



*ugr*

Universidad  
de **Granada**

---

# 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

<b>1</b>	<b>INTRODUCCIÓN.....</b>	<b>3</b>
1.1	Propósito.....	4
1.2	Partes .....	4
1.3	Justificación .....	5
1.4	Ámbito .....	5
1.5	Relevancia teórica y práctica de la investigación.....	6
1.6	Estado del arte .....	7
1.7	Descripción de la metodología de la investigación .....	8
<b>2</b>	<b>PROFUNDIZAR SOBRE EL LENGUAJE .....</b>	<b>9</b>
<b>3</b>	<b>OBJETIVO .....</b>	<b>11</b>
<b>4</b>	<b>DATOS.....</b>	<b>11</b>
4.1	Fuente.....	11
4.2	Limpieza .....	12
<b>5</b>	<b>FUNDAMENTOS TEÓRICOS .....</b>	<b>17</b>
5.1	Preprocesamiento .....	17
5.1.1	Puntuación.....	17
5.1.2	StopWords .....	18
5.1.3	Traducción .....	18
5.2	Extracción de características.....	18
5.2.1	Bolsa de Palabras .....	18
5.2.2	Modelos <i>N-gram</i> .....	20
5.2.3	Vectorización de documentos .....	20
5.2.4	TF-IDF .....	21
5.3	Selección de características .....	22
5.3.1	ANOVA.....	23
5.3.2	Kendall's .....	23
5.4	Reducción de la dimensión.....	24
5.4.1	LSA .....	24
5.4.2	NMF.....	25
5.5	Aprendizaje .....	25
5.5.1	Red Neuronal.....	25
5.5.2	Máquina de Vectores Soporte.....	28
<b>6</b>	<b>METODOLOGÍA DE EXPERIMENTACIÓN .....</b>	<b>30</b>

6.1	Exactitud o <i>accuracy</i> .....	31
6.2	Precisión .....	31
6.3	Exhaustividad o <i>recall</i> .....	31
6.4	Ajuste de hiperparámetros.....	31
6.5	Validación cruzada.....	31
7	EXPERIMENTOS .....	31
7.1	Modelo TF-IDF ANOVA MLP.....	31
7.1.1	Parámetros.....	35
7.1.2	Resultados .....	36
7.2	Modelo TF-IDF ANOVA SVM Lineal.....	36
7.2.1	Parámetros.....	36
7.2.2	Resultados .....	36
7.3	Modelo TF-IDF LSA MLP .....	37
7.3.1	Parámetros.....	37
7.3.2	Resultados .....	37
7.4	Modelo TF-IDF NMF MLP.....	37
7.4.1	Parámetros.....	37
7.4.2	Resultados .....	37
7.5	Modelo Embeddings LSTM .....	37
7.5.1	Parámetros.....	37
7.5.2	Resultados .....	37
7.6	Modelo Embeddings sepCNN .....	37
7.6.1	Parámetros.....	37
7.6.2	Resultados .....	37
8	COMPARATIVA.....	37
9	TECNOLOGÍA.....	38
9.1	Python .....	38
9.1.1	Pandas.....	38
9.1.2	Sklearn.....	39
10	CONCLUSIÓN.....	39
11	APÉNDICE .....	39
12	BIBLIOGRAFÍA .....	41

# 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 (NLP)*.
- 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 dado 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 automatizar 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 un Estudio de Métodos Modernos de Atribución de Autoría (*A Survey of Modern Authorship Attribution Methods*, Universidad de Aegean) y el efecto del tamaño del conjunto de autores y el tamaño de los datos en la atribución de autoría (*The effect of author set size and data size in Authorship Attribution*). Ambas investigaciones fueron sugeridas por los tutores del proyecto. La lectura del capítulo V y VI del libro Inteligencia Artificial un Enfoque Moderno (*Artificial Intelligence A Modern Approach* por Stuart Russell y Peter Norvig) 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 (Watch List Filtering) 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 *Machine Learning (ML)* inundan el mercado actual, dando 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

[Resumen del artículo de la encuesta y algo más]

Relevancia teórica

{Explicar los posibles avances dentro del campo}

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

- 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 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 del autor y a las 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 TFM).
  - Identificación de trastornos.
  - Psicología evolutiva y del aprendizaje.
- Desde la automatización de tareas, como es nuestro caso, tenemos algunos ejemplos.

- Documentación automática (nuestro caso).
- 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', un conjunto de 146 documentos de longitud variada escritos por tres autores diferentes. 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.

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 (*natural language processing*) 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.

(eficiencia CUDA)



## 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 de [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: de una característica única en forma de texto el conjunto de características de frecuencias de las palabras. 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 válido 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 Academia 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 comprensión 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 presentes 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 consiguiente, 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 en 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 de 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 puede 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 las 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. Algunas 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
...

```

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

$$\forall i \in (0, N) \text{ nombre.replace(cad}_i\text{,)}$$

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:

$$[^1 - 9a - zA - ZÁÉÍÓÚáéíóúñÑ.,;:\?¿\!;\(\)] \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:

" ¿ Esto es una prueba de [pausa] limpieza (3.0) ¿ !!"

Después de la limpieza:

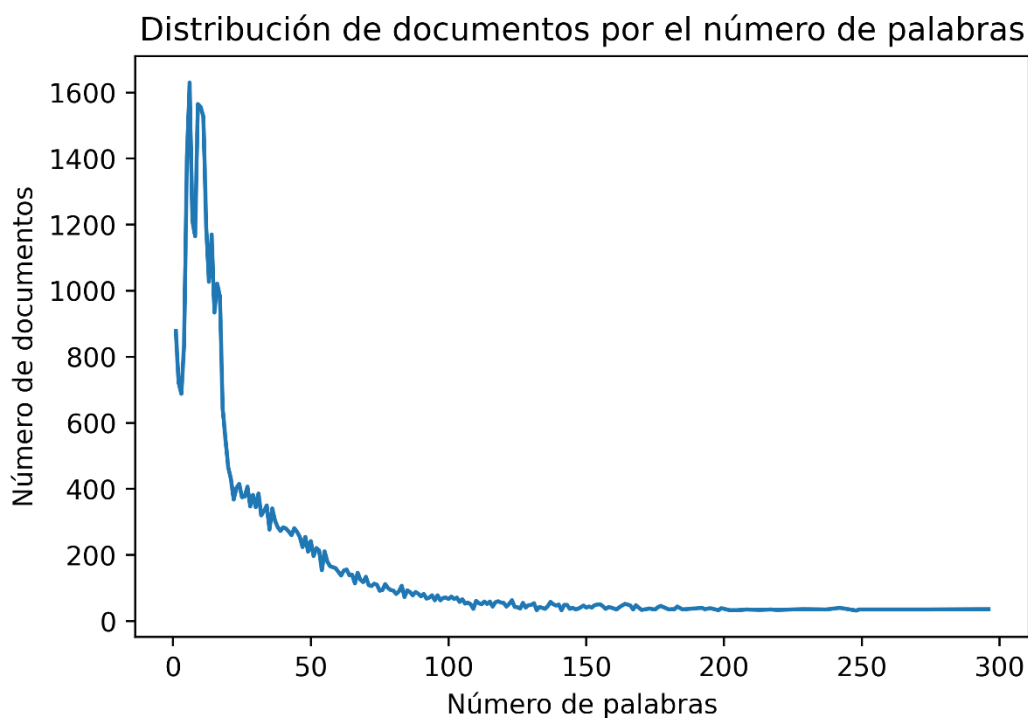
"¿ Esto es una pueba de limpieza (3.0) ¿ !!"

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.

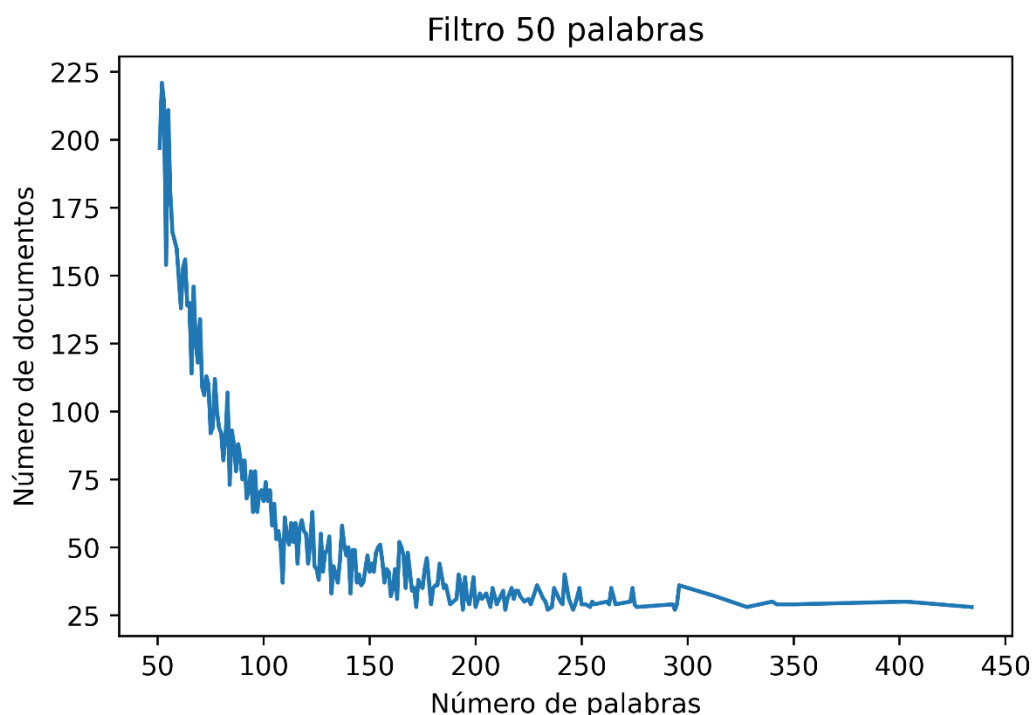
Como podemos ver existe una gran concentración de documentos de corta extensión entre 1 y 50 palabras. Esto tiene una explicación dentro del ámbito que se desarrolla el conjunto de datos. Al abrir un debate es usual encontrarse un par de intervenciones contundentes en tamaño seguidas de réplicas cortas. 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**.

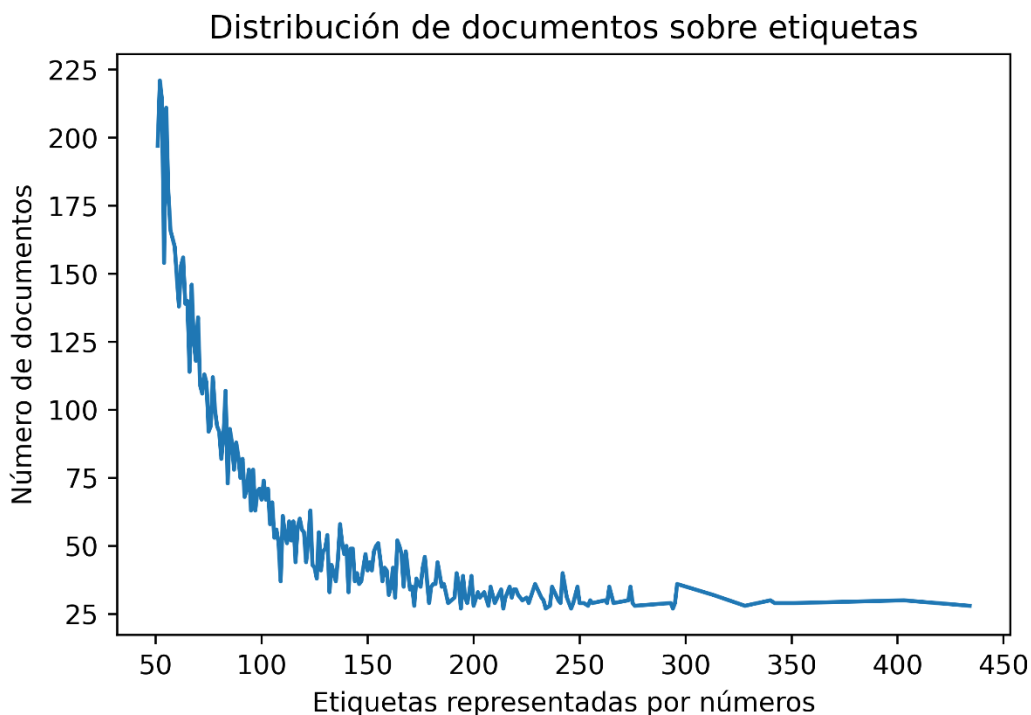
A continuación, vamos a realizar un conteo del número de documentos que tiene asociada cada etiqueta. Con ello perseguimos eliminar casos absurdos de aprendizaje sobre etiquetas que no presentan apenas muestras. Empecemos con algunos datos estadísticos para el número de documentos por etiqueta.

	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

## 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 abundan muchos nombres propios o de organizaciones puede llevar a un encarecimiento de la información extraña. El apellido “Pino” y el árbol “pino” pasarían a ser el mismo token dentro de la bolsa.
- Lematizador: mediante un algoritmo se extrae la palabra de la que deriva la original. Algoritmos como *WordNetLemmatizer* 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.
- Derivación: es una operación con el mismo objetivo que el lematizador. 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*

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*:

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

Los modelos *N-gram* son generalmente aplicados de tres 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 **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.

Normalmente, dentro de estos modelos basados en bolsas de palabras si se da el caso de un *N-gram*, los *(N-i)-gram* que lo contienen también forman parte del vocabulario, dando la posibilidad de detectar tokens concretos de forma separada, pero sabiendo que no se presentan juntos.

(Ampliar con ejemplo y aplicado a char y word)

(Russell & Norvig, N-gram character 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:

$$d_i = \text{"que es lo que quieres"}$$

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  $M_{n \times m}$  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.

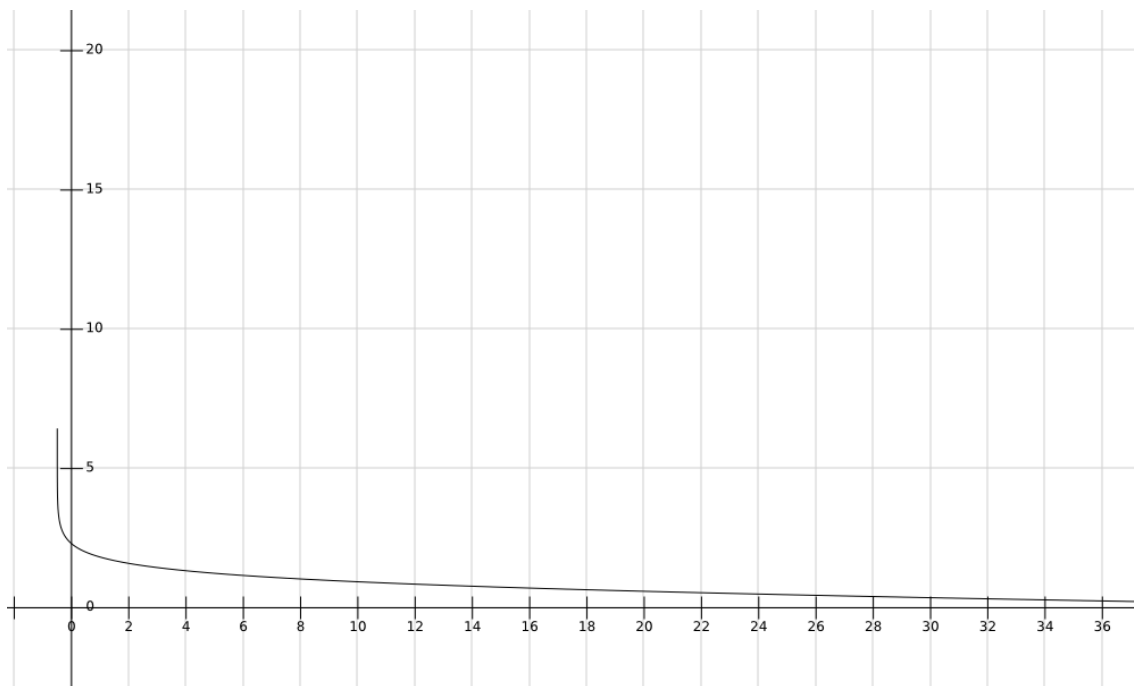
TF hace alusión a la frecuencia de un término dentro del corpus mientras que IDF refleja la inversa de la frecuencia sobre el documento. Se trata de una función de puntuación llamada **BM25**. Fue propuesta por Stephen Robertson y Karen Sparck Jones.

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.

### 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  $p - value$  válido.

- 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:

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

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 libertad 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.

(Chi Cuadrado)

### 5.3.2 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:

$$(x_i, y_i) \text{ concuerda con } (x_j, y_j) \text{ si } (x_i < x_j \ \& \ y_i < y_j) \mid (x_i > x_j \ \& \ y_i > y_j)$$

Donde  $n_d$  es el número de pares de puntos sobre las dos muestras que son 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:

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

Supongamos que  $M_{n \times m}$  tiene  $n$  filas y  $m$  columnas, entonces las matrices tendrán las siguientes dimensiones  $U_{n \times n}, \Sigma_{n \times m}, V_{m \times m}^*$  donde la diagonal de  $\Sigma$  refleja los valores singulares de  $M$  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 compoentes a calcular sustituyendo en la fórmula anterior  $m$  por un  $k$ :

$$M_k = \sum_{i=1}^k \sigma_i u_i v_i^*$$

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

[https://math.mit.edu/classes/18.095/2016IAP/lec2/SVD\\_Notes.pdf](https://math.mit.edu/classes/18.095/2016IAP/lec2/SVD_Notes.pdf)

<https://www.cs.cmu.edu/~venkatg/teaching/CStheory-infoage/book-chapter-4.pdf>

## 5.4.2 NMF

## 5.5 Aprendizaje

### 5.5.1 Red Neuronal

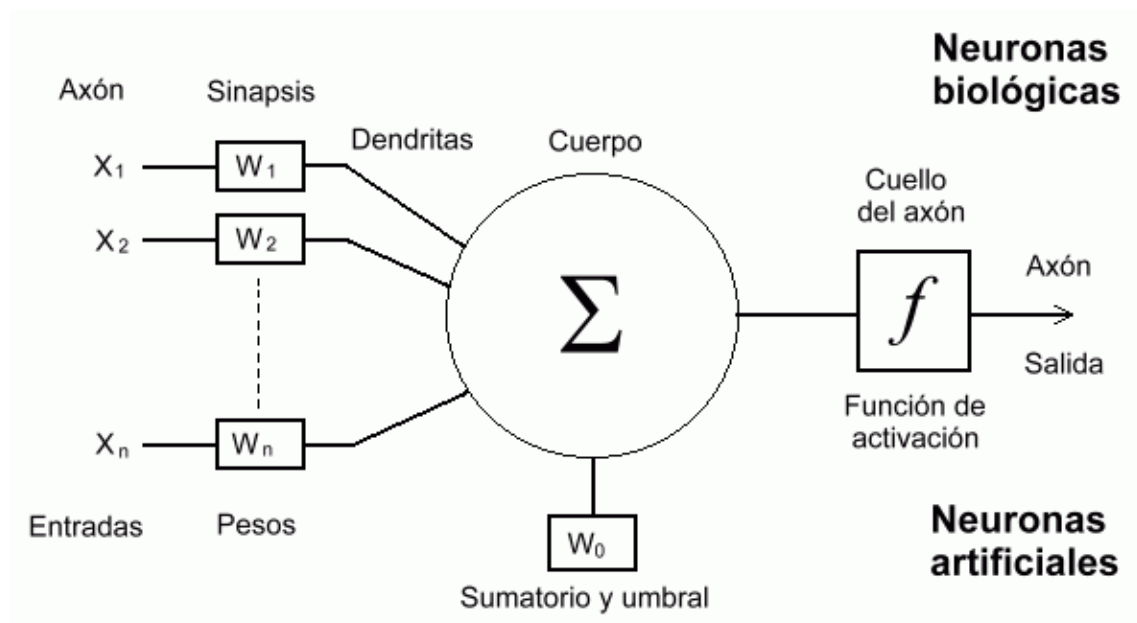
Uno de los modelos más conocidos dentro de Aprendizaje Automático son las redes neuronales artificiales. Un modelo inspirado en el comportamiento del cerebro humano que pretenden simular el intercambio de información entre las **neuronas**.

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  $w_{ij}$ .

Los modelos Multicapa Perceptrón son un subconjunto de los modelos basados en redes neuronales. Fueron los primeros en aparecer debido a la simplicidad de su arquitectura.

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.

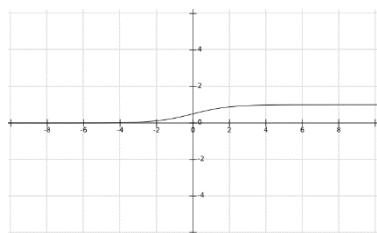
La expresión matemática que resume este proceso es la siguiente:

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

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.

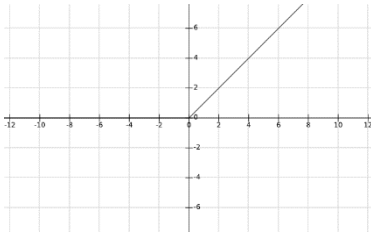
#### Función Sigmoide



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

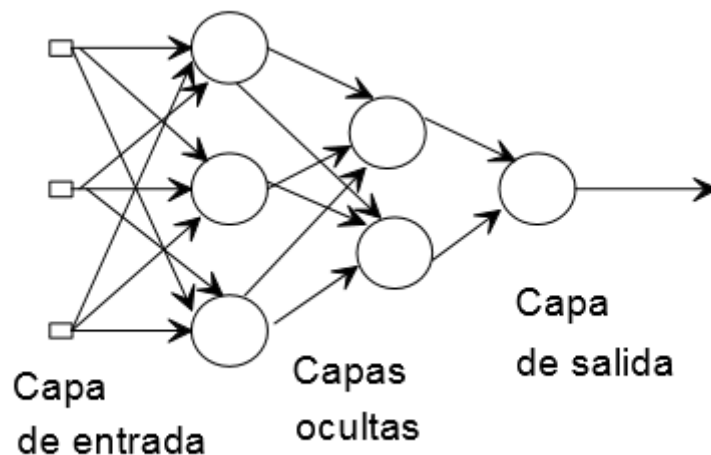
Función Softmax es comúnmente utilizada para clasificación de problemas multietiqueta dentro de entrenamiento de redes neuronales. Solemos encontrarla en la última capa de la red. La función devuelve un vector de  $K$  dimensiones, donde  $K$  es el número de clases de nuestro problema y cada competente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

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

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

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 forman estructuras visuales que comúnmente se representan mediante esquemas de grafos dirigidos:



- 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 overfit. Llamamos sobreajuste al efecto de sobreentrenar un modelo de tal forma que abandone su función de aprendizaje por la de memorizar. La técnica sin duda más usada es **dropout**. Un método de regularización que elimina unidades tanto de capas ocultas como visibles. Al eliminarlas, un porcentaje de ellas durante un proceso de entrenamiento, reducimos notablemente la complejidad.

Para realizar la búsqueda del mejor modelo de MLP se han simplificado los parámetros a cinco que se explican a continuación:

- Número de capas ocultas: se corresponde con el número de capas internas con las que se genera el modelo.
- Número de unidades: se corresponde con el número de neuronas que tendrá cada capa interna.
- Tasa dropout: es la tasa de enlaces entre dos capas que el algoritmo debe quitar.
- Dimensión input: dimensión de la primera capa.
- Dimensión output: dimensión de la última capa que debe concordar con el número de clases del problema.

#### **5.5.1.1 Densa**

#### **5.5.1.2 Dropout**

#### **5.5.1.3 Pooling**

#### **5.5.1.4 Embedding**

#### **5.5.1.5 Convolucionales 1D**

#### **5.5.1.6 LSTM**

#### **5.5.1.7 GRU**

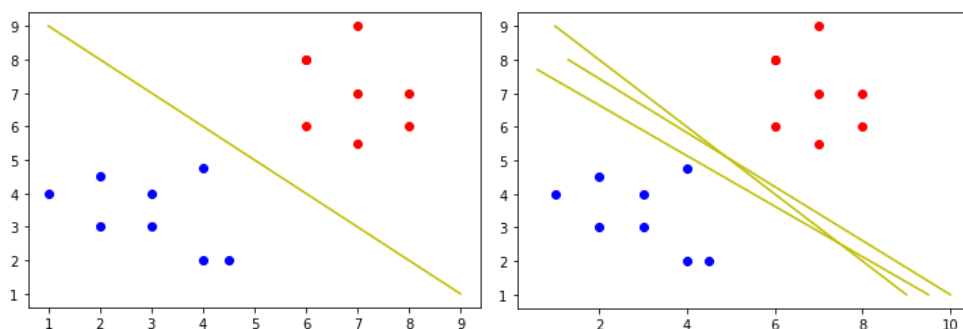
### **5.5.2 Máquina de Vectores Soporte**

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 denotará 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

Métricas

Validación cruzada

6.1 Exactitud o *accuracy*

6.2 Precisión

6.3 Exhaustividad o *recall*

6.4 Ajuste de hiperparámetros

Búsqueda en rejilla

Búsqueda aleatoria

Búsqueda evolutiva

6.5 Validación cruzada

## 7 Experimentos

### 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.

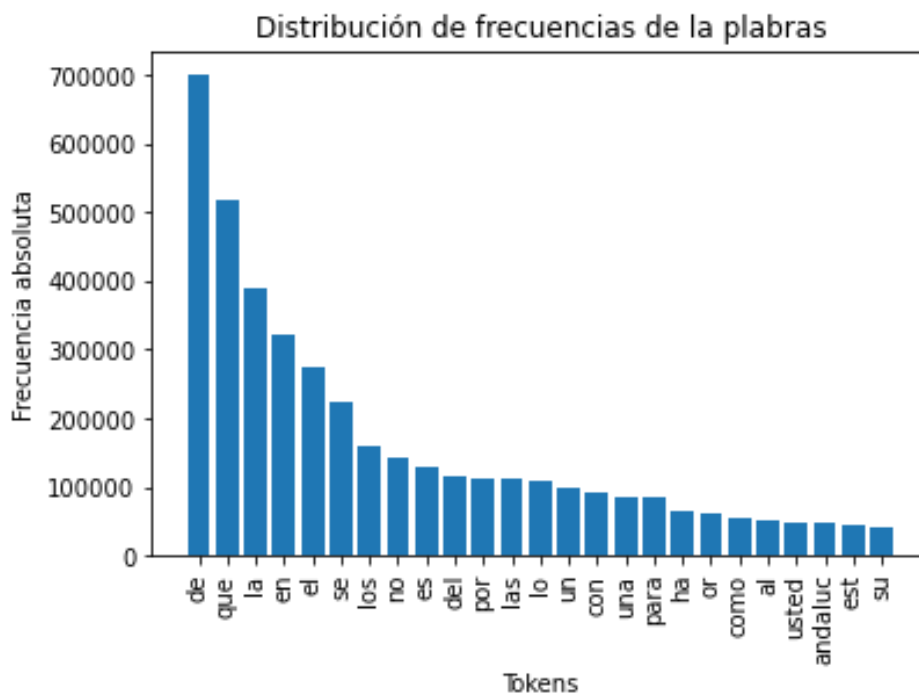
Bolsa de palabras



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*:

<i>N-gram</i>	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.

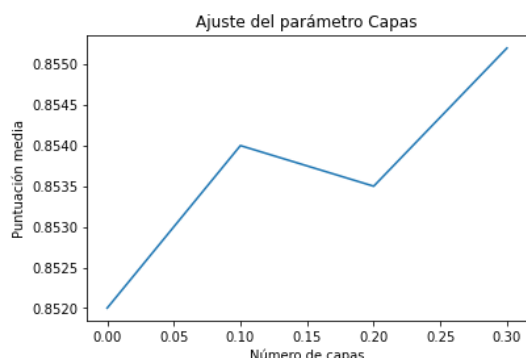
Para la aplicación de la vectorización, uno de los valores de referencia para la comparación de procesos de extracción de características sobre texto es la dispersión de los datos sobre la matriz vectorizada del corpus. En nuestro caso ese valor es el siguiente:

$$dispersión = 0,64\%$$

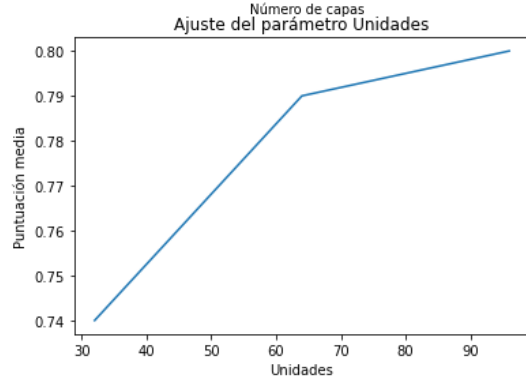
El valor se puede calcular fácilmente dividiendo el número de ternas de la matriz dispersa entre la multiplicación de las dimensiones de la matriz. La interpretación de estos datos es que el 0.64% de los valores de la matriz es distinto de 0. Esto no debe sorprendernos y debe ser interpretado positivamente. 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.

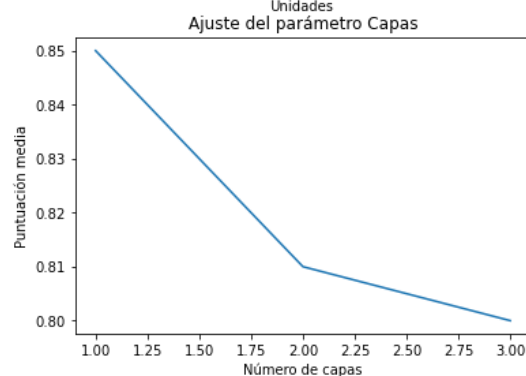




Valor	Puntuación
0,0	0,85
0,1	0,85
0,2	0,85
0,3	0,86



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
Bolsa de palabras	Frecuencia relativa mínima	0,001
	Frecuencia relativa máxima	0,999
	Máximo de características	100.000
	Stopwords	Spain+Puntuación
	N-Gram	(1,2)
BestKBest ANOVA	K	20.000
MLP	Número de capas ocultas	1
	Número de unidades por capa	96
	Tasa dropout	0,3
	Dimensión input	20.000
	Dimensión output	60

### 7.1.2 Resultados

División	1	2	3	4	5	Media	Desviación
Puntuación	0,85	0,85	0,86	0,86	0,86	0,86	0,0025

	Entrenamiento	Validación
Puntuación	1,00	0,87

Tiempo total de entrenamiento 1,3 minutos.

## 7.2 Modelo TF-IDF ANOVA SVM Lineal



### 7.2.1 Parámetros

Componente	Parámetro	Valor
Bolsa de palabras	Frecuencia relativa mínima	0,001
	Frecuencia relativa máxima	0,999
	Máximo de características	100.000
	Stopwords	Spain+Puntuación
	N-Gram	(1,2)
BestKBest ANOVA	K	20.000
SVM	C	1

### 7.2.2 Resultados

División	1	2	3	4	5	Media	Desviación
Puntuación	0,85	0,85	0,86	0,86	0,86	0,86	0,0025

	Entrenamiento	Validación
Puntuación	1,00	0,89

### 7.3 Modelo TF-IDF LSA MLP



#### 7.3.1 Parámetros

#### 7.3.2 Resultados

### 7.4 Modelo TF-IDF NMF MLP



#### 7.4.1 Parámetros

#### 7.4.2 Resultados

### 7.5 Modelo Embeddings LSTM

#### 7.5.1 Parámetros

#### 7.5.2 Resultados

### 7.6 Modelo Embeddings sepCNN

#### 7.6.1 Parámetros

#### 7.6.2 Resultados

## 8 Comparativa

Balanceados y no balanceado

Ajuste de hiperparametros

Graficas sobre hiperparametros

## 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 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

Tensorflow

Cuda

Pandas

Matplotlib

Git

## 10 Conclusión

## 11 Apéndice

Distribución de los documentos sobre las etiquetas 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

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

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/>

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

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