
# ESTUDIO DE IDENTIFICACIÓN DE AUTORÍA

Ciencias de la Computación e Inteligencia Artificial

12 DE MARZO DE 2019

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Tutores: Luis de Campos Ibáñez y Juan F. Miguel Huete
Autor: Ernesto Martínez del Pino

# Índice

# 1 Introducción

Vivimos un contexto tecnológico sin precedentes conocido como la era de la **información**. Cada vez somos más conscientes del poder que nos proporciona adquirirla y usarla. Por contraposición, al mismo tiempo vivimos una realidad en la que se nos hace más difícil responder de forma eficiente al volumen de datos que se crea. Cabe destacar que hay estudios que afirman que sólo se almacena menos de un 0,4% de la información que se produce y de ésta, más del 75% se encuentra desestructurada. Llamamos desestructura a aquella información que no puede ser procesada directamente por modelo, como documentos de texto o imágenes. Es frecuente que se nos presenten los datos cada vez más complejos e interconectados, incluso requiriendo de un preprocesamiento no trivial y en última instancia datos que deben ser inferidos a partir de otros.

Es en este último punto donde el Aprendizaje Automático, más conocido en inglés como *Machine Learning,* adquiere un papel clave al enlazarse y apoyarse con otras áreas del conocimiento. Nos permite responder a preguntas que o, por un lado, requerían de la supervisión de un humano o ni siquiera se sabía una respuesta.

Es indiscutible que muchos modelos de *Machine Learning* no tienen competidores actualmente que se asemejen en términos de eficiencia y eficacia. Algunos de los ejemplos más conocidos de aplicación son los siguientes:

- Diagnósticos médicos
- Procesamiento del lenguaje natural o *Natural Language Processing (NPL)*.
- Búsqueda online
- Coches inteligentes
- …

Con el paso del tiempo, la lista de aplicaciones se ha hecho interminable al mismo tiempo que la lista de publicaciones científicas o *papers* relacionadas con el tema ha crecido considerablemente. Este fenómeno, principalmente, se debe a dos motivos:

- Existe un componente **económico** que ha decidido apostar fuertemente por una industria relacionada con modelos predictivos. La necesidad de automatizar trabajos y explotar la información que se posee siempre ha tenido un rol fundamental en una empresa. En 2016, sólo el 8,6% de las empresas realizaba análisis masivos de datos; actualmente es un aspecto diferencial en la industria.
- El acceso a la **tecnología** para implementar, entrenar y validar modelos está prácticamente al alcance de todo el primer mundo. Esto se debe a la apuesta por igual que se ha hecho por la nube o más conocida como la *cloud*, que permite no disponer en local de los recursos en *hardware* necesarios para realizar estas tareas. Existe también un componente de desarrollo *software*, que ha posibilitado esta situación, dejando atrás la barrera técnica que encontraban muchos investigadores para testear sus soluciones.

Nos encontramos en una etapa dorada para la aplicación de todos estos conceptos que se han venido desarrollando en el último siglo de forma teórica y que empiezan a ver sus primeros frutos en el presente.

Existen infinidad de formas en las que nos podemos encontrar la información. Es sabido, que la dificultad de la predicción o respuesta a la pregunta que plantee el problema estará sumamente relacionada con el perfil de información que poseamos. Pues aun siendo la misma pregunta, el problema es distinto si el conjunto de características viene dato en forma de imágenes o textos. Para cada una de estas posibilidades se proponen diferentes estrategias que han conformado subramas dentro del Aprendizaje Automático.

En el caso que nos atañe en este trabajo, la información está representada por texto etiquetado por un autor. De las muestras disponibles que hay para un autor se debe inferir las características que lo definen. De este modo, llegado un texto nuevo el modelo predictivo debe discernir en base a las características extraídas a que autor pertenece. Este problema es conocido como identificación de la autoría o en inglés *authorship attribution problem*.

A lo largo de la historia se han producido numerosos debates sobre la autoría de obras transcendentales para el conocimiento humano. El hecho de conocer el autor de un contenido da un peso conceptual extra a sus palabras que, apoyado por su biografía y sus circunstancias, crean un marco decisivo desde el que poder abarcar cualquier estudio. Esto es debido a que, a veces, podría ser más conclusivo responder a la pregunta, quién desarrolló un contenido, que el contenido mismo.

## 1.1 Propósito

El propósito de este trabajo es estudiar la factibilidad de la identificación del autor de un documento sobre un conjunto de autores previamente definido. Se desarrollará y argumentará un estudio completo sobre que es diferencial y que no para etiquetar el autor de una obra textual.

## 1.2 Partes

El trabajo constará de 12 secciones definidas en el índice inicial y fácilmente consultables. El formato electrónico permite navegabilidad sobre el mismo gracias a la inserción de vínculos sobre las entradas. A lo largo del trabajo se pueden encontrar referencias en forma de vínculos al apartado bibliográfico que se encuentra en la última sección de este documento.

## 1.3 Justificación

He decido abordar este trabajo tras la lectura de una Estudio de Métodos Modernos de Atribución de Autoría de la Univerdidad de Aegean (Stamatatos, 2009) y El efecto del tamaño del conjunto de autores y el tamaño de los datos en la atribución de autoría (Luyckx & Daelemans, 2011). Ambas investigaciones fueron sugeridas por los tutores del proyecto. La lectura del capítulo V y VI del libro Inteligencia Artificial un Enfoque Moderno (Russell & Norvig, Learning, Communicating, perceiving, and acting, 2010) ha supuesto una notable aportación en este trabajo.

También existe un componente previo personal que me ha llevado a aceptar esta temática. Mi afición por la lectura ha ocupado gran parte de mi vida, haciendo un especial énfasis en las obras filosóficas. Esto unido a mi interés por la psicología humana me ha permitido incorporar algunas pincelas que considero interesantes en el desarrollo del trabajo.

Desde un punto de vista laboral, mi trabajo actual como Científico de Datos para la Prevención del Fraude y el Crimen Organizado en Deloitte comparte muchas áreas de conocimiento con el tema que nos ocupa. Trato desde la identificación de nombres sobre listas sancionadas (Nice Actimize Sanctions Screening & Watch List Filtering, s.f.) haciendo uso de emparejamiento por lógica difusa, hasta la identificación y clasificación de alertas sospechosas en los conceptos de las transferencias bancarias. Como ya se ha mencionado antes, las aplicaciones de Aprendizaje Automático inundan el mercado actual, danto la posibilidad de mantener arquitecturas de modelos semejantes en problemas aparentemente distintos. ML ha dado un paso de gigante en lo que a la abstracción de problemas se refiere.

Desde el punto de vista de las ciencias de la computación y la inteligencia artificial rama del conocimiento que estudio, este problema hace uso de muchos de los conceptos y herramientas que en ellas se explican. Debido a que ha sido una decisión propia el estudio de esta especialidad, es razonable la elección del proyecto.

## 1.4 Ámbito

El problema de la Identificación de Autoría o la Autoría de Documentos es una cuestión interdisciplinar que comparte actualmente lingüística, Recuperación de Información o *Information Retrieval* e Inteligencia Artificial. Siendo estos dos últimos desde los que se va a abordar el trabajo. Nos adentrándonos en la subrama Aprendizaje Automático perteneciente a la Inteligencia Artificial.

Podemos decir que los tres ámbitos específicos que predominan en este trabajo son el procesamiento del lenguaje natural, la recuperación de información y el aprendizaje automático. A continuación, daremos una definición básica de ambos conceptos:

El procesamiento del lenguaje natural (NLP) es un rango teóricamente motivado de técnicas computacionales para analizar y representar textos que ocurren naturalmente en uno o más niveles de análisis lingüístico con el fin de lograr el procesamiento del lenguaje humano en una variedad de tareas o aplicaciones.

La Recuperación de Información es la conversión de grandes volúmenes de texto en estructuras simplificadas y comprensibles para su uso posterior.

El Aprendizaje Automático es el estudio de algoritmos de computación que mejoran automáticamente por medio de la experiencia.

Si quisiéramos profundizar más, existen numerosas subcategorías dentro del problema al que nos enfrentamos, dependiendo del registro del lenguaje, permisibilidad de faltas de ortografía, tamaño de los textos, número de autores, número de muestras… Más adelante concretaremos estos conceptos en el apartado 2.

## 1.5 Relevancia teórica y práctica de la investigación

Realizando una búsqueda rápida en los principales gestores de publicaciones científicas podemos constatar que problema de la Atribución de la Autoría estás más vigente que nunca. (gestores de publicaciones)

Dentro la Minería Texto se puede considerar como una de las temáticas más tratadas junto al problema de Análisis de Sentimientos. Demostrando ser un entorno de investigación idóneo, como muchas publicaciones lo corroboran, para descubrir y proponer nuevas técnicas en Aprendizaje Automático.

A continuación, se presentan algunas aplicaciones que tiene el problema en la práctica:

- Desde la literatura nos encontramos con debates actuales sobre la identificación de autores de obras anónimas o puesta en duda de obras que fueron atribuidas sin estudio previo. Pero la realidad es que muchas obras continúan siendo anónimas actualmente ya sea por deterioro de la misma o por falta de candidatos.
    - Un ejemplo de la primera es la conocida obra del Lazarillo de Tormes que, tras una lista de candidatos, se resolvió su autoría gracias a un estudio lingüístico que atribuyó a Sebastián de Horozco su autoría.
    - Un ejemplo de la segunda es el debate que hubo en torno a las obras de Shakespeare debido a su estilo impropio de la cuna de la que provenía el autor, sumado a las numerosas lagunas de su biografía.

- Desde la criminología, derecho, psicología y psiquiatría nos encontramos un considerable número de problemas.
    - Peritaje de conversaciones electrónicas.
    - Autoría de cartas de suicidio.
    - Falsificación en las relaciones laborales.
    - Falsificación de estudios (TFGs y TFMs).
    - Identificación de trastornos.
    - Psicología evolutiva y del aprendizaje.

- Desde el Marketing:
    - Optimización de impacto de discursos
    - Comparaciones con la competencia
    - Generación de anuncios dirigidos

- Desde la automatización de tareas, como es nuestro caso, tenemos algunos ejemplos.
    - Documentación automática.
    - Detección de suplantaciones de identidad.

## 1.6 Estado del arte

El primer estudio que se realizó de esta materia fue en 1887 sobre las obras de Shakespeare, publicado por Thomas Corwin Mendenhall. Seguido medio siglo después por los trabajos estadísticos de Tule (1938:1944) and Zipf (1932). Aunque sin duda el trabajo más notorio y reconocido es el estudio realizado por Mosteller and Wallace (1964) construido a partir de 'The Federalist Papers' (Mosteller & Wallace, 1963), un conjunto de 146 documentos de longitud variada escritos por tres autores diferentes pertenecientes a un lenguaje formal. El método usado en este último estudio fue un modelo Bayesiano estadístico centrado sobre un grupo de palabras comunes en inglés.

Antes de este estudio, la capacidad de diferenciar autores se veía desde un punto de vista lingüístico dependiente del estilo literario de cada uno. Se definió un conjunto heterogéneo de mediciones -aproximadamente llegaron a ser un millar- sobre propiedades concretas y triviales del uso del lenguaje. Algunas de éstas fueron la frecuencia de palabras por frase, la frecuencia de caracteres por frase o el uso de palabras poco frecuentes.

La metodología de trabajo estaba bastante limitada tanto por los medios disponibles en el momento como por los problemas que se planteaban. La mayor parte de estos problemas cumplían las siguientes características:

- El documento a analizar usualmente se trataba de una obra completa o libro.
- El número de autores sobre los que se realizaba el estudio era pequeño (aproximadamente 2 o 3 autores).
- Existía un alto componente subjetivo en la evaluación de los métodos propuestos.
- La decisión de que método era el más apropiado para un problema partía de una ausencia notable de un banco de problemas resueltos.

Con el incremento de fuentes textuales electrónicas, el rango de uso del problema de la atribución de la autoría fue más heterogéneo. Encontramos trabajos aplicados a sistemas de inteligencia militar para la atribución de mensajes o proclamaciones de conocidos terroristas (Abbasi & Chen, 2005) en los que se usa procesos de extracción de raíces combinados con múltiples procedimientos de extracción de características; como pueden ser a nivel sintáctico, léxico, estructural... Un último extractor no tan genérico aplicado en esta investigación dentro de contextos de violencia, raza y nacionalidad. Aplicaciones digitales en el marco legal (Chaski, 2005) para determinar la autoría de mensajes de acoso. Desde el desarrollo del software, la identificación del autor de software malicioso (Yampolskiy & Govindaraju, 2009) a partir de distancias coseno de matrices de frecuencia. A medida que se iban proponiendo nuevos métodos también se realizaron trabajos de rendimientos comparativos (Joula & Sofko, 2004). Propiedades de los conjuntos de datos como la longitud palabras jugaron un papel

crucial introduciendo la aplicación de *Support Vector Machine* en el área (Marton, Wu, & Hellerstein, 2005). Experimentos forenses haciendo uso de modelos bigram, análisis sintáctico (Feiguina & Hirst, 2007).

A partir de la revolución de Internet, la metodología tradicional aplicada sobre los nuevos problemas se quedó obsoleta debido a la diversificación y volumetría de estos. Es en este punto, cuando el Procesamiento del Lenguaje Natural en conjunto con el Aprendizaje Automático y Recuperación de Información se imponen como áreas del conocimiento y modelos de trabajo para la identificación de la autoría.

Desde la Recuperación de Información se desarrollaron técnicas eficientes para la representación y clasificación de grandes volúmenes de información.

Desde el Aprendizaje Automático se desarrollaron algoritmos capaces de trabajar con problemas de alta dimensión. En 1992, con la publicación realizada por E. Boser, Isabelle M. Guyon y Vladimir N. Vapnik en la que sugirieron la aplicación del modelo 'máquina de vectores soporte' (SVM) sobre '*kernels'* no lineales.

Desde el procesamiento del lenguaje natural con el desarrollo de herramientas eficientes que analizaran características del lenguaje.

La revolución tecnológica en el entrenamiento de redes neuronales que han provocado los núcleos GPU Tensor introducido por Nvidia (A Navarro, Andrés Carrasco, J. Barrientos, & Vega, 2020) dan la posibilidad cruzar la barrera de rendimiento que imposibilitaba la implementación de ciertos modelos de Deep Learning.

## 1.7 Descripción de la metodología de la investigación

Unos de los mayores problemas en los trabajos de ciencia de datos es la metodología. Existen muchos puntos críticos y difícilmente detectables que pueden dar lugar a invalidar nuestra investigación y nuestros resultados. Resulta extremadamente complejo realizar estos pasos de forma ordenada. La principal causa es que se trata de un proceso **dinámico multiobjetivo**. A continuación, se explican los pasos se han seguido a lo largo de la investigación:

1. **Fuente de datos**: es el paso inicial donde se debe hacer especial atención al sistema de almacenamiento. Usualmente se pueden presentar dificultades técnicas con el sistema ya que pueden requerir de conocimientos profundos en otras áreas de conocimiento. Un ejemplo es el lenguaje de consultas SQL. Errores de formato y codificación son habituales cuando se trabaja con formatos de texto o imagen. En el caso de formatos de texto incrementa la dificultad si nos encontramos con varios idiomas. Algunas de las problemáticas que aplican a nuestro problema se ven reflejadas en el apartado Fuente.
2. **Limpieza de datos**: es un paso obligatorio, ya sea en su forma mínima como comprobación de que los datos concuerden con lo esperado o en su forma más compleja realizando una limpieza más sofisticadas como se ven en este proyecto. Las situaciones que se pueden dar son casos incoherentes que no se deban reflejar en el modelo, datos directamente erróneos que se deben quitar, casos extremadamente poco frecuentes que solo aportan ruido… La realidad es que no existe una guía definitiva de como limpiar los datos debido a la diversidad

de problemas que existen. En conclusión, termina siendo un compromiso del implementador con el grado de satisfacción que tenga sobre las pruebas o condiciones aplicadas. Algunas de las problemáticas de este paso se ven reflejadas en el apartado Limpieza. Es importante marcar un punto de homogeneidad a partir de aquí cuando se realizan comparaciones entre distintas arquitecturas de ML, pues dos arquitecturas solo son comparables si partes de los mismos datos.

3. **Preprocesamiento**: paso intermedio y fácilmente confundible con la limpieza y extracción de datos. Su principal de diferencia con la limpieza es que esta formalizado conceptualmente y se conocen el conjunto de preprocesamientos que le pueden aplicar a un determinado tipo de características. La diferencia con la extracción es que no va orientado generar información estructura sino a hacer más eficiente y posible el entrenamiento del modelo. En este proyecto por decisiones de implementación de las bibliotecas usadas el preprocesamiento ha ido estrechamente ligado a la extracción. Este paso se refleja en el apartado Bolsa de Palabras.
4. **Extracción de características**: este paso existe cuando se trabaja con modelos clásicos de Aprendizaje Automático que precisan de una orientación previa sobre qué características usar. Una característica extraída es aquella que se define como combinación de otras previas. Por ejemplo: a partir del texto se extrae el conjunto de características formado por las palabras junto a sus frecuencias. Este paso se refleja en el apartado Extracción de características.
5. **Selección de características**: de la misma forma que el paso anterior la selección de características es usualmente frecuente encontrarla en los modelos más clásicos. El conjunto de características seleccionadas es un subconjunto del original con el objetivo de determinar cuáles afectan al problema. Este paso se refleja en el apartado Selección de características.
6. **Elección del modelo**: dependiendo de si se han aplicado los pasos 4 y 5 y que técnicas se han empleado en ellos la disquisición de que modelo usar entraña un estudio teórico previo. La documentación relativa a este punto la podemos encontrar en Aprendizaje.
7. **Ajuste de hiperparámetros**: seleccionado el modelo queda encontrar los mejores parámetros para el mismo. La búsqueda de hiperparámetros suele ir acompañada de la validación cruzada. Los 3 métodos o formas de búsqueda más utilizados se pueden encontrar en el apartado Ajuste de hiperparámetros. Mientras que la descripción de la validación cruzada se encuentra en el apartado Validación cruzada.
8. **Validación de componentes**: es un proceso obligatorio que pretende hacer un seguimiento de todos los componentes que forman la arquitectura final del modelo. De esta forma, aunque el modelo se presente en forma de tubería o *pipeline* no pierda el seguimiento de las transformaciones de los datos. Una vez realizada esta comprobación del correcto funcionamiento de los componentes no es necesario validarla por cada ajuste de hiperparámetros. Este trabajo se ve reflejado en cada uno de los subapartados del apartado Experimentos.
9. **Validación de resultados**: es un proceso obligatorio que finalmente determina si es un modelo valido o se debe repetir el proceso usualmente desde el paso 4. La validación de resultados entraña saber si tenemos un *overfit* o *underfit*. Si cumple los requisitos del diseño de nuestro sistema de puntuación. En el caso de problemas desbalanceados este último paso tiene una importancia crucial a la hora de interpretar los datos. Teóricamente la validación de resultados se encuentra definida en la Metodología de experimentación. Experimentalmente

los datos se encuentran en el apartado Experimentos para cada uno de los modelos.

Decimos que trata de un proceso dinámico porque en la práctica puede resultar difícil pintar la frontera entre cada uno de los pasos descritos. Puede darse el caso de que la producción de datos transcurra al mismo tiempo que la elaboración del modelo obligando a repasar casuísticas e incoherencias en los mismos.

A lo largo del desarrollo de este trabajo, se han detectado pequeños errores en los datos relacionados con la codificación de las palabras o con la limpieza de las frases que han obligado a repetir muchas de estas fases.

# 2 Profundizar sobre el lenguaje

En esta sección se hará una breve introducción del problema de la identificación de la autoría utilizando las características del lenguaje como vía que contextualice nuestro conjunto de datos.

Acuñemos en primer lugar una definición de lenguaje:

> “Facultad del ser humano de expresarse y comunicarse con los demás a través del sonido articulado o de otros sistemas de signos.”

(Real Academia Española, 2020)

La producción que genera este lenguaje es ingrediente esencia de estudio al que llamaremos corpus. La definición formal de corpus es la siguiente:

> "Conjunto lo más extenso y ordenado posible de datos o textos científicos, literarios, etc., que pueden servir de base a una investigación."

(Real Academía Española, 2020)

Existen diferentes tipos de lenguajes que afectan de forma determinante a la clase de corpus sobre la que se realice la investigación. Si partimos de un conocimiento profundo sobre la fuente, podemos acotar algunas de sus propiedades antes de realizar un análisis genérico técnico. Seguidamente se ve reflejado un análisis superficial sabiendo a qué tipo de lenguaje atenernos.

En nuestro caso sabemos que se trata de lenguaje parlamentario o del foro sobre el cual tenemos algunas características que resaltar:

- Lenguaje oral: aunque muchas veces se puede presentar por escrito, siempre se produce pensando que va ser escuchado oralmente. La dificultad de compresión de lenguaje oral se ve limitada naturalmente por la fugacidad de las palabras. Es por ello que los conceptos recogidos en un discurso oral rara vez llegan a la complejidad que se da en su equivalente escrito. Es de destacar que en el entorno que se desarrolla ese lenguaje oral se registra normalmente un

lenguaje formal salpicado por expresiones coloquiales o momentos tensos del debate.

- Lenguaje argumentativo: tiene una tendencia explicativa recursiva con un fuerte apoyo en el grupo de sinónimos de la idea principal a medida que se va desarrollando el concepto. Estadísticamente, la idea principal se suele presentar al principio de la exposición de forma esquematizada, aunque esta estructura varía dependiendo del estilo argumentativo del hablante.
- Lenguaje de debate: función apelativa del lenguaje sin duda es una de las más presenten en el lenguaje parlamentario. Una práctica que se suele observar frecuentemente es la alusión a ideas reflejadas por el adversario político en intervenciones posteriores. Un elemento clave del debate que suele dar mayor rigurosidad al mensaje es el empleo de citas. Dentro de este lenguaje podemos encontrar citas de distinta índole como literarias, técnicas, coloquiales…

La característica del lenguaje oral tiene una fuerte influencia en el tamaño del vocabulario extraído sobre los documentos. Normalmente el idiolecto culto posee un menor número de recursos literarios que su registro estándar para una persona de cultura media. Por consecuente, es de esperar extraer un tamaño de vocabulario inferior a la media de corpus con volumetrías semejantes.

El lenguaje argumentativo desde su fuerte esquematización puede abrir una vía de análisis sintáctico desde el que enfrentarse a un conjunto de datos que tenga esta propiedad. Las estructuras rígidas subyacentes sugieren aplicar extractores de información que sitúen los elementos del lenguaje en su contexto.

El lenguaje de debate aporta una complejidad extra difícil de solventar de forma automática. Existe una dicotomía en los estudiosos del lenguaje. Ésta se refiere a si tener o no en consideración las citas como parte del análisis del estilo literario de un individuo. Desde el punto de vista teórico de este trabajo, la inclusión de frases en forma de cita pudiéndose dar el caso de ser producciones de individuos contenidos en el conjunto de datos, supone un límite de mejora en los resultados que debemos valorar.

Como veremos en el apartado de preprocesamiento la extracción de la raíz de una palabra es una práctica habitual es este tipo de estudios. La aparición de anglicismos o palabras todavía por acuñar en este tipo de lenguaje es muy frecuente. Los algoritmos de extracción raíces son dependientes de la lengua usada, en nuestro caso el castellano, puede verse afectados por estas casuísticas.

El tiempo es un punto a tener en cuenta. Pues un corpus se desarrolla siempre en un intervalo de tiempo en el que los sujetos protagonistas de análisis experimentan cambios en sus estilos de habla. Esta posible trayectoria del estilo del habla en algunos casos puedes ser imperceptible y otros sumamente perjudicial a la hora de inferir un modelo.

# 3 Objetivo

Estos son los objetivos del proyecto:

1. Realizar un **preprocesamiento** correcto de los **datos** para que puedan ser utilizados como entrada en un modelo de aprendizaje automático.
2. Proponer diversos **modelos** entrenados de clasificación multietiqueta que presenten buenos resultados.

3. Validar los modelos y compararlos entre sí.
4. Definir un criterio justificado para elegir el mejor modelo y presentarlo en un ambiente de producción.

# 4 Datos

## 4.1 Fuente

La fuente de datos utilizada son la Iniciativas Parlamentarias del Parlamento de Andalucía durante el año 2008. Estas iniciativas están recogidas en 5.260 ficheros en formato XML y codificación UTF8. El formato XML presenta la siguiente estructura:

```
< iniciativa_completa >
        ...
        < iniciativa >
                ...
                < intervencion >
                        < interviniente > Nombre </interviniente >
                        < discurso >
                                < parrafo > Texto </parrafo >
                                ...
                        </discurso >
                </intervencion >
        </iniciativa >
</iniciativa_completa >
```

El nombre se corresponde con la columna de las etiquetas y el texto está formado a partir de la unión de los párrafos con la columna de las características.

## 4.2 Limpieza

La extracción del **nombre** requiere de una complejidad mayor. Esto es debido a que no es exactamente el nombre como tal, sino la presentación que se le realiza al interviniente. Alguna de las dificultades para la extracción son las siguientes:

- El nombre viene acompañado del cargo político o puesto laboral que ocupa el interviniente.
- Sólo se aporta un apellido o un nombre en lugar del nombre completo del interviniente.
- Sólo se alude al cargo del interviniente. Por ejemplo: “el diputado”.
- Frases del manejo del debate. Por ejemplo: “toma la palabra”.
- Diferencia entre nombres en mayúsculas o minúsculas.

Algunas de estas dificultades se pueden resolver de forma automática, como es el caso de las mayúsculas y minúsculas, otras se han tenido que resolver manualmente con un proceso iterativo de visualización de los datos. Para ello, se ha definido un fichero de remplazamiento de cadenas de caracteres que se encuentra en la raíz del proyecto, llamándose 'replace.txt. El fichero tiene la siguiente estructura:

$$cadena1$$
$$cadena2$$
$$cadena3$$
$$\dots$$

Donde cada una de las cadenas que se lee del fichero se remplaza por la cadena vacía. En conclusión, una etiqueta de conjunto de datos seguirá la siguiente expresión:

$$\forall i \in (0, N)\ nombre.replace(cad_i, )$$

Donde $N$ es el número de cadenas contenidas en el fichero de remplazamiento que actualmente son 343.

Los nombres que tras los remplazamientos quedan vacíos se han eliminado.

Sobre el **texto** se han realizado cuatro transformaciones estándares de tipo remplazamiento en forma de expresión regular:

1. Eliminación de anotaciones sobre la entonación o falta de fragmento:

$$\backslash[[\backslash w\backslash W]^*] \rightarrow \varepsilon$$

2. Remplazamiento de caracteres fuera de la unión del abecedario castellano, el sistema básico de numeración y la puntuación:

$$[\hat{}1 - 9a - zA - ZÁÉÍÓÚáéíóúñÑ.,:;\backslash?¿\backslash!¡\backslash(\backslash)] \rightarrow ''$$

3. Remplazamiento de conjuntos de caracteres de separación de palabras como espacios, tabuladores y saltos línea por un único espacio.

$$[\backslash s\backslash t\backslash n]+ \rightarrow ''$$

4. Eliminación de los espacios laterales por medio de la conocida función $strip$.

Seguidamente se muestra un ejemplo de aplicación de estas tres transformaciones en el orden marcado:

$$"\ ¿\,Ésto\ es\ una \quad prueba\ de\ [pausa]\ limpieza\ (3.0) \quad ¿\,!!\,"$$

Después de la limpieza:

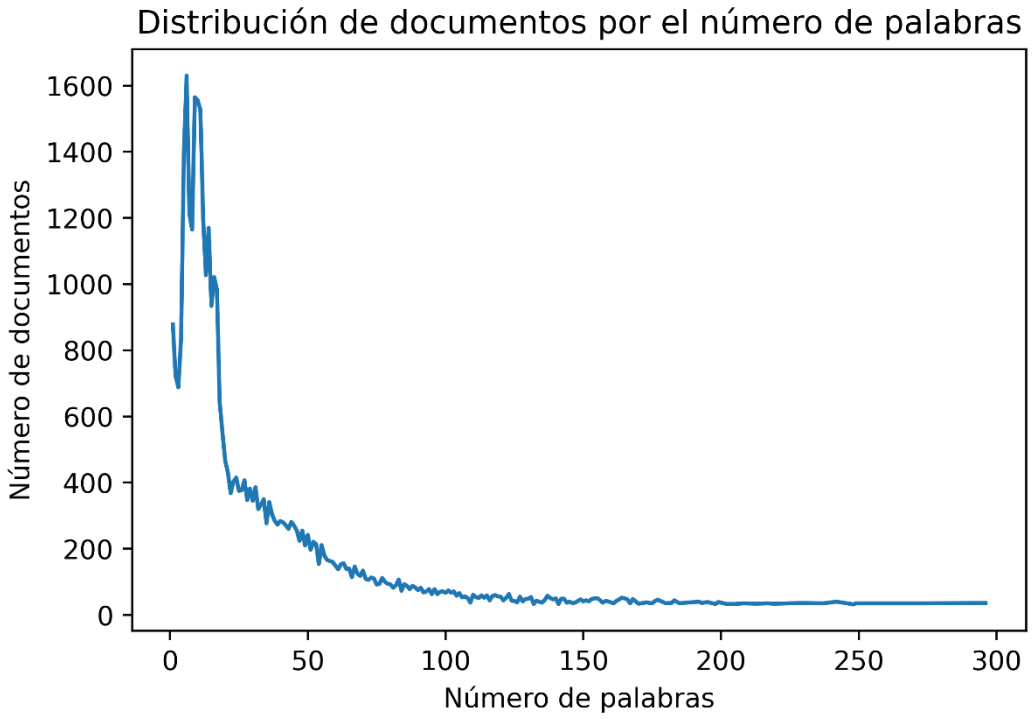
$$"¿\,Esto\ es\ una\ pueba\ de\ limpieza\ (3.0)\ ¿\,!!\,"$$

Los documentos que tras las transformaciones quedan vacíos se han eliminado.

Tras realizar la lectura de todos los ficheros y aplicar estas dos limpiezas sobre el nombre y texto obtenemos los siguientes resultados: tenemos un total de **594 nombres** o etiquetas diferentes con un total de **58.275 documentos**. A la vista de la conocida estructura de datos $dataframe$ en Ciencia de Datos quedaría de la siguiente forma:

| | name | text |
|---|---|---|
| 0 | COVES BOTELLA | Buenos días, señorías.. Se abre la sesión con ... |
| 1 | LETRADO MAYOR | Bueno, declaro constituida la Comisión de Gobi... |
| 2 | COVES BOTELLA | Pasamos al segundo punto del orden del día. Pr... |
| 3 | GARCÍA GARRIDO | Muchas gracias, Presidenta. Señorías.. Una vez... |
| 4 | COVES BOTELLA | Muchas gracias, señor García Garrido.. ¿Tienen... |

A continuación, debemos responder a la pregunta de si todos esos documentos son válidos para un modelo en el que se realice una extracción de características sobre el texto. Para ello nos fijaremos en el número de palabras por documento. En el siguiente gráfico se puede ver la distribución de los documentos frente al número de palabras que presentan:




El gráfico está truncado para mostrar sólo los 200 primeros valores sobre el eje x.

Como podemos ver existe una gran concentración de documentos de corta extensión entre 1 y 50 palabras. Esto tiene una explicación dentro del ámbito que se desarrolla el conjunto de datos. Al abrir un debate es usual encontrarse un par de intervenciones contundentes en tamaño seguidas de réplicas cortas. Muchas de estas replicas citan palabras textuales de las iniciales. Veamos algunos datos estadísticos asociados a esta propiedad:

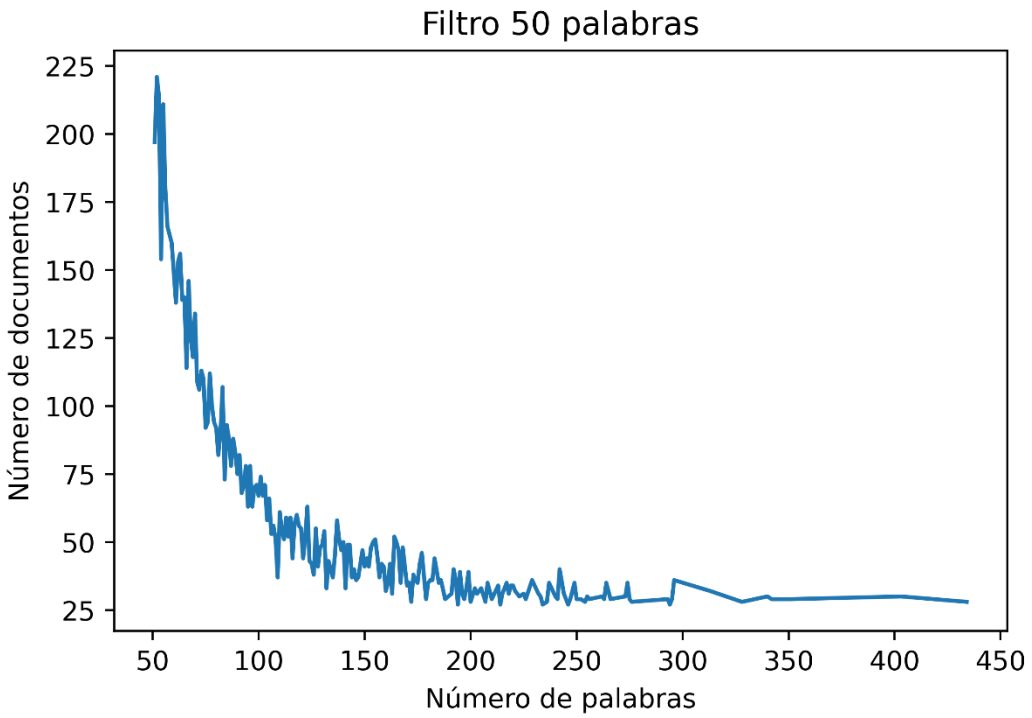
| | Valor |
|---|---|
| Conteo | 58.275 |
| Media | 323,68 |
| Desviación | 649,22 |

| | |
|---|---|
| Mínimo | 1 |
| Percentil 25% | 13 |
| Percentil 50% | 44 |
| Percentil 75% | 310 |
| Máximo | 12.474 |

Para poder abordar un aprendizaje lógico y factible se ha definido el siguiente filtro sobre el conjunto de datos:

"Se considerará como documento todo texto que contenga más de 50 palabras"

Por lo tanto, el gráfico anterior quedaría actualizado así:




El gráfico está truncado para mostrar sólo los 200 primeros valores sobre el eje x.

Actualizamos la tabla estadística anterior:

| | Valor |
|---|---|
| Conteo | 27.555 |
| Media | 665,10 |
| Desviación | 818,59 |
| Mínimo | 51 |
| Percentil 25% | 120 |
| Percentil 50% | 346 |
| Percentil 75% | 931 |
| Máximo | 12.474 |

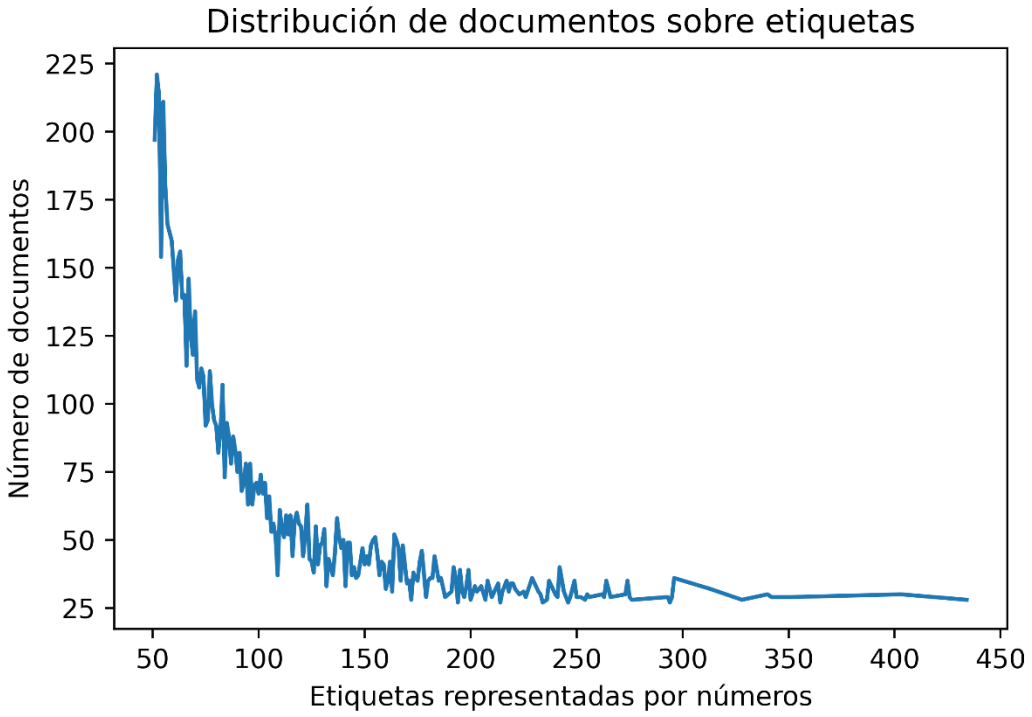
Podemos observar que el mínimo de palabras ha cambiado y por consecuencia la distribución de los percentiles.

Como conclusión a este filtro los datos que nos quedan después de su aplicación son los siguientes: **27.555 documentos** y **479 etiquetas**.

A continuación, vamos a realizar un conteo del número de documentos que tiene asociada cada etiqueta. Con ello perseguimos eliminar casos absurdos de aprendizaje sobre etiquetas que no presentan apenas muestras. Empecemos con algunos datos estadísticos para el número de documentos por etiqueta.

| | Valor |
|---|---|
| Conteo | 479 |
| Media | 57,53 |
| Desviación | 130,94 |
| Mínimo | 1 |
| Percentil 25% | 1 |
| Percentil 50% | 2 |
| Percentil 75% | 39 |
| Máximo | 1.583 |

Estos resultados arrojan datos interesantes. Vemos que existen un altísimo porcentaje de etiquetas con 2 o menos documentos asociados. Es importante conocer esta realidad sobre conjunto de datos ya que se verá reflejada en según que puntuación definamos para el problema. Vemos una gráfica asignando numeración a las etiquetas que muestre el descenso ordenado documentos asociados:




Del gráfico y la tabla estadística se puede concluir que existe un subconjunto mayoritario de etiquetas que exclusivamente aportarían ruido sobre la clasificación. Es por ello, que debemos aplicar un segundo filtro para eliminar aquellas etiquetas con una frecuencia baja. Observando los datos vemos que 424 de los 479 restantes tienen menos de 200 documentos. Por tanto, aplicamos el segundo filtro:

"Se considerará etiqueta a partir de los 200 documentos etiquetados a la misma fuente."

La tabla resultante de número de documentos asociados a las 55 etiquetas ordenados de mayor a menor se encuentra en el Apéndice.

En resumen, a la aplicación de los dos filtros los datos restantes son los siguientes: **19.113 documentos** y **55 etiquetas**. A partir de este momento, cuando nos referimos al conjunto de datos daremos por sentado que se han aplicado estos dos filtros para la implementación y ajuste de los modelos.

Los valores estadísticos a nivel de número de palabras resultantes son los siguientes:

| | Valor |
|---|---|
| Conteo | 19.113 |
| Media | 643,18 |
| Desviación | 818,74 |
| Mínimo | 51 |
| Percentil 25% | 111 |
| Percentil 50% | 326 |
| Percentil 75% | 867 |
| Máximo | 12.474 |

De esta tabla debemos resaltar la media que nos será muy útil para modelos que requieran un preprocesamiento basado en secuenciación.

Todo este proceso realizado se encuentra desarrollado en un *notebook* llamado "$desarrollo.ipynb$" paso a paso para su verificación.

Con el objetivo de realizar pruebas justas entre todos los modelos se ha fijado la semilla para la división entre entrenamiento y test. El tamaño de estos dos conjuntos se ha elegido siguiendo el estándar 80% entrenamiento y 20% test o validación. Resumiendo, quedan estos dos conjuntos:

- Entrenamiento: 15.290 entradas.
- Validación: 3.823 entradas.

# 5 Fundamentos teóricos

## 5.1 Preprocesamiento

El preprocesamiento en Aprendizaje Automático suele ser un apartado orientado a tareas de escalado, normalización, imputación, codificación entre otras muchas transformaciones aplicadas a las características. En nuestro caso, al tratarse de una sola en de forma texto tenemos un conjunto más limitado de operaciones.

### 5.1.1 Puntuación

Un paso previo a la extracción de características sobre texto es la eliminación de los signos de puntuación. El conjunto de signos de puntuación para el castellano es el siguiente:

!"#$%&′()∗+,−./:;<=>?@[\]^_`{|}~

Aplicar esta transformación no siempre asegura buenos resultados. Del mismo modo que en el lenguaje, la disposición de la puntuación en las frases aporta una comprensión diferente.

### 5.1.2 StopWords

Muchas de las bibliotecas de procesamiento del lenguaje natural ofrecen conjuntos de palabras por idioma llamados StopWords. Estos grupos recogen las palabras más frecuentes del lenguaje que sirven como nexo entre ideas o soporte de estructuras gramaticales más complejas. Por lo general, se sigue el criterio de que si se quitaran de una frase no se debería de ver mermada la información que se refleja. Dependiendo del conjunto de datos y problema, la eliminación de las StopWords puede aportar mejoras significativas en los resultados de la predicción al mismo tiempo que se reduce el tamaño del vocabulario.

El conjunto de palabras para el castellano propuesto se encuentra recogido en el apéndice.

### 5.1.3 Traducción

La traducción del conjunto de datos es una técnica frecuente que se aplica por dos motivos:

- Falta de recursos: usualmente muchas transformaciones de los datos como son las StopWords requieren de un trabajo de terceros que en ciertas ocasiones puede no estar disponible. Realizando la traducción a idiomas más comunes para los desarrolladores, como puedes ser el inglés, adquirimos todas esas ventajas.
- Ventajas del lenguaje: lenguajes como el castellano presentan una gran variedad de prefijos y sufijos que pueden aumentar desmesuradamente el vocabulario de nuestro corpus. Realizando una traducción a una lengua que no presente esta característica puede dar buenos resultados.

De la misma forma que las StopWords, la traducción suele ser un servicio externalizado en la tubería de pasos.

### 5.1.4 POS Tagging

El POS Tagging o el etiquetado gramatical es un componente importante de estudio de los sistemas de Procesamiento del Lenguaje Natural. Presenta una doble vertiente como problema de aprendizaje supervisado y no supervisado.

En el problema supervisado, que es el que aplica a este proyecto, se define para cada lenguaje un conjunto de $taqs$ que actúan como grandes grupos de palabras que reciben el nombre de categorías gramaticales. Las categorías gramaticales actúan como etiquetas finales del problema de clasificación. Para la resolución de este problema se han empleado modelos ocultos de Markov, programación dinámica y modelos de Aprendizaje Automático.

Es la siguiente imagen se pueden ver los resultados obtenidos por un gran número de publicaciones científicas:

| Language | Source | # Tags | O/O | U/U | O/U |
|---|---|---|---|---|---|
| Arabic | PADT/CoNLL07 (Hajič et al., 2004) | 21 | 96.1 | 96.9 | 97.0 |
| Basque | Basque3LB/CoNLL07 (Aduriz et al., 2003) | 64 | 89.3 | 93.7 | 93.7 |
| Bulgarian | BTB/CoNLL06 (Simov et al., 2002) | 54 | 95.7 | 97.5 | 97.8 |
| Catalan | CESS-ECE/CoNLL07 (Martí et al., 2007) | 54 | 98.5 | 98.2 | 98.8 |
| Chinese | Penn ChineseTreebank 6.0 (Palmer et al., 2007) | 34 | 91.7 | 93.4 | 94.1 |
| Chinese | Sinica/CoNLL07 (Chen et al., 2003) | 294 | 87.5 | 91.8 | 92.6 |
| Czech | PDT/CoNLL07 (Böhmová et al., 2003) | 63 | 99.1 | 99.1 | 99.1 |
| Danish | DDT/CoNLL06 (Kromann et al., 2003) | 25 | 96.2 | 96.4 | 96.9 |
| Dutch | Alpino/CoNLL06 (Van der Beek et al., 2002) | 12 | 93.0 | 95.0 | 95.0 |
| English | PennTreebank (Marcus et al., 1993) | 45 | 96.7 | 96.8 | 97.7 |
| French | FrenchTreebank (Abeillé et al., 2003) | 30 | 96.6 | 96.7 | 97.3 |
| German | Tiger/CoNLL06 (Brants et al., 2002) | 54 | 97.9 | 98.1 | 98.8 |
| German | Negra (Skut et al., 1997) | 54 | 96.9 | 97.9 | 98.6 |
| Greek | GDT/CoNLL07 (Prokopidis et al., 2005) | 38 | 97.2 | 97.5 | 97.8 |
| Hungarian | Szeged/CoNLL07 (Csendes et al., 2005) | 43 | 94.5 | 95.6 | 95.8 |
| Italian | ISST/CoNLL07 (Montemagni et al., 2003) | 28 | 94.9 | 95.8 | 95.8 |
| Japanese | Verbmobil/CoNLL06 (Kawata and Bartels, 2000) | 80 | 98.3 | 98.0 | 99.1 |
| Japanese | Kyoto4.0 (Kurohashi and Nagao, 1997) | 42 | 97.4 | 98.7 | 99.3 |
| Korean | Sejong (http://www.sejong.or.kr) | 187 | 96.5 | 97.5 | 98.4 |
| Portuguese | Floresta Sintá(c)tica/CoNLL06 (Afonso et al., 2002) | 22 | 96.9 | 96.8 | 97.4 |
| Russian | SynTagRus-RNC (Boguslavsky et al., 2002) | 11 | 96.8 | 96.8 | 96.8 |
| Slovene | SDT/CoNLL06 (Džeroski et al., 2006) | 29 | 94.7 | 94.6 | 95.3 |
| Spanish | Ancora-Cast3LB/CoNLL06 (Civit and Martí, 2004) | 47 | 96.3 | 96.3 | 96.9 |
| Swedish | Talbanken05/CoNLL06 (Nivre et al., 2006) | 41 | 93.6 | 94.7 | 95.1 |
| Turkish | METU-Sabanci/CoNLL07 (Oflazer et al., 2003) | 31 | 87.5 | 89.1 | 90.2 |

(Petrov, Das, & McDonald, 2011)

Para este proyecto se ha definido una transformación sobre el corpus realiza una sustitución de cada palabra por su etiqueta gramatical correspondiente. El vocabulario tras la transformación se ve reducido al número de etiquetas del castellano. La información resultante suele utilizarse para analizar estructuras gramaticales frecuentes por medio modelos N-gram de gran intervalo o modelos de secuencias de capas convolucionales de ventana grande.

### 5.1.5 Lemmatization

El proceso de Lemmatization es un proceso lingüístico que consiste en, dada una forma flexionada, hallar el lema correspondiente. Con lema nos referimos a la forma que por convenio se aceta como representante de todas las formas flexionadas de una misma palabra. Por ejemplo:

$$\{luminoso, lumínico, lucidez\} \rightarrow luz$$

### 5.1.6 Stemming

El proceso de Stemming es un método para reducir una palabra a su raíz o stem. Es un proceso muy común en Recuperación de Información. Es algunas ocasiones la raíz de la palabra puede coincidir con su palabra mínima. Por ejemplo:

$$\{campo, campesino, camposanto\} \rightarrow camp$$

## 5.2 Extracción de características

La extracción de características es el proceso de transformación de datos arbitrarios como texto o imágenes en características numéricas. Las características resultantes serán combinación de una o más característica de la matriz de entrada.

### 5.2.1 Bolsa de Palabras

La estructura de datos o representación Bolsa de Palabras o en inglés *Bag of Words* simplifica de forma eficiente el uso de texto en Aprendizaje Automático. Simplemente se basa en realizar un conteo de las apariciones de una palabra a lo largo del corpus. Esto descarta gran parte de la complejidad que aportan las estructuras del lenguaje como capítulos o párrafos. Por lo tanto, llamaremos bolsa a la distribución de las frecuencias absolutas de las palabras en el corpus.

$P(c_1 : N)$ probabilidad de la secuencia de $N$ caracteres en desde $c_1$ hasta $c_N$

Para computar la bolsa de palabras se siguen los siguientes pasos:

1. Token: definir la expresión que reconocerá una palabra para nuestro corpus. Ejemplo: una palabra es un conjunto de caracteres rodeados por espacios.
2. Tokenización: dividir cada documento en función del criterio que se haya definido en el token.
3. Construcción del vocabulario: realizar la unión algebraica de todos los conjuntos extraídos por la tokenización de cada documento.
4. Codificación: realizar el conteo de las apariciones de las palabras del vocabulario sobre los documentos.

Usualmente, para aumentar la probabilidad de emparejamiento de palabras y reducir la bolsa de palabras se siguen algunas pautas:

- Unificación de mayúsculas minúsculas: se realiza una transformación sobre los caracteres del corpus pasándolos todos a minúscula, por ejemplo. En algunos corpus en los que abunden muchos nombres propios o de organizaciones puede llevar a un encarecimiento de la información extraía. El apellido “Pino” y el árbol “pino” pasarían a ser el mismo token dentro de la bolsa.
- Lemmatization: mediante un algoritmo se extrae la palabra de la que deriva la original. Algoritmos como *WordNetLematizer* o *Lancaster,* entre otros, nos ayudan a realizar esa función. Este tipo de algoritmos parten de la idea de que existen agrupaciones de palabras que describen grafos dirigidos con grandes sumideros. Aprovechar esos sumideros a costa de perder cierto grado de información puede ser interesante si aumenta el número de emparejamientos. Se puede encontrar un ejemplo en el apartado de Lemmatization
- Derivación: es una operación que comparte el mismo objetivo lemmatization. La principal diferencia es que en este caso se realiza una extracción de la raíz.
- Eliminar acentuación: es una práctica muy común en los idiomas que provienen del latín.

Con el objetivo de eliminar palabras extremadamente poco frecuentes y muy frecuentes se definen dos umbrales:

- Frecuencia de documento mínima: se trata de la frecuencia relativa mínima que admitimos para considerar una palabra dentro de la bolsa de palabras.
- Frecuencia de documento máxima: se trata de la frecuencia relativa máxima que admitimos para considerar una palabra dentro de la bolsa de palabras.

Un tipo de umbral con el mismo propósito es el que limita el tamaño del vocabulario. Usualmente se seleccionan las $k$ tokens más frecuentes encontrados.

Muchas de las librerías de procesamiento del lenguaje natural ofrecen conjuntos de palabras por idioma llamados **StopWords**. Estos grupos recogen las palabras más frecuentes que sirven como nexos entre ideas. Por lo general se sigue el criterio de que si se quitaran de una frase no se debería de ver mermada la información que se refleja. Dependiendo del conjunto de datos y problema, la eliminación de las StopWords puede aportar mejoras significativas en los resultados de la predicción.

Una de las principales desventajas que tiene esta estructura es que perdemos la información de la posición que tiene cada palabra con respecto al resto. Esta propiedad tiene una relación directa con el lenguaje, ya que puede no tener el mismo significado una palabra al principio de una frase, en medio o al final. Este significado dependiente de la posición debemos trasladárselo al modelo. Una de las formas más frecuente de hacerlo es partir del parámetro *N-gram*. Consideras un conjunto de palabras extra que cumple las mismas propiedades anteriormente descritas pero que define su token como unión de dos o más palabras dependiendo del rango que fijemos. Los valores frecuentes para el rango de *N-gram* van desde *1-gram* hasta *3-gram* inclusive. Esto se debe a que se experimenta una fuerte disminución de la frecuencia cuando se usan valores mayores de tres.

Este último punto ha adquirido tal importancia que ha conformado un concepto de modelos dentro de la práctica.

### 5.2.2 Modelos *N-gram*

Una aproximación simple se basa en modelar y construir una distribución de probabilidad para una secuencia de caracteres del conjunto de datos.

Definimos $P(c_{i:N})$ como la probabilidad de que la secuencia de $N$ caracteres desde $c_1$ hasta $c_N$. De forma aclarativa podemos decir que lo siguiente en el lenguaje castellano:

$$P(\text{"}que\text{"}) > P(\text{"}agua\text{"})$$

A la secuencia de escrita de símbolos de longitud $n$ la llamamos *N-gram*. Los modelos que trabajan con la distribución de probabilidad sobre las secuencias de caracteres los llamamos **modelos *N-gram***.

Los modelos *N-gram* son generalmente aplicados de cuatro formas dependiendo la tokenización que se realice sobre el corpus:

- Realizado a nivel de **palabras** que se dividen por espacios o signos de puntuación. Los valores habituales no sobrepasan *3-gram*.
- Realizado a nivel de **sílabas**. Usualmente no utilizados debido al coste de análisis morfológico.
- Realizado a nivel de **caracteres**. En estos casos el rango está más abierto dependiendo de la aplicación que se le dé al modelo.
- Realizado a nivel de **caracteres dentro de una misma palabra**. Este caso es más frecuente y común que el anterior, ya que no suele tener mucho sentido que la cadena del vocabulario pertenezca a dos palabras diferentes de un texto.

Recordemos brevemente que una cadena de Markov es un modelo estocástico que describe una secuencia de posibles eventos donde la probabilidad de cada evento solo depende del estado alcanzado en el evento anterior.

Un *N-gram* se puede definir como una cadena de Markov de orden $n-1$, puesto que la probabilidad sobre el carácter $c_i$ depende exclusivamente de los caracteres que le preceden. Por lo tanto, para un *3-gram*:

$$P(c_i|c_{1:i-1}) = P(c_i|c_{i-2:i-1})$$

La probabilidad para una secuencia de caracteres $P(c_{i:N})$ para un modelo *3-gram* factorizando la regla de la cadena:

$$P(c_{i:N}) = \prod_{i=1}^{N} P(c_i|c_{1:i-1}) = \prod_{i=1}^{N} P(c_i|c_{i-2:i-1})$$

Justificándose $i-2$ como consecuencia del orden de la cadena de Marcov.

Una aplicación muy común de este modelo es en la identificación del idioma en el que está escrito un documento. Gracias al aprendizaje de los *N-gram* de cada lengua se puede estimar una probabilidad de pertenencia al mismo.

Seguidamente se describen las 2 tokenizaciones más comunes de la literatura.

#### 5.2.2.1 Modelos *N-caracteres*

Los modelos basados en *N-caracteres* tiene un rango de aplicación escaso para documentos de tamaño medio y largo debido a la explosión combinatoria que se produce al elevar el termino $N$. Pongamos un ejemplo de ello limitándonos exclusivamente a los símbolos contenidos en el abecedario del castellano.

El abecedario castellano está compuesto de 27 letras que conforman las siguientes cotas superiores de combinatoria:

| *N-gram* | Cota superior |
|---|---|
| 1 | 27 |
| 2 | 729 |
| 3 | 19.683 |
| 4 | 531.441 |

Estos números quedan reducidos por la imposibilidad de algunas combinaciones de letras.

(Russell & Norvig, N-gram character models, 2010)

#### 5.2.2.2 Modelos *N-palabras*

Los modelos basados en *N-palabras* presentan vocabularios muchos más extensos que los basados en caracteres. Esto se debe a la sencilla idea de que en el lenguaje existen menor número de símbolos únicos que palabras divididas por espacios o símbolos de puntuación.

Un riesgo adjunto del uso de estos modelos son las **palabras fuera del vocabulario**. Es muy común que puedan existir palabras en el conjunto de validación que no estén contenidas en el conjunto de entrenamiento. La forma de resolver este inconveniente pasa por crear una palabra para el conjunto que agrupe todas esas posibles palabras desconocidas.

También existen remplazamientos en forma de expresiones de regulares de todos los números o de emails por una palabra única del lenguaje.

Actualmente el uso de *1-palabras* ha quedado limitado para uso representativo y se considera acercamiento pobre como extracción de características.

(Russell & Norvig, N-gram word models, 2010)

### 5.2.3 Vectorización de documentos

El proceso de vectorización de un conjunto de documentos se resume en los siguientes puntos:

1. Construcción de un vocabulario a partir del corpus.
2. Siendo el número de componentes del vocabulario $N$, para cada documento $d_i$ extraemos su vector correspondiente $v_i$ que recoge las frecuencias de cada uno de los tokens en dicho documento.

Un ejemplo de aplicación de este proceso sobre la siguiente frase:

$$d_i = "que\ es\ lo\ que\ quieres"$$

Construcción del vocabulario:

| Palabra | Frecuencia |
|---|---|
| *que* | 2 |
| *es* | 1 |
| *lo* | 1 |
| *quieres* | 1 |

Vectorización del documento:

$$v_1 = (2,1,1,1)$$

Cuando se realiza este proceso sobre un número ingente de documentos el vocabulario que se obtiene genera vectores individuales con una alta frecuencia de valores 0. El conjunto de estos vectores se conforma en una matriz $M_{n\,x\,m}$ siendo $n$ el número de documentos y $m$ el tamaño del vocabulario. Esta matriz para ser almacenable en memoria requiere de una estructura de datos especial llamada **matriz dispersa**. Las matrices dispersas son un conjunto de ternas $(i,j,k)$ donde $i,j$ son los índices de posición de la matriz y $k$ el valor que toma esa posición. El resto de combinación de índices no recogida, se consideran 0.

El uso de matrices dispersas tiene como ventajas:

- Disminución en el uso de memoria
- Disminución del tiempo consumido de cómputo para ciertas operaciones

Pero también presenta una desventaja notable:

- Muchos de los modelos de Aprendizaje Automático no aceptan representaciones dispersas de las características, obligándonos a realizar entrenamientos parciales o incrementales tras una posterior transformación a matrices densas. Algunos modelos tampoco aceptan entrenamientos parciales.

### 5.2.4 TF-IDF

Es un uso más que extendido dentro del proceso de extracción de características sobre texto la aplicación de TF-IDF tras la vectorización de nuestro corpus.

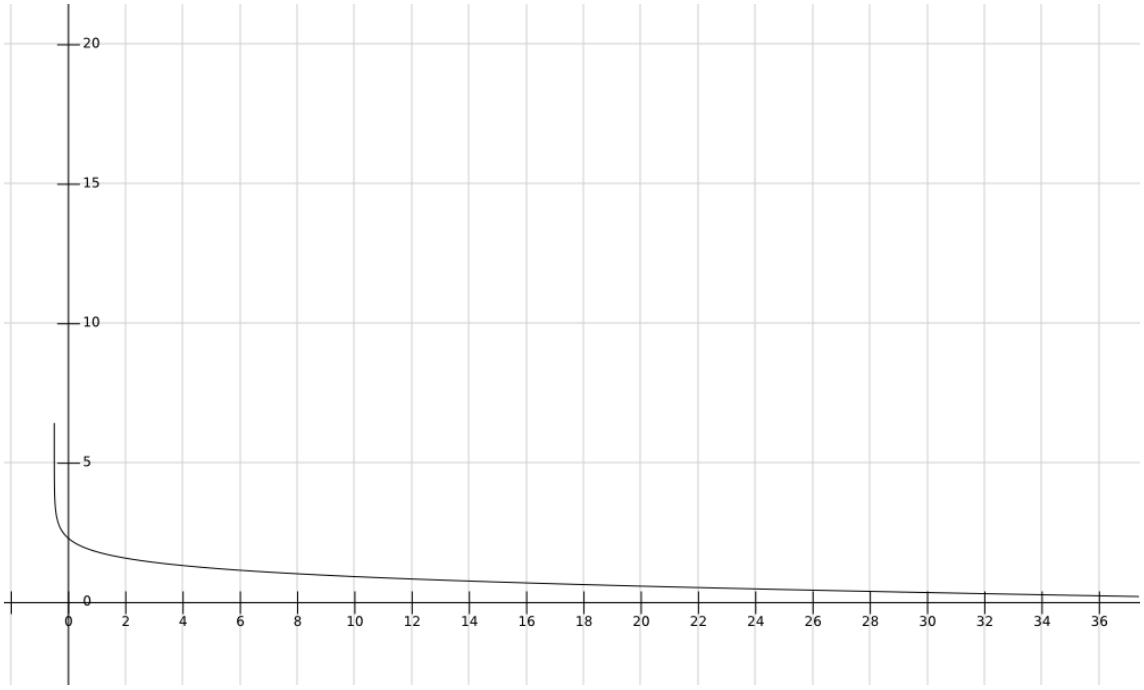
TF hace alusión a la frecuencia de un término dentro del corpus mientas que IDF refleja la inversa de la frecuencia sobre el documento. Se trata de una función de puntuación llamada **BM25**. Fue propuesta por Stephen Robertson y Karen Sparck Jones.

El objetivo TF-IDF es mezclar en una misma coordenada del documento la importancia del token sobre el corpus y sobre el documento. Hasta ahora el proceso de vectorización sólo recogía propiedades elegidas por el corpus con información exclusivamente del documento.

Definición de IDF:

$$IDF(q_i) = \log \frac{N - DF(q_i) + 0.5}{DF(q_i) + 0.5}$$

Donde $N$ es el número de documentos en el corpus y $DF(q_i)$ el número de documentos que contienen $q_i$. Analicemos el comportamiento de esta función mediante la siguiente gráfica para un valor de $N = 100$:




Es observable que conforme aumenta el número de documentos que contienen $q_i$ el valor de IDF cae logarítmicamente decayendo su importancia.

La función TF-IDF es una función en dos variables que se aplica sobre el documento $d_j$ el token $q_i$ y que tiene la siguiente expresión:

$$TF - IDF(d_i, q_{i:N}) = \sum_{i=1}^{N} IDF(q_i) \cdot \frac{TF(q_i, d_j) \cdot (k+1)}{TF(q_i, d_j) + k \cdot (1 - b + b \cdot \frac{|d_j|}{L}}$$

Donde $TF(q_i, d_j)$ es el número de veces que aparece el token $q_i$ en el documento $d_j$, $L$ es la media de tokens que aparecen por documento, $k = 2.0$ y $b = 0.75$ usualmente.

Un punto interesante de esta función es asignar el valor 0 donde el proceso de vectorización asigne un 0. De otro modo, la eficiencia en memoria de la representación dispersa de la matriz se volvería en nuestra contra. Esto es fácilmente comprobable gracias a la multiplicación del numerador de la fracción.

(cita)

### 5.2.5 Secuencias

La extracción de características basada en secuencias puede considerarse en muchos casos como una ampliación el conocido preprocesamiento llamado codificación.

Primeramente, se realiza una tokenización como en el apartado de los modelos *N-gram*. Existen tres tipos usuales de tokenización dependientes de propiedades del conjunto de datos:

- Tokenización a **nivel de caracteres**: de cada documento se extrae el conjunto de caracteres que lo forma. Usualmente aplicado para conjuntos de documentos con número de palabras medio pequeño.
- Tokenización a **nivel de sílabas**: de cada documento se extrae el conjunto de sílabas que lo forma. Usualmente aplicado cuando se posee una herramienta fiable de análisis morfológico para el lenguaje del corpus y el conjunto de documentos presenta un número de palabras pequeño. (Jacquemin & Tzoukermann, 2000).
- Tokenización a **nivel de palabras**: de cada documento se extrae el conjunto de palabras que lo forma. Usualmente aplicado sobre conjunto de variabilidad alta en el número de palabras por documento.

Este último tipo de tokenización es el que se ha aplicado en este trabajo. Se puede ver la justificación de su aplicación en el apartado de limpieza de datos.

Se realiza la unión de todos estos conjuntos de palabras y se les asigna una numeración ascendente empezando desde el número 1. De esta forma, se puede sustituir cada palabra por su codificación. Obtenemos los vectores de longitud variable para cada documento dependiendo el número de palabras que tengamos.

De cómo sea la varianza del número de componentes de estos vectores, dependerá la dimensión máxima elegida para todos ellos. Se suele completar con ceros aquellos que sean menores a esa dimensión máxima y truncar aquellos que la superen.

A continuación, veremos un ejemplo de su aplicación de este proceso para 2 frases:

| Frases |
|---|
| "***que es lo que quieres***" |
| "***que es lo que haces ahi***" |

Procedemos con la tokenización del conjunto de datos por palabras obteniendo la siguiente codificación y vectores:

| Palabra | Código |
|---|---|
| "***que***" | 1 |
| "***es***" | 2 |
| "***lo***" | 3 |
| "***quieres***" | 4 |
| "***haces***" | 5 |

| Frases | Vector |
|---|---|
| "***que es lo que quieres***" | $(1,2,3,1,4)$ |

| | |
|---|---|
| "***que es lo que haces***" | (1,2,3,5) |

Fijamos 5 como dimensión máxima de los vectores dando lugar a esta transformación:

| Frases | Vector |
|---|---|
| "***que es lo que quieres***" | (1,2,3,1,4) |
| "***que es lo que haces***" | (1,2,3,5,0) |

La propiedad principal que tiene este proceso de extracción de características que lo diferencia de la bolsa de palabras es la de respetar el orden de los datos conforme al origen. Esto da lugar a usar modelos que planteen aprendizajes en forma de ventana, como es el caso de las redes convolucionales.

Un factor decisivo en este proceso es la dimensión máxima del vector que siempre esta acotada superiormente por el documento que presente el vector de mayor dimensión. Siempre que los recursos computaciones lo permitan es conveniente elegir valores entre la media de dimensión de los documentos y la dimensión máxima.

## 5.3 Selección de características

El proceso de selección de características es aquel que elige un subgrupo de las características existentes en función de un criterio definido con el objetivo de discernir que características son influyentes en el problema.

Existe muchos algoritmos de selección de características. A continuación, mediante la siguiente tabla se resume la utilidad de algunos dependiendo de las características de entrada y del tipo de etiqueta.

| Input \ Output | Numérica | Categórica |
|---|---|---|
| **Numérica** | Pearson's, Spearman's | ANOVA, Kendall's |
| **Categórica** | ANOVA, Kendall's | Chi-Squared, Mutual Information |

En nuestro caso, teniendo en cuenta que la salida de una extracción de características sobre texto siempre es numérica, como se ha mencionado en el apartado anterior, nos limitaremos a explicar ANOVA y Kendall's.

### 5.3.1 ANOVA

El test ANOVA tiene las siguientes asunciones que deben satisfacerse para poder computar un $p - value$ valido.

- Las muestras deben ser independientes
- Cada muestra debe seguir una distribución normal

- La desviación estándar de la población de los grupos debe ser la misma. Esta es la propiedad conocida como homocedasticidad.

Si dice que un modelo es homocedástico si en todos los grupos de datos de una observación, la varianza del modelo respecto de las variables explicativas se mantiene constante.

El computo del F-Valor viene dado por la siguiente expresión:

$$F = \frac{\sum_{i=1}^{K} \frac{n_i(\overline{Y_i} - \bar{Y})^2}{K-1}}{\sum_{i=0}^{K} \sum_{j=1}^{n_i} \left(Y_{ij} - \bar{Y_i}\right)^2 / (N-K)}$$

Donde $K$ es el número de grupos; $\bar{Y_i}$ es la media de la columna, y $n_i$ el número de elementos en la columna. $K-1$, $N-K$ se corresponde con los grados de liberta de la distribución de Fisher.

Este test se realiza para cada una de las características extraídas por el modelo *N-gram* obteniendo las $k$ que presenten mejores resultados en el test.

### 5.3.2 T - Kendall's

En estadística, el coeficiente de correlación de Kendall es usado para medir la asociación ordinaria entre 2 mediciones cuantitativas.

Este coeficiente se puede expresar mediante la siguiente fórmula:

$$\tau = \frac{n_c - n_d}{\binom{n}{2}}$$

Donde $n_c$ es el número de pares de puntos sobre las dos muestras que son concordantes. Es decir, que son mayores o menores en sus dos coordenadas:

$$(x_i, y_i)\ concuerda\ con\ (x_j, y_j)\ si\ (x_i < x_j\ \&\ y_i < y_j)\ |\ (x_i > x_j\ \&\ y_i > y_j)$$

Donde $n_d$ es el número de pares de puntos sobre las dos muestras que con discordantes. Es decir, que son mayores en una coordenada y menores en otra.

Los puntos que son iguales en ambas coordenadas no se consideran ni concordantes ni discordantes.

Donde $\binom{n}{2}$ es el coeficiente binomial para la cantidad de formas de elegir dos elementos de $n$ elementos.

Definido el coeficiente de correlación se calcula la matriz de correlación de todas las características. Este proceso se suele aplicar de dos formas:

- Seleccionar aquellas características que no presenten correlación superior a un umbral con ninguna característica.
- Seleccionar las $n$ características que presente menor correlación con el conjunto.

## 5.4 Reducción de la dimensión

El proceso de reducción de la dimensión de un problema pretende realizar una simplificación de la matriz de características perdiendo la mínima información posible del problema. En algunas referencias este proceso se incluye en la extracción de características debido a su semejanza. Este proceso no se debe aplicar necesariamente cuando nos encontremos con problemas de alta dimensión, ya que existe modelos que tienen comportamientos aceptables en estos casos. La reducción de la dimensionalidad suele requerir en muchos casos un tiempo de cómputo demasiado grande. Es por esto que generalmente se han implementado métodos iterativos que aproximen a valores teóricos de publicaciones.

Sin duda, el método más conocido dentro de esta área es el Análisis de Componentes Principales o por sus siglas en inglés PCA. En este trabajo no se va a tratar éste debido a que no se puede aplicar sobre representaciones de datos dispersos. La representación de datos dispersa elimina la posibilidad de aplicar gran parte de los métodos disponibles como puede ser el Análisis de Independiente de Componentes o por sus siglas en inglés ICA. Los tres métodos más conocidos bajo estas restricciones son LDA, LSA y NMF. Nos centraremos en explicar estos dos últimos por sus buenos resultados.

### 5.4.1 LSA

El Análisis Semántico Latente es un proceso de reducción de la dimensionalidad estrechamente ligado a la extracción de características sobre texto. LSA es una técnica de Procesamiento de Lenguaje Natural al mismo tiempo que una técnica del aprendizaje no supervisado. Por tanto, como su nombre indica, está buscando conocimiento latente o inherente en los datos de por sí solos, por medio de representaciones del texto en términos de temáticas y palabras clave. La aplicación de LSA suele ir después de la vectorización y la transformación TF-IDF de nuestro corpus.

El primer paso para el computo de LSA es la Descomposición en Valores Singulares, o por sus siglas en inglés SVD, de la matriz de características. Resumiendo brevemente este proceso, expresamos la matriz original a partir de tres matrices:

$$\mathrm{M} = \mathrm{U}\Sigma\mathrm{V}^{*} = \sum_{i=1}^{m} \sigma_i u_i v_i^{*}$$

Supongamos que $M_{n\,x\,m}$ tiene $n$ filas y $m$ columnas, entonces las matrices tendrán las siguientes dimensiones $U_{n\,x\,n}, \Sigma_{n\,x\,m} V^{*}_{m\,x\,m}$ donde la diagonal de $\Sigma$ refleja los valores singulares de $\mathrm{M}$ ordenados de mayor a menor. Aplicándolo a un problema de ML, $m$ representa el número de características y $n$ el número de datos.

La forma más común de proceder es fijar un número máximo de compontes a calcular sustituyendo en la fórmula anterior $m$ por un $k$:

$$\mathrm{M}_k = \sum_{i=1}^{k} \sigma_i u_i v_i^*$$

A este cambio sobre el cálculo de la SVD se le conoce como SVD truncada.

(Edelman, 2016)

(Guruswami & Kannan, 2012)

### 5.4.2 NMF

La factorización no negativa de matrices en un proceso de reducción de la dimensionalidad con el objetivo de extraer características sobre un matriz ancha.

El proceso NMF asume que la matriz cumple para todos sus componentes ser positivo. Este requisito es idea para los modelos de clasificación de texto clásicos ya que la aplicación de NMF iría justo después del calculo de frecuencias. Puesto que no existen frecuencias de palabras que tengan un valor negativo su aplicación es siempre posible.

El modelo NMF viene dado por la siguiente expresión:

$$\mathrm{X} \approx \mathrm{WH}$$

Donde, $\mathrm{X}$ es $n \times p$, $\mathrm{W}$ es $n \times r$, $\mathrm{H}$ es $r \times p, r \leq p$.

Tras aplicar el modelo WH también serán matrices positivas en todos sus componentes.

Problema de minización:

$$L(\mathrm{W}, \mathrm{H}) = \sum_i \sum_u [X_{iu} \log(WH)_{iu} - (WH)_{iu}]$$

Se puede resolver mediante un proceso iterativo que actualiza las matrices $WH$ para obtener el máximo local de $L(\mathrm{W}, \mathrm{H})$:

$$w_{ik} \leftarrow w_{ik} \frac{\sum_{j=1}^{p} h_{kj} x_{ij} / (WH)_{ij}}{\sum_{j=1}^{p} h_{kj}}$$

$$h_{kj} \leftarrow h_{kj} \frac{\sum_{i=1}^{N} w_{ik} x_{ij} / (WH)_{ij}}{\sum_{i=1}^{N} w_{ij}}$$

(D. Lee & Sebastian Seung, 2001)

(Moitra, s.f.)

## 5.5 Aprendizaje

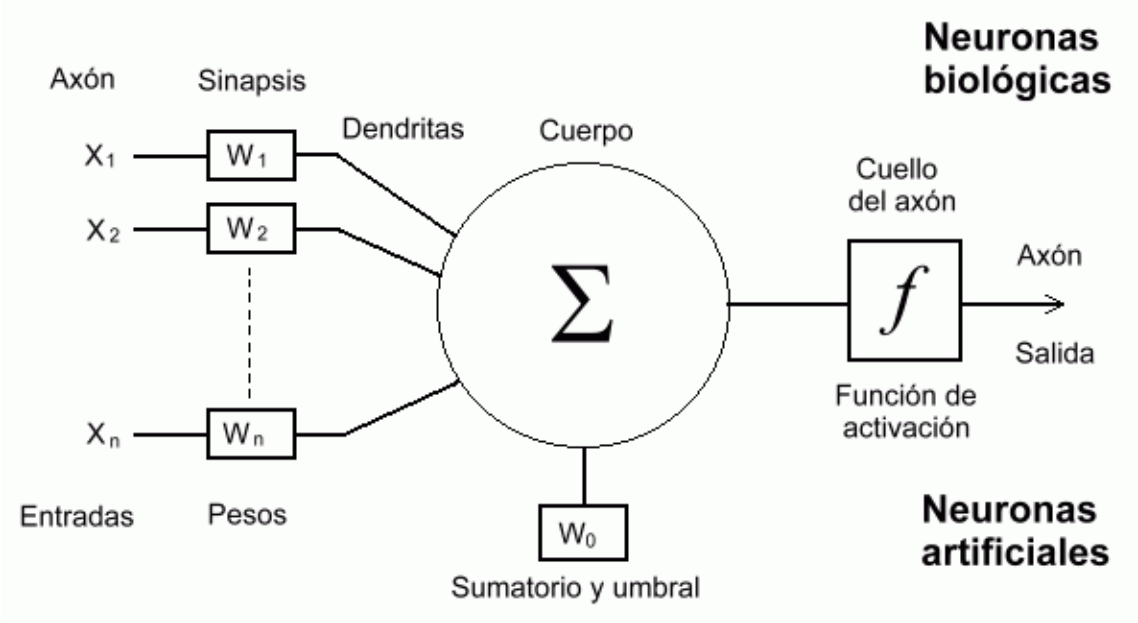
### 5.5.1 Red Neuronal

Uno de los modelos más conocidos dentro de Aprendizaje Automático son las redes neuronales artificiales. Un modelo inspirado en el comportamiento del cerebro humano que pretenden simular el intercambio de información entre las **neuronas**.

Las redes neuronales se estructuran en capas que forman un grafo dirigido que parte desde los inputs de las características hasta los outputs del etiquetado. Las capas intermedias que no son ni inputs ni outputs se les llama capas ocultas. Las capas están formadas por neuronas. Cada capa puede tener un número deferente de neuronas.

Usualmente a los nodos del grafo se les llama unidades. Al enlace que une la unidad $u_i$ con la unidad $u_j$ se le asocia un peso $w_{ij}$.

A continuación, se muestra la estructura que tendría una neurona o unidad dentro de estos modelos haciendo una comparativa bilógica:




- Entradas: se corresponde con información exterior del modelo o de neuronas conectadas ella.
- Pesos: como hemos descrito antes son los parámetros (usualmente números reales) que debe ajustar el modelo en su aprendizaje.
- Sumatorio: se produce tras multiplicar cada entrada por su peso. De esta forma pasamos de una entrada de dimensión $N$ a una entrada de dimensión 1.
- Función de activación: se aplica al resultado de la sumatoria cerrado el proceso de transformación de los datos.
- Salida: puede ser tanto el resultado final del modelo o las posibles conexiones a otra capa de neuronas.

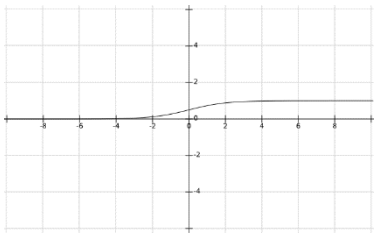
La expresión matemática que resume este proceso es la siguiente:

$$f(\sum_{i=1}^{K} X_i W_i + b)$$

Siendo $f$ la función de activación y $b$ usualmente llamada bias una constante de entrada a la neurona.

Existen varias propuestas de funciones de activación. A continuación, se explican las tres más conocidas y sus casos de aplicación.

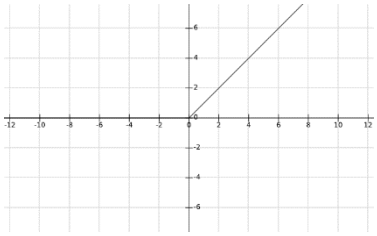
Función Sigmoide



La característica principal de esta función es su recorrido (0,1). Usada en la capa de salida para problemas de clasificación binaria. La expresión matemática que la define es la siguiente:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Función Relu



Función que posibilitó las redes neuronales profundas. Propuesta por Hahnoloser. Suele aplicarse en capas ocultas. La expresión matemática que la define es la siguiente:

$$f(x) = \max(0, x)$$

Función SoftMax

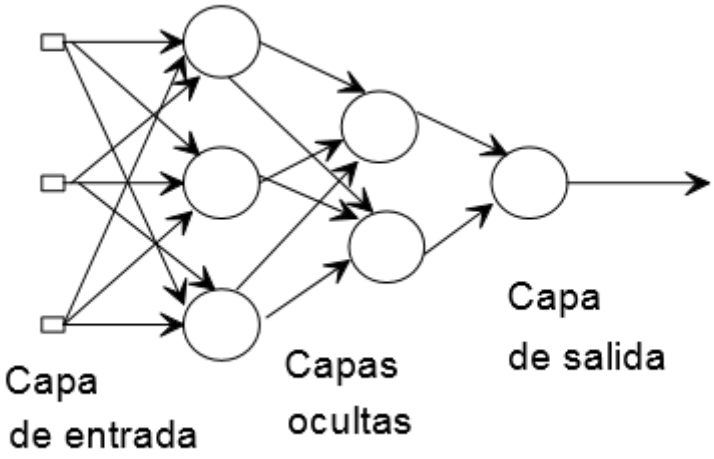
Función Softmax es comúnmente utilizada para clasificación de problemas multietiqueta dentro de entrenamiento de redes neuronales. Solemos encontrarla en la última capa de la red. La función devuelve un vector de $K$ dimensiones, donde $K$ es el número de clases de nuestro problema y cada competente la probabilidad de pertenencia a la clase.

$$f: \mathbb{R}^K \to \mathbb{R}^K$$

$$f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \; para\; i = 1, \dots, K \; y \; x = (x_1, \dots, x_K) \in \mathbb{R}^K$$

Otras funciones derivadas o con las mismas propiedades que éstas se han usado como pueden ser Selu, Elu o tangente.

Cuando muchas neuronas pertenecen a un mismo modelo formando capas se crean estructuras que puede visualizarse de la siguiente forma:




- Capa de entrada: se corresponde con los datos de entrada del problema.
- Capas ocultas: capas que reciben datos calculados por capas anteriores y conectan con capas posteriores.
- Capa de salida: se corresponde con la predicción del algoritmo.

Para realizar el proceso de entrenamiento se emplea el algoritmo de Propagación Hacia Atrás o en inglés **Back Propagation**, elaborado y popularizado por Rumelhart, Hinton y Williams. Resumidamente el algoritmo computa el gradiente de la función de pérdida para cada capa, iterando hacia atrás reutilizando los datos. Este esquema de reutilización es una estrategia de programación dinámica.

Uno de los problemas más difíciles de resolver dentro de estos modelos es el sobreajuste o en inglés **overfit**. Llamamos sobreajuste al efecto de sobreentrenar un modelo de tal forma que abandone su función de aprendizaje por la de memorización.

Algunas de las arquitecturas que se han propuesto en este trabajo se encuentran explicadas en conjunto con sus resultados en el apartado de experimentos.

En el mundo de las redes neuronales y sobre todo del **aprendizaje profundo** se han diseñado un gran número de tipos de capas. Algunos de estos están estrechamente relacionados con el tipo de entrada de datos que se procese.

Las posibles dimensiones de la entrada quedan abstraídas por el concepto matemático de **tensor**. Un tensor generaliza el concepto de escalar, vector y matriz de manera que sean independientes de cualquier sistema de coordenadas. De esta forma los sistemas de definición de redes neuronales (como puede ser TensorFlow) facilitan la entrada de imágenes, sonido, texto… como input.

- Un **escalar** sería un tensor de orden 0.
- Un **vector** sería un tensor de orden 1. Se corresponde con la representación del texto después de una extracción de características tanto por vectorización como por secuenciación.
- Una **matriz** sería un tensor de orden 2. Se corresponde con la representación de una imagen por pixeles.

A continuación, se puede encontrar una breve descripción de algunos de los componentes usados en las arquitecturas de redes neuronales propuestas:

#### 5.5.1.1 Densa

Las capas densas o capas completamente conectas son las más comunes de las redes neuronales. Se basan establecer enlaces una combinatoria completa de enlaces entre la entrada o capa anterior y el número de unidades de esta. De esta forma cada neurona recibe información de todas las entradas.

#### 5.5.1.2 Dropout

Dropout provee un método de regularización para evitar sobreajuste en redes neuronales que no representa un gran coste computacional.

Podríamos considerar como borrar una unidad, multiplicar por 0 la salida de la misma. La técnica dropout consiste en la simplificación del modelo por medio de la eliminación de algunas de las unidades que lo forman. Al tener un modelo más simple que explique nuestros datos aumenta la probabilidad de que se corresponda con la realidad reduciendo nuestro sobreajuste.

(Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014)

#### 5.5.1.3 Convolucionales 1D

Las capas de convolución usan la operación de convolución en lugar de la multiplicación de matrices general para transformar los datos que reciben. La operación de convolución sobre un tensor de orden 1 se pueden entender mediante el concepto de kernel o ventana. Esta ventana 1D del tamaño del kernel definido es capaz de detectar patrones por la relación de cercanía de unas palabras a otras.

(LeCun, Haffner, Bottou, & Bengio, 1989)

#### 5.5.1.4 Pooling

La capa pooling o de agrupación es el último paso requerido en una red neuronal convolucional. Realiza un remplazamiento de la salida por un resumen estadístico de las salidas cercanas. Este resumen estadístico suele presentarse como máximo o como media.

#### 5.5.1.5 Embedding

#### 5.5.1.6 LSTM

#### 5.5.1.7 GRU

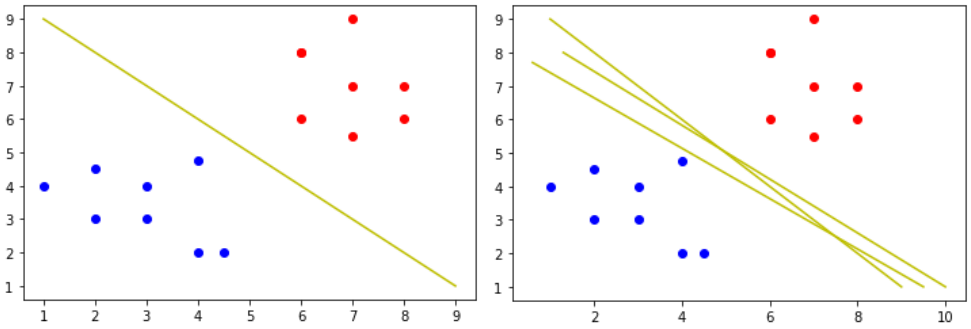
### 5.5.2 Máquina de Vectores Soporte

La Máquina de Vectores Soporte es uno de los modelos más usados dentro del Aprendizaje Automático supervisado. Se puede encontrar implementado en la mayor parte de bibliotecas genéricas de ML. Hay cinco propiedades que lo hacen diferenciarse al resto de modelos:

1. Pretende **maximizar el margen de separación** del conjunto de datos con el objetivo de tener un modelo más genérico y robusto.
2. Originalmente, muchos problemas presentan datos que no son linealmente separables para las dimensiones originarles. SVM haciendo uso de **kernels** aumenta las dimensiones del problema llevándolo a un espacio donde los datos son linealmente separables por un hiperplano.
3. SVM es un **modelo no paramétrico**. Ser no paramétrico quiere decir que la complejidad de su espacio de hipótesis crece según lo hace el número de datos de entrada. Esto representa una ventaja frente al sobreajuste y una desventaja frente a uso de los recursos. Está desaconsejado el uso de SVM para conjuntos de datos grandes.
4. SVM permite realizar entrenamientos con representaciones de datos dispersas, lo que lo hace un modelo ideal en combinación con una extracción de características sobre texto.
5. SVM debido a su característica no paramétrica **no** permite realizar **entrenamientos parciales o incrementales**. El uso actual para conjuntos de datos que no son posibles alojar en memoria requiere usar plataformas de entrenamiento más sofisticadas. En la publicación científica *'Incremental and Decremental Support Vector Machine Learning'* apuntan una modificación del algoritmo que permite el entrenamiento parcial. Por otro lado, la mayor parte de las guías concluyen en cambiar de modelo si se produjera esta situación.

**Margen**

Imaginemos la aplicación de un modelo SVM sobre un problema de clasificación binario linealmente separable, donde el número de hiperplanos posibles que dividan el espacio de las etiquetas sea superior a 1. Por lo tanto, existan infinitos hiperplanos posibles que realicen la separación.



En términos relativos, podríamos definir una “seguridad” en la predicción realizada diciendo que puntos más alejados de la frontera de decisión representan una mayor seguridad sobre datos más cercanos. En la gráfica de la izquierda se puede observar que la coordenada $(7,9)$ se encuentra más lejana de la frontera de decisión que la $(7,5)$.

Refiriéndonos a la gráfica de la derecha, cómo podemos saber cuál de los hiperplanos o separadores es el que mejor explica nuestro conjunto de datos:

- ¿El qué mayor número de puntos clasifique?
- ¿El que deje mayor distancia entre la frontera de decisión y el conjunto de puntos?

Generalizar estas dos ideas para todos los posibles hiperplanos que separan un conjunto de datos requiere una notación previa.

**Notación**

Consideraremos las etiquetas $y \in \{-1,1\}$ en lugar de $\{0,1\}$.

El parámetro independiente del clasificador se demorará de la siguiente forma:

$$h_{w,b}(x) = g(w^T x + b)$$

Donde $g(z) = 1$ si $z \geq 0$, si no $g(z) = -1$.

Problema de minimización

Resolución por Lagrange

Kernels

Regularización

Concepto de margen

Definamos el siguiente problema de se

Enlace: http://cs229.stanford.edu/notes/cs229-notes3.pdf

# 6 Metodología de experimentación

A continuación, se explican las métricas usadas en los experimentos para comparar modelos, estrategias de optimización de hiperparámetros y técnicas para garantizar la independencia de la partición de datos.

## 6.1 Métricas

Puesto que se trata de un problema clasificación multietiqueta, las métricas que se muestran descritas solo se centrarán en este tipo de problemas. Para la implementación de este proyecto se ha diseñado una métrica específica para el problema como combinación de otras, ya que existe una aparente homogeneidad en la distribución de las clases.

Definamos los siguientes dos vectores y un escalar:

- $y_{pred}$: conjunto de etiquetas predichas por un algoritmo.
- $y_{true}$: conjunto de etiquetas reales provenientes del conjunto de datos original.
- $n$: número total de datos

En el siguiente cuadro se encuentra resumida la explicación de verdadero (T), falso (F), positivo (P) y negativo (N) en su correcta combinación:

| | | Valor Actual | |
|---|---|---|---|
| | | Verdadero (T) | Falso (F) |
| Valor Predicho | Positivo (P) | TP | FP |

| | | | |
|---|---|---|---|
| | Negativo (N) | TN | FN |

Las tres últimas métricas definidas son aplicables a las clases individuales de la clasificación, por lo que es frecuente encontrar para problemas multietiqueta grandes estos valores expresados como media aritmética de todas las clases.

### 6.1.1 Exactitud o *accuracy*

Número total de predicciones correctas entre el total de predicciones realizadas:

$$acc = \frac{\sum_{i=1}^{k} y_{pred_i} == y_{true_i}}{k}$$

### 6.1.2 Precisión o *precision*

Número de verdaderos positivos entre la suma de los verdaderos positivos y los falsos positivos.

$$precision = \frac{TP}{TP + FP}$$

### 6.1.3 Exhaustividad o *recall*

Número de verdaderos positivos entre la suma de los verdaderos positivos y los falsos negativos.

$$recall = \frac{TP}{TP + FN}$$

### 6.1.4 F1 Score

Dos dividido entre la suma del inverso de la precisión y la exhaustividad.

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

## 6.2 Ajuste de hiperparámetros

El ajuste de hiperparámetros es el proceso que se sigue para encontrar los parámetros óptimos de un algoritmo a un determinado problema. Dependiendo de la anchura de la búsqueda se siguen 3 estrategias diferentes:

### 6.2.1 Búsqueda en rejilla

La búsqueda en rejilla sigue la estrategia de definir los dominios discretos de búsqueda para todos los parámetros a probar realizando una combinatoria exhaustiva (probando todas las posibilidades). Se suele implementar en conjunto con una validación cruzada. Su uso es frecuente cuando se pretender realizar pequeñas búsquedas.

Es la estrategia más usada a lo largo del proyecto una vez acotados los márgenes de búsqueda.

### 6.2.2 Búsqueda aleatoria

Como la estrategia anterior se definen los dominios de los parámetros discretos, pero en este caso solo se prueba un subconjunto de la combinatoria generada. De esta forma tenemos una idea de la influencia de los mismos sobre el problema sin necesidad de gastar un tiempo de cómputo mayor. Como desventaja existe la posibilidad de que la mejor combinación de parámetros sea la no probada.

Es la estrategia usada al principio del proyecto cuando solo se disponía un estudios teóricos o empíricos sobre otros conjuntos de datos.

### 6.2.3 Búsqueda genética

Este tipo de búsqueda está destinada a modelos con un campo de búsqueda combinatorio muy grande, como puede ser el caso de las redes neuronales. En este campo se define una vez más los dominios de los parámetros, aunque en este caso pueden no estar acotados e incluso ser continuos. Mediante una población inicial pequeña extraída de la combinatoria se definen operadores de mutación y cruce que convengan a la estructura de datos. En cada generación se van quedando los individuos más prometedores en funciones de las métricas definidas anteriormente. Se suele requerir de plataformas distribuidas de entrenamiento para esta estrategia.

Esta estrategia es la que se ha seguido de forma ‘simulada’ para los modelos de entrenamiento profundo.

(F. Barrero, Gonzalez-Pardo, Camacho, & Dolores R-Moreno, 2020)

## 6.3 Validación cruzada

Es una de las técnicas más usadas en combinación con el ajuste de hiperparámetros para reducir el sobreajuste del modelo. Se trata de realizar un $n$ particiones del conjunto

de entrenamiento. Se entrenan tantos modelos como particiones tomando como validación un conjunto diferente para cada uno y como entrenamiento el resto.

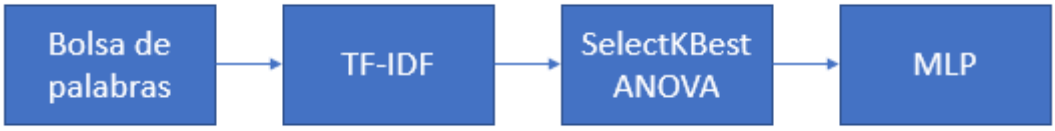
A continuación, se pueden ver estas divisiones e iteraciones para 5 particiones, que es el valor estándar usado y el que se ha seguido en este proyecto.

| | | | | |
|---|---|---|---|---|
| **División 1** | División 2 | División 3 | División 4 | División 5 |
| División 1 | **División 2** | División 3 | División 4 | División 5 |
| División 1 | División 2 | **División 3** | División 4 | División 5 |
| División 1 | División 2 | División 3 | **División 4** | División 5 |
| División 1 | División 2 | División 3 | División 4 | **División 5** |

# 7 Experimentos

A lo largo de este apartado se recogen algunas propuestas de arquitecturas haciendo uso de los conceptos teóricos explicados anteriormente.

## 7.1 Modelo TF-IDF ANOVA MLP




Modelo compuesto de 4 componentes:

1. Bolsa de palabras para realizar una extracción de características sobre texto.
2. TF-IDF transformación que mantiene las dimensiones del conjunto de características, pero aporta información extra.
3. Selección de las mejores características en función del test ANOVA.
4. Red neuronal multicapa perceptrón para la clasificación de las etiquetas a partir de las características extraídas y seleccionadas.

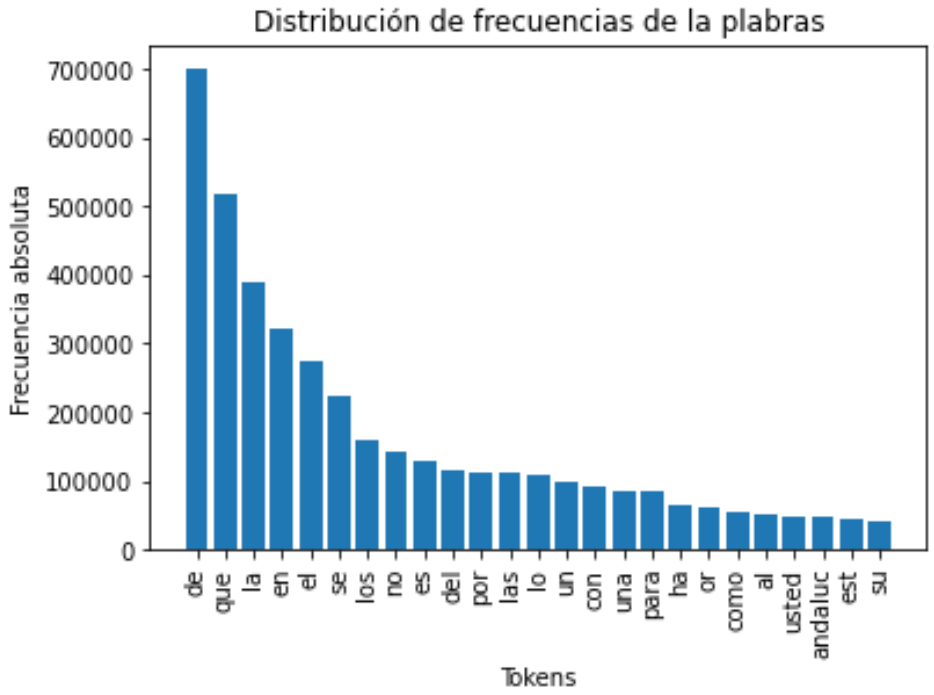
Bolsa de palabras

Para el conjunto de datos que nos atañe se ha seguido un análisis o tokenización a nivel de palabras. En la siguiente tabla se puede ver las 10 más frecuentes de todo el corpus:

| Palabra | Frecuencia absoluta |
|---|---|
| *de* | 700.842 |
| *que* | 518.938 |
| *la* | 389.484 |
| *en* | 321.489 |
| *el* | 272.735 |
| *se* | 223.706 |

| | |
|---|---|
| *los* | 158.813 |
| *no* | 141.399 |
| *es* | 128.702 |
| *del* | 116.744 |

Para este análisis sólo se han eliminado los acentos y se han pasado todas las palabras a minúscula, dando un total de 64.259 palabras o características *1-gram*. Se puede ver a continuación un diagrama de barras de las 25 primeras ordenadas por frecuencia:




Sobre nuestro conjunto de datos obtenemos las siguientes características para valores de *N-gram*:

| *N-gram* | Número de características |
|---|---|
| (1,1) | 64.259 |
| (2.2) | 1.360.480 |
| (3,3) | 4.451.936 |

Para la extracción de características de este modelo se ha usado el rango de *N-gram* (1,2) teniendo un total de 1.424.739 características. Ha este resultado se le ha aplicado tres filtros sobre la frecuencia:

- Frecuencia relativa mínima del documento: $\frac{1.0}{1000.0}$.
- Frecuencia relativa máxima del documento: $\frac{999.0}{1000.0}$.
- Número máximo de características: 100.000.

Tras la aplicación de estos tres filtros, el conjunto de características se reduce a 63.475.

Un último filtro mencionado en el apartado de bolsa de palabras es el conjunto de ***stopwords***. Un conjunto predefinido de tokens para los cuales sabemos que no suelen aportar información útil al modelo. Normalmente se suele incluir dos conjuntos:

- Signos de puntuación.
- Palabras comunes del lenguaje usado en el corpus.

Estas palabras se eliminan como posibles características del modelo reduciendo el ruido que provocan en la clasificación. En este caso hemos utilizado 345 de las cuales 32 son de puntuación y 313 comunes del lenguaje (castellano). Algunos ejemplos de palabras comunes son:

- Preposiciones: por, con, para…
- Determinantes: la, el…
- Conjunciones: y, o, ni…
- Formas verbales: estoy, está, habrán, tuviera…

Tras la aplicación del filtro por *stopwords* reducimos el conjunto de características a 34.344. Si no tuviéramos una representación en memoria dispersa nos sería imposible trabajar con ese volumen de datos.

Para la aplicación de la vectorización, uno de los valores de referencia para la comparación de procesos de extracción de características sobre texto es la dispersión de los datos sobre la matriz vectorizada del corpus. En nuestro caso ese valor es el siguiente:

$$dispersión = 0{,}64\%$$

El valor se puede calcular fácilmente dividiendo el número de ternas de la matriz dispersa entre la multiplicación de las dimensiones de la matriz. La interpretación de estos datos es que el 0.64% de los valores de la matriz es distinto de 0. Esto no debe sorprendernos y debe ser interpretado positivamente. Los documentos no comparten un gran número de características, pero tampoco un número muy pequeño como para que no podamos sacar ningún patrón común. Algunas publicaciones apuntan que sobrepasar el 1% de dispersión en este tipo de extracción de características se consideraría una mala extracción.

Para finalizar este apartado, una de las representaciones gráficas más comunes de las bolsas de palabras es la nube de palabras. El tamaño de la palabra refleja su frecuencia en el texto.

TF-IDF

Aplicamos la transformación TF-IDF sobre los datos ponderando como hemos apuntado antes a la baja a palabras muy comunes dentro del conjunto de datos. Esta transformación no altera la dimensión de nuestra matriz de características.

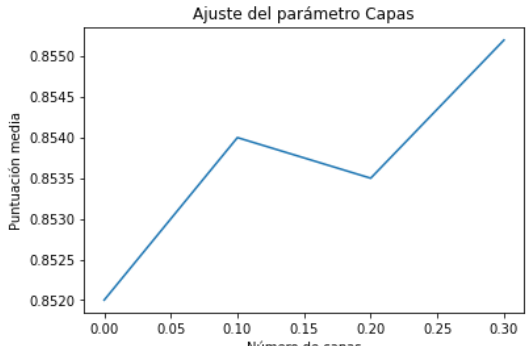
SelectKBest ANOVA

Llamamos al proceso de selección de características. Es de sobra conocido que, en la extracción de características sobre texto, al tener un gran número de columnas, en nuestro caso 34.344, muchas de ellas solo aporten ruido y confusión al problema. Para ello se han seleccionado las 20.000 mejores. El criterio de este número es el siguiente:

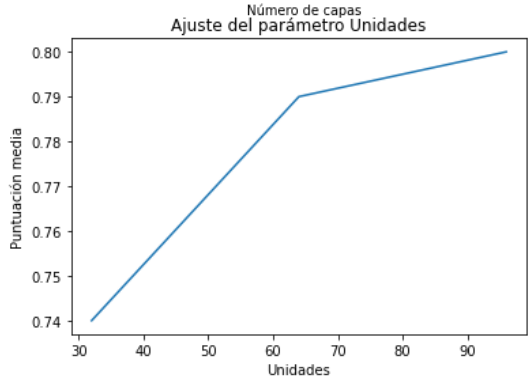
- Recomendación de la guía de implementación de este modelo.
- Pruebas realizadas con otros valores de $K$.

MLP

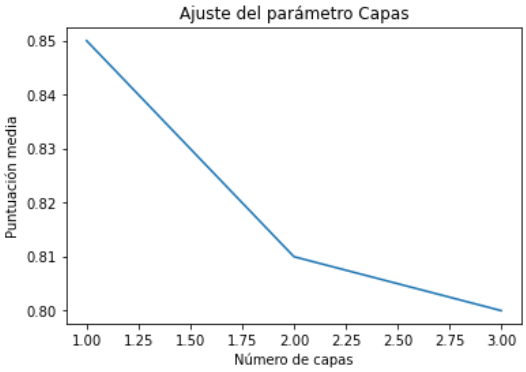
Los modelos MLP por defecto no admiten entrenamientos con matrices dispersas. Para resolver esta problemática se usa otra propiedad que, sí que tiene MLP, el entrenamiento parcial o incremental. Es tipo de entrenamiento permite pasar en lotes nuestro conjunto de datos por el modelo. El parámetro que define el tamaño del lote se llama $batch\_size$ y en nuestro caso toma el valor 1024.




| Valor | Puntuación |
|---|---|
| 0,0 | 0,85 |
| 0,1 | 0,85 |
| 0,2 | 0,85 |
| 0,3 | 0,86 |




| Valor | Puntuación |
|---|---|
| 32 | 0,79 |
| 64 | 0,84 |
| 96 | 0,86 |




| Valor | Puntuación |
|---|---|
| 1 | 0,85 |
| 2 | 0,81 |
| 3 | 0,73 |

### 7.1.1 Parámetros

| Componente | Parámetro | Valor |
|---|---|---|
| **Puntuación** | | |
| **StopWords** | language | spanish |
| **TfidVectorizer** | max_features | 100.000 |
| | ngram_range | (1,4) |
| | analyzer | word |
| | encoding | utf8 |
| | dtype | float32 |
| | min_df | 0,001 |
| | max_df | 0,9999 |
| | strip_accents | None |
| | decode_error | replace |
| | lowercase | False |
| **ANOVA** | k | 30.000 |
| **MLP** | input_shape | 30.000 |
| | num_classes | 55 |
| | dropout_rate | 0,1 |
| | regularization | $10^{-4}$ |
| | units | 1.536 |
| | layers | 1 |
| | total_params | 46.166.071 |
| | trainable_params | 46.166.071 |
| | non − trainable_params | 0 |
| **Fit** | batch_size | 64 |
| | epochs | 20 |
| | validation_split | 0,1 |
| | EarlyStopping(patience) | 3 |

### 7.1.2 Resultados

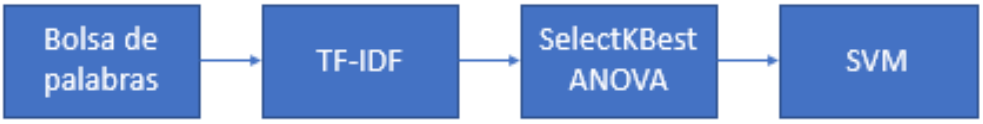
| | train | test |
|---|---|---|
| accuracy | 0,98 | 0,86 |

Reporte de clasificación para el conjunto test:

| | precisión | recall | F1 – score | support |
|---|---|---|---|---|
| accuracy | | | 0,86 | 3.823 |
| macro avg | 0,86 | 0.84 | 0.85 | 3.823 |
| weighted avg | 0,87 | 0.86 | 0.86 | 3.823 |

Tiempo total de entrenamiento 233.347s.

## 7.2 Modelo TF-IDF ANOVA LinearSVC




### 7.2.1 Parámetros

| Componente | Parámetro | Valor |
|---|---|---|
| **Puntuación** | | |
| **StopWords** | language | spanish |
| **TfidVectorizer** | max_features | 100.000 |
| | ngram_range | (1,4) |
| | analyzer | word |
| | encoding | utf8 |
| | dtype | float32 |
| | min_df | 0,001 |
| | max_df | 0,9999 |
| | strip_accents | None |
| | decode_error | replace |
| | lowercase | False |
| **ANOVA** | k | 30.000 |
| **LinearSVC** | C | 1 |
| | penalty | l2 |
| | dual | True |
| | loss | squared_hinge |
| | intercept_scaling | 1 |
| | max _iter | 10.000 |

### 7.2.2 Resultados

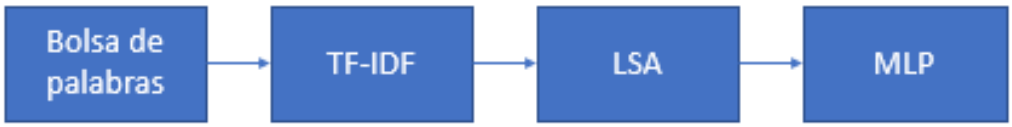
| | train | test |
|---|---|---|
| accuracy | 1,00 | 0,91 |

Reporte de clasificación para el conjunto test:

| | precisión | recall | F1 – score | support |
|---|---|---|---|---|
| accuracy | | | 0,91 | 3.823 |
| macro avg | 0,91 | 0.89 | 0.90 | 3.823 |
| weighted avg | 0,91 | 0.91 | 0.91 | 3.823 |

Tiempo total de entrenamiento 233.347s.

## 7.3 Modelo TF-IDF LSA LinearSVC




### 7.3.1 Parámetros

| Componente | Parámetro | Valor |
|---|---|---|
| **Puntuación** | | |
| **StopWords** | language | spanish |
| **TfidVectorizer** | max_features | 100.000 |
| | ngram_range | (1,4) |
| | analyzer | word |
| | encoding | utf8 |
| | dtype | float32 |
| | min_df | 0,001 |
| | max_df | 0,9999 |
| | strip_accents | None |
| | decode_error | replace |
| | lowercase | False |
| **ANOVA** | k | 30.000 |
| **LSA** | n_components | 1.000 |
| | n_iter | 15 |
| **LinearSVC** | C | 1 |
| | penalty | l2 |
| | dual | True |

| | | |
|---|---|---|
| | loss | squared_hinge |
| | intercept_scaling | 1 |
| | max _iter | 10.000 |

### 7.3.2 Resultados

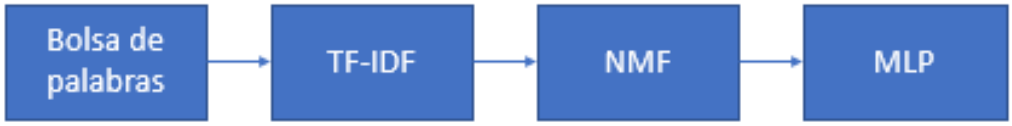
| | train | test |
|---|---|---|
| accuracy | 0,98 | 0,87 |

Reporte de clasificación para el conjunto test:

| | precisión | recall | F1 – score | support |
|---|---|---|---|---|
| accuracy | | | 0,87 | 3.823 |
| macro avg | 0,87 | 0.85 | 0.86 | 3.823 |
| weighted avg | 0,87 | 0.87 | 0.87 | 3.823 |

Tiempo total de entrenamiento 233.347s.

## 7.4 Modelo TF-IDF LSA MLP




### 7.4.1 Parámetros

| Componente | Parámetro | Valor |
|---|---|---|
| **Puntuación** | | |
| **StopWords** | language | spanish |
| **TfidVectorizer** | max_features | 100.000 |
| | ngram_range | (1,4) |
| | analyzer | word |
| | encoding | utf8 |
| | dtype | float32 |
| | min_df | 0,001 |
| | max_df | 0,9999 |
| | strip_accents | None |
| | decode_error | replace |
| | lowercase | False |
| **ANOVA** | k | 30.000 |
| **LSA** | n_components | 1.000 |
| | n_iter | 15 |

| | | |
|---|---|---|
| **MLP** | input_shape | 1.000 |
| | num_classes | 55 |
| | dropout_rate | 0,2 |
| | regularization | $3 * 10^{-5}$ |
| | units | 1.000 |
| | layers | 1 |
| | total_params | 1.056.055 |
| | trainable_params | 1.056.055 |
| | non − trainable_params | 0 |
| **Fit** | batch_size | 64 |
| | epochs | 20 |
| | validation_split | 0,1 |
| | EarlyStopping(patience) | 3 |

### 7.4.2 Resultados

| | train | test |
|---|---|---|
| accuracy | 0,98 | 0,86 |

Reporte de clasificación para el conjunto test:

| | precisión | recall | F1 − score | support |
|---|---|---|---|---|
| accuracy | | | 0,86 | 3.823 |
| macro avg | 0,86 | 0.84 | 0.85 | 3.823 |
| weighted avg | 0,87 | 0.86 | 0.86 | 3.823 |

Tiempo total de entrenamiento 233.347s.

## 7.5 Modelo Embeddings SC1DClassifier

### 7.5.1 Parámetros

| Componente | Parámetro | Valor |
|---|---|---|
| **Puntuación** | | |
| **Sequences** | num_words | 30. 000 |
| | maxlen | 512 |
| | padding | post |
| | truncating | post |
| **SC1DClassifier** | input_shape | 512 |
| | num_classes | 55 |
| | num_features | 30.000 |
| | embedding_dim | 512 |
| | embedding_trainable | False |

| | | |
|---|---|---|
| | filters | 512 |
| | kernel_size | 3 |
| | dropout_out | 0,3 |
| | regularization | $10^{-4}$ |
| | layers | 1 |
| **Fit** | batch_size | 128 |
| | epochs | 125 |
| | validation_split | 0,1 |
| | EarlyStopping(patience) | 3 |

### 7.5.2 Resultados

| | train | test |
|---|---|---|
| accuracy | 0,97 | 0,81 |

Reporte de clasificación para el conjunto test:

| | precisión | recall | F1 – score | support |
|---|---|---|---|---|
| accuracy | | | 0,81 | 3.823 |
| macro avg | 0,80 | 0.79 | 0.79 | 3.823 |
| weighted avg | 0,82 | 0.81 | 0.81 | 3.823 |

Tiempo total de entrenamiento 233.347s.

## 7.6 Modelo Embeddings C1DSingleClassifier

### 7.6.1 Parámetros

| Componente | Parámetro | Valor |
|---|---|---|
| **Puntuación** | | |
| **Sequences** | num_words | 30. 000 |
| | maxlen | 512 |
| | padding | post |
| | truncating | post |
| **C1DSingleClassifier** | input_shape | 512 |
| | num_classes | 55 |
| | num_features | 30.000 |
| | embedding_dim | 512 |
| | embedding_trainable | False |
| | filters | 512 |
| | kernel_size | 3 |
| | dropout_out | 0,2 |
| | regularization | $10^{-4}$ |

| | | |
|---|---|---|
| | layers | 1 |
| **Fit** | batch_size | 128 |
| | epochs | 125 |
| | validation_split | 0,1 |
| | EarlyStopping(patience) | 3 |

### 7.6.2 Resultados

| | train | test |
|---|---|---|
| **accuracy** | 0,98 | 0,82 |

Reporte de clasificación para el conjunto test:

| | precisión | recall | F1 – score | support |
|---|---|---|---|---|
| **accuracy** | | | 0,82 | 3.823 |
| **macro avg** | 0,81 | 0.80 | 0.80 | 3.823 |
| **weighted avg** | 0,82 | 0.82 | 0.82 | 3.823 |

Tiempo total de entrenamiento 620.213s.

## 7.7 Modelo Embeddings C1DMultiClassifier

### 7.7.1 Parámetros

| Componente | Parámetro | Valor |
|---|---|---|
| **Puntuación** | | |
| **Sequences** | num_words | 30.000 |
| | maxlen | 512 |
| | padding | post |
| | truncating | post |
| **C1DSingleClassifier** | input_shape | 512 |
| | num_classes | 55 |
| | num_features | 30.000 |
| | embedding_dim | 512 |
| | embedding_trainable | False |
| | fkl | (64,1,1)<br>(512,3,1)<br>(128,5,1) |
| | dropout_out | 0,2 |
| | regularization | $10^{-5}$ |
| **Fit** | batch_size | 128 |
| | epochs | 125 |
| | validation_split | 0,1 |
| | EarlyStopping(patience) | 3 |

### 7.7.2 Resultados

| | train | test |
|---|---|---|
| accuracy | 0,98 | 0,82 |

Reporte de clasificación para el conjunto test:

| | precisión | recall | F1 – score | support |
|---|---|---|---|---|
| accuracy | | | 0,82 | 3.823 |
| macro avg | 0,81 | 0.80 | 0.80 | 3.823 |
| weighted avg | 0,82 | 0.82 | 0.82 | 3.823 |

Tiempo total de entrenamiento 620.213s.

# 8 Comparativa

Balanceados y no balanceado

Ajuste de hiperparamentros

Graficas sobre hiperparametros

Tabla datos finales

# 9 Tecnología

Los lenguajes de programación más usados en 2019 dentro del ámbito del Aprendizaje Automático son Python, C++, JavaScript y Java.

(Elliott, 2019)

En este grupo se puede diferenciar 2 categorías: los que diseñan algoritmos de ML como C++ y Java, y los que implementan arquitecturas como Python y JavaScript. La realidad actual es que la mayor parte de científicos de datos usan entornos de desarrollo productivos, que permitan crear arquitecturas rápidas en conjunto con otras herramientas y por ende el segundo grupo mencionado. El lenguaje de programación que se ha usado para la implementación del proyecto ha sido **Python** en su versión 3.7

ya que no se requería el desarrollo de ningún modelo desde 0. En este caso la elección ha sido sencilla y se ha basado en criterios:

- Conocimientos previos: un fuerte bagaje de implementación de otros modelos y herramientas.
- Recursos disponibles: a día de hoy Python está consolidado como el lenguaje con mayor oferta de bibliotecas ML respaldadas por una gran comunidad de desarrolladores.

## 9.1 Python

Python es un lenguaje de programación interpretado programado en C y C++. La filosofía del mismo gira entorno a la legibilidad del código. Se trata de un lenguaje multiparadigma que soporta orientación a objetos. Según las estadísticas de GitHub en 2020 Python ocupa el tercer lugar en el ranking de lenguaje más usados por los desarrolladores. A continuación, mostraremos algunas de las bibliotecas más usadas durante la implementación de los modelos de este proyecto.

### 9.1.1 Pandas

Es conjunto de con Spark, Pandas es la biblioteca de Ciencia de Datos más usada en Python. Presenta dos estructuras de memoria principales:

- **Series**: Desde el punto de vista del Aprendizaje Automático suele hacer referencia a una característica. Desde el punto de vista matemático representan un vector indexado.
- **DataFrame**: Desde el punto de vista del Aprendizaje Automático suele hacer referencia a la matriz de características. Permite multiindexación, columnas con nombre y un rico conjunto de formas de acceder a los datos.

La principal característica de Pandas es que permite un nivel de abstracción a la programación que le ha abierto un sitio de estudio en gran parte de las ramas de ingenierías y ciencias puras.

### 9.1.2 Sklearn

Sklearn es una biblioteca de Python que provee de una colección de algoritmos entorno al aprendizaje supervisado y no supervisado. El código desarrollado en este trabajo ha seguido la estructuración que presenta esta biblioteca.

Destaca por su sencillez y versatilidad a la hora de implementar y desplegar modelos. Por otro lado, ofrece paralelización multinúcleo para muchos de sus algoritmos.

### 9.1.3 Tensorflow

Tensorflow es una biblioteca de código abierto desarrollada y mantenida por la empresa Google orientada a la implementación de modelos basados en redes neuronales. Su característica principal es la capacidad de estructurar los modelos en forma de **grafos dirigidos**. Un punto fuerte sin duda es su compatibilidad directa con la biblioteca Cuda que permite la aceleración por GPU.

Los modelos basados redes neuronales de este proyecto se han implementado sobre una capa abstracción superior llamada **Keras**. Keras empezó siendo un proyecto independiente a Tensorflow con el objetivo de agilizar los desarrollos.

### 9.1.4 Unittest

Unittest es una de las principales herramientas para realizar testeos sobre Python. Posee una integración total con el editor de código VSCode.

La implementación de los pipelines para cada modelo se encuentra en un módulo testeo aparte. Por lo tanto, cada modelo es se ha implementado en un test.

### 9.1.5 Matplotlib

Matplotlib es la biblioteca usada en el proyecto para presentar la mayoría de los gráficos que se muestran en el mismo.

## 9.2 Cuda

Cuda es una biblioteca desarrollada y mantenida por la empresa Nvidia que permite la comunicación de propósito general con las GPU de marca Nvidia. Sobre Cuda está desarrollada la biblioteca **cudnn** con el objetivo desarrollar redes neuronales optimizadas para GPU.

Para el desarrollo de este proyecto se ha realizado una instalación de estas dos bibliotecas mencionas sobre un sistema operativo Windows.

## 9.3 Git

Git es un sistema de control de versiones y herramienta prácticamente imprescindible en desarrollo de proyectos.

Para este proyecto se ha creado un repositorio en Github que reúne tanto el desarrollo del proyecto como la documentación. Puesto que es un proyecto relativamente pequeño solo se ha utilizado la rama por defecto.

https://github.com/rojo1997/Authorship

# 10 Conclusión

# 11 Apéndice

Distribución de los documentos sobre las etiquitas después aplicar los filtros al conjunto de datos:

| Etiqueta | Número de documentos |
|---|---|
| COVES BOTELLA | 1583 |
| GARCÍA RODRÍGUEZ | 779 |
| SÁNCHEZ GORDILLO | 625 |
| VAQUERO DEL POZO | 601 |
| VALDERAS SOSA | 534 |
| CASTILLO JIMÉNEZ | 476 |
| MONTERO CUADRADO | 435 |
| NAVARRO GARZÓN | 420 |
| CASTRO ROMÁN | 420 |
| ARENAL CATENA | 411 |
| MARTÍNEZ AGUAYO | 397 |
| RODRÍGUEZ DOMÍNGUEZ | 391 |
| MARTÍNEZ VIDAL | 385 |
| ÁVILA CANO | 363 |
| CEBRIÁN PASTOR | 361 |
| ÁLVAREZ DE LA CHICA | 357 |
| AGUILERA GARCÍA | 356 |
| FUENTES LOPERA | 355 |
| GRIÑÁN MARTÍNEZ | 353 |
| NIETO BALLESTEROS | 328 |
| ESPADAS CEJAS | 327 |
| SANZ CABELLO | 323 |
| NÚÑEZ ROLDÁN | 315 |
| SOLER MÁRQUEZ | 312 |
| GONZÁLEZ VIGO | 307 |
| RAYNAUD SOTO | 304 |
| MORENO RUIZ | 303 |
| DÍAZ TRILLO | 303 |
| RECIO MENÉNDEZ | 299 |
| FERNÁNDEZ GARCÍA | 291 |
| RAMOS AZNAR | 290 |
| MARISCAL CIFUENTES | 284 |
| JIMÉNEZ VÍLCHEZ | 281 |
| MESA CIRIZA | 278 |
| CÓZAR ANDRADES | 277 |
| ALONSO ALONSO | 268 |
| OÑA SEVILLA | 264 |
| BOTELLA SERRANO | 263 |

| | |
|---|---|
| **GARCÍA GIRALTE** | 261 |
| **CABALLOS MOJEDA** | 255 |
| **PLATA CÁNOVAS** | 248 |
| **CARRASCO GARCÍA** | 245 |
| **TORRES RUIZ** | 242 |
| **GALLEGO MORALES** | 237 |
| **CUENCA CABEZA** | 233 |
| **NAVARRO RODRÍGUEZ** | 231 |
| **MARTÍNEZ MARTÍN** | 230 |
| **ARENAS BOCANEGRA** | 229 |
| **GRACIA NAVARRO** | 222 |
| **PIZARRO MEDINA** | 221 |
| **MORO CÁRDENO** | 219 |
| **NARANJO MÁRQUEZ** | 209 |
| **ZOIDO ÁLVAREZ** | 208 |
| **VÁZQUEZ BERMÚDEZ** | 202 |
| **RUIZ-SILLERO BERNAL** | 202 |

Conjunto de StopWords propuesto para el lenguaje castellano:

de, la, que, el, en, y, a, los, del, se, las, por, un, para, con, no, una, su, al, lo, como, más, pero, sus, le, ya, o, este, sí, porque, esta, entre, cuando, muy, sin, sobre, también, me, hasta, hay, donde, quien, desde, todo, nos, durante, todos, uno, les, ni, contra, otros, ese, eso, ante, ellos, e, esto, mí, antes, algunos, qué, unos, yo, otro, otras, otra, él, tanto, esa, estos, mucho, quienes, nada, muchos, cual, poco, ella, estar, estas, algunas, algo, nosotros, mi, mis, tú, te, ti, tu, tus, ellas, nosotras, vosotros, vosotras, os, mío, mía, míos, mías, tuyo, tuya, tuyos, tuyas, suyo, suya, suyos, suyas, nuestro, nuestra, nuestros, nuestras, vuestro, vuestra, vuestros, vuestras, esos, esas, estoy, estás, está, estamos, estáis, están, esté, estés, estemos, estéis, estén, estaré, estarás, estará, estaremos, estaréis, estarán, estaría, estarías, estaríamos, estaríais, estarían, estaba, estabas, estábamos, estabais, estaban, estuve, estuviste, estuvo, estuvimos, estuvisteis, estuvieron, estuviera, estuvieras, estuviéramos, estuvierais, estuvieran, estuviese, estuvieses, estuviésemos, estuvieseis, estuviesen, estando, estado, estada, estados, estadas, estad, he, has, ha, hemos, habéis, han, haya, hayas, hayamos, hayáis, hayan, habré, habrás, habrá, habremos, habréis, habrán, habría, habrías, habríamos, habríais, habrían, había, habías, habíamos, habíais, habían, hube, hubiste, hubo, hubimos, hubisteis, hubieron, hubiera, hubieras, hubiéramos, hubierais, hubieran, hubiese, hubieses, hubiésemos, hubieseis, hubiesen, habiendo, habido, habida, habidos, habidas, soy, eres, es, somos, sois, son, sea, seas, seamos, seáis, sean, seré, serás, será, seremos, seréis, serán, sería, serías, seríamos, seríais, serían, era, eras, éramos, erais, eran, fui, fuiste, fue, fuimos, fuisteis, fueron, fuera, fueras, fuéramos, fuerais, fueran, fuese, fueses, fuésemos, fueseis, fuesen, sintiendo, sentido, sentida, sentidos, sentidas, siente, sentid, tengo, tienes, tiene, tenemos, tenéis, tienen, tenga, tengas, tengamos, tengáis, tengan, tendré, tendrás, tendrá, tendremos, tendréis, tendrán, tendría, tendrías, tendríamos, tendríais, tendrían, tenía, tenías, teníamos, teníais, tenían, tuve, tuviste, tuvo, tuvimos, tuvisteis, tuvieron, tuviera, tuvieras, tuviéramos, tuvierais, tuvieran, tuviese, tuvieses, tuviésemos, tuvieseis, tuviesen, teniendo, tenido, tenida, tenidos, tenidas, tened.

# 12 Bibliografía


A Navarro, C., Andrés Carrasco, R., J. Barrientos, R., & Vega, R. (2020). GPU Tensor Cores for fast Arithmetic Reductions.

Abbasi, A., & Chen, H.-c. (2005). Applying Authorship Analysis to Extremist-Group Web Forum Messages. *IEEE*, 67-75.

C. Müller, A., & Guido, S. (2016). *Introduction to Machine Learning with Python.* Sebastopol: O'REILLY.

Chaski, C. (2005). Who's At The Keyboard? Authorship Attribution in Digital Evidence Investigations. *International Journal of Digital Evidence*.

D. Lee, D., & Sebastian Seung, H. (2001). Algorithms for Non-negative Matrix Factorization. *CiteSeer*.

Edelman, A. (2016). *https://math.mit.edu/.* Obtenido de https://math.mit.edu/classes/18.095/2016IAP/lec2/SVD_Notes.pdf

Elliott, T. (24 de Enero de 2019). *Github*. Obtenido de The State of the Octoverse: machine learning: https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/

F. Barrero, D., Gonzalez-Pardo, A., Camacho, D., & Dolores R-Moreno, M. (2020). Distributed parameter tuning for genetic algorithms. *Computer Science and Information Systems*, 661-677.

Feiguina, O., & Hirst, G. (2007). Authorship attribution for small texts: Literary and forensic experiments. *Conference: Proceedings of the SIGIR 2007 International Workshop on Plagiarism Analysis*.

Guruswami, V., & Kannan, R. (17 de Enero de 2012). *https://www.cs.cmu.edu/.* Obtenido de https://www.cs.cmu.edu/~venkatg/teaching/CStheory-infoage/book-chapter-4.pdf

Jacquemin, C., & Tzoukermann, E. (2000). NLP for Term Variant Extraction: Synergy Between Morphology, Lexicon, and Syntax. *Natural Language Information Retrieval*, 25-74.

Joula, P., & Sofko, J. (2004). Proving and Improving Authorship Attribution Technologies.

LeCun, Y., Haffner, P., Bottou, L., & Bengio, Y. (1989). Object Recognition With Gradient-Based Learning.

Luyckx, K., & Daelemans, W. (2011). The effect of author set size and data size in authorship attribution. *Literary and Linguistic Computing*, 35-55.

Marton, Y., Wu, N., & Hellerstein, L. (2005). On Compression-Based Text Classification. *Lecture Notes in Computer Science*, 300-314.

Moitra, A. (s.f.). *https://ocw.mit.edu/*. Obtenido de https://ocw.mit.edu/courses/mathematics/18-409-algorithmic-aspects-of-machine-learning-spring-2015/lecture-notes/MIT18_409S15_chapp2.pdf

Mosteller, F., & Wallace, D. (1963). Inference in an Authorship Problem. *Journal of the American Statistical Association*, 275-309.

*Nice Actimize Sanctions Screening & Watch List Filtering*. (s.f.). Obtenido de https://www.niceactimize.com/anti-money-laundering/watch-list-filtering/

Petrov, S., Das, D., & McDonald, R. (2011). A Universal Part-of-Speech Tagset.

*Real Academia Española*. (13 de Mayo de 2020). Obtenido de https://dle.rae.es

*Real Academía Española*. (13 de Mayo de 2020). Obtenido de https://dle.rae.es/corpus

Russell, S., & Norvig, P. (2010). Artificial Neural Networks. En S. Russell, & P. Norvig, *Artificial Intelligence A Modern Approach* (págs. 727-736). Harlow: PEARSON.

Russell, S., & Norvig, P. (2010). Learning, Communicating, perceiving, and acting. En *Artificial Intelligence A Modern Approach* (págs. 693-1010). Harlow: Pearson.

Russell, S., & Norvig, P. (2010). N-gram character models. In S. Russell, & P. Norvig, *Artificial Intelligence A Modern Approach* (pp. 861-862). Harlow: PEARSON.

Russell, S., & Norvig, P. (2010). N-gram word models. En *Artificial Intelligence A Modern Approach* (págs. 864-865). Harlow: PEARSON.

Russell, S., & Norvig, P. (2010). Support Vector Machines. En S. Russell, & P. Norvig, *Artificial Intelligence A Modern Approach* (págs. 744-747). Harlow: PEARSON.

Singh, P., & Manure, A. (2020). *Learn TensorFlow 2.0.* Bangalore: APRESS.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research 15*, 1929-1958.

Stamatatos, E. (2009). A Survey of Modern Authorship Attribution Methods. *Journal of the American Society for Information Science and Technology*, 538-556.

Yampolskiy, R., & Govindaraju, V. (2009). Taxonomy of Behavioural Biometrics.