Prototipo de análisis del lenguaje utilizado en los trabajos de fin de grado de la EII

Pablo Morales Gómez

Supervisado por:

Modesto Fernando Castrillón Santana

José Javier Lorenzo Navarro

José Daniel Hernández Sosa

1 de julio de 2022

Agradecimientos

Primero a mi familia por tener paciencia conmigo. También a mis amigos y compañeros por acompañarme en esta travesía, especialmente a Christian, a Carlos, a Jorge y, por supuesto, a Pablo, a Pablo y a Pablo.

Resumen

El objeto de este proyecto es realizar un estudio del tipo de vocabulario que se emplea en la realización de un Trabajo de Fin de Título. Con este objetivo en mente, primero se ha creado un corpus con todas las memorias de los proyectos entregados en los últimos años en la Escuela de Ingeniería Informática de la ULPGC. Este conjunto de textos ha sido sometido a una serie de tratamientos de procesamiento de lenguaje natural (Natural Language Processing, NLP) para posteriormente poder ser utilizado por el algoritmo Word2Vec. Finalmente, esta técnica NLP ha sido usada para clasificar los títulos de los distintos proyectos en varios grupos con características comunes.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En el año 2011, el artista de datos canadiense Jer Thorp autor de numerosos proyectos relacionados con el tratamiento y representación de datos, llevó a cabo un proyecto conocido en conjunto como "NYT Word Frequencies". Este trabajo artístico muestra en una serie de representaciones creativas del uso y aparición de ciertas palabras en el prestigioso diario neoyorquino The New York Times desde el año 1981 hasta el 2010. Este mismo artista es el autor de otros proyectos famosos, como la organización de nombres en el memorial de Nueva York del ataque terrorista de las torres gemelas en la que se distribuyeron de forma que se encuentran en lo que denominó adyacencias significativas, es decir, cada uno está próximo a aquellos con los que tenía una mayor relación personal y se encontraban cerca del lugar en el que fallecieron.

Algunas de las figuras más famosas que surgieron de esta creación son, por ejemplo, RGB – NYT Word Frequency (figura 1.2) o Hope/Crisis – NYT Word Frequency (figura 1.1). En la primera, figura 1.2, se hace una representación de las palabras red, green y blue en tres anillos superpuestos, uno por cada color, que pueden ser leídos en dirección horaria. Este trabajo es interesante, ya que no se centra solo en la manifestación de las palabras ya mencionadas, sino que tiene en cuenta también la aparición de agrupaciones, equipos o empresas que son representados con esos tonos como puede ser en el caso del azul General Motors Corp o en el del rojo los Detroit Red Wings. El segundo de los trabajos mencionados de este proyecto, Hope/Crisis – NYT Word Frequency (figura 1.1), nos muestra en un formato parecido al primero la frecuencia de aparición de las palabras esperanza, en azul, y crisis, en grafito. Lo que nos permite de alguna forma apreciar la interconexión de estos dos términos en las complejidades del mundo en el que vivimos en el contexto del rotativo estadounidense.

Fue este macroproyecto el que inspiró este trabajo de fin de grado. Logró que, sin aspirar a conseguir las representaciones gráficas que tanto caracterizan a Jer Thorp, nos preguntáramos que resultados se podrían obtener si sometiéramos el conjunto de memorias que se han ido entregando a lo largo de los años en la Escuela de Ingeniería Informática (EII) de la Universidad de Las Palmas de Gran Canaria (ULPGC) a un proceso de procesamiento de lenguaje natural.

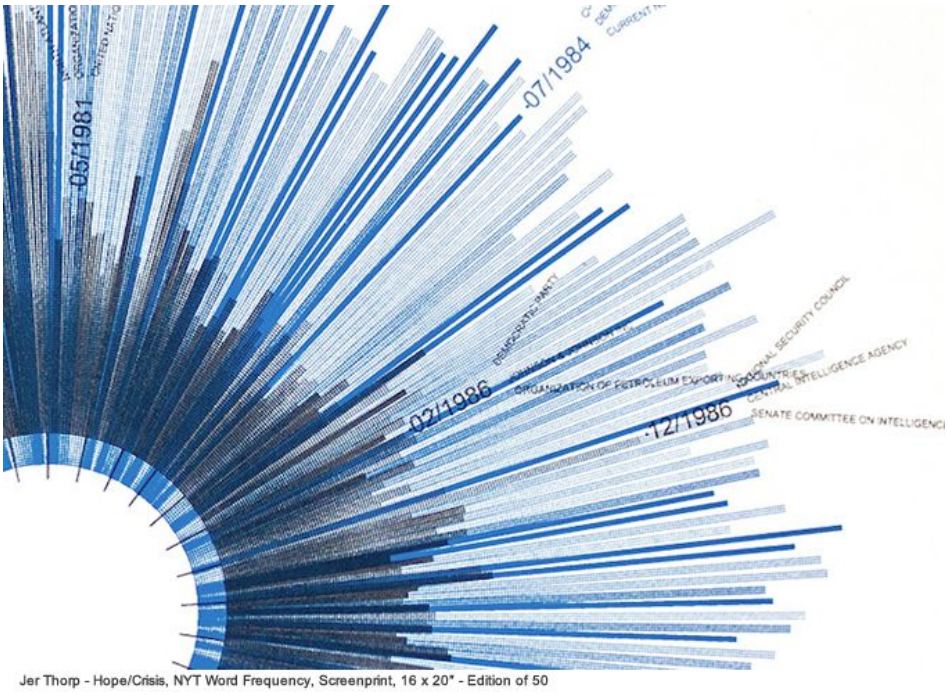


Figura 1.1: Visualización del proyecto Hope/Crisis – NYT Word Frequency.Imagen obtenida de https://i.pinimg.com/736x/3d/5b/ab/3d5bab3e1157ca1aae7ecfa2159d816b–business-design-data-visualization.jpg

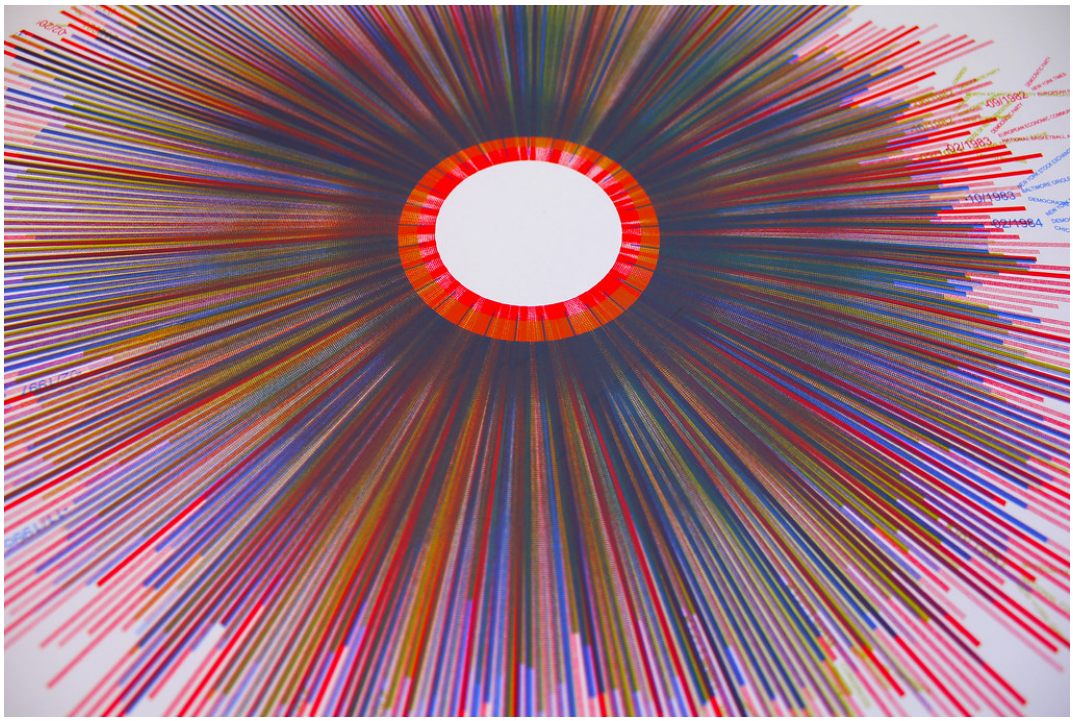


Figura 1.2: Visualización del proyecto RGB – NYT Word Frequency. Imagen obtenida de https://www.flickr.com/photos/randomnumbernu/5362267633

1.1. Competencias

En este subapartado recorreremos las competencias específicas del grado de Ingeniería Informática que han sido aplicadas en la realización del proyecto.

CP03

"Capacidad para evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos."

Este proyecto, por su propia naturaleza, ha requerido del uso de grandes cantidades de datos y por tanto de potencia computacional. Por este motivo, en la aplicación de algunos algoritmos ha sido necesario cambiar la aproximación inicial (computacionalmente más costosos) y darles un enfoque de programación dinámica, para así poder completar la tarea en un tiempo aceptable.

CP04

"Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación."

La espina central que da forma a este trabajo de fin de grado es la aplicación de los mencionados fundamentos, paradigmas y técnicas. Estos, han servido para analizar, diseñar y construir un sistema capaz de arrojar una serie de resultados concernientes al estudio del vocabulario usado en los TFT.

CP07

"Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos."

En este trabajo hemos puesto en práctica una de las técnicas, relacionadas con el NLP, más populares de los últimos años. Esta necesita de grandes volúmenes de datos para poder hacer una extracción del conocimiento que en ellos se encierra y, gracias a eso, construir los estudios y aplicaciones que deseemos.

Capítulo 2

Estado del arte

El campo del procesamiento del lenguaje natural, conocido y referido popularmente como NLP, es un subcampo de los sistemas inteligentes que trabaja con algoritmos computacionales para representar y procesar de forma automática distintas variantes de entradas de lenguaje natural, para así poder establecer una comunicación con una interfaz tipo HCI (Human-Computer-Interface).

Este proceso implica numerosas fases intermedias, entre la que podríamos destacar algunas como el análisis léxico, sintáctico y semántico. La complejidad y versatilidad que ofrece esta transformación permite que las áreas de aplicación de esta tecnología sean de elevado interés hoy en día, por ejemplo, el reconocimiento del habla, Optical Character Recognition (OCR), traductores y chatbots.

El comienzo del desarrollo de este subcampo de la Inteligencia Artificial podemos datarlo en torno a mediados del siglo veinte (un poco después de la segunda guerra mundial) cuando la gente comenzó a reconocer la importancia de poder traducir de un lenguaje a otro y esperaba poder desarrollar una máquina con esta capacidad. Podríamos establecer el test de Turing (1950) como uno de los primeros trabajos que dependen en gran medida del NLP. Ya fue en 1957 cuando los investigadores empezaron a desarrollar conjuntos de reglas basadas en estructuras sintácticas para cimentar la tecnología. Hubo que esperar hasta los años 90 para que con la lenta mejora de la capacidad computacional y la posterior introducción del Aprendizaje Automático se comenzaran a desechar estos complejos sistemas de reglas escritas a mano con un vocabulario reducido que constreñían la disciplina.

Actualmente, gracias a los algoritmos de Aprendizaje Automático, los procesamientos

de lenguaje natural pueden recibir como entrada enormes cantidades de ejemplos de textos escritos por humanos. Esto les permite entender el 'contexto' de las palabras y formas de expresión que usamos tanto al hablar como al escribir, lo cual mejora ampliamente los resultados que son capaces de dar los procesos NLP.

Hasta 1990, por lo ya comentado, la mayor parte del trabajo de investigación que se realizaba estaba centrado en establecer los conceptos del NLP y la creación de una máquina traductora. Las tecnologías de Aprendizaje Profundo han cambiado drásticamente esta situación. Los temas que ahora se estudian a menudo se solapan con la Inteligencia Artificial, en general, y el aprendizaje profundo. La inclusión de estos solapamientos ha logrado aumentar la eficiencia de las tareas que desempeña el procesamiento de lenguaje natural, por este motivo podemos encontrar gran cantidad de nuevos campos que se ven afectado por el NLP. La lista es larga, en ella podemos encontrar algunos ejemplos como: Diálogos y Sitemas Interactivos, Análisis de Documentos, Generación de Textos, Extracción de Información y Minado de Textos, Teorías Lingüísticas, Traducción Automática, Fonología, Análisis de Sentimientos, etc.

Aunque los campos de investigación y estudio son muy numerosos y el objetivo en última instancia de la disciplina sea equiparar la comunicación humano-máquina a la humano-humano hasta el punto de que sean indistinguibles, en la actualidad ya está recibiendo muchas aplicaciones industriales. Las principales aplicaciones industriales que realiza el NLP pueden ser clasificadas en general en 3 campos: Sistemas Conversacionales, Análisis de Textos y Traducción Automática.

Sistemas Conversacionales

Los sistemas conversacionales nos permiten establecer una conversación con un sistema automático a través del uso del lenguaje natural mediante voz o texto. Permiten automatizar complejos flujos de trabajo en empresas con atención al cliente 24 horas.

Los más comunes hoy en día son los conocidos como Chatbots y Asistentes Virtuales. Extensamente usados por bancos, comercio electrónico, redes sociales y otros servicios de venta de productos o servicios online.

Análisis de Texto

EL análisis de texto busca la obtención de información relevante de cualquier formato de texto, desde un tuit hasta novelas o textos académicos. En el presente se usa principalmente en el contexto de las redes sociales para obtener distintas métricas y realizar análisis. Aunque también es conocido su uso por parte de empresas financieras y bancos para realizar un minado de textos en tiempo real, para conocer los últimos cambios que se puedan haber producido en los mercados. Esta información, además de en las redes sociales, la extraen de periódicos electrónicos y de portales relacionados con las finanzas y la economía.

Traducción Automática

La traducción automática fue uno de los primeros campos a los que se aplicó la tecnología NLP y busca resolver la tarea de traducir un idioma a otro de forma automática preservando el significado del texto original.

Una de las aplicaciones más famosas que desmpeña esta labor es el conocido Google Translator. Aunque existen muchos otros proyectos software menos famosos que son también usados en la traducción. Uno de ellos, es el llamado DeepL que en la actualidad está recibiendo muy buenas críticas.

Capítulo 3

Objetivos

La cantidad de objetivos que podrían marcarse en un trabajo de esta clase son muchos y de muy variadas condiciones. Por ello, los propósitos que vamos a describir que conciernen a este proyecto son aquellos que fueron previamente especificados en el documento TFT01.

- Creación de un corpus de TFTs de la EII para su uso con técnicas de NLP -

En los últimos 11 años en la Escuela de Ingeniería Informática de la ULPGC se han entregado de media al año unos 50 trabajos de fin de título. En un mundo donde tecnologías como el aprendizaje profundo, muy unido actualmente al NLP, tienen un uso tan extendido que, al menos en el contexto del campo de la informática, casi podemos hablar de cotidianidad por la relativa facilidad que presenta aplicar estos algoritmos, esa ingente cantidad de datos representa una oportunidad que no se puede desaprovechar.

Esperamos que con la organización de estos datos puedan llegar en el futuro más proyectos que aprovechen esta cantidad de textos que permanece hasta ahora prácticamente inexplotada.

- Estudio de las tecnologías NLP para el español -

El campo del procesamiento del lenguaje natural ha sufrido en los últimos años un increíble desarrollo y una rápida popularización. Desgraciadamente, como suele ocurrir con la tecnología, las herramientas más populares están desarrolladas en inglés para que su uso pueda internacionalizarse de una forma más sencilla.

Esto, normalmente, no supone ningún tipo de inconveniente, pero en el caso que nos ocupa (al ser este un trabajo tan relacionado con el idioma español) supone una desventaja. Por ello, parte de este trabajo será la localización de herramientas compatibles con el castellano.

- Diseño de flujos de interacción -

Todos los proyectos software tienen un flujo de interacción, aunque el de este trabajo no estará necesariamente centrado en el diseño de una aplicación para el uso de un tercero. Esto no quita que se organizará el código para que pueda ser estudiado de forma compartimentada.

- Integración de elementos -

La fuente de la que extraeremos estos datos es el portal web asociado a la ULPGC, AccedaCRIS. Sin embargo, aunque todos los datos son extraídos de una misma fuente, la realidad es que son datos desestructurados que hay que revisar y limpiar para posteriormente intergrarlos dentro del corpus que usaremos para realizar los distintos estudios.

- Pruebas y Validación -

Como cualquier proyecto que implica el desarrollo de un software, a lo largo de todo el proceso, a medida que se van generando distintas funciones y elementos todas se irán testeando para asegurar su correcto funcionamiento.

Capítulo 4

Conceptos teóricos

La realización de este trabajo de fin de grado ha requerido del uso de una serie de conceptos teóricos, técnicas y algoritmos relacionados con el mundo del Aprendizaje Profundo y el NLP. En este capítulo esperamos poder realizar una explicación de cómo funcionan de forma que no se produzcan confusiones cuando aparezcan mencionados en otras secciones.

4.1. Stopwords

Las palabras “de”, “y” o “la” están entre las 10 palabras más usadas en prácticamente cualquier tipo de redacción en español. Estos vocablos, aunque sean fundamentales a la hora de establecer una correcta comunicación entre personas e incluso entre máquinas y humanos, cuando son estudiadas desde el prisma del análisis de textos carecen de valor, puesto que no aportan ninguna clase significado. No solo es así, sino que pueden alterar negativamente el examen de los documentos opacando la importancia de otras palabras que sí aportan significado y valor al texto, aunque su ratio de aparición sea considerablemente menor.

Las stopwords, también conocidas en español como palabras vacías, son este conjunto de palabras que aportan una baja, o nula, cantidad de información y significado a una frase. El procedimiento habitual en los procesos de limpieza de textos es retirarlas del conjunto para que no enturbien los resultados.

En el caso de este trabajo, hemos incluido en esta lista de palabras que se han retirado del conjuto de textos a las preposiciones, determinantes de todas las clases, conjunciones,

algunos tipos de adverbio como los de cantidad y pronombres. Además de una serie de conjugaciones de los verbos “ser”, “tener”, “estar” y “haber”.

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

Figura 4.1: Lista de Stopwords usadas en el proyecto.

4.2. Lemmatization y stemming

Lematization y Stemming son las denominaciones de dos algoritmos ampliamente usados en el contexto del procesamiento de lenguaje natural para normalizar los escritos antes de someterlos a procesos de aprendizaje profundo. Ambos son empleados para reducir el vocabulario de los textos al tiempo que evitamos la dispersión de la información de las palabras, por ejemplo, en una obra que va a seguir este procedimiento, las palabras ’niño’ y ’niñito’, en términos de significado en una frase, equivalen a lo mismo. Sin embargo, para los algoritmos estas palabras son tan distintas como podrían ser ’dragón’ y ’mermelada’. El objetivo de estas dos técnicas es reducir ambas palabras a una raíz común para que así ’niño’ y ’niñito’ sean tratados como si fueran el mismo vocablo.

El ’stemming’ elimina prefijos y sufijos de las palabras para terminar con la que considera es la raíz que comparten todas. De forma que si tenemos los términos ’necesidad’, innecesario’ y ’necesitado’, las transformaría a todas en la raíz ’neces’. En general, es un algoritmo muy dependiente de reglas del idioma inglés, por lo que es raro encontrar implementaciones válidas para otras lenguas, aunque en el desarrollo de este trabajo se han encontrado algunas compatibles con el español.

Sin embargo, el ’stemming’ puede presentar un problema llamado ’sobreestemización’

que se da cuando le asigna la misma raíz a palabras que no están relacionadas. Un ejemplo podrían ser las palabras 'universidad' y 'universo' que aunque compartan raíz no están relacionadas entre sí. Alternativamente existe el gemelo opuesto a este problema, que es la 'subestemización' donde el algoritmo es incapaz de asignar la misma raíz a dos palabras que claramente la comparten.

Por estos motivos, en este proyecto hemos usado la 'lematización' que puede ser considerada una evolución del stemming. Esta busca agrupar las distintas declinaciones y variaciones que pueda poseer una palabra en una palabra común. Además, otra diferencia que incorpora es que añade contexto a las palabras, pudiendo incluso juntar dos palabras con significado casi idéntico en una misma. En el caso inicial que planteamos, las palabras 'niño', 'niñito' o 'niñería' serían transformadas en 'niño'.

4.3. Word2Vec

Word2Vec [Wor] es una técnica para el procesamiento de lenguaje natural publicada en 2013. Esta emplea un modelo de red neuronal para establecer asociaciones entre palabras a partir de un gran conjunto de escritos. El método que usa para establecer estas relaciones se conoce como 'word embedding'.

El 'word embedding' es una técnica mediante la cual podemos transformar palabras individuales en representaciones vectoriales numéricas que representan el término, en el caso que nos ocupa hablamos de vectores de hasta 100 dimensiones. Estos vectores intentan almacenar las distintas características de los vocablos; hablamos de su semántica, el contexto en el que es usado, su definición, etc.

Derivada de estas representaciones numéricas viene la efectividad de Word2Vec, su capacidad de identificar las similitudes y diferencias entre palabras, usando para ello la distancia de los vectores entre sí, para poder agrupar juntos vectores que correspondan a palabras similares. Si se posee un corpus lo suficientemente grande, esta técnica puede hacer estimaciones bastante buenas sobre el significado de una palabra en función de sus ocurrencias en un texto, además de su relación con el resto de palabras. Uno de los ejemplos clásicos que se suele emplear para poner de manifiesto esta potencia de asociación, es que puedes realizar una consulta a la red del estilo: 'Dime la palabra que es a Mujer lo mismo que Rey es a Hombre'. La respuesta a esta consulta, siempre que se haya entrenado con el dataset correcto a la red, será 'Reina'. Esto es así porque Word2Vec es capaz de establecer que los vectores

correspondientes a las palabras 'Rey' y 'Reina' son muy similares entre sí, y, a través de una serie de operaciones algebraicas internas, puede establecer que la proximidad de 'Hombre' a 'Rey' es muy próxima a la de 'Mujer' y 'Reina'.

4.4. Distancia Levenshtein

La distancia Levenshtein es un algoritmo desarrollado en 1965 por Vladimir Levenshtein. También conocida como "Distancia Edit", cuenta la catidad mínima de modificaciones caracter a caracter (inserciones, eliminaciones y sustituciones) a las que hay que someter a una palabra para que se transforme en otra.

La aplicación principal por la que son útiles este tipo de algoritmos es para resolver un problema común dentro del NLP, que es la localización de similitudes entre textos. Un ejemplo de este uso puede ser la utilización del servicio Turnitin, que asegura que los escritos que entregan los alumnos han sido hechos por ellos y no plagiados.

La distancia Levenshtein acumula la suma del número de ediciones a las que hay que someter a los caracteres para transformar una palabra en otra (el funcionamiento del algoritmo requiere de dos vocablos, uno como referencia y el otro que es modificado). Cuanto mayor sea este número más distante estará una palabra de otra, por ejemplo, las palabras 'tener' y 'perder' tendrán una distancia entre sí de 3, porque si queremos convertir 'tener' en 'perder' tendremos que sustituir la 't' por la 'p', la 'n' por la 'r' y, además, insertar una 'd'. En caso de que la transformación quisieramos hacerla en el otro sentido, la distancia sería idéntica, solo tendríamos que invertir los pasos y sustituir la inserción de la 'd' por su eliminación. Por otro lado, la distancia entre 'tener' y 'tener' es 0, ya que se trata del mismo vocablo.

En la figura 4.2 podemos apreciar la representación matemática de la distancia edit. Donde a y b corresponden a las dos strings que se toman como entrada, $|a|$ y $|b|$ son la longitud de cada una de esas ristras, la palabra 'tail' de una string se refiere a la ristra que esté referenciando excepto su primer caracter y a[0] y b[0] al primer caracter de cada string respectivamente.

$$\operatorname{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \operatorname{lev}\big(\operatorname{tail}(a), \operatorname{tail}(b)\big) & \text{if } a[0] = b[0] \\ 1 + \min \begin{cases} \operatorname{lev}\big(\operatorname{tail}(a), b\big) \\ \operatorname{lev}\big(a, \operatorname{tail}(b)\big) \\ \operatorname{lev}\big(\operatorname{tail}(a), \operatorname{tail}(b)\big) \end{cases} & \text{otherwise,} \end{cases}$$

Figura 4.2: Representación matemática de la distancia Lehvenstein. Imagen obtenida en https://en.wikipedia.org/wiki/Levenshteindistance.

4.5. K-Means

K-Means es un algoritmo de agrupamiento ampliamente usado en el mundo de la informática. Concretamente es un algoritmo no supervisado, esto significa que aprende patrones de datos sin etiquetar. Por ello, puede ser usado para entrenar un modelo que genera grupos de cualquier conjunto de datos para una K predefinida sin, teóricamente, tener que poseer ninguna clase de familiaridad con las muestras.

Cada dato solo puede pertenecer a un único clúster. Estos consisten en agrupaciones de datos que comparten similitudes entre ellos, lo cual significa que los otros valores (los no incluidos en el grupo) pertenecen a otros conjuntos que tienen unas características distintas a este.

El funcionamiento en sí es bastante simple. Primero hay que definir un número K de clúster que formar. Después se asignan un número K de puntos aleatorios para formar el centroide del grupo. Una vez tenemos el centroide asignamos los puntos restantes al agrupamiento del centroide más cercano y calculamos el nuevo centroide medio. Este último paso lo repetimos hasta que no se produzcan cambios en los agrupamientos.

Capítulo 5

Tecnologías

El desarrollo de este trabajo se ha llevado a cabo en su totalidad en el lenguaje de programación de alto nivel Pyhton. Concretamente, la IDE (Integrated Development Environment) usada fue Pycharm Community Edition 2021.2.2 con el entorno de Conda. Se escogió esta opción por las facilidades que incorpora a la hora de gestionar y controlar las distintas bibliotecas que fueron necesarias para la correcta realización del proyecto, además de que el alumno ya estaba familiarizado con el entorno, por lo que no fue necesario un periodo de aprendizaje con la herramienta. A continuación, llevaremos a cabo un listado de las distintas bibliotecas y módulos que fueron necesarios para hacer este trabajo de fin de grado y expondremos brevemente cual fue su función concreta en el conjunto del proyecto.

5.1. Multiprocessing

Este paquete permite la generación de procesos usando una sencilla API (Application Programming Interface) para poder desarrollar aplicaciones en las que es necesario realizar programación concurrente o para mejorar el aprovechamiento de los múltiples procesadores que pueda tener una máquina determinada. En el caso de este proyecto que nos ocupa, se utlizó para determinar cuántos núcleos de nuestro procesador se encontraban disponibles, para poder dedicarlos en el entrenamiento del modelo de red neuronal que se utilizó y de esta forma reducir considerablemente el tiempo de espera [mul].

Figura 5.1: Logo de Python. Imagen extraída de https://shorturl.at/LMUY4

5.2. Os

Este módulo está pensado para poder interactuar con diversas funcionalidades que dependen del sistema operativo del ordenador. En el contexto de este trabajo, eran muchos los documentos con datos necesarios que por las distintas naturalezas de la información que almacenaban, se encontraban organizados en diversos directorios y subdirectorios. Por este motivo se empleó os, para poder lidiar con los accesos a todas estas carpetas de una forma ordenada y controlada [os].

5.3. Re

Este módulo consigue proveer de operaciones de emparejamiento con expresiones regulares similares a las que se encuentran el Perl. En este proyecto, por su propia esencia, hemos tenido que trabajar con gran cantidad de textos provenientes del conjunto de memorias publicadas por la escuela. Estas son publicadas en PDF (Portable Document Format) y para poder procesarlas tuvieron que ser transformadas en texto plano. En el proceso de conversión se producen errores que generan caracteres especiales que no son interesantes y pueden perjudicar el transcurso del proyecto. Usábamos re precisamente para hacer una limpieza inicial (de caracteres especiales, números, etc.) en los textos [re].

5.4. Numpy

Esta biblioteca permite gestionar de forma cómoda, contrastada y efectiva operaciones de matrices y vectores con alta dimensionalidad. El modelo de inteligencia artificial que hemos usado en este trabajo convierte todas las palabras que aprende en vectores de cien dimensiones. Con el objetivo de dividir los títulos en varios clúster necesitamos hallar el vector medio que representa cada uno, todas estas operaciones fueron manejadas gracias a Numpy. Este es uno de los varios ejemplos que podemos poner sobre los diversos usos que se le pueden dar a esta biblioteca [Num].

Figura 5.2: Logo de NumPy. Imagen extraída de https://shorturl.at/GHLP0

5.5. Pandas

Originalmente concebida como una expansión del ya mentado Numpy, sección 5.4, ofrece una forma simple, flexible y rápida de realizar manipulación y análisis de datos. En nuestro caso lo empleamos para almacenar, guardar y manipular los datos que leemos de los ficheros y, de esta forma, poder acceder cómodamente a ellos en los diversos momentos de la ejecución en los que son requeridos. Además, los usamos para filtrar por valor los datos cuando es necesario y para evitar tener que procesar los ficheros que los almacenan originariamente cada vez que queramos procesarlos [Pan].

Figura 5.3: Logo de pandas. Imagen extraída de https://shorturl.at/jmrU3

5.6. NLTK

Es una plataforma que se centra en la construcción de programas relacionados con el procesamiento de lenguaje natural. Esta contiene numerosas bibliotecas para el procesamiento

de textos a los que podemos tokenizar, clasificar, etiquetar, analizar, etc. En nuestro trabajo de fin de grado uno de los principales usos que le hemos dado a este conjunto de bibliotecas fue la obtención del conjunto de stopwords del español, necesarias para una correcta limpieza de los textos [NLT].

Figura 5.4: Logo de NLTK. Imagen extraída de https://medium.com/100daysofmlcode/day-40-of-100daysofml-edfcef9a161e

5.7. Spacy

Se trata de otra biblioteca que se centra en el procesamiento de lenguaje natural e incorpora gran cantidad de funcionalidades con objeto de simplificarlo. En este caso, su uso fue interesante para nosotros, porque incorporaba un algoritmo de lematización compatible con el español cosa vital para poder completar adecuadamente la limpieza de los datos antes de incorporarlos como entrada en el algoritmo Word2Vec [Spa].

Figura 5.5: Logo de Spacy. Imagen extraída de https://shorturl.at/abgv7

5.8. Scikit-Learn

Esta biblioteca es una herramienta simple y eficiente para la predicción y análisis de datos, además de contar con gran cantidad de opciones en lo referente al aprendizaje automático. A nosotros nos fue de gran utilidad porque dentro de su repertorio incorpora, entre otros, el algoritmo de agrupamiento K-means. Este lo usamos para llevar a cabo la clasificación de los diversos títulos en varios grupos en función del vector medio de cada uno [Sci].

Figura 5.6: Logo de Scikit-Learn. Imagen extraída de https://shorturl.at/eopq0

5.9. Gensim

Esta es una de las bibliotecas más importantes que incorporamos, esta se centra en modelado de temas, la indexación de documentos y en la recuperación de similitudes en grandes corpus de datos. Actualmente es ampliamente usado por la comunidad de procesamiento de lenguaje natural y recuperación de información. En nuestra situación fue relevante principalmente porque nos proveyó con la implementación del algoritmo Word2Vec, una de las piezas centrales en todo este trabajo [Gen].

Figura 5.7: Logo de gensim. Imagen extraída de https://radimrehurek.com/gensim/intro.html

5.10. Matplotlib

Biblioteca que nos permite la creación de visualizaciones estáticas, animadas e interactivas. Ampliamente usada a lo largo de este proyecto para mostrar los resultados obtenidos de los distintos tratamientos realizados al conjunto de los datos [Mat].

Figura 5.8: Logo de Matplotlib. Imagen extraída de https://shorturl.at/jnvZ0

5.11. Circlify

Este paquete está en realidad basado en el ya mencionado Matplotlib. Desarrollado por el usuario de GitHub elmotec, esta bilioteca permite la implementación de gráficos de paquetes de círculos. Se decidió utilizar para poder ver, de una forma alternativa a los clásicos diagramas de barras, cuáles eran las palabras más populares y usadas a lo largo del tiempo [Cir].

Capítulo 6

Desarrollo

Una vez, hemos conseguido establecer las bases de todo los elementos que han tomado mayor o menor parte en el transcurso de este trabajo de fin de grado y hemos desgranado los distintos conceptos formales y/o teóricos que lo envuelven. Vamos a proceder a relatar cómo ha sido concretamente el desarrollo del mismo.

Lo primero que habría que poner de relieve es la naturaleza del proyecto. Este no surge con el ánimo de crear una aplicación con una interfaz que sea fácilmente manipulable por un conjunto determinado de usuarios, esto es un trabajo de ciencia de datos y como tal, el relato de cómo se ha ido realizando va a estar dividido en las distintas partes que caracterizan a un proyecto de esta índole.

6.1. Recolección de los datos

Como todo trabajo de ciencia de datos que se precie, el primero de los pasos que tuvimos que dar fue la obtención del conjunto de datos con los que íbamos a trabajar. Estos podían ser conseguidos a través del portal web de la universidad, "accedaCRIS".

En esta página uno puede encontrar todas las publicaciones que ha realizado la ULPGC, ya que, según se definen ellos mismos, es el "Portal de investigación científica que recopila la producción científica de la Universidad de Las Palmas de Gran Canaria (ULPGC) y parte del Current Research Information System (CRIS) de esta Universidad. " . Concretamente, para este proyecto estábamos interesados en la sección de publicaciones académicas que ofrece la

web. En esta podemos encontrar los trabajos de fin de título de todas las carreras que ofrece la universidad.

Para obtener los datos que necesitábamos para el estudio tuvimos que aplicar un filtro que nos permitía discriminar la búsqueda (figura 6.1) y así solo obtener los resultados del "Grado en Ingeniería Informática". Esto unido a un nuevo filtro nos permitió poder hacer una rápida separación de los Trabajo de Fin de Título (TFT) en función de su año de publicación.

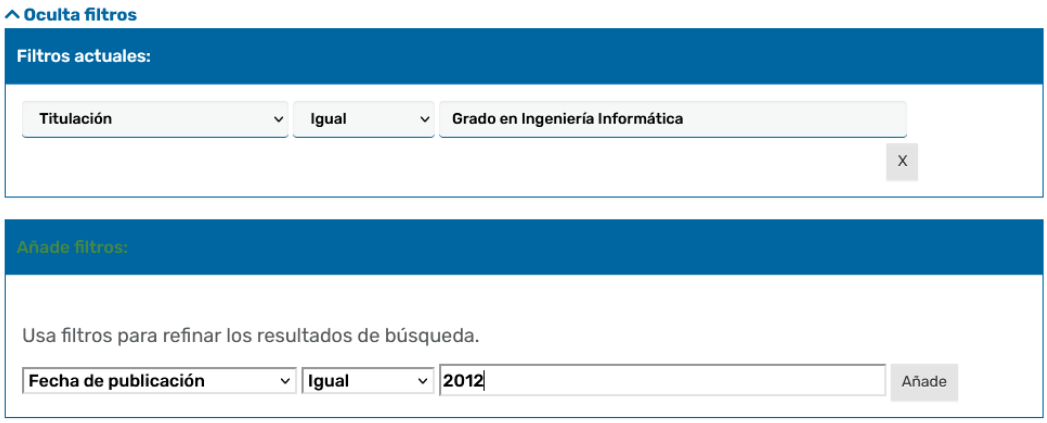


Figura 6.1: Ejemplo de uso de los filtros del portal web accedaCRIS

Un aspecto importante que debemos especificar es que no necesariamente por haber una entrada de un proyecto, este va a tener una memoria subida dentro de la página. Desde el año 2014, se viene aplicando en la EII una política al criterio de publicación de un trabajo en este portal web. Se necesita alcanzar una nota mínima de un 9, además del consentimiento explícito del alumno y los tutores para que el documento sea subido. Solo si se cumplen estos dos requisitos aparecerá la memoria almacenada. Por el contrario, aunque no se alcancen estos requerimientos se genera una entrada donde aparece el título del TFT, su autor, sus tutores y alguna información extra.

Durante la extracción de los datos, en un estadío inicial, se decidió simplemente recuperar las memorias que estaban disponibles, ya que estas iban a ser el cimiento sobre el que construiríamos el resto del trabajo. Aunque se consideró usar una herramienta que permitiese la extracción automática de los datos, la idea se acabó desechando porque, aunque en un primer vistazo pueda no parecerlo, la forma de acceder a las memorias varía en función de la publicación.

Figura 6.2: Portal accedaCRIS, publicaciones del año 2013 del Grado en Ingeniería Informática

En la figura 6.2, podemos ver un resultado normal tras aplicar los filtros necesarios para obtener los TFT de la Escuela de Ingeniería Informática. Como podemos ver, en la primera de las entradas, en el lado derecho se puede apreciar la memoria accesible para cualquiera que desee leerla o descargarla. La deducción que podríamos hacer al ver esta imagen es que las dos siguientes publicaciones carecen de una tesis adjunta, sin embargo, esto no es así. Los dos siguientes sí tienen un informe almacenado, pero para poder acceder a ellos uno tiene que registrarse, lo que hace que no aparezca el icono e induzca a pensar que no hay nada subido.

Este no pasa únicamente con aquellos proyectos con un acceso restringido, a menudo los TFT incluyen aparte de una memoria una serie de documentos extras. En ocasiones es el código de la aplicación desarrollada, anexos, resúmenes, etc. Cuando en primer lugar dentro de la lista de documentos subidos no se encuentra el PDF correspondiente al informe, aunque el proyecto no tenga restricciones repecto al acceso, tampoco aparecerá ningún icono que indique que se ha subido una memoria.

Por este motivo se tuvo que revisar manualmente todas y cada una de las entradas de la página web “accedaCRIS”. Hay que añadir que dentro de cada una de las publicaciones existía la opción de descargar los datos que contenía en varios formatos (CSV, excel, PDF, etc), pero solo se aplicaba a los datos de esa entrada en concreto y, por la complejidad y sobrecarga de los recursos que se estimó que supondría lidiar con todo ese número de documentos a posteriori, se decidió no emplearlo.

Una vez se hubo revisado todos los trabajos de fin de grado del Grado en Ingeniería Informática, que abarca desde el año 2011 hasta el 2022, se habían obtenido 274 memorias de 553 trabajos, es decir, de todas las publicaciones solo el 49,3 % iba a formar parte de este proyecto. Este número, aunque ya de por sí suficiente para fundar un corpus sustancial con el que hacer un trabajo de ciencia de datos, se quiso hacer mayor uniendo al dataset los proyectos del antecesor de este grado, la carrera de Ingeniero en Informática. Antes de esta carrera la Universidad de Las Palmas de Gran Canaria ofrecía una diplomatura, "Diplomado en Informática", de la que se decidió prescindir en el estudio por remontarse demasiado atrás y no aportar demasiadas nuevas entradas al corpus.

Esta nueva inmersión en el portal accedaCRIS, en la que también hubo que hacer comprobaciones a mano de cada entrada, resultó en una adición de 348 memorias extra de 376 publicaciones, es decir el 92.6 % estaban almacenadas. Estas habían sido presentadas entre 1989 y 2019 y junto con las que ya poseíamos alcanzamos el número de 622 textos de 929 publicaciones, el 67 %. Además, debemos añadir que entre esta adición de nuevos textos y la original se decidió recoger en un conjunto aparte los nombres de los tutores de cada proyecto y el título, este en un segundo set distinto del de los tutores, para poder realizar estudios con ellos más adelante.

	Memorias	Publicaciones	Memorias/Publicaciones
TFT	271	550	49.3 %
Proyectos	351	379	92.6 %
Total	622	929	67 %

Cuadro 6.1: Resumen de las memorias y publicaciones obtenidas

En el cuadro 6.1 podemos apreciar el resumen de las cantidades obtenidas tras hacer la recolección de los datos. La columna "Memorias" hace referencia a todas las publicaciones que incluían una tesis, la columna "Publicaciones" indica el total de entradas que existía en el momento de hacer esta colecta de información y "Memorias/Publicaciones" es el porcentaje sobre el total de memorias publicadas. Respecto a las filas, "TFT" simboliza al Grado en Ingeniería Informática y "Proyectos" a la carrera Ingeniero en Informática.

Año	Memorias_TFT	TFT	Memorias_Proyecto	Proyectos	Memorias/Total
1989	0	0	2	2	100 %
1991	0	0	1	1	100 %
1992	0	0	2	2	100 %
1994	0	0	1	1	100 %
1995	0	0	2	2	100 %
1996	0	0	1	1	100 %
1999	0	0	2	2	100 %
2000	0	0	5	5	100 %
2001	0	0	6	6	100 %
2002	0	0	10	10	100 %
2003	0	0	14	14	100 %
2004	0	0	10	10	100 %
2005	0	0	12	13	92.3 %
2006	0	0	15	16	93.8 %
2007	0	0	21	24	87.5 %
2008	0	0	23	23	100 %
2009	0	0	34	35	97.1 %
2010	0	0	42	43	97.7 %
2011	30	33	35	37	92.9 %
2012	23	25	27	28	94.3 %
2013	42	42	15	19	100 %
2014	41	43	25	27	94.3 %
2015	12	39	15	16	49.1 %
2016	6	57	6	9	18.2 %
2017	26	58	12	18	50 %
2018	21	47	2	2	46.9 %
2019	27	58	7	17	45.3 %
2020	33	79	0	0	41.8 %
2021	10	53	0	0	18.9 %
2022	0	16	0	0	0 %

Cuadro 6.2: Desglose de las memorias y proyectos que se incluyeron en el trabajo

En el cuadro 6.2 podemos apreciar en orden de izquierda a derecha por columnas: el año en el que se publicaron, el número de memorias del grado de Ingeniería Informática subidas, el total de publicaciones del grado de Ingeniería Informática, el número de memorias almacenadas de la carrera, la cantidad de entradas de la carrera y, finalmente, el porcentaje del total de memorias publicadas frente al total de proyectos aprobados.

Además, en la tabla podemos apreciar cosas como la popularización de los estudios de informática a lo largo del tiempo, o si se prefiere, la digitalización de los TFT de los alumnos. Comenzamos sin llegar a las 2 cifras antes de los 2000 y se alcanza un pico en 2020 con 79 publicaciones. Otra cosa que es claramente visible es la entradda en vigor del criterio de publicación de las memorias en la web accedaCRIS en el 2014. Se aprecia claramente el brusco descenso en el porcentaje de memorias almacenadas.

6.2. Limpieza y organización de los datos

El siguiente paso que tuvimos que dar en este proceso, fue la organización y limpieza de los datos extraídos de accedaCRIS. Gran parte de la organización se hizo a medida que se extraían, los textos fueron agrupados en directorios que recibían como nombre el año en el que se publicaron. Asimismo, el nombre de cada uno de los documentos fue el título del proyecto.

Por otro lado, los títulos también se incluyeron en un documento, uno por cada año, para poder tenerlos concentrados en un mismo lugar y no tener que leer grandes cantidades de ficheros para extraerlos todos. En lo que se refiere a los tutores de cada uno de los proyectos, fueron incuidos, al igual que los títulos, en un documento aparte agrupados por año. El orden en el que se guardaron los títulos dentro de cada documento es idéntico al orden en el que se almacenaron los tutores, con esto logramos simplificar el acceso a los datos dentro del programa. Por ejemplo, los tutores del trabajo publicado en el 2017 cuyo título aparece en primera posición dentro del documento de títulos, se encuentran, a su vez, en la primera posición del documento de tutores del año 2017. Cuando se lean ambos ficheros y, pongamos, se almacenen los títulos y tutores en dos arrays para luego trabajar con ellos, en cada uno de los índices de ambas listas, que se sobreentiende son del mismo tamaño, se encontran los contenidos referentes a un mismo proyecto.

6.2.1. Limpieza de las denominaciones de los tutores

Una de las labores que nos llevo más tiempo completar fue la homogeneización de los nombres de los tutores. Podría parecer en una primera instancia excesivo, pero la realidad fue que nos encontramos con una situación que no podíamos ignorar.

Los nombres de los tutores son almacenados en cada publicación, al igual que el resto de información, presumiblemente por un bibliotecario o encargado y como tal, están sujetos a la aparición de errores propios de la naturaleza humana. La cuestión es que un mismo tutor podía aparecer con hasta 3 o 4 variaciones de la forma en la que se escribe su nombre: su nombre completo seguido de sus apellidos, los apellidos seguido de su nombre completo, su nombre y después solo su primer apellido, su primer apellido seguido de su nombre, la aparición o no de sus segundo nombre si lo tuviese, etc. Esto nos presentaba una gran cantidad de inconsistencias que dificultan enormente el tratamiento y agrupación de los apelativos de los tutores (nótese que en la lista no hemos incluido la aparición de erratas, también muy comunes), pero, desgraciadamente, esta no era toda la casuística con respecto a los nombres con la que nos tuvimos que enfrentar.

Encontramos una considerable cantidad de casos extremos y particulares con los que hubo que lidiar que ahora listaremos para que puedan hacerse una pequeña idea de la gran cantidad de inconsistencias, duplicidades y errores que localizamos. En el año 2016 aparecía un trabajo de fin de grado que directamente carecía de tutor. En los años 2012 y 2013 localizamos dos proyectos con prácticamente el mismo título, el mismo autor y mismos directores, sus memorias en un primer vistazo dan la impresión de ser idénticas, la mayor diferencia es que en 2013 adjuntaron un nuevo documento donde se incorporaban un abstract y un resumen. Otro caso relacionado con la aparición de duplicidades se ubicó en el año 2014, donde dos TFT claramente relacionados, eran el desarrollo del front-end y el back-end de una misma aplicación, resultaron ser el mismo trabajo subido dos veces, ambos trataban de la implementación del backend. Aunque esta parecía tener una explicación plausible ya que existía un tercer proyecto que relataba la implementación del front-end, así que suponemos que se subió incorrectamente el título de la duplicidad que falsamente indicaba que era el desarrollo del front-end y cuando se corrigió y se cambió el título por el correcto (el del back-end) nunca se llego a retirar la entrada incorrecta. Podríamos continuar mentando una larga serie de situaciones similares a las ya mencionadas, pero como último ejemplo de errores localizados pondremos lo que denominamos “nombres frankenstein”, llamados así por estar conformados claramente por la combinación de nombres de varios tutores, no era raro

localizarlos, por ejemplo vimos a "Fernando González Ripoll Navarro, Ángel Manuel Ramos García" o a "Teresa Morant De Diego E Inmaculada Carretero Moreno" entre otros.

Todos estos errores fueron recopilados y reportados por las vías pertinentes para que pudieran ser corregidos con la mayor brevedad posible y así subsanar la página. Esto fue considerado como un resultado del TFT que no se había considerado al inicio. Además, nos consta que ha sido de utilidad a accedaCRIS para la eliminación de los errores de las entradas, lo que ha hecho que todo este trabajo haya tenido una utilidad real.

Algunos de estos errores eran claramente apreciables a medida que se recolectaban los datos, pero rápidamente se hizo patente que la tarea de detectar todas estas inconsistencias y errores a lo largo de 33 años de publicaciones a mano era irrealizable, especialmente aquellos relacionados con la ortografía. Por esa razón, se decidió agrupar todas las variaciones de los nombres en un mismo conjunto y se procedió a comparar unas con otras usando para ello la distancia Levenshtein.

La aplicación de este algoritmo presentó sus propias dificultades, puesto que como se puede apreciar en la figura 4.2 el algoritmo tiene una naturaleza recursiva. Esto hace que tener que comprobar letra a letra de cada uno de los apelativos si se debe sustituir, añadir o eliminar para que sea pueda ser transformado en el nombre con el que se le está comparando implique un tiempo de cómputo inaceptable. Para más inri debemos mencionar que había un total de 251 nombres, lo que hace un total de más de treinta y un mil combinaciones distintas con nombres de longitudes variables que en su mayoría excedían o se aproximaban a 20 caracteres.

La solución a esta situación vino de la mano de la localización en GitHub de una implementación de este algoritmo por el usuario "bdebo236" usando para ello programación dinámica, específicamente estamos hablando de una implementación "tabulation" del problema. Tras localizar este código y someterlo a una serie de tests para comprobar su correcto funcionamiento, se observó que los superaba sin problemas y se resolvió emplear la implementación. Una vez teníamos el código en funcionamiento solo tuvimos que aplicarle un filtro para que no nos mostrase aquellas distancias que eran muy grandes y que, por tanto, los apelativos implicados no tenían mucha relación entre sí.

Cuando se terminó este proceso solo tuvimos que sustituir las erratas y variaciones por el que se estableció debía ser el estándar. Para tratar de respetar la consistencia que parece debía seguir el portal web accedaCRIS, se estableció como modelo los apellidos, seguidos de una coma y, a continuación, el nombre completo.

6.2.2. Transformación del formato de los textos

Los textos, una vez estuvieron localizados dentro de sus correspondientes directorios tuvieron que ser sometidos a un proceso de transformación. La extensión de todos ellos era la de PDF (Portable Document Format) lo cual nos obligaba a convertirlos a un formato que nos permitiese manipularlos comodamente a nuestro antojo, este fue por su puesto el texto plano TXT.

Para transformarlos se probaron varios métodos, algunos conversores online e incluso el conversor a texto que tiene integrado, el programa de manipulación de PDF por excelencia, Adobe Acrobat. Los resultados variaban en función del documento, sin embargo, en términos generales, se encontró que la página pdftotext arrojaba de media mejores resultados que todo el resto de métodos, además de que permitía la conversión de varios PDF de forma conjunta.

No obstante, nos encontramos con una dificultad que no se supo preveer con anterioridad. El problema en las conversiones lo encontramos con los escritos correspondientes a la carrera de Ingeniero en Informática. Estos proyectos habían sido entregados originalmente en papel, almacenados en los archivadores correspondientes y, posteriormente, con la aparición del portal accedaCRIS escaneados y subidos a la web.

Esta forma de creación del documento, hizo que al contrario de lo que ocurría cuando el PDF era originario de un documento ya digital, los conversores tuvieran grandes dificultades a la hora de detectar correctamente las palabras escritas. Esto hacía que muchos de los ficheros al ser transformados solo incluyeran caracteres no alfabéticos, metacaracteres, que se encontrasen completamente vacíos o en los casos donde lograba extraer algunos párrafos, como se puede ver en la figura 6.3, estos aparte de incompletos estuviesen repletos de palabras llenas de faltas de ortografía o combinadas con caracteres numéricos y de otra índole.

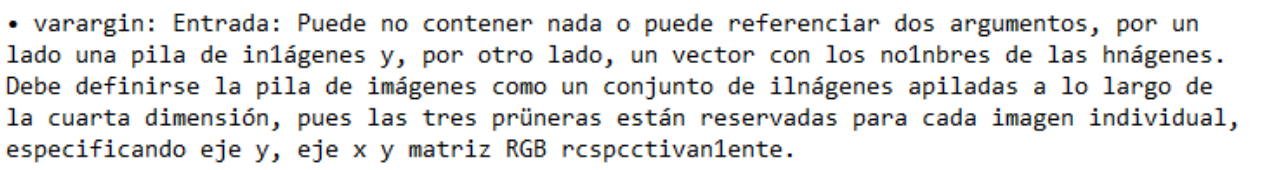


Figura 6.3: Ejemplo de conversión de un párrafo de un texto perteneciente a la carrera de Ingeniero en Informática

Por este motivo, la dificultad y complejidad de lograr una conversión satisfactoria para

poder ser empleadas en un estudio con fundamento, se decidió prescindir de las memorias presentadas en la titulación de Ingeniero en Informática. Se retiraron del total del estudio unos 351 ficheros, que representaba en torno a la mitad de todo el corpus en lo que se refiere a los textos. Sin embargo, los títulos y tutores se conservaron, puesto que al haber sido extraídos directamente de accedaCRIS, no presentaban los mismos problemas.

6.2.3. Limpieza general

Finalmente, en lo que a limpieza se refiere hubo que someter al conjunto de títulos, y también a los textos, a una serie de tratamientos antes de someterlos al conjunto de estudios por los que pasaron posteriormente. Esta limpieza se realizó frase a frase, por lo que en primer lugar se dividieron los textos usando para ello los puntos que delimitan las oraciones.

Después, valiéndonos para ello del módulo de Python re mencionado en la sección 5.3, llevamos a cabo una limpieza que autodenominamos "superficial" de los textos. Retiramos los caracteres especiales, retiramos los caracteres numéricos y eliminamos aquellas palabras que tras haber sido sometidas a este tratamiento solo estaban formadas por un único caracter. Además, sustituimos todos aquellos espacios múltiples que se pudieron haber generado al realizar estas cribas, por un solo espacio, quitamos los que podrían exitir al comienzo y al final de las frases y eliminamos aquellas frases que se habían quedado vacías después del proceso.

Hay que añadir que todas las palabras fueron convertidas a minúsculas y que durante todo este trabajo se han respetado las tildes, diéresis y caracteres como la letra 'ñ' que se usan comúnmente en el español. Es vital la conservación de estas letras, puesto que si las eliminamos o sustituimos las vocales con tildes por su alternativa sin ella, estamos contaminando el contexto y el significado de las frases. Un ejemplo claro y rápido de esto es la palabra "publico", sin tilde es una palabra llana que proviene del verbo publicar y tiene un tiempo presente, si le añadimos una tilde y la transformamos en aguda, "publicó" el tiempo cambia a pasado (cosa que en realidad no va a tener efecto debido a la lematización), pero si la volvemos esdrújula, "público", se refiere a un conjunto de personas.

Finalmente, hicimos una limpieza puramente relacionada con el NLP. Comenzamos eliminando las stopwords, mencionadas en la sección 4.1, y así retirando la trivialidad de los resultados que nos puede ocasionar someter estos datos a los procesos sin haber retirado antes los pronombre, preprosiciones, verbos auxiliares, determinantes, etc. Una vez retiradas,

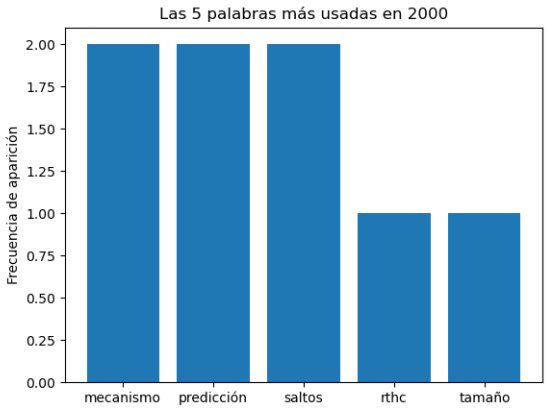
sometimos al conjunto de los datos a un proceso de lematización, detallado en la sección 4.2, con la que conseguimos eliminar las declinaciones de las palabras y conjugaciones de los verbos. Como paso extra, aplicado solo para las memorias retiramos aquellas frases que al final de la limpieza estaban formadas por dos palabras o menos, por la poca información que aportan a los modelos.

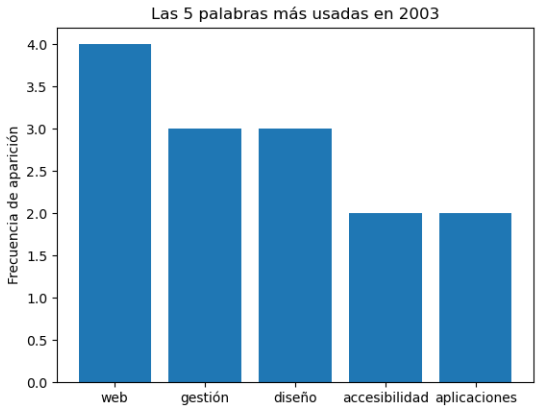
6.3. Estudio preliminar

Esta parte del desarrollo se realizó simultáneamente junto con la limpieza y organización de los datos, nos sirvió como toma de contacto de las posibilidades que podía ofrecer este corpus. La idea inical que se planteó fue sencilla, queríamos ver cuáles eran las palabras más comunes que se usaban cada año en los títulos de los trabajos de fin de título.

Como ya teníamos organizados los títulos por año, no resultó muy difícil realizar los conteos necesarios para establecer las palabras más populares anualmente, por suspuesto fueron sometidas a una limpieza preliminar. Estas se decidieron representar en histogramas y en burbujas. La representación en burbujas requirió del uso de la biblioteca Circlify, comentada en la sección 5.11.

En la figura 6.7 podemos ver una pequeña selección de algunos de los histogramas que se generaron. Claramente se puede apreciar como las palabra 'aplicación' se vuelve la más popular a partir del año 2015. Puesto que antes era alternado por un set más amplio de palabras, destacamos algunas como 'sistema' o 'desarrollo'. Además, deberíamos hacer una mención especial a la palabra 'web' que parece nunca pasar de moda y de algún modo colarse en la gran mayoría de tops durante un periodo de 20 años.





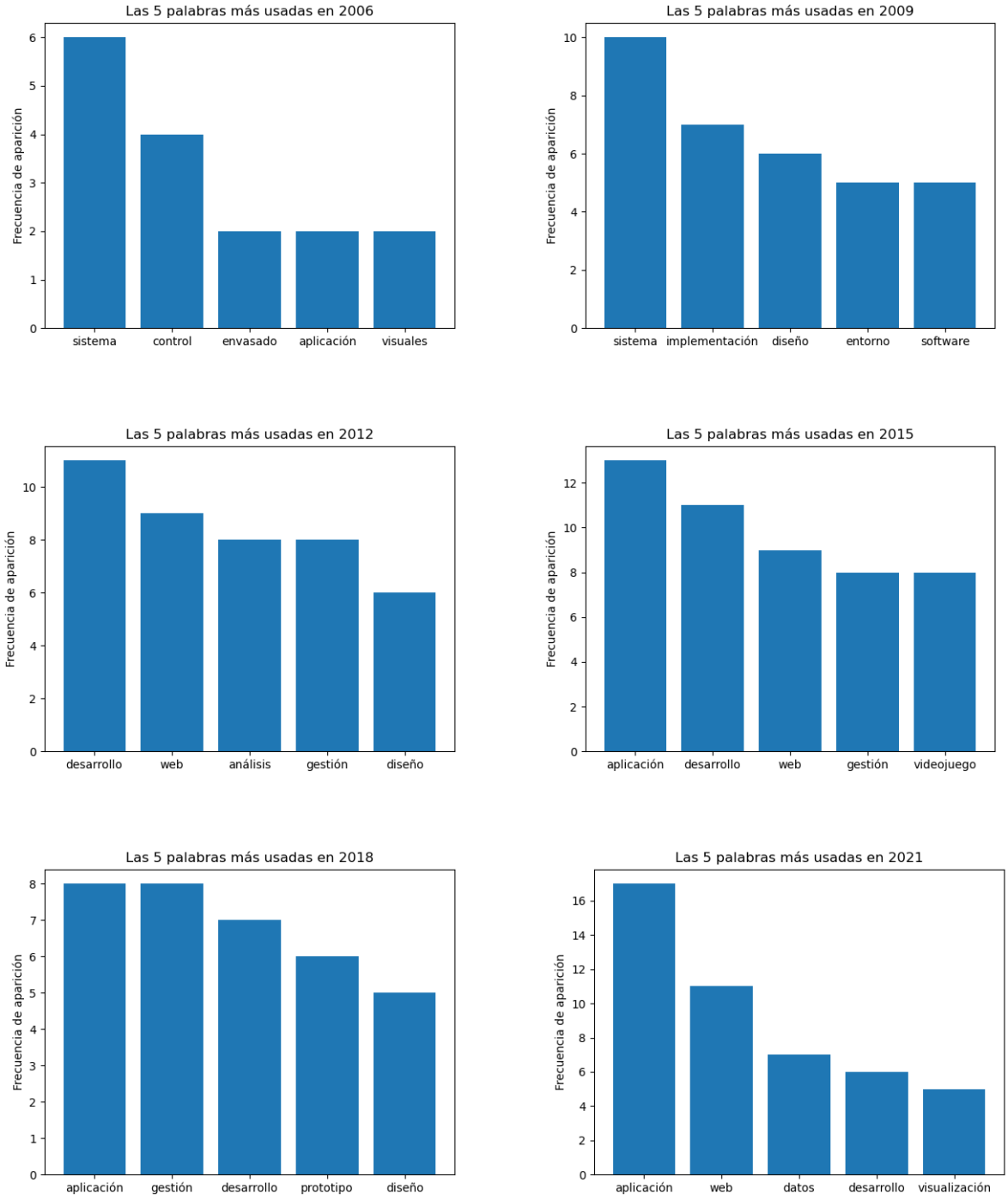


Figura 6.7: Selección de gráficos de barras con las palabras más usadas en los títulos de TFT

El uso de los gráficos de burbujas, como los que se encuentran en la figura 6.9, nos permitió ver desde una nueva perspectiva algo más intuitiva, sin necesidad de tener en cuenta cantidades numéricas, el uso de las palabras más populares comparadas entre sí. Este tipo de

representaciones pensamos que son más beneficiosas para un público general, ya que permite hacerse una idea de la situación sin la necesidad de conocer los detalles.

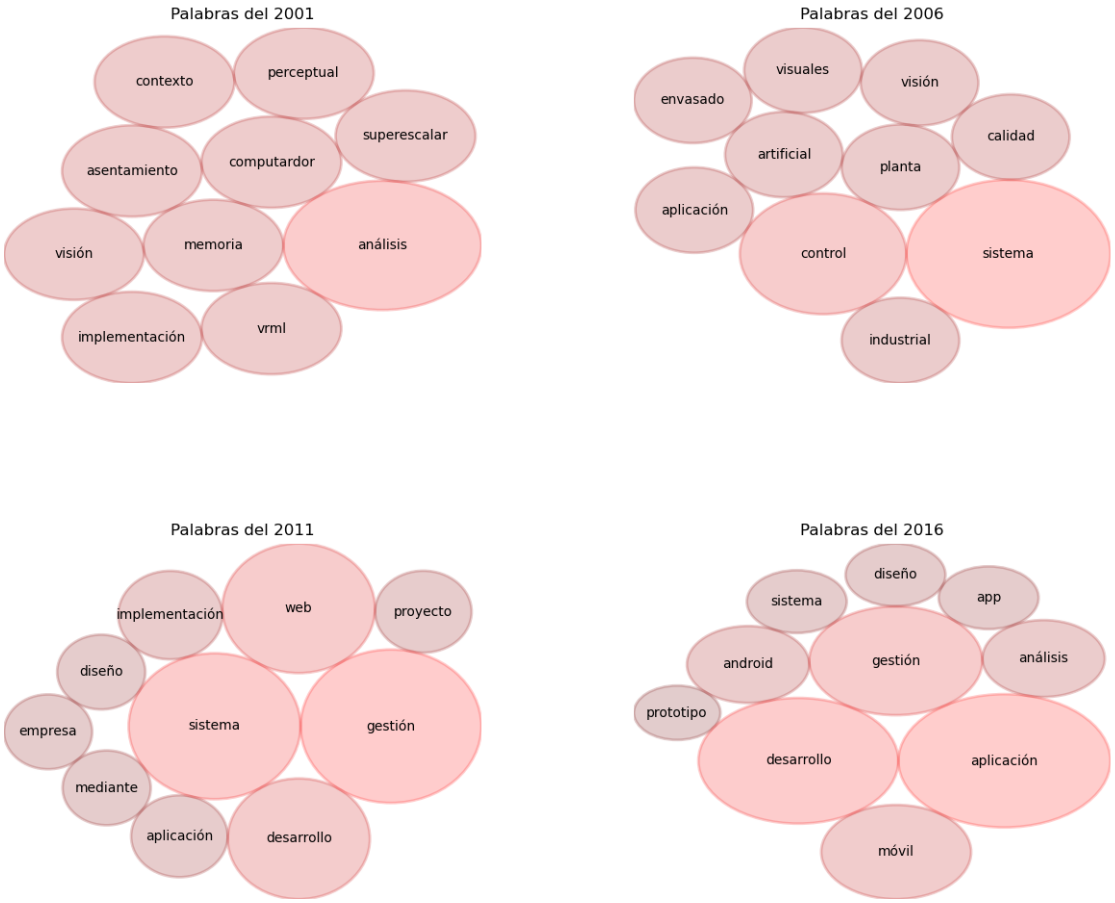


Figura 6.9: Selección de gráficos de burbujas con las palabras más usadas en los títulos de TFT

6.3.1. Palabras favoritas de los tutores

Completada esta primera fase, surgió una pregunta que pensamos que sería interesante estudiar. Cada año se presentan como vimos en el cuadro 6.2, especialmente desde el 2011, entre 50 y 60 trabajos de fin de título. Estos son tutorizados, generalemente, por el personal docente de la escuela, con la excepción de algún tutor de empresa, y esto presentaba una cuestión clara: ¿existe algún tipo de preferencia o relación por parte de los profesores con las palabras que se usan en los títulos de los proyectos que tutorizan?

La respuesta rápida es que sí. A lo largo de los años, el personal docente de la escuela ha supervisado la realización de, en ocasiones, decenas de proyectos distintos. A pesar de esto, una y otra vez las mismas palabras son usadas, ya que las temáticas sobre las que versan los TFT suelen compartir un patrón.

Esto puede ocurrir por una serie de razones. Una de ellas se da cuando los tutores particionan un gran proyecto como puede ser "La identificación de dorsales de corredores en eventos deportivos a través de tecnologías de identificación de imagen por computador" en una serie de módulos más pequeños con cierta independencia entre si. Otro motivo, es que cuando los alumnos solicitan a un profesor que supervise su TFT suelen escoger los profesores en base a la materia que ellos imparten. Con esto nos referimos a que si, digamos, se quiere presentar una aplicación como TFT, los estudiantes solicitaran a los profesores que les han impartido asignaturas relacionadas con la ingeniería del software o el desarrollo de aplicaciones que les tutoricen, de la misma forma si el trabajo de fin de título implica el desarrollo de, por ejemplo, bases de datos o sistemas inteligentes, por lo general, el docente escogido será aquel que les haya dado materias relacionadas con el tema. Este criterio a la hora de escoger tutor, lógica por una parte, hace que los nombres de los proyectos se parezcan, pues los elementos que caracterizan a todos estos trabajos suelen ser, sino iguales, al menos muy parecidos.

Como una pequeña muestra de los resultados obtenidos hemos incluido en la figura 6.12 algunas de las gráficas que se obtuvieron con respecto a las palabras de los tutores de este trabajo y algunos de los miembros originales del tribunal que lo iban a juzgar. Además, debemos añadir que como este estudio se hizo al tiempo que se organizaban y limpiaban los datos, algunos de los criterios menionados con anterioridad como la lematización o la eliminación de las palabras formadas por una sola letra no se habían añadido.

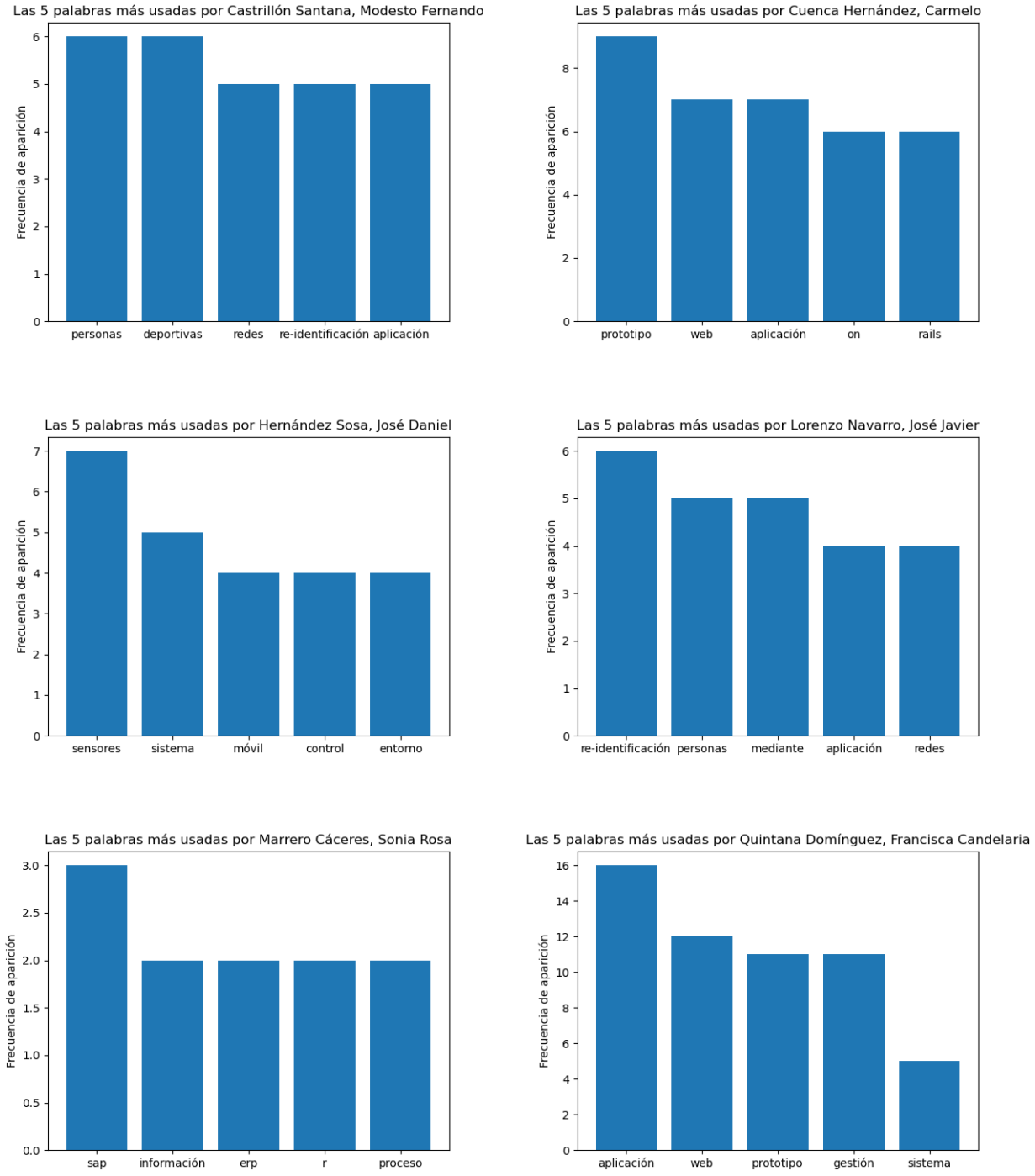


Figura 6.12: Palabras más empleadas por los tutores en los títulos de los TFT que supervisan a lo largo de los años

6.4. Entrenamiento Word2Vec

Ya con la limpieza y organización de los datos completada y con la práctica que habíamos obtenido haciendo los estudios previos, nos decidimos por incluir un nuevo elemento al proyecto que estábamos generando. La nueva adición fue la técnica para el procesamiento de datos Word2Vec, que detallamos en la sección 4.3.

Pensamos que podríamos sacar ventaja de que esta técnica emplease "word embedding", lo que aporta muchas posibilidades a la hora de trabajar con grandes conjuntos de textos, de ahí su popularidad.

6.4.1. Establecimiento de los parámetros y entrenamiento

Tras hacer varias pruebas y consultar con diversas fuentes a través de la web, establecimos los siguientes hiperparámetros:

- ✔ **min_count = 20**. Con este parámetro solo tendríamos en consideración palabras que aparecen más de 20 veces, pudiendo descartar fácilmente erratas y casos similares del corpus de textos.
- ✔ **window = 2**. Así hacemos que cuando el modelo vaya a predecir una palabra, este se encuentre a un máximo de 2 posiciones de la actual. El objetivo de este trabajo nunca fue predecir frases, pero si es interesante ver que palabras se suelen encontrar próximas entre sí.
- ✔ **sample = 6e-5**. Siempre existen una serie de palabras con una frecuencia de aparición muy alta, que pueden eclipsar al resto de vocablos. Por este motivo, de forma aleatoria este modelo reduce la aparición de estas palabras, con sample establecemos un "threshlod" para ver cuáles de estas palabras pueden ser sometidas a esto.
- ✔ **alpha = 0.03**. La tasa de aprendizaje inicial de Word2Vec.
- ✔ **min_alpha = 0.0007**. Cantidad máxima a la que puede ser reducidad la tasa de aprendizaje en el transcurso del entrenamiento.
- ✔ **negative = 20**. Cuando este valor es mayor que 0, especifica el número de palabras que el modelo considera que solo aportan ruido al conjunto de los datos que Word2Vec retirará a medida que progresa el entrenamiento. El valor recomendado es entre 5 y 20,

pudimos observar que cuando lo manteníamos en el valor más alto era cuando mejor se comportaba el modelo.

- ✔ **workers = cores - 1**. Este último parámetro nos sirve para especificar cuántos de los cores de nuestro PC serán usados al mismo tiempo cuando se utilice Word2Vec. En nuestro caso, previamente habíamos almacenado en la variable "cores" cuántos núcleos poseía nuestra máquina y se decidió asignarle al proceso todos menos uno, para acelerar lo máximo posible el trámite.

Los hiperparámetros fueron establecidos antes de comenzar el entrenamiento de la red. Entre el establecimiento de estas opciones y el propio entrenamiento, ejecutamos una función que recibe el nombre de "build_vocab". Este método, como su nombre parece indicar, comienza la construcción del vocabulario del modelo sin necesidad de entrenarlo. La realidad es que cuenta la frecuencia de aparición de las palabras y organiza listas para así poder aplicar los hiperparámetros antes indicados con mayor eficiencia. Para esto, hay que pasarle como parámetro el conjunto de frases que van a ser usadas en el entenamiento.

6.4.2. Pruebas y comprobaciones posteriores

Una vez completados todos estos pasos previos, procedimos a la ejecución de la función "train" del modelo Word2Vec. El proceso de entrenamiento de la red constó de treinta épocas, cantidad de épocas que tras varias pruebas se consideró arrojaba unos resultados aceptables.

Con el modelo ya entrenado pudimos hacer una serie de experimentos con las funciones que ya incorpora la biblioteca para ver si todo el proceso estaba dando resultados comprensibles, lógicos y aceptables. Una de las funciones que incorpora es "wv.most_similar()", con ella podemos pasarle una palabra cualquiera que se encuentre dentro del vocabulario y nos devolverá aquellas que son usadas contextualmente de la forma más parecida. En nuestro caso le pedimos que nos buscase las palabras más similares a "web", los resultados que obtuvimos fueron: "aplicación", "sitio", "portal", "página", "móvil" y "servidor". También se obtuvieron otras como "través" y "dinámico" que guardan un parecido, que diríamos al menos en primera instancia, menor, pero esto es normal puesto que el procedimiento te devuelve las 10 más parecidas.

Esta no fue la única de las funciones con las que probamos los resultados. Otra de

ellas es la denominada "wv.similarity()", a la que le pasas dos palabras y te dice lo próximas que están entre sí, concretamente devuelve un número entre 0 y 1, cuanto más próximo este a la unidad más parecidas serán. Por ejemplo, probamos con las palabras "web" y "aplicación" cuya similitud es de 0.742, o buscando vocablos que estuviesen más distantes entre sí escogimos "web" y "tener" que comparten un parecido de 0.21.

Adicionalmente, se probó la función "wv.doesnt_match()" en la que se debe pasar como parámetro una lista de palabras del vocabulario de las que luego te devuelve la que tenga menor relación con el resto. Uno de los experimentos más evidentes que hicimos fue cuando le pasamos en la lista "tener", "web" y "servicio". Evidentemente la palabra que resultó ser la menos pareida fue "tener".

Finalmente, pusimos a prueba el modelo con el ejemplo que se usa por excelencia para hablar de Word2Vec: "Que palabra es a Mujer lo mismo que Hombre es a Rey", cuya respuesta evidentemente es "Reina". Solo que nosotros le dimos una pequeña vuelta y usamos un ejemplo más relacionado con nuestro campo de estudio: "Que palabra es a usuario lo mismo que Servicio es a Web". Una consulta de este estilo se formula de la siguiente forma: "wv.most_similar(positive=['usuario', 'web'], negative=['servicio'], topn=3)". Los resultados fueron, lo decimos en plural porque como se puede ver solicitamos las tres respuestas más parecidas, "página" (página de usuario), "portal" (portal de usuario) y "visualizar" (visualizar/visualización de usarios). Como se puede ver son todos resultados plausibles propios de expresiones que se pueden localizar en muchos TFT relacionados con el desarrollo de una aplicación, red o base de datos.

6.5. Clustering

El uso del "word embedding" que caracteriza a Word2Vec inspiró esta parte del proyecto. Como se puede intuir, el criterio que usa el modelo para establecer como se relaciona un vocablo con el resto (si son parecidos, si no lo son, etc) es la posición de estos puntos que determinan los vectores que representan las palabras, con respecto del resto. Entonces, al encontramos con un conjunto de puntos con estas características, nos preguntamos si podíamos juntarlos en una serie de grupos distintos.

En específico quisimos ver si podíamos extrapolar este pensamiento y, en vez de realizar los agrupamientos solo con palabras, intentar establecer una serie de grupos con los títulos. Para esto primero tuvimos que obtener lo que llamamos el "Vector Medio" de una de las

frases. Este véctor es la media aritmética de todos los vectores de las palabras del título que pertenezcan al vocabulario del modelo.

Una vez disponemos de los vectores medios solo tenemos que aplicar el algoritmo Kmeans, comentado en la sección 4.5. Este algoritmo, que conformará todos los grupos que le especifiquemos como K, nos dejó con la pregunta de qué K deberíamos escoger. Como el conjunto de vectores es demasiado grande como para ir comprobando una a una tantas K como elementos hay (recordemos que en el caso de los títulos estamos trabajando con el total de las publicaciones, que son 929), necesitábamos un método que nos permitiese dividir al conjunto de nombres de los proyectos en un número de grupos aceptables.

6.5.1. Método del codo

Esta forma de escoger la cantidad de grupos consiste en representar gráficamente la inercia que nos proporciona Kmeans para tomar la decisión. La inercia es la suma de la distancia al cuadrado de cada una de las muestras con respecto a su centroide más próximo, por supuesto, cuantas mayor sea la K, mayor será el número de centroides y, por tanto, menor será la inercia del conjunto.

La representación de esta medida genera una curva que se asemeja a un codo, de ahí su nombre. La idea es escoger aquella K que en la representación hace que la gráfica comience a aplanarse.

El mayor problema que presenta esta forma de escoger la K, es que según el criterio del observador se puede decidir que la curva se aplana en distintos momentos, especialmente si en la representación no se aprecia con mucha evidencia. Este, desgraciadamente, fue nuestro caso como se puede apreciar en la figura 6.13, en la que se podría argumentar que la curva se comienza a aplanar en cualquier punto entre una K con valor 5 o 12, o incluso más adelante.

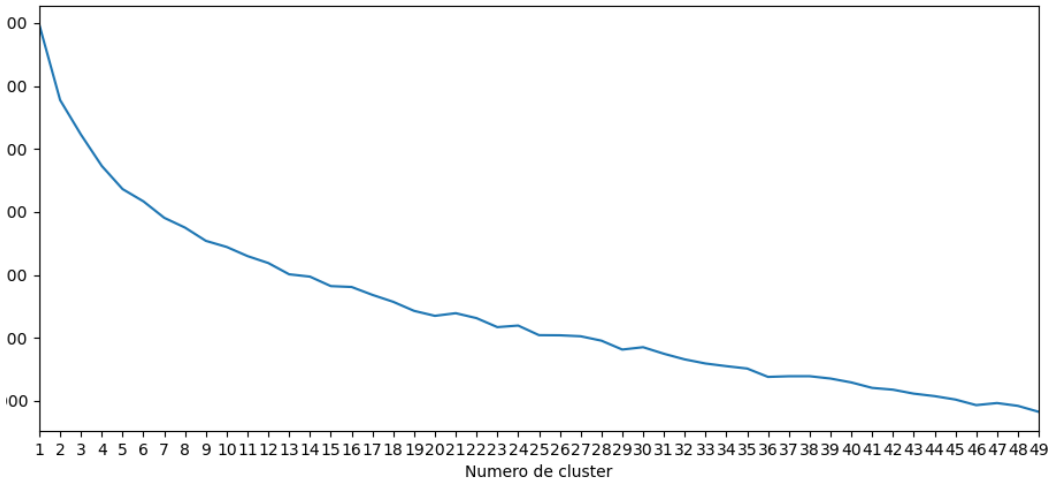


Figura 6.13: Método del Codo aplicado a nuestro dataset

6.5.2. Silhouette score

Aunque el primero de los métodos acotó hasta cierto punto las posibilidades a la hora de escoger una K, no obtuvimos un resultado conluyente, por lo que quisimos aplicar esta nueva estrategia. La aproximación escogida fue el "Silhouette Score", que aspira a medir como de densos y separados están los agrupamientos de puntos para una determinada K.

La forma habitual de poner en práctica esta técnica es usando los llamados "Silhouette Plots", sin embargo nosotros hemos optado por emplear directamente la "Silhouette Score" que representa la media de todas las puntuaciones para los disitintos cumulos de datos generados para una K en concreto.

Al representar estas "score" en una gráfica, optaremos por aquella que tenga una puntuación mayor. Como se puede ver en la figura 6.14 esta es 6, el 2 fue descartado porque las temáticas dentro de los grupos con esta opción eran demasiado variadas e inconexas entre si.

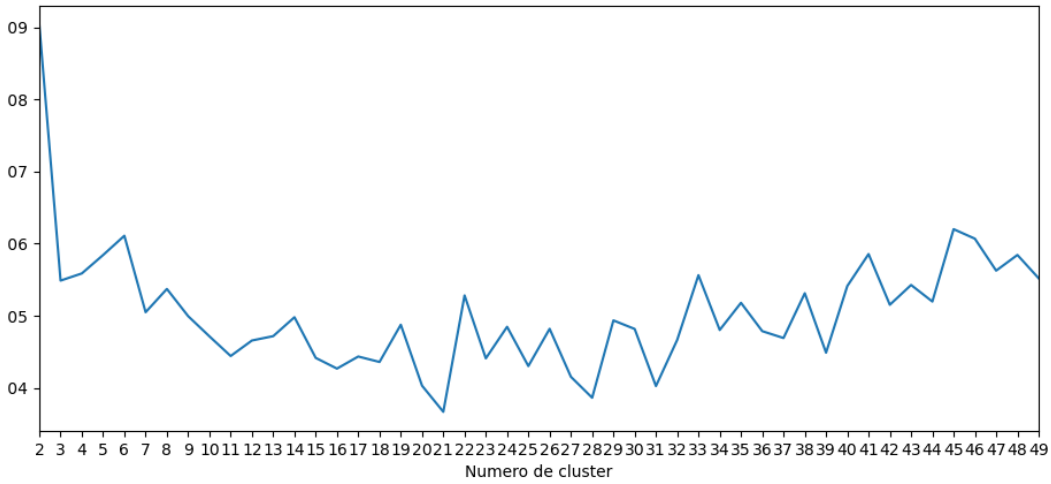


Figura 6.14: Silhouette Score aplicado a nuestro dataset

6.5.3. Decisión final

Inicialmente, optamos por fijar la K en 6, según nos indicaba el método de la silueta. Este resultado ya nos permitía apreciar claros patrones dentro de las agrupaciones de los títulos, sin embargo las temáticas seguían siendo bastante amplias. Por este motivo se decidió aumentar la K hasta 8, un resultado subóptimo dentro de la figura 6.14.

El primero de los grupos es uno con aparentemente una mayor variedad, sin embargo si uno lo observa con detenimiento puede apreciar como se empiezan a notar la aparición de algunas palabras como “servicio” o “sistema”, además de temáticas comunes. Dentro de estos temas tenemos que destacar que en su mayoría son aplicaciones, servicios web o análisis de organizaciones. Concretamente hablamos de empresas, dentro de este amplio grupo resaltamos un gran contenido relacionado con el turismo, la restauración y los transportes. Adicionalmente, debemos mencionar la aparición de muchos proyectos relacionados con el campo de la sanidad.

El segundo grupo es mucho más sencillo de delimitar. Aquí encontramos que prácticamente la totalidad de los títulos menciona el desarrollo de un videojuego o una consola. Las tecnologías, métodos y objetivos que detallan los nombres de los proyectos son variopintos, pero la temática principal es el mundo de los videojuegos.

La siguiente agrupación se encuentra muy relacionada con la mención de computación. Aquí podemos detectar un gran número de trabajos relacionados con los sistemas inteligentes,

el desarrollo de robots y bots de distinta clase, la visión por computador y el tratamiento de imágenes en general y, por último, proyectos altamente relacionados con NLP.

El cuarto conjunto se centra en proyectos claramente marcados por el mundo académico y la investigación científica. Gran parte de ellos directamente incluyen las palabras “ULPGC”, “instituto”, “enseñanza”, “alumno”, “profesorado”, “investigación”, “universidad”, “científico”, “colegio”, etc.

El grupo consecutivo a este último está profundamente enlazado con las redes. En primer lugar, el vocablo que aparece con una mayor fercuencia es “web”, además podemos apreciar gran cantidad de trabajos finales relacionados con servidores y servicios propios de servidores, conexiones o sistemas online. No podemos olvidarnos de mencionar una tendencia clara a la aparición de proyectos sobre móviles o tecnologías móviles.

El sexto conjunto es el más pequeño de todos, en este solo aparecen TFT basados en NLP, pero con la característica común de que todos son conversores. En especial destacamos la formula usada numerosamente como título de “Conversor de números a texto en (idioma)”.

El siguiente es un grupo bastante especial. Los títulos que se observan son todos de un tamaño muy pequeño, en su mayoría compuestos por una sola palabra. Esta palabra suele ser una palabra inventada que representa el nombre propio con el que el alumno denominó a la aplicación que ha desarrollado. Asimismo, podemos ver títulos algo mayores, pero estos o están completamente en inglés o poseen uno de estos nombres ya mencionados.

La última agrupación es la más complicada de clasificar. Esta está formada por una combinación de muchos de los temas que ya hemos mencionado sin que destaque ninguno excesivamente por encima del resto. Anecdóticamente podemos decir que hay una especial aparición de palabras como “arquitectura” y “plataforma”.

Capítulo 7

Conclusiones y trabajo futuro

A lo largo de este trabajo hemos logrado establecer que el conjunto de trabajos finales entregados en los últimos 30 años en la Escuela de Ingeniería Informática ofrece un amplio abanico de posibilidades a la hora de realizar estudios relacionados con el NLP y el Aprendizaje Profundo. Este corpus de textos puede servir como base para conocer cuales son las tendencias e intereses de los alumnos a lo largo de los años, por no mencionar la lectura que se puede realizar del desarrollo tecnológico en general durante el tiempo.

Esta segunda vida que se le ha dado a los proyectos demuestra que el uso de tecnologías como el NLP, nos permite entender y localizar detalles que de otra forma habrían pasado desapercibidos. Las tendencias de los tutores a la hora de escoger las palabras de los trabajos que supervisan o la creación de agrupamientos con claras característica comunes dentro del conjunto de todos los títulos, son solo unos pocos ejemplos de las muchas posibilidades que encierran estos textos.

- Trabajo Futuro -

- ✔ Encontrar la forma de convertir los PDF de la titulación de Ingeniero en Informática a texto plano para incorporarlos al corpus de datos.
- ✔ Estudiar las palabras más populares dentro de cada uno de los grupos usando técnicas de minería de datos.
- ✔ Estudiar la posibilidad de usar el corpus para entrenar a un sistema inteligente que sea capaz de generar proto-TFG.
- ✔ Realizar un análisis en mayor profundidad de la aparición, uso y desaparición de términos y temáticas a lo largo del tiempo.
- ✔ Crear una representación gráfica de los términos al estilo de Jer Thorp.
- ✔ Realizar un estudio parecido con los trabajos de fin de grado de otras facultades.

Bibliografía

[Gen] Manual de referencia para gensim. https://radimrehurek.com/gensim/apiref.html.

[Mat] Manual de referencia para matplotlib. https://matplotlib.org/stable/api/index.html.

[NLT] Manual de referencia para nltk. https://www.nltk.org/api/nltk.html.

[Num] Manual de referencia para numpy. https://numpy.org/doc/stable/reference/.

[Pan] Manual de referencia para pandas. https://pandas.pydata.org/docs/reference/index.html.

[Sci] Manual de referencia para scikit-learn. https://scikit-learn.org/stable/modules/classes.html.

[Spa] Manual de referencia para spacy. https://spacy.io/api.

[mul] Página de referencia del módulo multiprocessing. https://docs.python.org/es/3.9/library/multiprocessing.html.

[os] Página de referencia del módulo os. https://docs.python.org/es/3.10/library/os.html.

[re] Página de referencia del módulo re. https://docs.python.org/3/library/re.html.

[Cir] Repositorio github de donde se extrajo la librería. https://github.com/elmotec/circlify.

[Wor] Word2vec explained. https://towardsdatascience.com/word2vec-explained-49c52b4ccb71.

@book JerThorp, Title = Living in Data, Author = Jer Thorp, Journal = Machine Vision and Applications, Year = 2021

@article NLP, title = Advances in Natural Language Processing - A Survey of Current Research Trends, Development Tools and Industry Applications, author = Krishna Prakash Kalyanathaya, D. Akila and P. Rajesh, howpublished = https://shorturl.at/afmW2

@misc NLP, title = What Is Natural Language Processing?, howpublished = https://shorturl.at/BCE79

@misc NLP, title = NLP History, howpublished = https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/nlp/overview_history.html

@misc re, title = Página de referencia del módulo re, howpublished = https://docs.python.org/3/library/re.html

@misc multiprocessing, title = Página de referencia del módulo multiprocessing, howpublished = https://docs.python.org/es/3.9/library/multiprocessing.html

@misc os, title = Página de referencia del módulo os, howpublished = https://docs.python.org/es/3.10/library/os.html

@misc NumPy, title = Manual de referencia para NumPy, howpublished = https://numpy.org/doc/stable/reference/

@misc Pandas, title = Manual de referencia para Pandas, howpublished = https://pandas.pydata.org/docs/reference/index.html

@misc NLTK, title = Manual de referencia para NLTK, howpublished = https://www.nltk.org/api/nltk.html

@misc Spacy, title = Manual de referencia para Spacy, howpublished = https://spacy.io/api

@misc Scikit-Learn, title = Manual de referencia para Scikit-Learn, howpublished = https://scikit-learn.org/stable/modules/classes.html

@misc Gensim, title = Manual de referencia para Gensim, howpublished = https://radimrehurek.com/gensim/apiref.html

@misc Matplotlib, title = Manual de referencia para Matplotlib, howpublished = https://matplotlib.org/stable/api/index.html

@misc Circlify, title = Repositorio GitHub de donde se extrajo la librería, howpublished = https://github.com/elmotec/circlify

@misc EditDistance, title = Implementación de la distancia edit con Tabulation, howpublished = https://github.com/bdebo236/edit-distance

@misc Stopwords, title = Text pre-processing: Stop words removal using different libraries, howpublished = https://shorturl.at/apXY5

@misc Stopwords, title = Lista de la RAE de stopwords, howpublished = https://apps2.rae.es/CORPES2/estad/1000_elementos.txt

@misc Lemmatization, title = Stemming vs. Lemmatization in NLP, howpublished = https://towardsdatascience.com/stemming-vs-lemmatization-in-nlp-dea008600a0

@misc Word2VecExplanation, title = Word2Vec Explained, howpublished = https://towardsdatascience.com/word2vec-explained-49c52b4ccb71

@misc Word2Vec, title = Word2Vec, howpublished = https://es.wikipedia.org/wiki/Word2vec

@misc LevenshteinDistance, title = Text Similarity w/ Levenshtein Distance in Python, howpublished = https://shorturl.at/BG089

@misc LevenshteinDistance, title = Levenshtein Distance, howpublished = https://en.wikipedia.org/wiki/Levenshtein_distance

@misc KMeans, title = K-Means Explained, howpublished = https://towardsdatascience.com/k-means-explained-10349949bd10

@misc KMeans, title = Understanding K-means Clustering in Machine Learning, howpublished = https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e6

@misc Clusters, title = How Many Clusters?, howpublished = https://towardsdatascience.com/how-many-clusters-6b3f220f0ef5

@misc ElbowMethod, title = Explicación e implementación del método del codo, howpublished = https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/

@misc Silhouette, title = KMeans Silhouette Score Explained With Python Example, howpublished = https://dzone.com/articles/kmeans-silhouette-score-explained-with-python-exam

@misc Silhouette, title = Silhouette (clustering), howpublished = https://es.wikipedia.org/wiki/Silhouette_(clustering)