conjunto de datos, las hojas representan las clases del problema y cada rama es el conjunto de decisiones, respecto a las características, que lleva a escoger una clase u otra.

Son un modelo base muy utilizado en métodos de *ensemble*, o agrupación. Los llamados modelos fuertes, o *strong learners*, están compuestos por diversos modelos débiles, o *weak learners*, y en el caso de que la composición sea por árboles, se habla de bosques. Estos árboles pueden ser poco profundos, reduciendo la *variance* pero aumentando el *bias*, o profundos, que por el contrario tienen bajo *bias* pero gran varianza, por lo que son los más utilizados para la implementación de *strong learners* ya que, el principal objetivo de estos, es reducir la varianza.

Los llamados *Random Forests* son una serie de técnicas de *bagging* donde los árboles profundos, repartidos en agregaciones de *bootstrap*, se combinan para producir una única salida con baja varianza. Además, para lograr una menor correlación entre los árboles, en las agrupaciones no solo se utilizan las muestras, sino que las características de las muestras, por lo que, a la hora de construir cada árbol, este solo tiene un subconjunto aleatorio del conjunto de entrenamiento y de las características.

Mediante este proceso se consigue que no todos los árboles tengan acceso a la misma información a la hora de hacer una predicción por lo que reduce la correlación entre cada salida predicha. Además, esto aumenta la robustez del modelo ya que es menos sensible a la perdida de información. Finalmente, para generar una solución única, se toma la predicción de cada árbol y mediante técnicas de consenso o estadísticas, como se puede observar en la Figura 2.6, se generar la salida final.

La implementación que se ha utilizado corresponde a la ofrecida por la librería *Scikit-learn* y, en concreto, la implementación *RandomForest*, [9] la cual utiliza el promedio de las soluciones particulares para generar una solución final para aumentar la precisión y controlar el *over-fitting*.

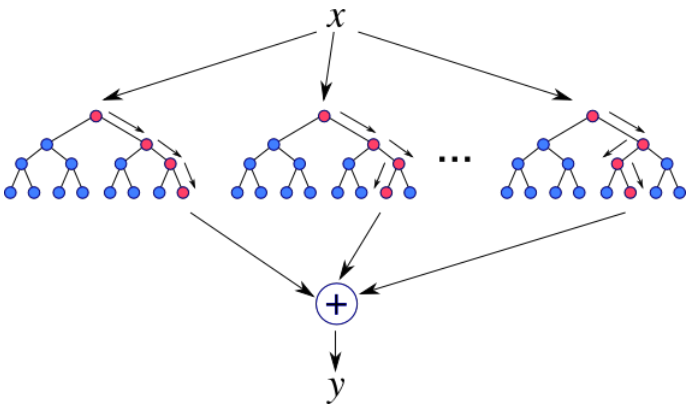

Figura 2.6: Representación gráfica de un *Random Forest*

2.5 Redes neuronales

Las redes neuronales artificiales se inspiran en el cerebro humano ya que este es un procesador de información natural que cuenta con características únicas que nos permite realizar determinadas tareas de forma automática, como pude ser la visión, el reconocimiento del habla o el aprendizaje como tal. Conseguir desentrañar los misterios de cómo funciona realmente permitiría realizar tareas muy complejas, típicamente únicamente humanos, por máquinas.

Aun así, el cerebro es bastante diferente a cualquier ordenador ya que estos están compuestos, normalmente, por un único procesador mientras que el cerebro humano cuenta con más de mil neuronas, todas ellas trabajando en paralelo. Pese a ello, una neurona es mucho más simple y lenta que cualquier procesador actual, la diferencia es su conectividad ya que, de media, cada neurona está conectado con otras casi cien. Otra diferencia importante es la memoria, mientras que en una máquina esta se sitúa separada del procesador, en el cerebro se cree que esta forma parte de la misma red formando las conexiones entre neuronas.

De forma análoga, una red neuronal artificial está formado por diversos nodos basados en el algoritmo perceptrón [10] que se sitúan en capas, propagando la información entre ellas escalando su valor por unos pesos internos y aplicando una función de activación lineal a la salida. Según los tipos de conexiones, número de capas y arquitectura se pueden distinguir varios tipos, los cuales serán descritos en capítulos posteriores.

En el proyecto se ha utilizado un clasificador basado en redes neuronales implementado en la librería FastText [11]. Según los desarrolladores [12], la arquitectura del modelo se basa en una modelo de clasificación lineal, estructura que se puede observar en la Figura 2.7, con evaluación por *hierarchical softmax* [13] que permite realizar de forma eficiente el entrenamiento, sobre todo con tamaños de *embeddings* elevados y un gran número de clases.

Adicionalmente, este modelo cuenta con vectores de representación preentrenados de talla cien y trescientos para diversos idiomas, de los cuales se hablarán en capítulos posteriores, así como la posibilidad de entrenar modelos de forma supervisada o no supervisada. Además de la típica parametrización de hiperparámetros, el modelo incluye la posibilidad de utilización de n-gramas para utilizar una unidad mínima de representación distinta a la palabra.

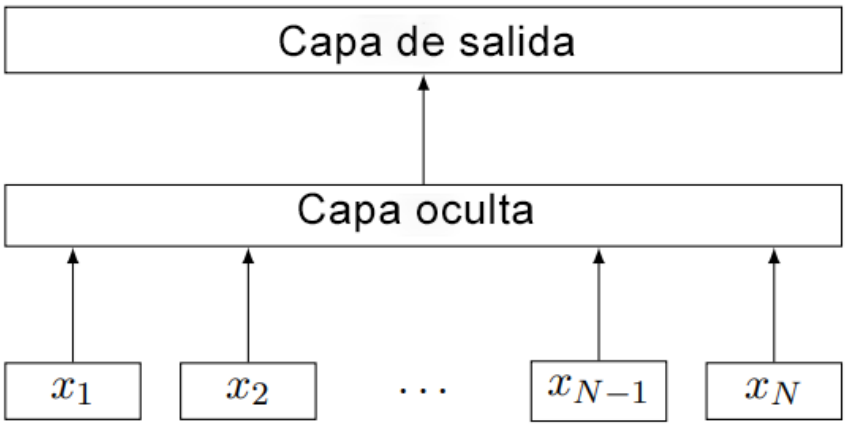

Figura 2.7: Representación gráfica del modelo *FastText* para su variante CBOW

2.6 Métricas y evaluación

Existen diversas métricas y técnicas para evaluar cualquier modelo de aprendizaje automático pero en la presente sección definiremos algunas de ellas que han sido utilizadas en el transcurso del proyecto y son útiles para clasificación. Por otra parte, también se describirán diferentes técnicas para estimar el error producido por los modelos.

Existen cuatro términos comunes a prácticamente todas las métricas de clasificación y los cuales son utilizados en posteriores secciones para definirlas:

- **Verdadero positivo**: o *true positive*, para una determinada clase, cuando la clase predicha y la real son la misma.
- **Verdadero negativo**: o *true negative*, para una determinada clase, cuando la clase predicha y la real son la misma y no son esta.
- **Falso positivo**: o *false positive*, para una determinada clase, se predice que una muestra pertenece a ella cuando en realidad pertenece a otra clase.
- **Falso negativo**, o *false negative*, para una determina clase, se predice que la muestra no pertenece a ella cuando realmente si es esa clase.

2.6.1. *Accuracy*

La *accuracy*, o exactitud, es el ratio de muestras bien etiquetadas entre el número total de muestras 2.4. Está métrica solo aporta valor real si el número de muestras entre cada clase esta balanceado. Además, esta métrica tampoco es indicada cuando el peso real de cada clase no es el mismo. Por ejemplo, predecir si un paciente tiene una enfermedad, el modelo puede ser muy bueno detectando casos negativos, pero no positivos, llevando a un falso valor de *accuracy*.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$

Existe una variante de esta métrica llamada *accuracy-at-k* siendo k un número menor o igual al número de clases del problema. La mayoría de clasificadores no tienen como salida la etiqueta de la clase a predecir, sino un vector de pesos con la probabilidad de

que la muestra pertenezca a cada clase. Esta métrica mide las veces que un modelo ha predicho correctamente la clase entre las k etiquetas con mayor probabilidad.

2.6.2. *Precision*

La precisión representa el porcentaje de muestras que se han identificado como una clase y efectivamente pertenecían a esa clase 2.5. En problemas de clasificación multiclase, esta métrica corresponde con el *accuracy*.

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

2.6.3. Recall

El *recall*, o recubrimiento, se define como el ratio de muestras bien etiquetadas entre el número de muestras relevantes 2.6, es decir, los positivos verdaderos y los falsos negativos.

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

Con esta métrica se intenta simular el porcentaje de muestras que ha logrado ver un modelo por cada etiqueta, por lo que es una métrica muy utilizada en clasificación multiclase. Para calcular el *recall* de un modelo de este tipo existen principalmente dos formas, el micro *average*, que realiza el cálculo respecto al número de muestra totales y no las de cada clase, y el macro *average*, que realiza el *recall* para cada clase 2.7 y posteriormente se hace la media. Esta última es más indicado para problemas de clasificación multiclase donde existe desbalanceo entre ellas, sobre todo cuando existen clases muy minoritarias.

$$Recall_c = \frac{TP_c}{TP_c + FN_c} \tag{2.7}$$

2.6.4. Matriz de confusión

La matriz de confusión, o matriz de error, permite representar de manera visual el comportamiento de un sistema. En ella, el eje de abscisas representa la clase real y el de ordenadas representa la clase predicha por el modelo. Si seguimos el mismo orden $X(C_n) = Y(C_n)$ para enumerar las clases en cada eje obtenemos que el valor de la diagonal son los *true positives* para cada clase, los valores que se encuentra fuera de la diagonal y a la vez fuera de la fila representaran los *true negatives*, los *false positives* vendrán dado por los valores de la fila ,menos el de la diagonal, y los valores que estén en la columna, menos el de la diagonal, serán los *false negatives*.

Mediante esta técnica es posible obtener también las métricas anteriormente descritas, así como poder definir otras más complejas. Es habitual encontrarse los valores de la matriz ponderados al número total de muestras del conjunto de entrenamiento, en tal caso no estaremos hablando de cuantas muestras son de cada tipo, sino del porcentaje. En la Figura 2.8 se puede observar una matriz de confusión extraída del propio proyecto.

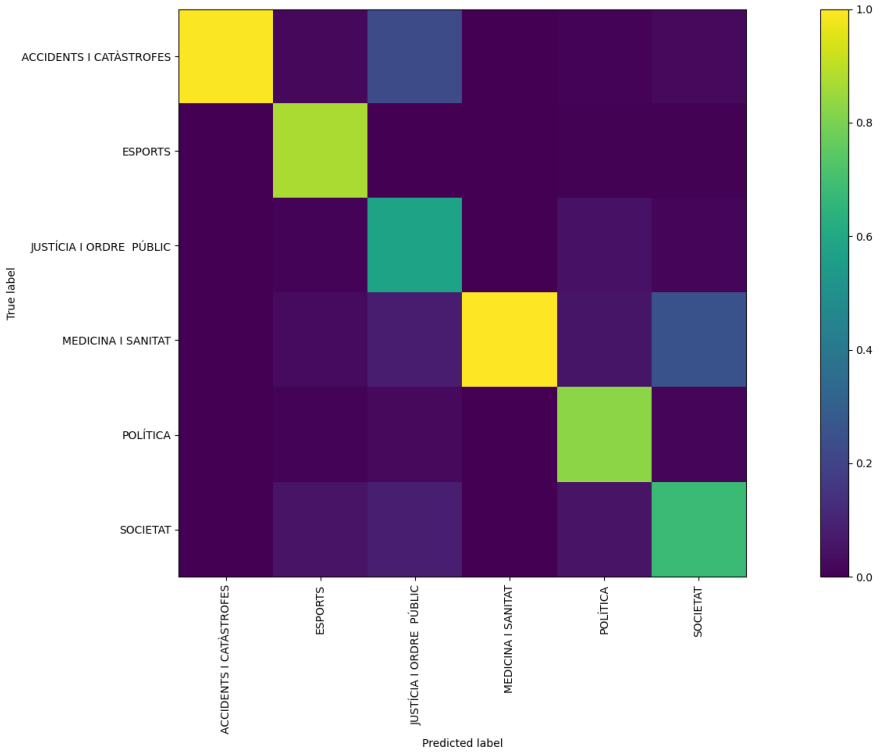

Figura 2.8: Matriz de confusión normalizada con $\mathcal{C} = 3$

2.6.5. Estimación de error y técnicas de partición de datos

Tras el desarrollo de cualquier modelo de *machine learning* es esencial evaluar su rendimiento o, por el contrario, cuál es su error. Si $f : \mathcal{X} \rightarrow \mathcal{Y}$ es la función estimada por nuestro modelo y sea $P(x)$ la distribución real de probabilidad de cualquier conjunto de datos $x \in \mathcal{X}$, podemos definir la probabilidad de error, o esperanza de error, como la Ecuación 2.8.

$$E_x[\text{error}(f(\mathbf{x}))] = \sum_{x \in \mathcal{X}} \text{error}(f(\mathbf{x}))p(\mathbf{x})dx \tag{2.8}$$

El problema se encuentra en que la verdadera probabilidad de error, $P(x)$, no se suele conocer en los problemas reales, por lo que tampoco es posible calcular la verdadera esperanza de error. Es por ello que, en la práctica, en los problemas de aprendizaje supervisado, se suele estimar este valor mediante diversas técnicas que utilizan conjuntos de datos etiquetados, es decir, con el valor real. Algunas de las técnicas más utilizadas son:

- **Resustitución**: o *resubstitution*, todos los datos son utilizados para evaluar el modelo, tanto en entrenamiento como para test, siendo una técnica muy optimista ya que se evalúa con datos ya visto e introduce *overfitting*.

- **Partición**: o *holdout*, consiste en la partición del total de datos en dos conjuntos, uno para entrenamiento y otro para test, siendo habitual que el de entrenamiento sea más grande, como por ejemplo, una relación 80-20. En caso de que la tarea sea de clasificación y multiclase, se suele utilizar la variante *stratified* que realiza una partición equitativa de cada clase presente en el conjunto. El inconveniente de esta técnica es que no se dispone de todos los datos para entrenar el modelo.

- **Validación cruzada**: o *K-Fold Cross-Validation*, el conjunto de datos se particiona en K bloques, o *folds*, y se utiliza cada uno para validación, utilizando el resto de datos para entrenamiento, siendo el error total la media de los errores de los K experimentos. La ventaja de este método es que todos los datos se utilizan para entrenar, pero si K es bajo, se utilizaran pocos datos para entrenar, pudiendo sufrir $underfitting$.
- **Exclusión individual**: o *Leave-One-Out*, consiste en realizar tantos experimentos como datos haya en el conjunto, utilizando todos los datos menos uno para entrenar y utilizando este excluido para test. Equivale a realizar validación cruzada con $K =$ número de datos. Está técnica es útil para tareas de regresión o clasificación binaria, pero poco representativa para clasificación multiclase. Además, si el conjunto de datos es grande y el entrenamiento del modelo lento, es temporalmente inviable de realizar.
- **Submuestreo aleatorio**: o *random subsampling*, consiste en crear varias particiones seleccionado los datos de forma aleatoria para validación y el resto para entrenamiento, calculando el error total como la media de cada experimento. Es posible que cada conjunto contenga datos que ya han pertenecido a otros conjuntos.
- **Muestreo por agregaciones**: o *bootstrapping*, el conjunto de datos se particiona en bloques de igual tamaño y posteriormente se seleccionan un número aleatorio de estos para entrenamiento, utilizando el resto para validación, por lo que el tamaño de cada conjunto puede variar entre iteraciones, tal como se ilustra en la Figura 2.9. El error final se obtiene con la media de cada iteración. Es una técnica utilizada para evitar el *overfitting*.

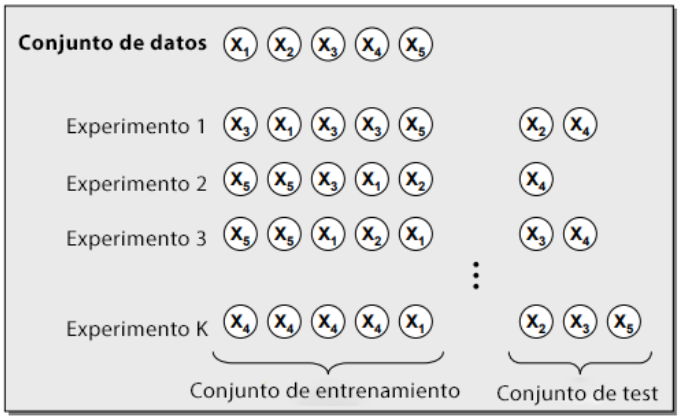

Figura 2.9: Técnica de validación *bootstraping*

CAPÍTULO 3
Representación numérica del lenguaje

En el desarrollo de un proyecto de *machine learning* es tan importante una buena selección del algoritmo de aprendizaje como el modelo de representación de los datos. En el capítulo actual se exponen diversos modelos con los que representar el lenguaje natural de forma numérica ya que cualquier modelo espera, directa o indirectamente, una entrada de este tipo.

3.1 Modelos vectoriales básicos

Los modelos definidos en la presente sección aportan conceptos básicos necesarios para el entendimiento y desarrollo de modelos de representación más avanzados.

3.1.1. *One-hot enconding*

Un vector *one-hot* es una matriz de tamaño $1\,x\,N$, siendo N el número de palabras únicas del vocabulario, el cual su utiliza para diferenciar una palabra de entre todas las demás. El vector está completo a 0 a excepción de la casilla correspondiente a la palabra, que contendrá un 1. En modelos de *machine-learning*, esta representación aporta la misma importancia a cada palabra independientemente de su posición, por lo que una palabra que su vector represente el 1 en la posición primera, tendrá la misma importancia que el que la represente en la última. Por lo tanto, un documento mediante representación *one-hot*, estará representado por una matriz con tantas columnas como palabras únicas tenga el documento.

Esta representación es totalmente arbitraria y no tienen en cuenta las relaciones entre palabras ni las relaciones del contexto. Además, se vuelve demasiado ineficiente cuando se cuenta con vocabularios con una talla elevada, por lo que en la práctica no se suele utilizar salvo en tareas reducidas y específicas.

3.1.2. *Bag-of-words*

La representación *bag-of-words*, o bolsa de palabras, modela un documento como un vector de tamaño N, siendo este el tamaño total del vocabulario, es decir, como un conjunto sin repeticiones. Dicho vector puede representar la aparición o no de la palabra en el documento, con ceros y unos, o como la frecuencia en cada documento, con cero si no aparece y con un número mayor de cero indicando cuantas ocurrencias tiene.

Este modelo tampoco tiene en cuenta la posición de las palabras en el texto, pero sí la frecuencia, por lo que es utilizado como base para otros modelos.

3.2 *TF-IDF*

Como paso previo a la definición del modelo *TF-IDF*, es necesario definir dos tipos de matrices [14] que son utilizadas por dicho modelo. La primera de ella es la matriz de palabra-documento, o *term-document matrix*, donde se representa la ocurrencia de cada palabra del vocabulario en el conjunto de documentos. La fila i indica la palabra en la posición i del vocabulario, mientras que la columna j indica el documento j del conjunto total. Por tanto, la posición $M_{i,j}$ indicara el número de ocurrencias de la palabra i en el documento j, siendo este valor mayor o igual que 0. Por lo tanto, la dimensión de la matriz será de $V \times |D|$, siendo V la talla del vocabulario y $|D|$ el número total de documentos.

Por su parte la matriz de palabra-palabra, o *term-term*, indica para el *corpus* completo la relación directa entre dos pares de palabras y puede ser construida de diferentes formas. Una de estas implementaciones se basa en que las filas representan las palabras del vocabulario, el cual es de talla V, y las columnas indican tres valores por palabra, por lo tanto, la matriz será de dimensión $V \times 3V$. Estos valores son las ocurrencias de una palabra i precedida directamente por una palabra j, las ocurrencias de una palabra i totales y las ocurrencias de una palabra i seguida de una palabra j. Esta matriz es extrapolable a cualquier conjunto n-gramas.

El modelo *TF-IDF*, o *term-frecuency - inverse document frecuency*, es un técnica que permite modelar la relevancia de una palabra para un documento en un *corpus* de forma numérica. Para una palabra, dicho valor es incrementado proporcionalmente con el número de ocurrencias del término en un texto y posteriormente suavizado según el número de documentos que lo contengan, lidiando así con el hecho de que algunas palabras son más frecuentes que otras.

La técnica se divide a su vez en dos partes. La primera es el *term-frecuency*, o frecuencia del término, que indica la frecuencia de ocurrencia de un término en un único documento. Para solventar el problema de que un documento puede ser de una longitud muy diferente a otro en el *corpus*, y por tanto más posibilidad de que la palabra esté repetida, se suelen utilizas frecuencias logarítmicas o normalizadas. En el caso de una frecuencia es un escala logarítmica, podemos definir el *TF* como la Ecuación 3.1, donde t es un término, d es un documento y $f(t,d)$ es la frecuencia de un término en un documento.

$$\mathrm{tf}(t,d) = \begin{cases} 1 + log(f(t,d)) & \text{si } df(t,d) > 0 \\ 0 & \text{si } df(t,d) = 0 \end{cases} \tag{3.1}$$

Por otra parte, la *inverse document frecuency*, o frecuencia inversa del documento, representa el ratio entre el número total de documentos representado por $|D|$ y el número de documentos donde aparece un término $df(t)$. Su definición se puede ver en la Ecuación 3.2 y, como en el caso de *TF*, se suele utilizar una escala logarítmica para suavizar el valor en *corpus* con un gran número de documentos. Esta operación da valores más altos a términos poco frecuentes en el conjunto.

$$\mathrm{idf}(t,D) = log(\frac{|D|}{\mathrm{df}(t)}) \tag{3.2}$$

Finalmente, el valor del modelo se obtiene como el producto escalar 3.3 del *TF* y el *IDF*. Un valor alto indica que un término t es frecuente en un documento d y a la vez poco frecuente en el conjunto D.

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D) \tag{3.3}$$

3.3 *Latent Semantic Analysis*

El análisis de semántica latente, o *Latent Semantic Analysis* (LSA), son una serie de técnicas basadas en espacio de representación vectorial que intentan modelar la semántica del lenguaje mediante factorización de matrices para así encontrar factores latentes en palabras y documentos con lo que estimar la similitud entre estos.

Para ello, se aplica una técnica algebraica llamada descomposición de valores singulares en la matriz de *word-document* para encontrar factores no correlacionado, tanto para palabras como para documentos. De aplicar este proceso a la matriz se desprenden tres matrices resultantes: la matriz E donde cada fila representa una palabra, la matriz diagonal Z que representa los valores singulares y la matriz D, cuyas filas representan cada documento. Entonces, podemos definir la similitud entre dos palabras w_i y w_j como la Ecuación 3.4.

$$sim(w_i, w_j) = M_i, M_j^T = w_i Z^2 w_j \tag{3.4}$$

Mientras que el coste de calcular Z no es excesivamente costoso, el de E y D sí que lo es, sobre todo para *corpus* con muchos documentos y con una talla de vocabulario elevada ya que el coste computacional es $min(m^2n, mn^2)$, siendo n y m el tamaño de la matriz. Es por ello que existen técnicas alternativas para calcular dichas matrices, como el indexado aleatorio que asigna a cada documento un vector ternario disperso, generado aleatoriamente, que es ortogonal al vector que representa una palabra.

3.4 *Latent Dirichlet Allocation*

En los modelos de asignación Latente de Dirichlet, *o Latent Dirichlet Allocation* (LDA), son utilizados para analizar el contenido de los documentos y el significado de las palabras. Estos modelos asumen que cada texto está compuesto por varias categorías, las cuales se definen como una distribución de probabilidad sobre un conjunto de vocabulario fijo, y que han sido definidos antes de la generación de cualquier *corpus*.

El modelo actúa en dos fases: en la primera generar una distribución aleatoria de las categorías y en la segunda, para cada palabra del documento asignar aleatoriamente una categoría y a su vez asignar una palabra de la categoría anterior. Con ello se quiere simular que un documento está formado por varias categorías mediante una distribución de probabilidad que indique cuanta proporción de cada categoría hay presente en cada documento.

Así pues, podemos definir la probabilidad de las distribuciones como la Ecuación 3.5, donde β_k representa la distribución sobre el vocabulario, θ_{dk} es la proporción de la

categoría k en el documento d, Z_{dn} es la categoría asignada a la palabra n en el documento d y finalmente V representa el vocabulario.

$$P(\beta_{1:k}, \theta_{1:D}, Z_{1:D} \mid V_{1:D}) = \frac{P(\beta_{1:k}, \theta_{1:D}, Z_{1:D}, V_{1:D})}{P(V_{1:D})} \tag{3.5}$$

La probabilidad del denominador se puede calcular, teóricamente, como la suma de las uniones de todas las posibles muestras de cada tópico. En la práctica, determinarlo es casi imposible por lo que se opta por modelar esta probabilidad mediante una seria de distribuciones parametrizadas para cada categoría, transformando así el problema en uno de optimización. Mediante la relajación de algunas asunciones previas del modelo, así como con la incorporación de meta-datos, como por ejemplo la relación entre documentos, se pueden construir modelos LDA más eficientes y precisos.

3.5 Redes neuronales prealimentadas

Las redes neuronales prealimentadas, o *feed-forward*, es una red acíclica donde la información se propaga en una única dirección.

El input, las palabras de entrada, son representadas mediante *word embeddings*. Estas representaciones son vectores donde se mapea la representación distribuida del lenguaje en un espacio predefinido de dimensión n. De esta forma, se intenta representar que las palabras relacionadas semánticamente están próximas en este espacio de representación haciéndolas sensibles al contexto donde se encuentran.

El modelo concatena la representación individual de cada palabra formando así un único vector con el que alimentar la red neuronal de tamaño $n \times k$, siendo k el número de palabras del texto a representar, mientras que la salida será una capa con un número de neuronas igual a la talla del vocabulario. Entre estas dos capas se sitúa una capa oculta que es la encargada de mapear la distribución cuyas funciones de activación suelen ser del tipo sigmoide. Los pesos de las neuronas de esta capa intermedia son aprendidos durante el entrenamiento.

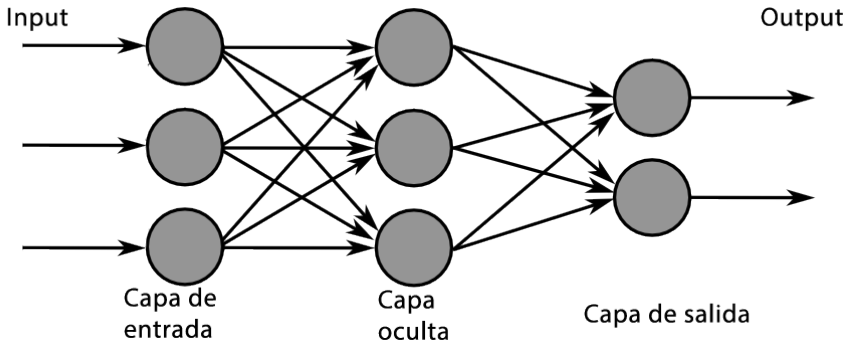

Figura 3.1: Red neuronal artificial *feed-forward* [15]

Finalmente, la función de propagación hacia la capa de salida se realiza mediante una función de activación *softmax* ya que el resultado esperado es un vector que representa la distribución de probabilidad total.

Normalmente, los parámetros de esta red son aprendidos por *back-propagation* mediante la técnica de optimización de descenso por gradiente y, adicionalmente, el *embedding* individual de cada palabra también puede ser aprendido durante el proceso, aunque también se suelen utilizar *embeddings* preentrenados, como por ejemplo *Word2Vec*.

Como todos los modelos estadísticos, este también tiene ciertas limitaciones, ya que tallas de vocabulario enormes pueden desembocar en entrenamientos temporalmente inviables y, además, las palabras vistas en una posición dentro de un texto solo llegan a influenciar a palabras cercas de ella, por lo que las dependencias espaciales y temporales se suelen perder en el contexto.

Finalmente, se puede realizar un modelo que tenga precalculado el *embedding* para cierta palabra precedida de un número determinado m de palabras, el problema es que el uso de memoria crece exponencialmente con m. Por contra parte, el número de parámetros de este modelo sería menor ya que escala de forma lineal m, de la forma $m * k * h + h * n$, siendo h el número de neuronas de la capa oculta.

3.6 Redes neuronales recurrentes

Las redes neuronales recurrentes, o *recurrent neural networks* (RNN), intentan solventar el problema de las redes con retroalimentación modelando las dependencias temporales de una palabra con otra, sin depender de *embeddings* preentrenados únicamente para una cierta distancia.

Se trata de una red con ciclos donde la entrada del modelo tiene la forma de un vector x y produce otro vector de forma y, la diferencia recae en que el modelo espera recibir una secuencia de vectores, uno en cada instante de tiempo t, y produce otra secuencia de vectores en el mismo orden.

El vector $y(t)$ obtenido en un instante de tiempo determinado no está influenciado única y exclusivamente por $x(t)$, sino que lo está por toda la secuencia de vectores x vistos por la red. Almacenar en memoria todas las secuencias vistas es inviable en términos de memoria, sobre todo para textos grandes, por lo que en vez de ello, se guarda este historia de secuencias en un estado que sea capaz de representarla, el cual se define como $h(t) = f(h(t-1), x(t))$, donde se actualiza el estado antes de realizar la predicción. Por tanto, podemos definir la salida del modelo para un instante determinado como $g(h(t))$.

Finalmente, queda definir las funciones f y g. La primera calcula el siguiente estado de la red como una combinación lineal entre el estado actual y el siguiente vector x y se puede definir como $f(h, x) = A1(Whh\dot{h} + Wxh\dot{x})$. La segunda función calcula el vector de salida y como una transformación linear del nuevo estado y se define como $g(h) = A1(Why * h)$. En ambos casos A es una función de activación, normalmente de tipo sigmoide $A1$ y la función *softmax* para $A2$.

En cuenta a la estructura de una RNN, es similar a una red *feed-forward*. El tamaño de la capa de entrada dependerá de la representación y su tamaño, el tamaño de la capa oculta será arbitrario y el número de neuronas en la capa de salida dependerá del tamaño del vocabulario.

Ahora bien, en la práctica, este tipo de redes suele tener un rendimiento menor al de la redes prealimentadas y se utiliza implementaciones basadas en esta estructura como son las *Long Short-Term Memory* (LSTM). Estas capturan mejor las dependencias temporales a largo plazo ya que están formadas por tres tipos de celdas, las dos de input y output clásicas, y las celdas de memoria que son capaces de recordar cierto rango de valores para determinados periodos de tiempo. Dicha estructura se puede observar en la Figura 3.2

Además, solventan los problemas de desvanecimiento de gradiente y el de gradiente explosivo. El problema de desvanecimiento es típico de las redes neuronales y está relacionado con la técnica de descenso de gradiente ya que con esta se llega a alcanzar valores muy pequeños, decrece exponencialmente, y al depender los pesos de los nodos de este valor, se actualizan de forma lenta y desigual, sobre todo si la capa tiene varias

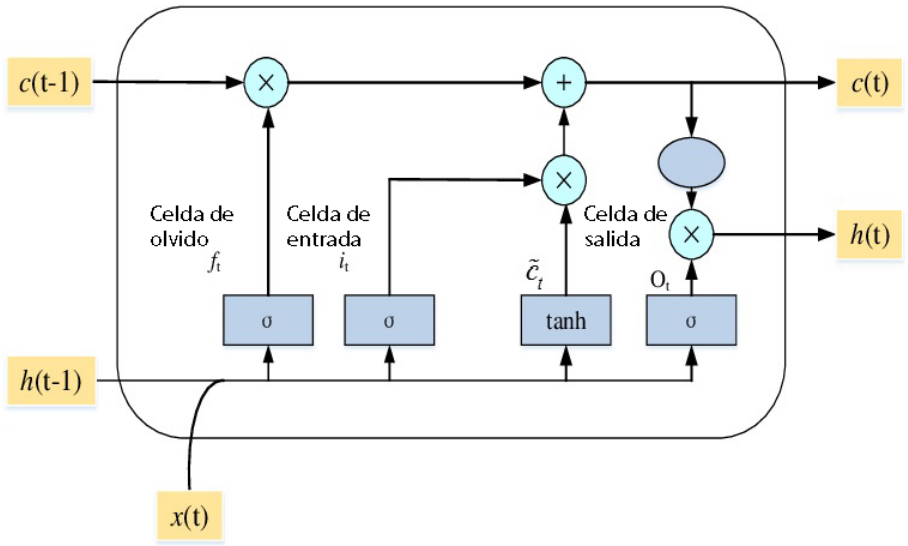

Figura 3.2: Arquitectura de una RNN LSTM [16]

capas ocultas. Por el contrario, el problema de gradiente explosivo, también está relacionado está relacionado con la técnica de descenso de gradiente, pero en este caso por la utilización de funciones de activación cuyo rango de valores de salida puedan tomar valores altos y, inevitablemente, el tamaño de la gradiente crezca exponencialmente. Este problema está más relacionado con las RNN.

3.7 CBOW

El modelo CBOW, o *Continuous Bag-of-Words*, intenta predecir una palabra a partir del contexto, sin tener en cuenta la posición en el texto ni la historia [17]. Su arquitectura es similar a la de las redes *feed-forward* con la diferencia que las capa o capas ocultas no lineales son eliminadas, compartiendo la capa de proyección en la misma posición, obteniendo los vectores mediante promediado.

El tamaño de la capa de entrada vendrá dado por la talla del vocabulario y por el número de palabras a utilizar como *input*, la capa de proyección será del tamaño de representación del *emebedding* y la capa de salida será igual a la talla del vocabulario. Dicha estructura se puede observar en la Figura 3.3.

El resultado final del modelo será la obtención de la palabra más probable según el contexto. Según [18] el funcionamiento del modelo se pude desglosar en los siguientes pasos:

1. Generar los vectores *one-hot* de las palabras de entrada.
2. Transformar estos vectores en *embeddings* según el contexto.
3. Realizar el promedio de todos los *embeddings* obtenidos de la entrada.
4. Generar un vector de calificaciones para al vector anterior.
5. Convertir el vector de calificaciones en uno de probabilidades mediante una función de activación, por ejemplo, la función *softmax*.

Finalmente, se puede definir el coste computacional Q de entrenar este modelo como la Ecuación 3.6, donde N es el tamaño de la capa de entrada definido por el número de

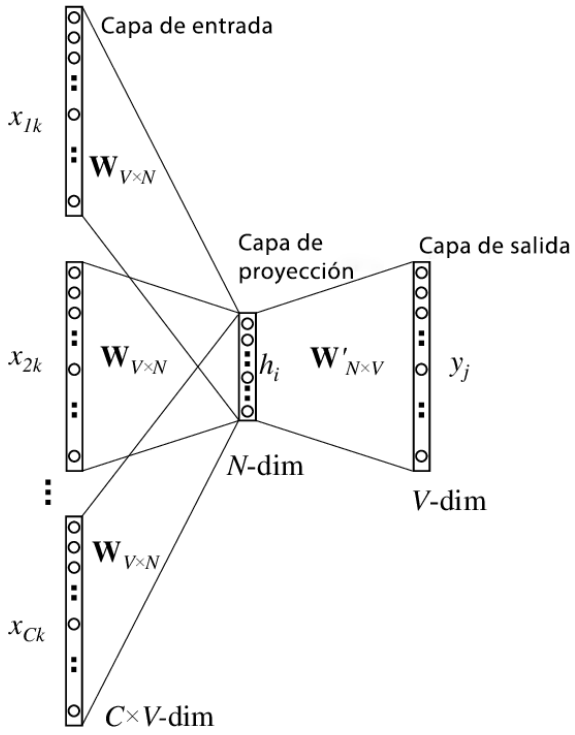

Figura 3.3: Arquitectura de red neuronal CBOW [18]

palabras, H es el número de neuronas de la capa de proyección común definido por la talla de los *embeddings* y V es la talla del vocabulario.

$$Q = N * H + H * log2(V) \tag{3.6}$$

3.8 Skip-Gram

Por su parte, el modelo *Skip-Gram*, es el opuesto del modelo *CBOW* pese que comparten una estructura similar. Este modelo intenta predecir el contexto de una palabra, es decir, dada una palabra como entrada, devuelve un conjunto de palabras próximas en el espacio de representación.

Por lo tanto, su arquitectura consta de una capa de entrada de tamaño igual al vocabulario, una capa de proyección común cuyo tamaño vendrá dado por la dimensión del *embedding* y una capa de salida cuyo tamaño viene definido por el número de palabras del contexto a predecir y la talla del vocabulario. Esta estructura está definida en la Figura 3.4.

el resultado final será la probabilidad de cada palabra de pertenecer al contexto, pero normalmente solo se obtiene como salida las C palabras más probables. El funcionamiento del modelo puede definirse generalmente según [18] como los siguientes pasos:

1. Generar el vector de entrada *one-hot* de la palabra a utilizar.
2. Transformar este vector en un *embedding* según el contexto.
3. Como no hay que realizar el promediado, el *embedding* generado es el propagado.
4. Generar tantos vectores de calificaciones como palabras tenga el vocabulario.

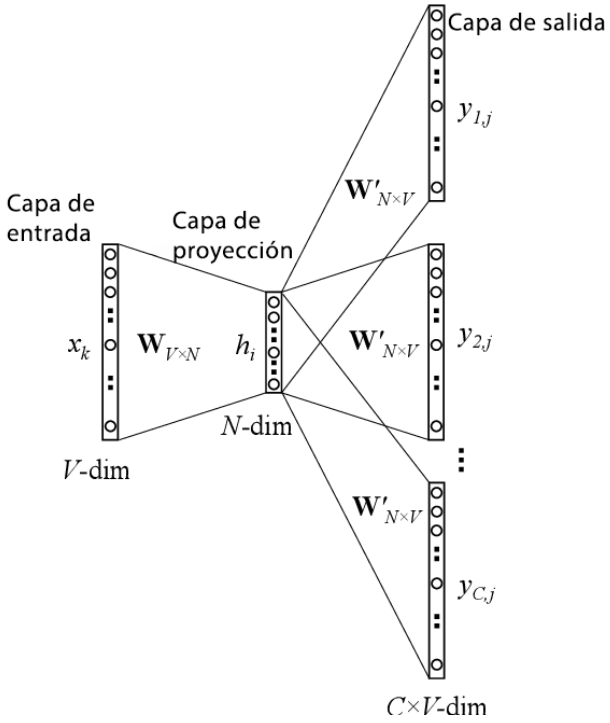

Figura 3.4: Arquitectura de red neuronal *Skip-Gram* [19]

5. Convertir los vectores de calificaciones en vectores de probabilidades mediante una función de activación, por ejemplo, la función *softmax*.

Finalmente, se puede definir el coste computacional Q de entrenar este modelo como la Ecuación 3.7, donde C es el número de palabras del contexto a predecir, H es el número de neuronas de la capa de proyección común definido por la talla de los *embeddings* y V es la talla del vocabulario.

$$Q = C * (H + H * log2(V)) \tag{3.7}$$

3.9 FastText

FastText es una librería [11] que proporciona implementaciones de clasificadores RNN del tipo LSTM así como una representación del lenguaje en *embeddings* basada en *CBOW* y *skip-gram* junto a la técnica, que a continuación se describirá, llamada *subword information*. Según los propios desarrolladores [20], FastText implementa un modelo capaz de aprender la representación teniendo en cuenta la morfología de las palabras. Esta morfología no toma la palabra completa como unidad, sino que cada palabra es representada como la suma de unas *subword units*, siendo estás la división de la misma palabra en unidades de k caracteres. Por lo tanto, lo que se intenta modelar en este modelo es la importancia de los prefijos y sufijos para la representación del lenguaje.

Cada palabra es representada como una *bag-of-characters* de k-gramas, de forma análoga como se realizaría con un modelo *bag-of-words*. Estos k-gramas no son de cualquier tamaña, sino que únicamente se extraen los que su talla este comprendida entre los tres y seis caracteres. Adicionalmente, se añade la misma palabra en este conjunto como una secuencia especial. De esta forma se aprende tanto las palabras únicas del lenguaje como su representación en k-gramas.

Basándonos en la implementación *skip-gram* [19], podemos definir la función de puntuación que puntúa una palabra en un contexto como 3.8, donde w es la palabra, c es el número de palabras del contexto, $g \in \mathcal{G}_w$ representa cada k-grama de la palabra y V el vector de salida que representa el contexto.

$$s(w, c) = \sum_{g \in \mathcal{G}_w} Z_g^T V_c \tag{3.8}$$

Finalmente, el modelo realiza su entrenamiento como un problema de optimización mediante descenso por gradiente.

CAPÍTULO 4
Corpus CVMC y preprocesamiento

El *corpus* CVMC, nombrado así por las siglas de Corporació Valenciana de Mitjans de Comunicació, tiene su origen en el sistema de documentación y catalogación de dicha entidad. El sistema lleva en funcionamiento desde su implementación en 1999 y cuenta con registros desde este año hasta el cierre de Canal 9 en 2013, y desde la reapertura de la cadena como À Punt en 2018 hasta la actualidad. Se trata de un *corpus* privativo.

En el sistema se almacena información de cada noticia que acontece en los telediarios de la cadena, así como algunas que no se han llegado a emitir. En él se guardan tanto lo dicho por el presentador como por los reporteros, si incluye directo la emisión, así como la voz en *off*. Adicionalmente, también se almacena otra información descrita a continuación.

Por lo tanto, en este capítulo se describe el modelo de datos del sistema y las diferentes técnicas para transformarlo en un *corpus* utilizable por un modelo de aprendizaje automático, así como las estadísticas de este.

4.1 Descripción del *corpus*

No se cuenta con acceso directo al sistema, sino que ha sido la misma Corporació la que ha proporcionado los datos de su sistema en un conjunto de ficheros portables a partir de los cuales se ha trabajado.

El conjunto de datos se encuentra dividido en un total de 21 ficheros con extensión .xlsx, es decir, hojas de cálculo de Microsoft Excel, siguiendo el nombre de cada fichero un patrón determinado por "PI ANY *año número* assets.xlsx" siendo *año* el año al que hace referencia el archivo, en el rango de 1999 a 2013 y de 2018 a 2020, y por otra parte *número*, haciendo referencia al número de entradas totales de cada documento.

A su vez, también se proporciona un archivo adicional "Model de dades Documentació 20180411.xlsx" donde se define cada columna y sus características de la siguiente forma:

- **Código identificativo**: utilizado para obtener más información de la columna accediendo a una hoja distintiva en el mismo documento.
- **Nombre del campo**: un identificador descriptivo en lenguaje natural.
- **Tipo**: la codificación, la cual puede ser alfanumérico, numérico, fecha, hora o una combinación de estos tipos.
- **Extensión**: valor descriptivo indicando la longitud, en caracteres, de los valores dentro del campo.

- **Libre/Tesauro/Enumerado**: columna técnica útil para los usuarios del sistema de catalogación.
- **Multivalor**: utilizado para indicar si el valor en la columna puede estar compuesto a su vez de varios.
- **Obligatorio**: otra columna técnica para señalar si puede o no estar vacío el valor del campo.

El documento adicional cuenta con un total de 87 entradas pese a que los archivos anteriores únicamente contienen 17 de estas. Finalmente, de estas columnas, solo se utilizan 2 para elaborar el *corpus*, siendo estas:

- **Descripción**: contiene la noticia en sí, así como información adicional y ruido.
- **Clasificación**: la clasificación interna de la Corporació de cada registro que elaboran.

Se ha optado a reducir el conjunto útil de columnas a este par ya que las restantes se han considero ruido al contener únicamente valores estadísticos, como la duración de la emisión o la fecha exacta, números identificativos, ubicaciones o las personas que intervienen.

El conjunto resultante de datos sigue estando desestructurado y repleto de ruido, siendo inviable el entrenamiento de cualquier modelo de aprendizaje automático, por lo que debe ser procesado para tal fin.

4.2 Técnicas de preprocesamiento

La complejidad de generar un *corpus* a partir de este conjunto reside en la variación del estilo de descripción a lo largo de los años. Cabe recordar que el conjunto de datos cuenta con registros desde 1999 y que a su vez ha sido elaborado por dos canales de televisión, Canal 9 y À Punt, habiendo ocurrido cambios de estilo dentro de estas organizaciones, todo ello sumado a las peculiaridades únicas de cada redactor.

Es por ello que, como primer paso, se han implementado una seria de técnicas de preprocesamiento para lograr un conjunto de datos con un estilo homogéneo y, a su vez, que sirvan para futuras agregaciones de datos. Como se ha comentado en secciones anteriores, existe ruido en las muestras, por lo que la elaboración de dichas técnicas tiene en cuenta la creación de filtros específicos para homogeneizar aún más el conjunto.

Como consideración, se ha convertido todos los textos en minúscula, para reducir la talla del vocabulario, y, para todas las técnicas de preprocesamiento, se ha considerado como *token*, o unidad mínima de información, la palabra, siendo esta de longitud mínima de un carácter.

4.2.1. Filtro de apóstrofes y *pronoms febles*

Una característica de la lengua catalana es la *apostrofación* de los artículos en palabras que empiezan por vocal, por la letra "h" o siglas, entre otros casos. Otra peculiaridad son los *pronoms febles* que pueden ir solos, apostrofados o unidos por un guion al verbo. Así mismo, también pueden encontrar como combinación de varios pronombres.

Estas palabras carecen de significado propio y únicamente aumentan el tamaño del vocabulario de forma artificial ya que, si están unidas a otra, crean variaciones cuyo significado real es el mismo pero que las representaciones del lenguaje las tratan, normalmente, como totalmente distintas. Por ejemplo, existen en algunas muestras la palabra "*aniversari*" pero en otras también "*l'aniversari*", ambas hacen referencia al mismo concepto, pero al estar apostrofada la segunda, se considerarían totalmente diferentes.

El primer filtro realizado consiste en recorrer cada muestra, sustituyendo en esta todas las ocurrencias de artículos apostrofados y *pronoms febles* en su forma apostrofada y con guion, de una lista predefinida, por la cadena vacía. Dicha lista se puede consultar en el apéndice A. La sustitución o no de los *pronoms febles* en su forma libre se comentará en la siguiente sección.

4.2.2. Filtro de *stopwords*

Las *stopwords* o palabras vacías son aquellas que carecen de significado propio como artículos, preposiciones, etc., y, además, suelen ser palabras muy frecuentes en el lenguaje natural. Es por ello que estás palabras dependerán del idioma en el que se hallen los textos y no existe una lista única, sino que está dependerá en mayor o menor medida del *corpus* único que se utilice.

La lista de *stopwords* que se ha elaborado para este proyecto parte de la sugerida [21] por el profesor Lluís de Yzaguirre i Maura de la Universitat Pompeu Fabra. Esta lista contiene originalmente 701 palabras, a la cual se han sumado 19 palabras, listadas en un apendice, consideradas *stopwords* para este problema en concreto, ya que eran muy frecuentes en las muestras, eran palabras en inglés o se han considerado ruido al ser metadatos del sistema de la Corporació. La lista final se puede consultar en el apéndice A. Este segundo filtro funciona de manera similar al anterior, sustituyendo en cada muestra cada *stopword*, por la cadena vacía también.

Ahora bien, no es siempre recomendable eliminar estas palabras, sobre todo para determinadas tareas, por lo que la utilización o no de este filtro es opcional y es por ello que se crearan dos *corpus* similares, uno con su aplicación y otro sin, analizando en futuros capítulos si, para este proyecto, mejora o empeora las métricas de clasificación.

4.2.3. Filtro de ruido

Este último filtro es el más exhaustivo de los tres ya que es el encargado de limpiar todas las peculiaridades de las muestras armonizando los estilos de las diferentes cadenas de televisión y documentalistas, logrando así un *corpus* homogéneo en cuanto al formato.

El filtro está compuesto a su vez por varias expresiones regulares o *regex* que se ejecutan en un orden concreto para conseguir así una limpieza ascendente, reduciendo así la eliminación de información útil de forma colateral.

El primer paso ha sido eliminar todos los espacios en blanco y los caracteres ocultos, como "$\backslash n$", "$\backslash t$", "$\backslash r$", etc., tanto al principio como al final de cada muestra. Este filtro evita problemas de codificación.

El siguiente filtro es el encargado de eliminar los comentarios realizados por los documentalistas y que no aportan valor a los textos. Estos siguen un patrón predefinido compuesto por un texto envuelto en doble paréntesis, tanto inicial como final. Algunos de los comentarios más comunes son "((Entradeta))", "((Durada, cua i peu text))" o "((Historia - OFF))". Para tratar con ellos se ha elaborado la *regex* 4.2.3, el cual sustituye por un espacio en blanco todas las ocurrencias que casen con el patrón descrito.

```
pattern_comments = re.compile(r'\(+(.*?)\)+')
re.sub(pattern_comments, ' ', sample)
```

Otro tipo de comentario más discreto, pero igualmente común, son palabras pertenecientes a los meta-datos con uno o más guiones, tanto delante como atrás. También se utilizan como separadores de secciones sin aportar ningún valor. Por otra parte, también se hayan en palabras compuestas, como por ejemplo *"Xàtiva-Ontinyent"* aumentando el tamaño del vocabulario de forma artificial. Esta *regex* 4.2.3 realiza también una sustitución por un espacio en blanco por cada combinación de uno o más guiones.

```
pattern_ray = re.compile(r'-+')
re.sub(pattern_ray, ' ', sample)
```

El siguiente patrón está relacionado con los puntos. En gran parte de las muestras existe, seguramente por algún motivo interno del sistema de catalogación, dos patrones relacionados con combinaciones de puntos. El primero corresponde a dos o más puntos alternando espacios en blanco mientras que al segundo corresponde a una combinación de dos o más puntos seguidos. Ambos casos se resuelven utilizando la *regex* 4.2.3 para sustituir las ocurrencias por la cadena vacía.

```
pattern_multiple_dot = re.compile(r'\.\s(\.\s)+|\.\.+')
re.sub(pattern_multiple_dot, '', sample)
```

Existe también otro tipo de comentarios pero que suelen contener información útil para la muestra, por lo que se ha optado por mantenerlos realizando una limpieza menos exhaustiva. Estos suelen venir entre paréntesis o corchetes simples y, dentro de estos, comenzar con palabras de meta-datos, por ejemplo "(* AMBIENT 6")" o "[* NO comment 14]". En un primer filtro se sustituyen estos símbolos por un espacio y, posteriormente, se eliminan las palabras clave. Las *regex* 4.2.3 realiza está función.

```
pattern_parentesis = re.compile(r'\(+|\)+')
pattern_claudators = re.compile(r'\[+|\]+')
re.sub(pattern_parentesis, ' ', sample)
re.sub(pattern_claudators, ' ', sample)
```

En las muestras hay presentes secuencias de dos o más espacios en blanco, o bien porqué han sido introducidas como ruido por el sistema, o bien porqué han sido generadas por los filtros anteriores. Para evitar problemas de *tokenización* y facilitar la lectura, se ha convertido estás secuencias a un único espacio en blanco mediante la *regex* 4.2.3.

```
pattern_m_spaces = re.compile(r'\s+')
re.sub(pattern_m_spaces, ' ', sample)
```

Los siguientes filtros corresponde a la detección y eliminación de las palabras correspondiente a los meta-datos, las cuales son muy comunes y no aportan significado al contenido. Además, muchas de estas vienen escritas en ingles por lo que aumentan el vocabulario con extranjerismos. Algunos de estos filtros son los que se pueden observar en el fragmento de código 4.2.3.

```
pattern_news = re.compile(r'$\s*new .*$')
re.sub(pattern_news, '', sample)
sample.replace('/ retol/', ' ')
sample.replace('entradeta de seguretat', '')
```

Los últimos filtros realizados están relacionados con los símbolos no alfabéticos y los números. El sistema introduce de forma errónea etiquetas HTML a los textos, por lo que se ha substituido cualquier secuencia de caracteres contenida entre "<" y ">",

símbolo inicial y final de una etiqueta de este tipo. Posteriormente, se ha procedido a eliminar cualquier símbolo que no sea una palabra, ello incluye signos de interrogación, exclamación, puntos, comas, etc. Por último, se ha eliminado cualquier dígito presente en los textos ya que la mayoría eran anotaciones, como la duración de un vídeo, o datos numéricos únicos de la noticia al que el texto hace referencia y no aportan nada a la clasificación. Las *regex* 4.2.3 realizan estás funciones.

```
pattern_html = re.compile(r'<.*?>')
pattern_signs = re.compile(r"[?!@*+-/\\#$\.,'%&:|]")
remove_digits = str.maketrans('', '', digits)
re.sub(pattern_html, '', sample)
re.sub(pattern_signs, '', sample)
sample.translate(remove_digits)
```

Finalmente, después de todos los procesamientos, se eliminan las muestras que, o bien tengan la columna "Clasificación" vacía, o bien tengan la columna "Descripción" vacía. Adicionalmente, para evitar problemas ocasionados por fallos del sistema, se ha añadido un filtro para eliminar las posibles muestras repetidas. Como resultado se obtienen dos ficheros CSV, o *comma-separated values*, con dos columnas, una con el valor de la etiqueta de la muestra y la otra con el texto en sí. Como se ha comentado en secciones anteriores, el filtro de *stopwords* es opcional, por lo que un fichero contiene el *corpus* con estas y el otro sin.

4.2.4. Modelo de n-gramas

En secciones anteriores se ha comentado que la unidad mínima de información para la realización de filtros ha sido la palabra, pero existen otras representaciones que toman en cuenta otro tipo de unidad y cantidad, son los modelos de n-gramas [22].

Un n-grama es una secuencia de n unidades, pudiendo ser estas palabras, sílabas, letras, etc., las cuales son interpretadas como una única unidad. Si la unidad es la palabra, los tres casos más comunes son los uni-gramas, que hace referencia a una única palabra, y los bi-gramas y tri-gramas, que contienen dos y tres palabras respectivamente. También es común la utilización de esta técnica con n caracteres, lo que permite extraer información semántica de las palabras relacionadas con los prefijos y sufijos.

Según este modelo, la representación de la frase "*Les populars mascletaes de fogueres alacant*" extraída directamente del *corpus* con *stopwords* sería:

- **1-grama:** $, Les, populars, mascletaes, de, fogueres, alacant, $
- **2-grama:** ($ Les), (Les populars), (populars mascletaes), (mascletaes de), (de fogueres), (fogueres alacant), (alacant $)
- **3-grama:** ($ Les populars), (Les populars mascletaes), (populars mascletaes de), (mascletaes de fogueres), (de fogueres alacant), (fogueres alacant $)

En el ejemplo anterior, el símbolo $ hace referencia a la cadena vacía situada al principio y final de una oración, ya que hay algunos modelos que la tienen en cuenta a la hora de representar los n-gramas. Por otra parte, existen otras variaciones donde para representar una frase también se utilizan la representación de la frase en tallas de n menores. Por ejemplo, para este caso con 2-gramas, la representación quedaría como su representación en 2-grama más su representación en 1-grama: *{$, Les, populars, mascletaes, de, fogueres, alacant, $ ($ Les), (Les populars), (populars mascletaes), (mascletaes de), (de fogueres), (fogueres alacant), (alacant $)}*.

Finalmente, este modelo no ha sido utilizado para realizar los filtros pero sí para representar el lenguaje una vez procesado el *corpus* en diferentes modelos, como *TF-IDF* o *embeddings*, tal y como se comentará en posteriores secciones.

4.3 Estadísticas del *corpus*

Una vez realizado el preprocesamiento completo, se ha procedido a extraer estadísticas de este para así conocer sus características y saber exactamente al problema que estamos tratando. Toda estadística está "duplicada" ya que, como se ha comentado en la sección anterior, se cuenta con dos *corpus* con los que tratar.

El primer análisis corresponde al porcentaje de muestras útiles que existe en el *corpus* final ya que, con el preprocesamiento, se ha eliminado un gran número de muestras que estaban vacías o carecían de contenido útil. Como se puede observar en la Tabla 4.1, más de un tercio de las muestras se pierde; originalmente hay 654.306 muestras. En versiones anteriores de los filtros este porcentaje de perdida era incluso mayor por lo que se optó por suavizar algunos filtros, sobre todo los correspondientes a los comentarios. La diferencia de muestras entre ambos *corpus* se debe a que 129 muestras han resultado únicamente contener *stopwords*.

Muestras	% útil	*Stopwords*
397.804	60,79 %	✓
397.684	60,77 %	✗

Tabla 4.1: Número de muestras y porcentaje útil del *corpus*

El segundo análisis concierne a la columna "Clasificación". Esta columna, originalmente, contenía 8.198 muestra con valores nulos, así como 17.669 entradas con multietiqueta, hasta 5, habiendo un total de 1.413 combinaciones posibles. Es por este motivo por lo que se ha optado por eliminar las muestras etiquetadas con más de una clase, ya que aumentan la dificultad del problema, y únicamente trabajar con las 38 etiquetas simples. La eliminación de estas muestras ya se incluye en las estadísticas de la Tabla 4.1. Con esta amputación de datos se puede perder información útil, pero, al representar estos un porcentaje tan bajo del total, se ha optado por ello para conseguir un sistema más preciso. El nombre de cada etiqueta simple se puede consultar en la Tabla 4.2.

En un problema de clasificación multiclase, idealmente, cada clase contiene un número de muestras similar para que los modelos de aprendizaje automático no se especialicen o den más importancia a unas clases que otras. En los problemas reales, pocas veces ocurre esta repartición equitativa. Tal como se puede contemplar en la Tabla 4.2, dónde la primera columna representa el nombre de la clase, la segunda el número de muestras de esa clase con *stopwords*, la tercera el porcentaje de muestras de esa clase respecto el total con *stopwords*, la cuarta el número de muestras de la clase sin *stopwords* y la quinta el porcentaje de muestras de la clase respecto el total sin *stopwords*, existe un gran desbalanceo entre clases.

La clase "ESPORTS" representa más de una cuarta parte del total, un 28,17 %, mientras que las 10 clases más frecuentes suman casi el 80 % de todo el conjunto. Por otra parte, las 15 clases menos frecuentes no llegan a representar ni el 8 % del total. Estos datos anticipan que realizar un modelo altamente preciso será una tarea complicada.

El tercer análisis hace referencia a la columna "Descripción" la cual contiene la información utilizada por los modelos de *machine learning*. En la Figura 4.1 se muestran

Clase	Muestras	% total	Muestras	% total
ESPORTS	112061	28,17 %	112027	28,17 %
JUSTÍCIA I ORDRE PÚBLIC	43749	11,00 %	43736	11,00 %
POLÍTICA	36897	9,28 %	36873	9,27 %
SOCIETAT	26219	6,59 %	26215	6,59 %
ACCIDENTS I CATÀSTROFES	22468	5,65 %	22468	5,65 %
MEDICINA I SANITAT	16158	4,06 %	16148	4,06 %
FESTES I TRADICIONS	14700	3,70 %	14697	3,70 %
ECONOMIA, COMERÇ I FINANCES	13594	3,42 %	13584	3,42 %
MEDI AMBIENT	8975	2,26 %	8971	2,26 %
TRANSPORTS I COMUNICACIONS	8925	2,24 %	8925	2,24 %
AGRICULTURA, RAMADERIA I PESCA	8879	2,23 %	8876	2,23 %
FENÒMENS NATURALS	8626	2,17 %	8624	2,17 %
TERRORISME	6253	1,57 %	6252	1,57 %
CONFLICTES ARMATS	6247	1,57 %	6245	1,57 %
CULTURA	5341	1,34 %	5342	1,34 %
MÚSICA	4962	1,25 %	4959	1,25 %
TURISME	4757	1,20 %	4757	1,20 %
TREBALL	4625	1,16 %	4625	1,16 %
URBANISME I HABITATGE	4541	1,14 %	4538	1,14 %
CINEMA	4295	1,08 %	4295	1,08 %
OBRES PÚBLIQUES I INFRAESTRUCTURES	4114	1,03 %	4114	1,03 %
ENSENYAMENT	3970	1,00 %	3970	1,00 %
ARTS	3915	0,98 %	3916	0,98 %
RELACIONS INTERNACIONALS	3375	0,85 %	3375	0,85 %
CIÈNCIA I INVESTIGACIÓ	3336	0,84 %	3332	0,84 %
RELIGIÓ	3090	0,78 %	3090	0,78 %
ESPECTACLES	2678	0,67 %	2679	0,67 %
MITJANS DE COMUNICACIÓ	2558	0,64 %	2558	0,64 %
INDÚSTRIA	2277	0,57 %	2277	0,57 %
ENERGIA	1437	0,36 %	1437	0,36 %
DEFENSA	1301	0,33 %	1301	0,33 %
GASTRONOMIA	1119	0,28 %	1119	0,28 %
HISTÒRIA	800	0,20 %	800	0,20 %
LITERATURA	777	0,20 %	774	0,19 %
MISCEL·LÀNIA	278	0,07 %	278	0,07 %
ACTUALITAT ROSA	194	0,05 %	194	0,05 %
RECURSOS TERRITORIALS	185	0,05 %	185	0,05 %
TAUROMÀQUIA	128	0,03 %	128	0,03 %

Tabla 4.2: Número de muestras y porcentaje del total del *corpus* con y sin *stopwords*

tanto las palabras únicas como los bi-gramas y los tri-gramas únicos para cada conjunto obtenidos mediante la técnica de *bag-of-words*.

Cabe recordar que el tiempo de entrenamiento de algunos modelos crece linealmente con la talla del vocabulario mientras que otros lo hacen logarítmicamente, por lo que tallas grandes de este se ven reflejadas en el tiempo. Como era de suponer, la talla de n-gramas únicos para el conjunto sin *stopwords*, es menor que la del conjunto que sí contiene. Así mismo, la talla con n-gramas mayores a uno es mucho mayor ya que, al tratarse

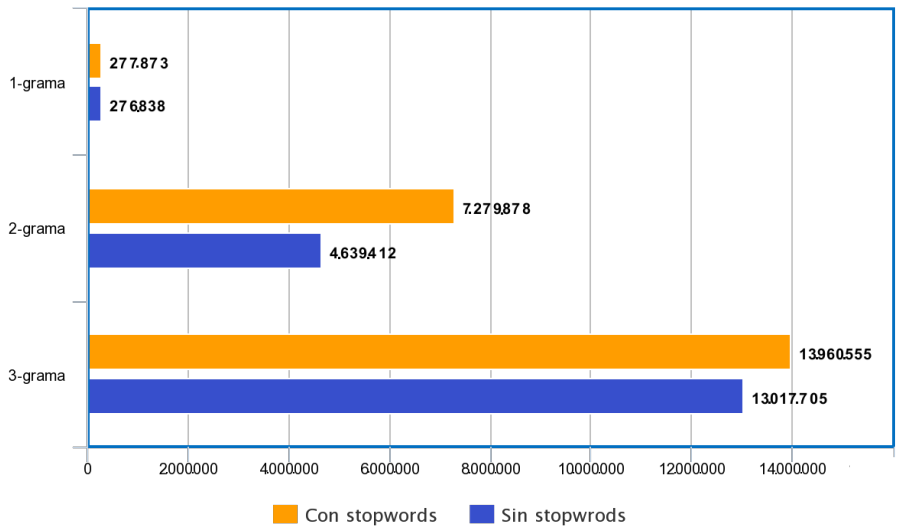

Figura 4.1: Representación gráfica de los n-gramas únicos

de un conjunto sin repetidos, los bi-gramas y tri-gramas crearan más combinaciones únicas.

Como se ha comentado, los n-gramas ayudan a captar información semántica que los 1-gramas no logran modelar, así como estructuras internas de las palabras en caso de que la unidad de estos sea menor que la palabra. Es por ello que se ha extraído información relativa al número medio de palabras de cada muestra por clase, así como la longitud mediana de las palabras también por clase. Ambas estadísticas se pueden observar en la Tabla 4.3, donde la columna dos y tres hacen referencia al *corpus* con *stopwords* y la cuatro y cinco al *corpus* sin *stopwords*. Como también era de esperar, la longitud media de los textos sin *stopwords* resulta menor, pero, por el contrario, la longitud media de las palabras aumenta ya que la mayoría de *stopwords* son palabras cuya longitud es menor a los 5 caracteres. Un primer análisis de estas estadísticas denota que sí será posible utilizar bi-gramas y tri-gramas para el entrenamiento de los modelos y muy posiblemente aumentar su precisión.

En la Figura 4.2 se puede apreciar el histograma de las palabras más frecuentes en el *corpus* y, como se puede observar, la mayoría son verbos o sustantivos que denotan el contexto regional de la CVMC. Adicionalmente, en la Tabla 4.4 se muestran los n-gramas más frecuentes de cada clase para n igual a 1, 2 y 3. Se puede observar que el n-grama más frecuente para uni-gramas y bi-gramas se solapa entre algunas clases, pero para tri-gramas, el más frecuente, es único para cada una. Este hecho reafirma la hipótesis anterior de que con la utilización de n-gramas se podrá lograr una mayor precisión.

Como análisis adicional, se puede observar que el *corpus* está muy especializado en el concepto político de la Comunitat Valenciana, con términos como *"comunitat"*, *"València"*, o *"comunitat valenciana"*, en términos políticos regionales como *"partit popular"* o *"president generalitat francesc"* y en lugares propios de la región como *"universitat politècnica valència"*, *"hospital fe valència"* o *"factoria ford almussafes"*. De este análisis se desprende que el *corpus* está muy especializado en los sucesos de la Comunitat Valenciana y que seguramente no será extrapolable a otros territorios de lengua catalana como les Illes Balears o Catalunya.

Finalmente, se ha hecho un análisis de los n-gramas únicos por clase para conocer el grado de especialización de cada una. En la Tabla 4.5 se puede ver el porcentaje de n-gramas únicos para uni-gramas, bi-gramas y tri-gramas del *corpus* con *stopwords*, mientras que en la Tabla 4.6 se observan las del *corpus* sin *stopwords*. En general, la especia-

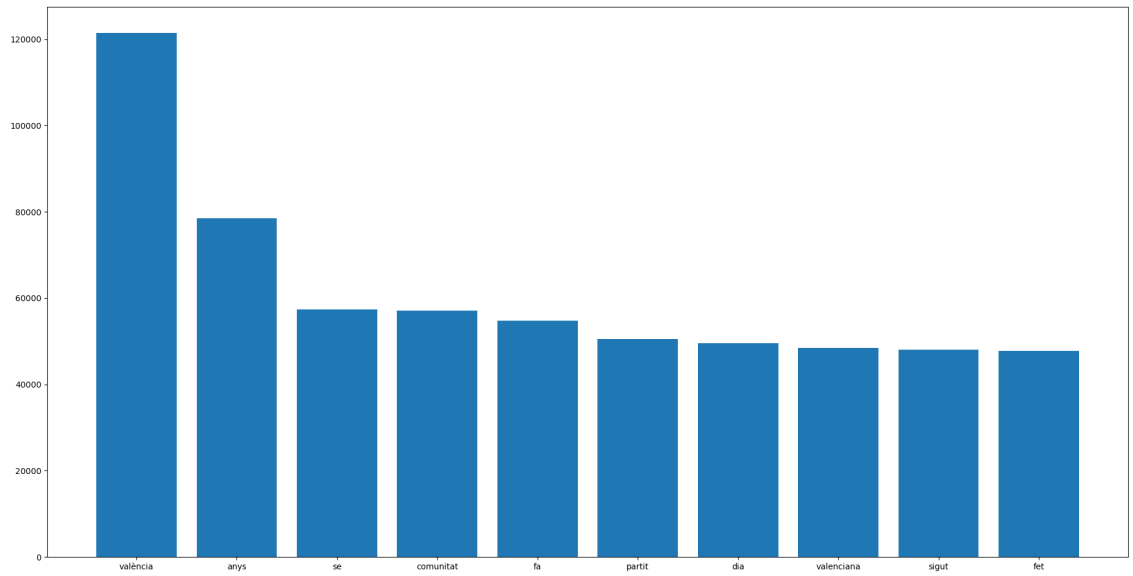

Figura 4.2: Histograma de las palabras más frecuentes

lización del *corpus* con *stopwords* es menor para cualquier n-grama. Conforme aumenta la talla de n el número de n-gramas únicos va aumentando logrando cifras mayores al 80 % con tri-gramas sin *stopwords*. Se destaca el elevado porcentaje asociado a la clase "*ESPORTS*", que sumado a su gran número de muestras, se traduzca en una elevadas métricas locales para la clase.

4.4 Partición de entrenamiento y test

En el próximo capítulo se procederá a utilizar el *corpus* para entrenar diversos modelos con diferentes configuraciones y representación del lenguaje. Es por ello que se ha elaborado una partición del cada conjunto en dos subconjuntos para poder comparar los resultados entre cada experimentación.

Se ha optado por una partición estándar del 70 % del total para el conjunto de entrenamiento y del restante 30 % para el conjunto de test. Como se trata de un conjunto muy desbalanceado, realizar una partición aleatoria puede llegar a ser contraproducente ya que se pueden a llegar a generar un conjunto de entrenamiento que no contenga ninguna o casi ninguna muestra de las clases minoritarios, o por el contrario, que el conjunto de validación sea el que no contenga de alguna clase y las métricas obtenidas no sean representativas.

Por tanto, esta partición se ha realizado de forma estratificada, o más comúnmente llamada *stratified*, la cual preserva la misma proporción de cada clase en cada conjunto. Por ejemplo, si la clase mayoritaria del *corpus* representa el 30 % del total, la relación anteriormente nombrada realizara una partición donde contenga el 70 % de ese 30 % en entrenamiento y el 30 % del 30 % total en test, es decir, el 21 % del total irá a entrenamiento y el 9 % a validación.

Además, las muestras de ambos conjuntos, tanto con *stopwords* como sin *stopwords*, han sido generadas en el mismo orden, para que la replicación del experimento entre modelos sea lo más exacta posible.

Esta repartición equitativa ayuda a lidiar parcialmente con el problema de desbalanceo entre clases ya que existen muestras de todas ellas en cada conjunto, pero el porcenta-

Clase	*x* palab.	*x* carac.	*x* palab.	*x* carac.
ACCIDENTS I CATÀSTROFES	368	13	235	13
ACTUALITAT ROSA	363	5	175	8
AGRICULTURA, RAMADERIA I PESCA	376	9	185	10
ARTS	361	9	189	9
CIÈNCIA I INVESTIGACIÓ	338	9	206	10
CINEMA	357	8	250	12
CONFLICTES ARMATS	357	7	184	9
CULTURA	361	8	194	9
DEFENSA	345	8	178	8
ECONOMIA, COMERÇ I FINANCES	361	9	208	10
ENERGIA	349	6	190	9
ENSENYAMENT	347	12	183	12
ESPECTACLES	350	8	196	9
ESPORTS	373	13	205	13
FENÒMENS NATURALS	377	12	189	12
FESTES I TRADICIONS	380	11	212	11
GASTRONOMIA	340	7	180	9
HISTÒRIA	351	7	200	9
INDÚSTRIA	357	8	178	9
JUSTÍCIA I ORDRE PÚBLIC	379	11	202	13
LITERATURA	358	7	200	9
MEDI AMBIENT	364	12	189	12
MEDICINA I SANITAT	366	12	201	12
MISCEL·LÀNIA	296	13	159	13
MITJANS DE COMUNICACIÓ	360	9	183	9
MÚSICA	360	10	201	10
OBRES PÚBLIQUES I INFRAESTRUCTURES	361	10	190	10
POLÍTICA	380	10	217	12
RECURSOS TERRITORIALS	346	7	180	9
RELACIONS INTERNACIONALS	354	8	181	9
RELIGIÓ	357	8	185	9
SOCIETAT	365	12	212	12
TAUROMÀQUIA	350	6	169	8
TERRORISME	353	9	186	9
TRANSPORTS I COMUNICACIONS	366	10	182	10
TREBALL	356	8	186	10
TURISME	377	9	196	9
URBANISME I HABITATGE	354	10	188	10

Tabla 4.3: Promedio de palabras y caracteres por palabra del *corpus* con y sin *stopwords*

je de muestras de las clases mayoritarias seguirán siendo muy elevado comparado con el de las minoritarias. Se podría optar a realizar *undersampling*, lo que implicaría equiparar el porcentaje de cada clase al de las minoritarias eliminando muestras de clases mayoritarias de forma aleatoria, pero se ha descartado esta opción ya que llevaría información sensible y útil para el problema. Por otra parte, tampoco se ha realizado *oversampling* ya que la generación de muestras sintéticas para problemas de procesamiento de lenguaje natural es una tarea complicada, ya que implicaría la generación automática de texto para este *corpus* concretamente y esto a su abarcaría un proyecto totalmente distinto y único.

Clase	1-grama	2-grama	3-grama
ACCIDENTS I CATÀS.	persones	persones mort	persones perdut vida
ACTUALITAT ROSA	anys	nova york	penélope cruz javier
AGRICUL., RAM. I PES.	comunitat	comunitat valenci.	transvasament tajo segura
ARTS	València	siguen feliços	museu belles arts
CIÈNCIA I INVESTIGACIÓ	València	estació espacial	estació espacial intern.
CINEMA	pellícula	festival cine	ciutat llum alacant
CONFLICTES ARMATS	mort	nord americans	soldats nord americans
CULTURA	València	fa anys	museu belles arts
DEFENSA	militar	forces armades	dia forces armades
ECO., COMERÇ I FIN.	euros	millions euros	banc central europeo
ENERGIA	energia	comunitat valenci.	president gene. francesc
ENSENYAMENT	alumnes	comunitat valenci.	alejandro font mora
ESPECTACLES	teatre	teatre principal	teatre principal València
ESPORTS	València	vila real	vila real cf
FENÒMENS NATURALS	neu	comunitat valenci.	litres metre quadrat
FESTES I TRADICIONS	festa	mare déu	festes moros i cristians
GASTRONOMIA	cuina	quique dacosta	universitat politècnica valè.
HISTÒRIA	anys	fa anys	guerra civil espanyola
INDÚSTRIA	empresa	comunitat valenci.	factoria ford almussafes
JUSTÍCIA I ORDRE PÚB.	anys	guàrdia civil	demana anys pressó
LITERATURA	premi	miguel hernández	isabel clara simó
MEDI AMBIENT	aigua	medi ambient	conselleria medi ambient
MEDICINA I SANITAT	persones	comunitat valenci.	hospital fe València
MISCEL·LÀNIA	reuters	comunitat valenci.	banda musica fotos
MITJANS DE COMU.	punt	radiotelevi. valenci.	valenciana mitjans comu.
MÚSICA	música	palau música	palau música València
OBRES PÚBLIQUES I INF.	obres	milions euros	transva. xúquer vinalopó
POLÍTICA	govern	partit popular	luis rodríguez zapatero
RECURSOS TERRITORIALS	aigua	comunitat valenci.	confed. hidrogràfica xúquer
RELACIONS INTER.	president	unió europea	josé maría aznar
RELIGIÓ	papa	joan pau	joan pau ii
SOCIETAT	València	comunitat valenci.	conselleria benestar social
TAUROMÀQUIA	bous	plaça bous	plaça bous València
TERRORISME	eta	banda terrorista	banda terrorista eta
TRANSPORTS I COMU.	València	comunitat valenci.	direcció general trànsit
TREBALL	treball	comunitat valenci.	enquesta població activa
TURISME	comunitat	comunitat valenci.	ciutat arts ciències
URBANISME I HABITATGE	València	comunitat valenci.	vivendes protecció oficial

Tabla 4.4: N-gramas más frecuentes del *corpus* sin *stopwords*

Clase	% 1-gramas	% 2-gramas	% 3-gramas
ACCIDENTS I CATÀS.	16,97 %	40,28 %	64,94 %
ACTUALITAT ROSA	2,59 %	21,54 %	52,07 %
AGRICUL., RAM. I PES.	12,68 %	36,84 %	62,72 %
ARTS	10,09 %	34,55 %	61,35 %
CIÈNCIA I INVESTIGACIÓ	9,60 %	31,64 %	59,54 %
CINEMA	14,86 %	42,09 %	67,60 %
CONFLICTES ARMATS	11,38 %	35,80 %	61,90 %
CULTURA	9,17 %	30,85 %	56,91 %
DEFENSA	4,29 %	22,67 %	50,99 %
ECO., COMERÇ I FIN.	12,98 %	36,66 %	62,04 %
ENERGIA	4,72 %	23,48 %	51,39 %
ENSENYAMENT	8,08 %	29,57 %	57,46 %
ESPECTACLES	8,92 %	31,65 %	58,97 %
ESPORTS	36,02 %	65,80 %	83,28 %
FENÒMENS NATURALS	11,19 %	35,01 %	61,82 %
FESTES I TRADICIONS	14,90 %	40,00 %	65,26 %
GASTRONOMIA	8,86 %	33,37 %	62,82 %
HISTÒRIA	5,69 %	26,04 %	55,53 %
INDÚSTRIA	6,96 %	28,13 %	55,13 %
JUSTÍCIA I ORDRE PÚB.	23,06 %	49,18 %	71,25 %
LITERATURA	6,17 %	28,62 %	57,62 %
MEDI AMBIENT	12,64 %	35,25 %	61,03 %
MEDICINA I SANITAT	15,81 %	39,14 %	64,28 %
MISCEL·LÀNIA	3,05 %	16,57 %	42,07 %
MITJANS DE COMU.	6,91 %	26,90 %	54,32 %
MÚSICA	13,35 %	38,29 %	63,65 %
OBRES PÚBLIQUES I INF.	6,07 %	24,69 %	51,25 %
POLÍTICA	17,54 %	44,25 %	67,27 %
RECURSOS TERRITORIALS	2,28 %	17,77 %	48,59 %
RELACIONS INTER.	5,97 %	26,82 %	53,94 %
RELIGIÓ	7,81 %	28,49 %	55,81 %
SOCIETAT	18,00 %	41,04 %	64,09 %
TAUROMÀQUIA	3,04 %	20,79 %	51,25 %
TERRORISME	10,11 %	31,57 %	56,96 %
TRANSPORTS I COMU.	9,70 %	31,18 %	57,26 %
TREBALL	6,88 %	27,08 %	53,66 %
TURISME	8,73 %	29,34 %	56,41 %
URBANISME I HABITATGE	6,98 %	26,79 %	53,76 %
Media	10,37 %	32,36 %	58,85 %

Tabla 4.5: Porcentaje de n-gramas únicos del *corpus* con *stopwords*

Clase	% 1-gramas	% 2-gramas	% 3-gramas
ACCIDENTS I CATÀS.	17,38 %	60,77 %	92,04 %
ACTUALITAT ROSA	2,66 %	52,34 %	90,83 %
AGRICUL., RAM. I PES.	12,78 %	60,40 %	92,16 %
ARTS	10,16 %	59,29 %	90,84 %
CIÈNCIA I INVESTIGACIÓ	9,69 %	57,81 %	91,54 %
CINEMA	14,95 %	66,55 %	94,46 %
CONFLICTES ARMATS	11,50 %	59,62 %	92,59 %
CULTURA	9,22 %	53,91 %	87,88 %
DEFENSA	4,38 %	48,71 %	88,33 %
ECO., COMERÇ I FIN.	13,06 %	57,65 %	90,86 %
ENERGIA	4,80 %	48,79 %	87,67 %
ENSENYAMENT	8,15 %	53,68 %	90,26 %
ESPECTACLES	8,99 %	56,14 %	90,46 %
ESPORTS	36,10 %	81,64 %	97,80 %
FENÒMENS NATURALS	11,52 %	56,76 %	91,43 %
FESTES I TRADICIONS	14,97 %	62,20 %	92,82 %
GASTRONOMIA	8,95 %	61,82 %	93,76 %
HISTÒRIA	5,78 %	54,25 %	90,68 %
INDÚSTRIA	7,05 %	52,17 %	88,64 %
JUSTÍCIA I ORDRE PÚB.	23,45 %	67,61 %	93,62 %
LITERATURA	6,29 %	57,69 %	91,02 %
MEDI AMBIENT	12,71 %	58,57 %	91,02 %
MEDICINA I SANITAT	15,91 %	60,93 %	92,42 %
MISCEL·LÀNIA	3,13 %	38,45 %	80,25 %
MITJANS DE COMU.	6,99 %	51,33 %	88,81 %
MÚSICA	13,43 %	61,84 %	92,85 %
OBRES PÚBLIQUES I INF.	6,14 %	47,04 %	84,62 %
POLÍTICA	17,69 %	63,74 %	91,88 %
RECURSOS TERRITORIALS	2,39 %	46,96 %	87,82 %
RELACIONS INTER.	6,22 %	50,79 %	88,43 %
RELIGIÓ	7,89 %	55,39 %	90,27 %
SOCIETAT	18,07 %	59,54 %	90,90 %
TAUROMÀQUIA	3,27 %	51,68 %	89,64 %
TERRORISME	10,34 %	53,19 %	88,83 %
TRANSPORTS I COMU.	9,78 %	52,25 %	88,84 %
TREBALL	6,97 %	49,12 %	88,21 %
TURISME	8,80 %	51,13 %	89,57 %
URBANISME I HABITATGE	7,04 %	50,12 %	88,38 %
Media	10,49 %	56,10 %	90,33 %

Tabla 4.6: Porcentaje de n-gramas únicos del *corpus* sin *stopwords*

CAPÍTULO 5
Dominio del problema, análisis de resultados y aplicaciones

Una vez realizadas todas las aproximaciones teóricas al *machine learning* así como la representación del lenguaje y la elaboración del *corpus*, resta realizar la experimentación final. En el presente capítulo se presenta el entorno y las tecnologías utilizadas durante el proyecto para realizar la experimentación del mismo. Así mismo, se focaliza el problema y se exponen los diversos resultados obtenidos así como sus análisis para finalmente relacionar estos con la tarea inicial requerida por la Corporació Valenciana de Mitjans de Comunicació.

5.1 Tecnologías utilizadas

En este sección se presenta el entorno en el cual se ha desarrollado la experimentación del proyecto así como las diferentes librerías utilizadas, algunas de las cuales han sido ya nombradas y explicadas superficialmente en capítulos anteriores. Finalmente, se exponen una serie de librerías que en la planificación inicial del proyecto se contemplaba su utilización y que finalmente han sido descartadas, así como el porqué de su no utilización.

5.1.1. Entorno de trabajo

El desarrollo de la experimentación ha sido llevado a cabo en diferentes dispositivos, pero todos ellos con un entorno común. Como sistema operativo se ha utilizado Ubuntu 20.04, la última versión de este sistema operativo de software libre y código abierto basado en Linux.

Como lenguaje de programación se ha optado por la última versión estable disponible a fechas de realización del proyecto, la cual es Python 3.8 [25], en un entorno virtual gestionado por el módulo PyEnv [26] para evitar problemas de compatibilidad entre versiones de librerías de diferentes proyectos. La elección de este lenguaje se fundamenta en la gran importancia que ha tenido en los últimos años el lenguaje en el desarrollo del *machine learning* y el tratamiento de datos, así como en el gran número de librerías con las que cuenta para estas tareas, que serán descritas a continuación.

Finalmente, se ha escogido Visual Studio Code como entorno de desarrollo, o también llamado IDE, por su rapidez y versatilidad a la hora de gestionar proyectos de este tipo.

5.1.2. Librerías utilizadas

La librería NumPy [27] añade soporte para grandes conjuntos de datos representados en vectores y matrices multidimensionales, así como una serie de funciones para operar sobre ellos. De esta forma se solventa el problema de eficiencia de Python base ya que no tiene soporte para matrices multidimensionales, sino que internamente son representadas como listas de listas.

Pandas [28] es una librería para la manipulación y análisis de datos basada en las estructuras desarrolladas por NumPy, la cual permite representar conjunto de datos de forma estructurada como un *dataframe*, el cual permite una gestión más estandarizada, intuitiva y eficaz de estos.

Una librería ya nombrada anteriormente es Scikit-learn [29], o también conocida como Sklearn. Se trata de una librería para Python orientada al *machine learning* y al análisis de datos que incluye algoritmos de clasificación, regresión y *clustering* así como diferentes técnicas para el tratamiento de datos y su representación. Internamente, también utiliza la librería NumPy para representar las estructuras y cuenta con soporte para la utilización de *dataframes* de Pandas.

La librería FastText [11] también ha sido comentada en anteriores capítulos. Se trata de una biblioteca para la representación del lenguaje mediante *embeddings* así como la clasificación de textos, tanto de manera supervisada como no supervisada, mediante clasificadores basados en redes neuronales.

Una librería auxiliar no utilizada directamente para la obtención de resultados, sino para la representación de estos es Matplotlib [30]. Ofrece la posibilidad de generar diferentes tipos de gráficos tanto estáticos como dinámicos a partir de conjuntos de datos. Como las demás librerías, utiliza internamente NumPy para un uso optimizado de los recursos.

Seaborn [31] es otra librería gráfica basada en Matplotlib para la parte de visualización y en Pandas para la parte de representación de los datos. Ofrece gráficas avanzadas aplicadas a grandes conjuntos de datos que permiten explorarlos de forma más intuitiva, así como su mejor comprensión.

5.1.3. Librerías descartadas

En la planificación inicial del proyecto se contaba con diversas librerías que finalmente no han podido ser utilizadas. En esta sección se describen brevemente el propósito de estas y el motivo de su descarte para realizar la experimentación del proyecto.

Natural Language Toolkit [32], o simplemente NLTK, es la principal librería en procesamiento del lenguaje natural la cual cuenta con más de cincuenta *corpus* y recursos léxicos para diversos idiomas, así como algoritmos de clasificación y técnicas de procesamiento como *tokenización*, *stemming* y razonamiento semántico, entre otras técnicas. Pese a su gran potencial, esta librería ha sido descartada por qué no cuenta con soporte para la lengua catalana.

Mientras que NLTK está orientada al mundo académico y la investigación, spaCy [33] es una librería de procesamiento del lenguaje natural orientada para el desarrollo de *software* y para el despliegue de modelos en producción. Actualmente, soporta más de 64 lenguajes, entre ellos el catalán, y cuenta con 55 *pipelines* de producción. Además, cuenta con diversos modelos preentrenados de *machine learning* basados en la técnica de redes neuronales de *transformers* así como diversos *embeddings* también preentrenados. Por el contrario, su soporte al catalán se ve limitado a tareas básicas ya que ninguna de

las *pipelines* ni modelos preentrenados son para esta lengua, por lo que finalmente se ha decidido descartar la librería.

Finalmente, la librería HuggingFace [34] proporciona modelos preentrenados vanguardistas basados en *transformers* para diversas tareas específicas y lenguajes. Cuenta con modelos para catalán para tareas como reconocimiento del habla, pero no para clasificación por lo que se ha tenido que descartar finalmente.

5.2 Problema de clasificación multiclase

Los problemas de clasificación supervisada son uno de las tareas más comunes dentro del campo del aprendizaje automático. Tener un conjunto de datos etiquetados nos permite a priori conocer el nombre exacto de clases y por tanto abordar el problema de forma más exacta. El problema más simple es el de clasificación binaria en el que una muestra se etiqueta en una clase u otra, pero en la mayoría de problemas no se cuenta únicamente con dos, sino que el número de clases suele ser mayor; hablamos de entonces de problemas de clasificación multiclase.

La forma de abordar este tipo de problemas es similar a la binaria ya que la mayoría de algoritmos de aprendizaje automático contempla la existencia de más de dos clases o existen variantes de estos que abordan el problema de forma especial como, por ejemplo, dividiendo el problema en subproblemas de clasificación binaria. Así mismo, las métricas utilizadas por los modelos de clasificación binaria son extrapolables a los de clasificación multiclase. Es más, son utilizadas métricas específicas para dichos modelos que describen mejor el comportamiento del sistema ante los datos.

Idealmente, el número de muestras entre clases debería mantener una proporción similar. En la práctica es habitual encontrase con lo contrario, que el porcentaje de muestras de cada clase varía, la cual cosa no debería suponer un gran problema para los algoritmos de aprendizaje automático. El problema surge cuando esta proporción es demasiado distinta entre clases, es entonces un problema de clasificación multiclase desbalanceado.

El principal inconveniente de este problema es la especialización que pueden sufrir los modelos a entrenar con conjuntos de datos etiquetados donde la mayoría de estos pertenezcan a una o más clases ya que, al especializarse en un subconjunto de clases, alcanzará un error de *bias* elevado. La elección de las métricas correctas es vital para detectar y subsanar este comportamiento.

Además, si no se realiza una correcta partición de los datos para los conjuntos de entrenamiento y test, es posible que en alguno de ellos no existan muestras de alguna de las clases, por lo que el modelo o bien no las tendrá en cuenta para modelar el problema al entrenar, o bien no se estará evaluando el rendimiento del problema con el dominio completo de este.

5.3 Experimentación y análisis de resultados

En la presente sección se encuentran los resultados obtenidos durante la experimentación del proyecto. La intención del mismo no es obtener los mejores resultados posibles, sino realizar una primera aproximación sobre la que poder realizar futuros experimentos, así como el sistema de ayuda a la clasificación para la CVMC.

Los diferentes experimentos realizados varían el modelo utilizado, la talla del n-grama, la representación del lenguaje, el número de clases y la utilización o no de *stopwords*. Los parámetros utilizados en cada modelo son los por defecto en cada implemen-

tación y no se ha realizado *hyperparameter tuning*, o ajuste de hiperparámetros, ya que, como se ha comentado, la experimentación es una aproximación inicial al problema. Como se ha comentado en capítulos anteriores, las diversas experimentaciones se han realizado sobre las mismas particiones de entrenamiento y test para así poder comparar resultados de forma más exacta.

La experimentación comienza con modelos más clásicos del aprendizaje automático y representando los datos con el modelo TF-IDF. El modelo utilizado de NB es el *Gaussian Naive Bayes* [6] con un valor de *variance smoothing*, o suavizado de varianza, por defecto de 1e-09. Por su parte, el modelo de SVM utilizado es el *Stochastic Gradient Descent* SVM [8] con función de pérdida del tipo *hinge*, con término de regularización *l2* y valor de *alpha* de 0.0001.

La Tabla 5.1 se presentan los resultados obtenidos de *recall* macro de los dos anteriores modelos con representación TF-IDF, con diferentes tallas de n-gramas y número de clases para el *corpus* con *stopwords*. En la primera columna se indica el clasificador y la talla de n-gramas mientras que en las tres restantes se muestran los resultados para 4, 6 y 38 clases. Estás 4 clases no son aleatorias, sino que son las mayoritarias del *corpus*, las cuales corresponden a "ESPORTS", "JUSTÍCIA I ORDRE PÚBLIC", "POLÍTICA" y "SOCIETAT". Así mismo, las 6 clases son las más mayoritarias por lo que son las anteriores más "ACCIDENTS I CATÀSTROFES" y "MEDICINA I SANITAT". Finalmente, las 38 clases representan al *corpus* completo. De forma análoga, la Tabla 5.2 muestra las estadísticas para el *corpus* sin *stopwords*.

Clasificador	$\mathcal{C}$=4	$\mathcal{C}$=6	$\mathcal{C}$=38
Naive Bayes (uni-gramas)	81,74 %	74,98 %	14,22 %
Naive Bayes (bi-gramas)	72,32 %	61,41 %	10,72 %
Naive Bayes (tri-gramas)	75,79 %	64,43 %	11,45 %
SVM (uni-gramas)	**88,84 %**	87,74 %	53,09 %
SVM (bi-gramas)	80,16 %	**79,77 %**	56,03 %
SVM (tri-gramas)	53,15 %	61,82 %	**70,32 %**

Tabla 5.1: *Recall* macro por modelo, por n-grama y por clases del *corpus* con *stopwords*

Clasificador	$\mathcal{C}$=4	$\mathcal{C}$=6	$\mathcal{C}$=38
Naive Bayes (uni-gramas)	84,66 %	80,50 %	19,13 %
Naive Bayes (bi-gramas)	84,79 %	76,11 %	15,56 %
Naive Bayes (tri-gramas)	77,44 %	62,24 %	9,44 %
SVM (uni-gramas)	**88,59 %**	87,52 %	54,35 %
SVM (bi-gramas)	55,14 %	55,71 %	72,55 %
SVM (tri-gramas)	25,54 %	**89,08 %**	**85,53 %**

Tabla 5.2: *Recall* macro por modelo, por n-grama y por clases del *corpus* sin *stopwords*

Con la reducción del problema a 4 y 6 clases se pretendía probar en fases iniciales de la experimentación si era posible diferenciar las clases con más ocurrencias. El comportamiento del clasificador NB ha sido el esperado ya que, con conjuntos reducidos de clases, el *recall* macro es superior al 70 %, mejorando en el *corpus* sin *stopwords* pero empeorando al aumentar la talla de n-gramas en este. Por el contrario, al entrenar con el *corpus* completo, la métrica se desploma hasta valores inferiores al 20 %.

Por su parte, el comportamiento de la SVM ha resultado ser bastante anómalo. Para un número de clases determinado y una talla de n-gramas observamos que a veces mejora o empeora el resultado, sin ninguna correlación aparente. Así mismo, se obtiene un

valor para 4 clases, con tri-gramas y sin *stopwords* de 25,54 % mientras que, para la misma configuración y 38 clases, se alcanza un 85,53 % de *recall* macro. En el primer caso se produce seguramente porque el modelo se especializa en la clase mayoritaria mientras que en el segundo parece ser que el modelo ha sufrido *overfitting*. Un análisis más exhaustivo de este comportamiento será parte de un trabajo futuro.

De esta primera experimentación se extrae que a *priori* sí que es posible realizar una diferenciación entre clases, pero es necesario otros algoritmos y representaciones de lenguaje para alcanzarla con el conjunto total de clases. Para las próximas experimentaciones se ha descartado el uso del clasificador NB ya que sus resultados son paupérrimos e inferiores a la SVM en prácticamente todas las configuraciones.

En las siguientes experimentaciones se ha dejado de utilizar la representación TF-IDF para utilizar la generada para el modelo FastText. El modelo ha sido entrenado directamente con el *corpus* para aprender la representación en *embeddings*, sin utilizar modelos preentrenados. Se ha utilizado un tamaño de vector 300 ya que, como se ha comentado en capítulos anteriores, la talla media de las muestras para el conjunto con *stopwords* es de entre 320 y 370 palabras y de entre 175 y 150 para el conjunto sin *stopwords*, por lo que se ha considerado que la talla escogida era suficiente para captar las relaciones del lenguaje sin aumentar el tiempo de entrenamiento de forma excesiva que implica aumentar la talla.

En los próximos resultados se incluye también la métrica *accuracy-at-k* con k entre 2 y 5, a parte de la ya utilizada *recall* macro, para así conocer el porcentaje de veces que la clase correcta está entre las k primeras opciones predichas por un modelo. De tal manera que un modelo puede devolver la clase predicha para una muestra, se le puede solicitar que devuelva un vector de probabilidades de que esa muestra pertenezca a cada clase.

En la Tabla 5.3 se pueden observar las métricas para la SVM con representación por *embeddings* para diferentes tallas de n-grama del *corpus* con *stopwords* y para únicamente 4 clases. De forma análoga se pueden observar las mismas estadísticas para el *corpus* sin *stopwords* en la Tabla 5.4. En comparación con la representación por TF-IDF, los resultados obtenidos son más coherentes. Eliminar las *stopwords* mejora la precisión y aumentar la talla de los n-gramas también mejora las métricas. Además, el porcentaje de *recall* es mayor en todos los casos. El *accuracy-at-k* de 4 y 5 aporta unas métricas del 100 % ya que es un número mayor o igual al número de clases consideradas.

Clasificador	Recall	Acc-at-2	Acc-at-3	Acc-at-4	Acc-at-5
SVM (uni-gramas)	91,57 %	98,97 %	99,81 %	100,00 %	100,00 %
SVM (bi-gramas)	94,46 %	99,29 %	99,83 %	100,00 %	100,00 %
SVM (tri-gramas)	94,56 %	99,24 %	99,81 %	100,00 %	100,00 %

Tabla 5.3: Estadísticas del clasificador SVM con *embeddings*, con 4 clases y por n-gramas del *corpus* con *stopwords*

Clasificador	Recall	Acc-at-2	Acc-at-3	Acc-at-4	Acc-at-5
SVM (uni-gramas)	93,87 %	99,26 %	99,87 %	100,00 %	100,00 %
SVM (bi-gramas)	96,99 %	99,53 %	99,90 %	100,00 %	100,00 %
SVM (tri-gramas)	97,04 %	99,53 %	99,89 %	100,00 %	100,00 %

Tabla 5.4: Estadísticas del clasificador SVM con *embeddings*, con 4 clases y por n-gramas del *corpus* sin *stopwords*

En la Tabla 5.5 se pueden observar los resultados de la misma configuración para 6 clases para el conjunto con *stopwords* y de forma semejante en la Tabla 5.6 para el conjunto

sin *stopwords*. Como se observa, los resultados se comportan de forma similar a los resultados obtenidos para 4 clases, mejorando sin *stopwords* y, generalmente, aumentando la talla de n-gramas.

Clasificador	Recall	Acc-at-2	Acc-at-3	Acc-at-4	Acc-at-5
SVM (uni-gramas)	88,52 %	98,20 %	99,40 %	99,75 %	99,94 %
SVM (bi-gramas)	91,98 %	98,80 %	99,58 %	99,83 %	99,94 %
SVM (tri-gramas)	92,10 %	98,64 %	99,44 %	99,77 %	99,92 %

Tabla 5.5: Estadísticas del clasificador SVM con *embeddings*, con 6 clases y por n-gramas del *corpus* con *stopwords*

Clasificador	Recall	Acc-at-2	Acc-at-3	Acc-at-4	Acc-at-5
SVM (uni-gramas)	91,54 %	98,75 %	99,59 %	99,84 %	99,96 %
SVM (bi-gramas)	95,96 %	99,21 %	99,68 %	99,86 %	99,94 %
SVM (tri-gramas)	95,70 %	99,19 %	99,67 %	99,86 %	99,95 %

Tabla 5.6: Estadísticas del clasificador SVM con *embeddings*, con 6 clases y por n-gramas del *corpus* sin *stopwords*

Finalmente, para la SVM, se ha obtenido las estadísticas para el conjunto completo de clases para las mismas configuraciones anteriores. En la Tabla 5.7 se observan las estadísticas con *stopwords* donde el *recall* baja al 46,41 % con uni-gramas y, a diferencia de los anteriores casos, este porcentaje baja al aumentar la talla de n-gramas mientras que el *accuracy-at-k* ronda entre el 75 % y el 89,50 %, por lo que el modelo casi siempre predice la clase mayoritaria entre las 5 más probables. De forma análoga se pueden observar las estadísticas para el conjunto sin *stopwords* en la Tabla 5.8 donde las métricas se comportan de igual manera pero con un ligero incremento de estas, logrando alcanzar casi un *recall* macro de 55 % para uni-gramas y más del 90 % para *accuracy-at-4*.

Clasificador	Recall	Acc-at-2	Acc-at-3	Acc-at-4	Acc-at-5
SVM (uni-gramas)	46,41 %	85,98 %	88,15 %	88,97 %	89,47 %
SVM (bi-gramas)	38,10 %	81,62 %	86,45 %	87,72 %	88,50 %
SVM (tri-gramas)	31,16 %	75,71 %	80,37 %	82,15 %	83,31 %

Tabla 5.7: Estadísticas del clasificador SVM con *embeddings*, con 38 clases y por n-gramas del *corpus* con *stopwords*

Clasificador	Recall	Acc-at-2	Acc-at-3	Acc-at-4	Acc-at-5
SVM (uni-gramas)	54,87 %	88,67 %	88,72 %	90,61 %	91,57 %
SVM (bi-gramas)	52,29 %	89,73 %	91,04 %	91,77 %	92,15 %
SVM (tri-gramas)	45,41 %	85,93 %	88,29 %	89,91 %	90,41 %

Tabla 5.8: Estadísticas del clasificador SVM con *embeddings*, con 38 clases y por n-gramas del *corpus* sin *stopwords*

En la Figura 5.1 y la Figura 5.2 se puede ver la representación gráfica de las matrices de confusión para los resultados del modelo para el *corpus* con *stopwords* y sin *stopwords* respectivamente. Idealmente, la diagonal debería teñirse de colores claros próximos al amarillo mientras que el resto de casillas tener un color oscuro. Se puede observar que hay algunas clases que se predicen bastante bien como "CINEMA", "ESPORTS" o "MEDICINA I SANITAT" mientras que otras como "TAUROMÀQUIA" o "RECURSOS TERRITORIALS" son incapaces para el modelo de detectar. También se observa que la

mayoría de clases se confunden parcialmente con "JUSTICIA I ORDRE PÚBLIC" y "POLÍTICA".

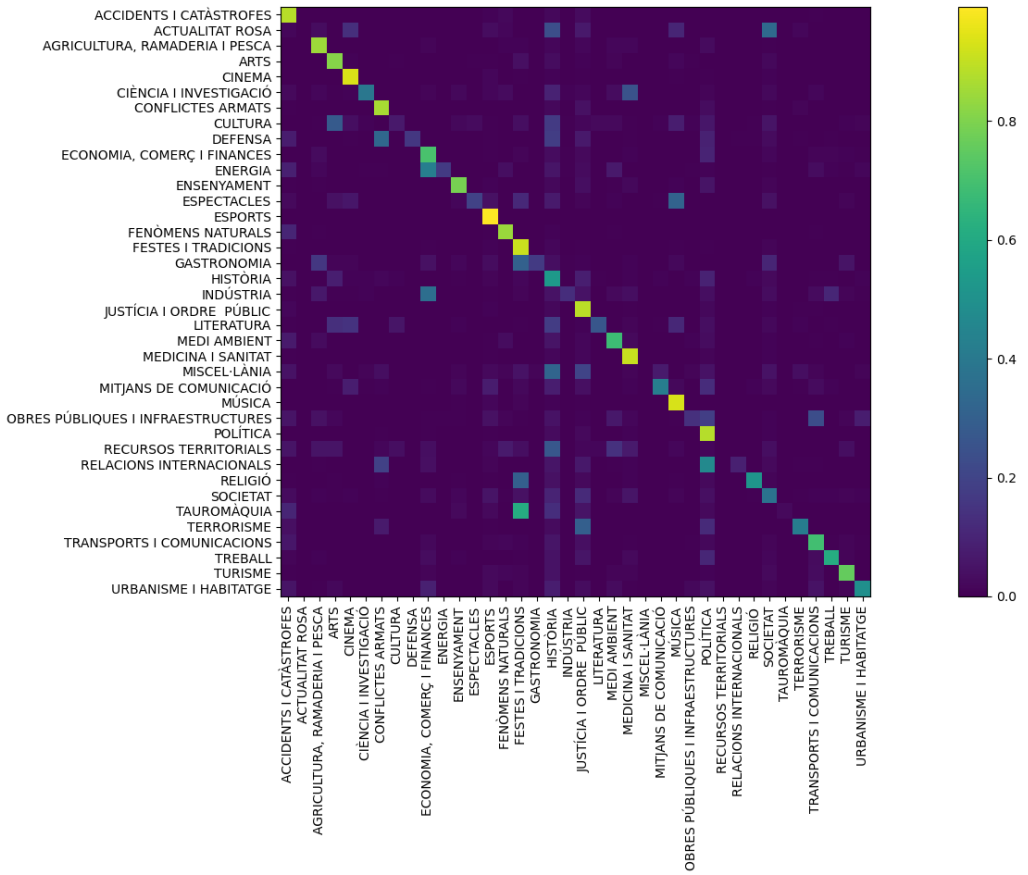

Figura 5.1: Matriz de confusión para el clasificador SVM con *stopwords*

Los próximos resultados a analizar corresponden a los obtenidos por el modelo propio de FastText con parámetros por defecto, pero con talla de representación de los *embeddings* de 300, como el modelo anterior. En la Tabla 5.9 se observan diversas estadísticas extraídas entrenando con el conjunto con *stopwords* y de forma análoga para el conjunto sin *stopwords* en la Tabla 5.10.

El *recall* macro mejorar alrededor de un 9 % respecto al anterior modelo y el *accuracy-at-k* también mejora de forma general un 5 %. El comportamiento es similar al anterior modelo ya que al aumentar el tamaño de n-gramas empeoran las métricas, pero al utilizar el conjunto sin *stopwords* el resultado mejora considerablemente aumentando más del 9 % el *recall* y llegando al 63,58 %. Por su parte, se llega a alcanzar un 98,48 % de *accuracy-at-5*, por lo que el modelo casi siempre detecta la clase correcta entre sus primera cinco opciones.

Clasificador	Recall	Acc-at-2	Acc-at-3	Acc-at-4	Acc-at-5
FastText (uni-gramas)	54,47 %	89,75 %	93,35 %	95,21 %	96,32 %
FastText (bi-gramas)	49,17 %	87,33 %	90,96 %	93,10 %	94,60 %
FastText (tri-gramas)	40,88 %	83,11 %	87,45 %	90,01 %	91,85 %

Tabla 5.9: Estadísticas del clasificador FastText con *embeddings*, con 38 clases y por n-gramas del *corpus* con *stopwords*

En la Figura 5.3 y la Figura 5.4 se puede observar la representación gráfica de las matrices de confusión para los resultados del modelo para el conjunto con *stopwords* y

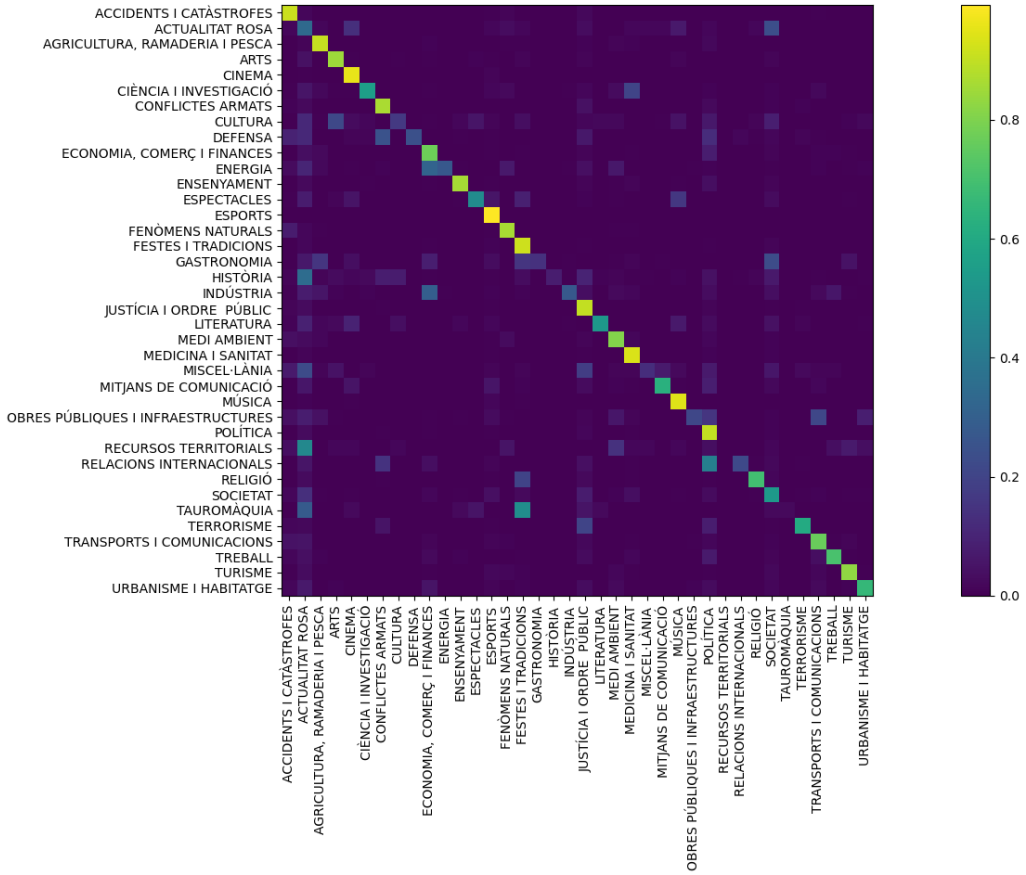

Figura 5.2: Matriz de confusión para el clasificador SVM sin *stopwords*

Clasificador	Recall	Acc-at-2	Acc-at-3	Acc-at-4	Acc-at-5
FastText (uni-gramas)	63,58 %	94,19 %	96,80 %	97,91 %	98,48 %
FastText (bi-gramas)	60,02 %	94,45 %	96,57 %	97,66 %	98,24 %
FastText (tri-gramas)	54,11 %	92,76 %	95,19 %	96,45 %	97,19 %

Tabla 5.10: Estadísticas del clasificador FastText con *embeddings*, con 38 clases y por n-gramas del *corpus* sin *stopwords*

sin *stopwords* respectivamente. Se observa que la diagonal toma ligeramente colores más claros, reflejando así el aumento de métricas, y por su parte se puede apreciar que la clase "SOCIETAT" se confunde de forma general.

Finalmente, se ha probado un modelo *Random Forests* con representación mediante *embeddings* de FastText, de tamaño 300 como los anteriores casos. En la Tabla 5.11 se muestran los resultados del conjunto con *stopwords* y en la Tabla 5.12 para el *corpus* sin *stopwords*, de forma similar a las anteriores tablas. Sin duda alguna, este modelo ha sido el que mejores resultados ha aportado de forma más consistente, no como los obtenidos por la SVM con representación TF-IDF.

Se puede apreciar que para este modelo no influye especialmente el uso o no de *stopwords* ni la talla de n-gramas. El mejor valor de *recall* macro se consigue con el uso de bi-gramas y sin *stopwords*, obteniendo un 89,30 %, siendo esta la mejor métrica de todos los experimentos para el conjunto completo de clases. Además, el *accuracy-at-2* ronda en todos los casos el 97 % por lo que el modelo casi siempre llega a predecir la clase correcta entre sus dos opciones más probables. Para el caso del *accuracy-at-5* se llega a alcanzar casi el 99 %.

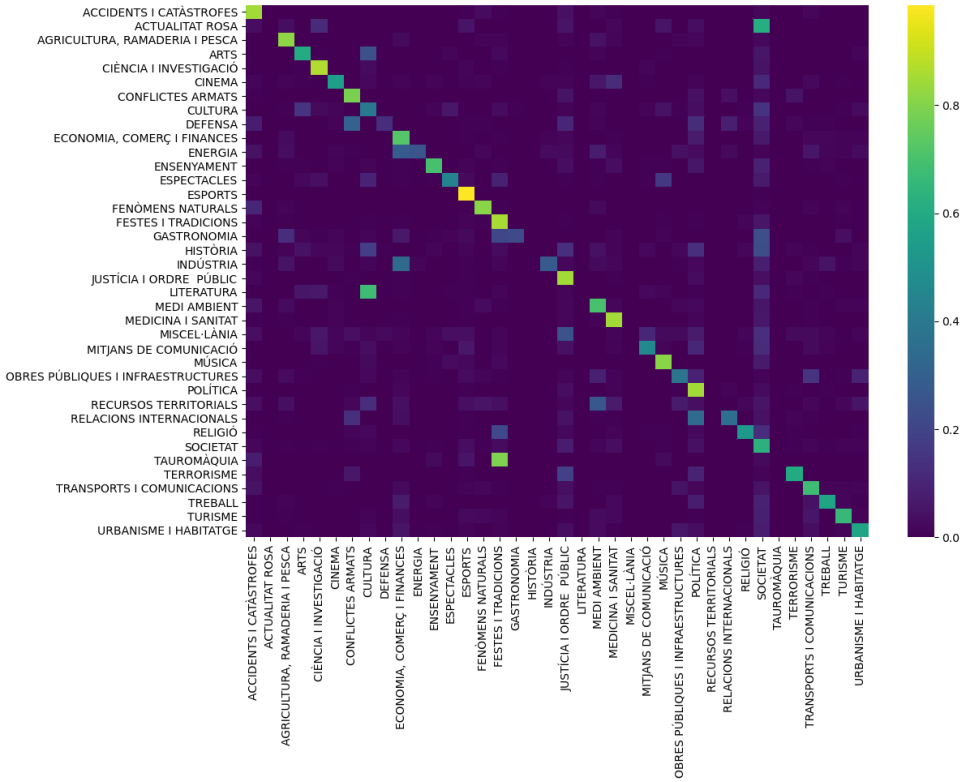

Figura 5.3: Matriz de confusión para el clasificador FastText con *stopwords*

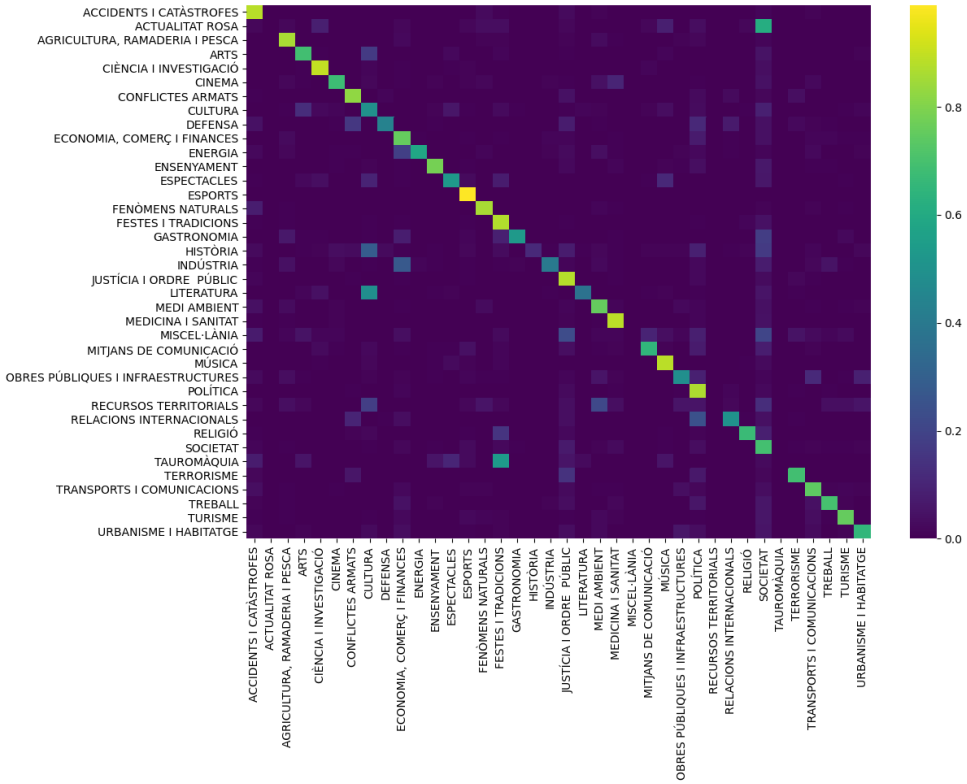

Figura 5.4: Matriz de confusión para el clasificador FastText sin *stopwords*

En la Figura 5.5 y la Figura 5.6 se observan las representación gráficas de las matrices de confusión para los resultados del modelo*Random Forests* para el conjunto con *stopwords*

Clasificador	Recall	Acc-at-2	Acc-at-3	Acc-at-4	Acc-at-5
Random Forests (uni-gramas)	88,10 %	97,23 %	98,24 %	98,75 %	99,03 %
Random Forests (bi-gramas)	87,90 %	97,18 %	98,16 %	98,67 %	98,94 %
Random Forests (tri-gramas)	87,40 %	96,95 %	97,91 %	98,38 %	98,70 %

Tabla 5.11: Estadísticas del clasificador *Random Forests* con *embeddings*, con 38 clases y por n-gramas del *corpus* con *stopwords*

Clasificador	Recall	Acc-at-2	Acc-at-3	Acc-at-4	Acc-at-5
Random Forests (uni-gramas)	88,70 %	97,09 %	98,06 %	98,60 %	98,92 %
Random Forests (bi-gramas)	89,30 %	97,21 %	98,14 %	98,58 %	98,81 %
Random Forests (tri-gramas)	88,90 %	97,05 %	97,95 %	98,38 %	98,63 %

Tabla 5.12: Estadísticas del clasificador *Random Forests* con *embeddings*, con 38 clases y por n-gramas del *corpus* sin *stopwords*

y sin *stopwords* respectivamente. En este último modelo se aprecia que la diagonal se tiñe casi en su totalidad de colores claros muy próximos al amarillo mientras que la tonalidad general fuera de la diagonal se suaviza y oscurece, respaldando así las métricas obtenidas. Como en los anteriores modelos, la clase "SOCIETAT" sigue siendo la que más se confunde entre clases.

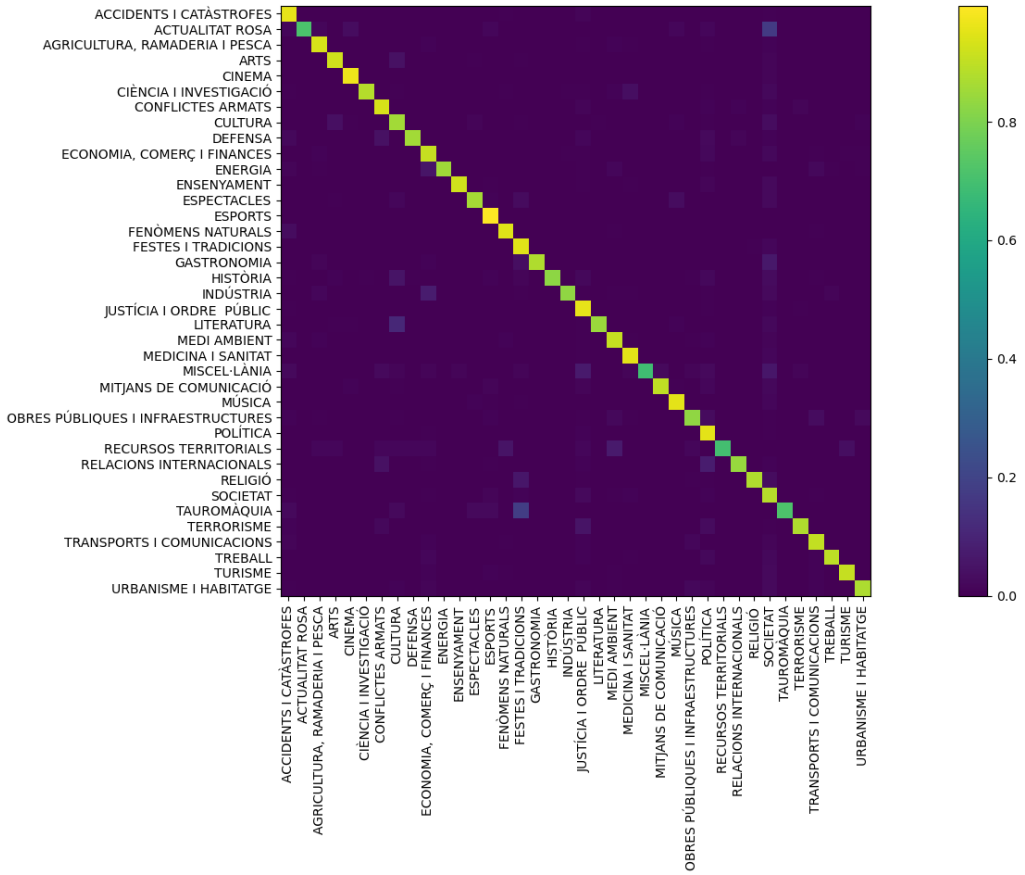

Figura 5.5: Matriz de confusión para el clasificador *Random Forests* con *stopwords*

Los resultados de la mejor métrica de *recall* macro obtenida para cada modelo y conjunto se puede contemplar de forma visual en la Figura 5.7, donde cada entrada indica el modelo al que corresponde y entre paréntesis la talla de n-gramas y el uso o no de *stopwords*, respectivamente.

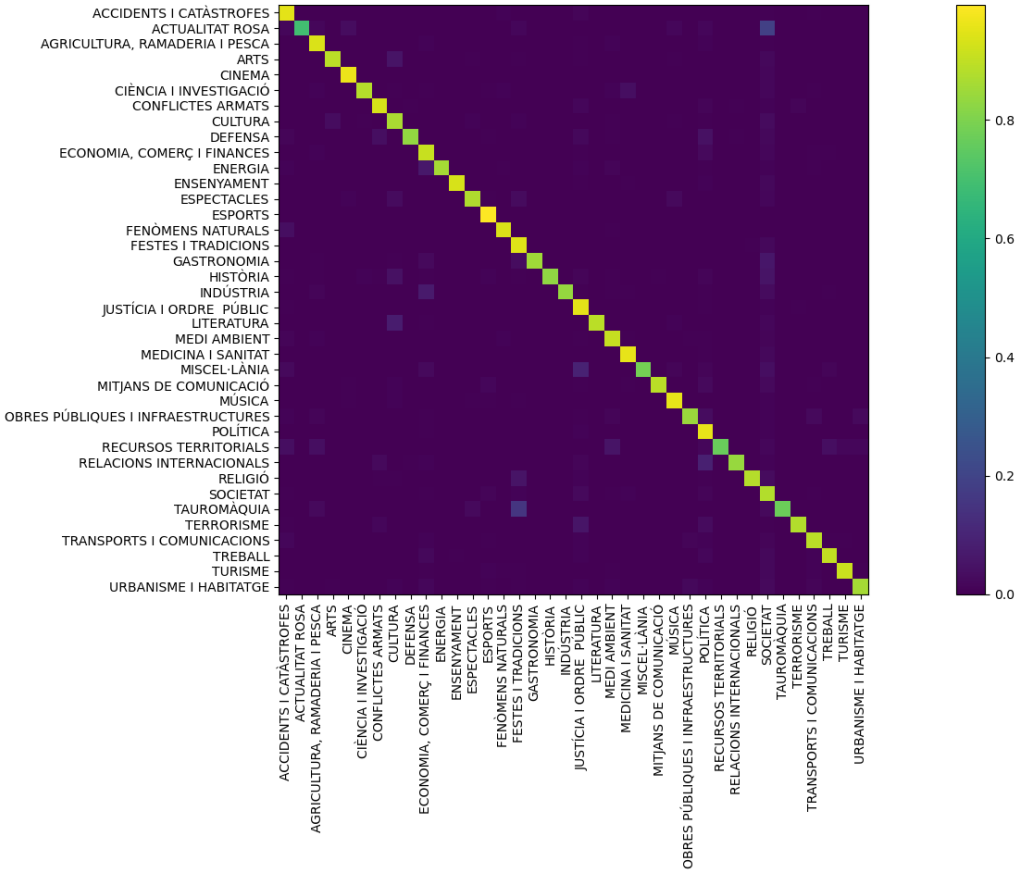

Figura 5.6: Matriz de confusión para el clasificador *Random Forests* sin *stopwords*

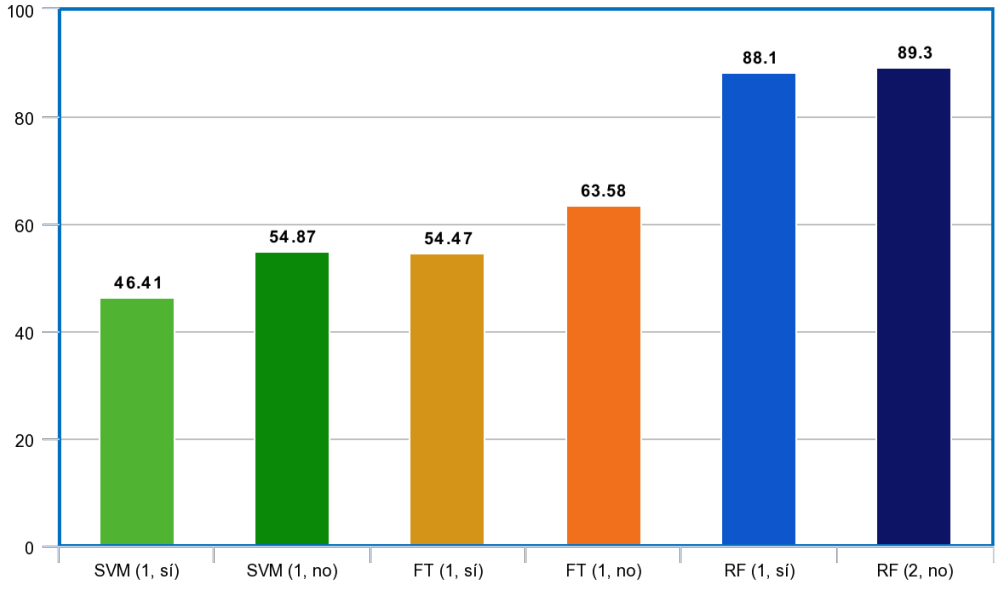

Figura 5.7: Comparativa de la mejor métrica obtenida por modelo

5.4 Sistema de ayuda a la clasificación

El objetivo final del proyecto es determinar si es posible realizar o no un sistema de ayuda a la catalogación para la CVMC para su futura implementación. Del análisis de resultados anterior se desprende que sí es posible la realización de dicha herramienta de

apoyo para el personal encargada de la tarea de catalogación, pero no así un sistema que sea capaz de llevar su función de forma autónoma.

Todos los modelos han aportado una métrica *accuracy-at-k* suficiente elevada que permite en la gran mayoría de casos encontrar la clase correcta entre las primeras cinco opciones con mayor probabilidad de cada modelo. Por el contrario, el *recall* no es suficientemente alto para poder realizar la tarea de forma autónoma pese que el modelo *Random Forests* obtiene una métrica bastante elevada de este, por lo que es una opción interesante para realizar futuras experimentaciones.

La herramienta propuesta recibe como entrada una noticia en el mismo formato que se almacena en el sistema interno de catalogación a la que se aplica los filtros de preprocesamiento desarrollados en capítulos anteriores. Posteriormente se predice con uno o diversos modelos para obtener un vector de probabilidades, en caso de que la predicción se haga con más de un modelo, se realizaría la suma de probabilidades o promedio entre ellas. Posteriormente, se devolvería a la persona que esté realizando la catalogación una lista ordenada por probabilidad de todas aquellas clases que superen un umbral. Dicho umbral no se escogería de forma aleatoria, sino que sería fruto de futuras experimentaciones para conseguir un valor óptimo que permita obtener una lista donde esté la clase correcta con gran fiabilidad pero que, a su vez, esta lista no contenga demasiadas clases y acabe entorpeciendo la labor del personal.

CAPÍTULO 6
Conclusiones y trabajo futuro

En este capítulo final se presentan las conclusiones obtenidas tras la realización completa del proyecto, así como una serie de propuestas y posibles mejoras a realizar en futuros trabajos que tenga como propósito seguir profundizando en este sistema.

6.1 Conclusiones

Este trabajo fin de grado ha tenido como objetivo descubrir si es posible la realización de un sistema de apoyo a la catalogación para la CVMC a partir de un conjunto de noticias cedido por la propia organización. Durante la realización del proyecto se ha implementado un sistema que es capaz de procesar el conjunto de noticias de la corporación y generar un *corpus* resultante con el que entrenar diversos modelos y obtener así unos primeros resultados para el problema.

Se ha logrado entrenar diversos modelos de aprendizaje automático tanto con una representación convencional como con una representación basada en redes neuronales como son los *embeddings*.

Como resultado de las diversas experimentaciones se ha concluido que sí es posible la realización de la herramienta descrita para sugerir al personal encargada de la tarea el conjunto de clases más probables para nuevas noticias, pero no así la realización de esta tarea de clasificación de forma automática ya que los resultados obtenidos no son lo suficientemente altos para permitir la clasificación sin supervisión manual.

Finalmente, la realización de este proyecto ha servido como una primera aproximación personal al campo del procesamiento de lenguaje natural, así como una forma de afianzar de manera práctica los conocimientos teóricos adquiridos durante la carrera y que resultarán útiles tanto en futuros proyectos personales como en el mundo laboral.

6.2 Trabajo futuro

La realización de un proyecto de *machine learning* sobre un conjunto de datos nunca antes utilizados para este propósito tiene sus ventajas e inconvenientes. Por una parte, no existe ningún tipo de análisis ni resultados previos con los que compararse ni por los que guiarse, por lo que la realización del proyecto se ha realizado a ciegas según el criterio propio y el de los tutores. Por otra parte, este escenario es el más común que se encuentra un *data scientist* en la empresa, cuando comienza un nuevo proyecto y no se cuenta con ningún resultado para comparar para ese proyecto en concreto.

Por el contrario, no contar con resultados previos, abre todo un abanico de posibilidades para probar cualquier modelo, representación y técnica que se desee. Además, también se pueden mejorar los resultados obtenidos y seguir ahondando en las peculiaridades de este *corpus*.

En primer lugar, una nueva experimentación a realizar está relacionada con las anomalías detectadas con los modelos de NB y SVM y la representación TF-IDF. Averiguar si el origen de estas está relacionado con la naturaleza del problema o la implementación de los modelos puede arrojar más información al problema o conocer mejor las limitaciones de estas librerías para trabajos futuros.

En segundo lugar, la realización de *hyperparameter tuning* es otra tarea común desempeñada en proyectos de aprendizaje automático para conseguir mejores resultados a los obtenidos por los modelos base. Por ello como futuro trabajo se contempla un ajuste de hiperparámetros para obtener mejores métricas, controlando siempre los errores de *bias* y *variance* para no acabar sufriendo *overfitting*.

En tercer lugar, en relación con los *embeddings*, se ha probado únicamente una talla de 300. El tamaño de estos vectores puede llegar a influir considerablemente en las métricas obtenidas, por lo que un ajuste de este parámetro en concreto es susceptible de ser fruto de trabajo futuro. Así mismo, la única representación por *embeddings* obtenida ha sido la producida por FastText por lo que se puede probar a generar representaciones con modelos distintos que generen dicha representación densa utilizando otras técnicas.

Además, existen infinidad de modelos e implementaciones diferentes con los que experimentar para intentar obtener mejores resultados. Es por ello, que en cuarto lugar se propone de una diversificación de los clasificadores a utilizar para obtener más resultados bases que comparar con especial interés en los basados en redes neuronales y, posteriormente, realizar también su *hyperparameter tuning*. Sería de especial interés poder abordar el problema con el estado del arte en procesamiento en lenguaje natural con los llamados *transformers* pero normalmente estos se basan en modelos preentrenados y, como se ha comentado en anteriores capítulos, el uso de *machine learning* para textos en catalán no está tan extendido como para textos en inglés o español.

En quinto lugar, relacionado con los filtros de preprocesamiento, se puede realizar diversas modificaciones. La creación de nuevos filtros, la modificación de los ya realizados o procesamientos avanzados, como puede ser la detección de nombres propios o combinaciones de estos para tratarlos como un token único en el texto y dotarles de la distinción que tienen. Así mismo, el texto se ha realizado convirtiendo todo el vocabulario a minúscula, por lo que realizar pruebas sin realizar este procesado es una opción posible.

Finalmente, la implementación del producto final, el cual es la herramienta de apoyo a la catalogación. Pese que esta implementación no entre dentro del alcance del proyecto, se puede realizar un prototipo de esta herramienta para realizar un proyecto software completo que abarque el campo del aprendizaje automático. Así mismo, la búsqueda del umbral de probabilidad que indique si la herramienta tiene o no que mostrar una clase es también una experimentación futura posible a realizar.

Bibliografía

[1] Aditya Koli, Diksha Khurana, Kiran Khatter, Sukhdev Singh, "Natural Language Processing: State of The Art, Current Trends and Challenges". 2017.

[2] Objetivos y metas de desarrollo sostenible. https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/.

[3] F. Casacuberta Nolla, E. Vidal Ruiz, "Tema 1: Introducción al Aprendizaje Automático" de Aprendizaje Automático. 2020-2021.

[4] Ejemplo de regresión lineal de scikit-learn. https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html.

[5] Jason Brownlee, "Gentle Introduction to the Bias-Variance Trade-Off in Machine Learning". https://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning/

[6] Scikit-learn: Machine Learning in Python, "GaussianNB". https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[7] F. Casacuberta Nolla, E. Vidal Ruiz, "Tema 4: Máquinas de vectores soporte" de Aprendizaje Automático, 2020-2021.

[8] Scikit-learn: Machine Learning in Python, "SGDClassifier". https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html#sklearn.linear_model.SGDClassifier

[9] Scikit-learn: Machine Learning in Python, "RandomForestClassifier". https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[10] Perceptron algorithm. https://en.wikipedia.org/wiki/Perceptron

[11] FastText: "Library for efficient text classification and representation learning". https://fasttext.cc/

[12] Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov, "Bag of Tricks for Efficient Text Classification", 2016.

[13] Abdul Arfat Mohammed, Venkatesh Umaashankar, "Effectiveness of Hierarchical Softmax in Large Scale Classification Tasks". 2018.

[14] Long Ouyang, Lera Boroditsky, "Semantic Coherence Facilitates Distributional Learning of Word Meanings". 2016.

[15] Feed-forward neural network. https://en.wikipedia.org/wiki/Feedforward_neural_network

[16] Xiaofeng Yuan, Yalin Wang, Lin Li, "Nonlinear Dynamic Soft Sensor Modeling With Supervised Long Short-Term Memory Network". 2019.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, "Efficient Estimation of Word Representations in Vector Space". 2013.

[18] Rohit Francois Chaubard, Richard Socher Mundra, "CS 224D: Deep Learning for NLP, Chapter 4.2". 2016.

[19] Rohit Francois Chaubard, Richard Socher Mundra, "CS 224D: Deep Learning for NLP, Chapter 4.3". 2016.

[20] Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov, "Enriching Word Vectors with Subword Information". 2017.

[21] L. de Yzaguirre i Maura, "Llista de mots buits del català". http://latel.upf.edu/morgana/altres/pub/ca_stop.htm

[22] Matthieu Jimenez, Maxime Cordy, Yves Le Traon, Mike Papadakis, "On the Impact of Tokenizer and Parameters on N-Gram Based Code Analysis". 2018.

[23] Enrique J. Carmona Suárez, "Tutorial sobre Máquinas de Vectores Soporte (SVM)". 2016.

[24] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jegou, Tomas Mikolov, "FastText.zip: Compressing text classification models". 2016.

[25] Python. https://www.python.org/

[26] PyEnv. https://github.com/pyenv/pyenv

[27] NumPy. https://numpy.org/

[28] Pandas. https://pandas.pydata.org/

[29] scikit-learn. https://scikit-learn.org/stable/

[30] Matplotlib. https://matplotlib.org/

[31] Seaborn. https://seaborn.pydata.org/

[32] Natural Language Toolkit. https://www.nltk.org/

[33] spaCy. https://spacy.io/

[34] HuggingFace. https://huggingface.co/

APÉNDICE A
Listas utilizadas en preprocesamiento

Este apéndice contiene las lista utilizadas en los distintos preprocesamientos.

A.1 Lista de *pronoms febles*

[d', l', s', n', t', 'l, 'm, 't, 's, 'l, 'ls, 'n, me'n, t'ho, te'ls, t'hi, ns-en, ls-ho, m'hi, me'l, -hi, -ho, -ne]

A.2 Lista de *stopwords*

[a, abans, abans-d'ahir, abintestat, ací, adesiara, adés, adéu, adàgio, ah, ahir, ai, aitambé, aitampoc, aitan, aitant, aitantost, aixà, això, així, aleshores, algun, alguna, algunes, alguns, algú, alhora, allà, allèn, allò, allí, almenys, alto, altra, altre, altres, altresí, altri, alça, al·legro, amargament, amb, ambdues, ambdós, amunt, amén, anc, andante, andantino, anit, ans, antany, apa, aprés, aqueix, aqueixa, aqueixes, aqueixos, aqueixs, aquell, aquella, aquelles, aquells, aquest, aquesta, aquestes, aquests, aquèn, aquí, ara, arran, arrera, arrere, arreu, arri, arruix, atxim, au, avall, avant, aviat, avui, açò, bah, baix, baldament, ballmanetes, banzim-banzam, bastant, bastants, ben, bis, bitllo-bitllo, bo, bé, ca, cada, cal, cap, car, caram, catorze, cent, centes, cents, cerca, cert, certa, certes, certs, cinc, cinquanta, cinquena, cinquenes, cinquens, cinquè, com, comsevulla, contra, cordons, corrents, cric-crac, d, daixonses, daixò, dallonses, dallò, dalt, daltabaix, damunt, darrera, darrere, davall, davant, de, debades, dedins, defora, dejorn, dejús, dellà, dementre, dempeus, demés, demà, des, desena, desenes, desens, després, dessobre, dessota, dessús, desè, deu, devers, devora, deçà, diferents, dinou, dins, dintre, disset, divers, diversa, diverses, diversos, divuit, doncs, dos, dotze, dues, durant, ecs, eh, el, ela, elis, ell, ella, elles, ells, els, em, emperò, en, enans, enant, encara, encontinent, endalt, endarrera, endarrere, endavant, endebades, endemig, endemés, endemà, endins, endintre, enfora, engir, enguany, enguanyasses, enjús, enlaire, enlloc, enllà, enrera, enrere, ens, ensems, ensota, ensús, entorn, entre, entremig, entretant, entrò, envers, envides, environs, enviró, ençà, ep, ep, era, eren, eres, ergo, es, escar, essent, esser, est, esta, estada, estades, estan, estant, estar, estaran, estarem, estareu, estaria, estarien, estaries, estaré, estarà, estaràs, estaríem, estaríeu, estat, estats, estava, estaven, estaves, estem, estes, esteu, estic, estiguem, estigueren, estigueres, estigues, estiguessis, estigueu, estigui, estiguin, estiguis, estigué, estiguérem, estiguéreu, estigués, estiguí, estos, està, estàs, estàvem, estàveu, et, etc, etcètera, ets, excepte, fins, fora, foren, fores, força, fos, fossin, fossis, fou, fra, fui, fóra, fórem, fóreu, fóreu, fóssim, fóssiu, gaire,

gairebé, gaires, gens, girientorn, gratis, ha, hagi, hagin, hagis, haguda, hagudes, hagueren, hagueres, haguessin, haguessis, hagut, haguts, hagué, haguérem, haguéreu, hagués, haguéssim, haguéssiu, haguí, hala, han, has, hauran, haurem, haureu, hauria, haurien, hauries, hauré, haurà, hauràs, hauríem, hauríeu, havem, havent, haver, haveu, havia, havien, havies, havíem, havíeu, he, hem, heu, hi, ho, hom, hui, hàgim, hàgiu, i, igual, iguals, inclusive, ja, jamai, jo, l, la, leri-leri, les, li, lla, llavors, llevat, lluny, llur, llurs, lo, los, ls, m, ma, mai, mal, malament, malgrat, manco, mant, manta, mantes, mantinent, mants, massa, mateix, mateixa, mateixes, mateixos, me, mentre, mentrestant, menys, mes, meu, meua, meues, meus, meva, meves, mi, mig, mil, mitges, mitja, mitjançant, mitjos, moixoni, molt, molta, moltes, molts, mon, mos, més, n, na, ne, ni, ningú, no, nogensmenys, només, noranta, nos, nosaltres, nostra, nostre, nostres, nou, novena, novenes, novens, novè, ns, nòs, nós, o, oh, oi, oidà, on, onsevulga, onsevulla, onze, pas, pengim-penjam, per, perquè, pertot, però, piano, pla, poc, poca, pocs, poques, potser, prest, primer, primera, primeres, primers, pro, prompte, prop, prou, puix, pus, pàssim, qual, quals, qualsevol, qualsevulla, qualssevol, qualssevulla, quan, quant, quanta, quantes, quants, quaranta, quart, quarta, quartes, quarts, quasi, quatre, que, quelcom, qui, quin, quina, quines, quins, quinze, quisvulla, què, ran, re, rebé, renoi, rera, rere, res, retruc, s, sa, salvament, salvant, salvat, se, segon, segona, segones, segons, seguida, seixanta, sempre, sengles, sens, sense, ser, seran, serem, sereu, seria, serien, series, seré, serà, seràs, seríem, seríeu, ses, set, setanta, setena, setenes, setens, setze, setè, seu, seua, seues, seus, seva, seves, si, sia, siau, sic, siguem, sigues, sigueu, sigui, siguin, siguis, sinó, sis, sisena, sisenes, sisens, sisè, sobre, sobretot, sol, sola, solament, soles, sols, som, son, sos, sota, sots, sou, sovint, suara, sí, sóc, són, t, ta, tal, tals, també, tampoc, tan, tanmateix, tant, tanta, tantes, tantost, tants, te, tercer, tercera, terceres, tercers, tes, teu, teua, teues, teus, teva, teves, ton, tos, tost, tostemps, tot, tota, total, totes, tothom, tothora, tots, trenta, tres, tret, tretze, tu, tururut, u, uf, ui, uix, ultra, un, una, unes, uns, up, upa, us, va, vagi, vagin, vagis, vaig, vair, vam, van, vares, vas, vau, vem, verbigràcia, vers, vet, veu, vint, vora, vos, vosaltres, vostra, vostre, vostres, vostè, vostès, vuit, vuitanta, vuitena, vuitenes, vuitens, vuitè, vés, vàreig, vàrem, vàreu, vós, xano-xano, xau-xau, xec, érem, éreu, és, ésser, àdhuc, àlies, ça, ço, òlim, ídem, últim, última, últimes, últims, únic, única, únics, úniques, del, al, dels, el, la, als, este, tall]