UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Creación de un corpus de artículos de prensa y generación automática de resúmenes

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Fernando Alcina Sanchis
Cotutor: Encarnación Segarra Soriano
Cotutor: Lluís Felip Hurtado Oliver

Curso 2018-2019

Resum

La generació automàtica de resums és un camp molt atractiu i vigent dins de l'àrea del processament del llenguatge natural. Per aquesta raó, en aquest treball, s'ha decidit analitzar i comparar distintes tècniques per a la generació automàtica de resums, tant basades en xarxes neuronals com basades en algoritmes clàssics. Per a l'avaluació d'aquestes tècniques es fan servir mètriques sintàctiques utilitzades en la major part d'estudis d'aquest camp i mètriques semàntiques propostes en aquest treball. Mitjançant aquest estudi, es podran observar les diferències que existeixen entre les dues avaluacions comentades anteriorment i la millora que aporta la mesura semàntica proposta.

Per a l'ús d'aproximacions de resum basades en xarxes neuronals i per a l'avaluació de tots els sistemes, es requereix un corpus de documents i resums. Per aquest motiu, mitjançant un procés de *crawling*, s'ha elaborat un corpus de notícies que provenen de distints llocs web de premsa digital i per als idiomes de castellà i català.

Finalment, el treball conté un extens anàlisi dels resultats experimentals obtinguts en el corpus. En aquest es podran observar les diferències que existeixen entre les notícies que contenen ambdós corpus emprant distintes mesures. A més, es compararan els corpus generats amb altres corpus que formen part del estat de l'art.

Paraules clau: corpus d'articles de premsa, crawling, resum automàtic, xarxes neuronals, embeddings

Resumen

La generación automática de resúmenes es un campo muy atractivo y vigente dentro del área del procesamiento del lenguaje natural. Por esta razón, en este trabajo, se han decidido analizar y comparar distintas técnicas para la generación automática de resúmenes tanto basadas en redes neuronales como basadas en algoritmos clásicos. Para la evaluación de estas técnicas, se utilizan métricas sintácticas usadas en la mayoría de estudios de este campo y métricas semánticas propuestas en este mismo trabajo. Mediante este estudio, se podrán observar las diferencias que existen entre las dos evaluaciones comentadas anteriormente y la mejora que aporta la medida semántica propuesta.

Para la utilización de aproximaciones de resumen basadas en redes neuronales y para la evaluación de todos los sistemas, se requiere un corpus de documentos y resúmenes. Por este motivo, mediante un proceso de *crawling*, se ha elaborado un corpus de noticias que provienen de distintos sitios web de prensa digital y para los idiomas de castellano y catalán.

Finalmente, el trabajo contiene un extenso análisis de los resultados experimentales obtenidos en los corpus. En este, se podrán observar las diferencias que existen entre las noticias que contienen ambos corpus utilizando para ello distintas medidas. Además, se compararán los corpus generados con otros corpus que forman parte del estado del arte.

Palabras clave: corpus de artículos de prensa, crawling, resumen automático, redes neuronales, embeddings

Abstract

Automatic summary generation is a current and very attractive field in the area of natural language processing. For this reason, in this work, it has been decided to analyse and compare different techniques for automatic abstract generation both based on neural net-work and based on classical algorithms. For the evaluation of these techniques, syntactic metrics used in most studies of this field and semantic metrics proposed in this work have been used. Through this study, differences between the two evaluations discussed above can be seen. Moreover, improvements introduces by the semantic measure proposal are also presented.

For the use of summary approaches based on neural networks and for the evaluation of all systems, a corpus of documents and summaries is required. For this reason, through a crawling process, a corpus of news that come from different digital press websites has been prepared in both Spanish and Catalan languages.

Finally, the work contains an extensive analysis of the experimental results obtained with the corpus. In it, the existing differences between the news contained in both corpus can be observed. At the same time, a comparison of the different measures has been included. In addition, the generated corpus will be compared with other corpus that take part of the state of the art.

Key words: news corpus creation , crawling, automatic summarization, neural network, embeddings

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

A mis tutores Lluís y Encarna por haberme aconsejado y corregido durante todo este tiempo.

A mi familia y amigos por motivarme y ayudarme durante todo el desarrollo del trabajo.

A Jose Ángel por toda la ayuda prestada en la parte de la generación del corpus y por poder utilizar el sistema automático de resumen que forma parte de su tesis doctoral.

«Aequam memento rebus in arduis servare mentem» - Horacio

CAPÍTULO 1
Introducción

En este trabajo, se compararán distintas técnicas para la generación automática de resúmenes de documentos y se llevará a cabo una evaluación de las mismas. Algunas de las técnicas comparadas están basadas en modelos de aprendizaje supervisado y estos a su vez, necesitan una gran cantidad de datos para su entrenamiento; es por ello, que se requiere un corpus grande de documentos para llevar a cabo el análisis comentado. Este tipo de corpus existe para el inglés, pero sin embargo, no para el catalán y el castellano. Por esta razón, el trabajo consta de dos partes; en la primera, se han creado dos corpus, uno para catalán y otro para castellano, de documentos lo suficientemente grandes como para entrenar modelos supervisados y en la segunda se ha llevado a cabo un análisis de las distintas técnicas de resumen automático elegidas.

Dado que el proceso de la generación del corpus podía considerarse un trabajo por si solo, tanto Vicent Ahuir Esteve en [1] como el presente trabajo compartían la necesidad de la generación de un corpus y ambos teníamos un tutor común del TFG, se decidió desarrollar conjuntamente esta parte del trabajo.

El resumen de un documento es otro documento más corto que el original pero que mantiene el significado del mismo. Este resumen ayuda a obtener los conceptos e ideas que contiene un escrito con la simple lectura del resumen. A pesar de que el campo de los resúmenes automáticos ha sido estudiado durante varias décadas, está creciendo el interés por este en los últimos años. Este creciente interés es debido fundamentalmente al notable aumento del contenido en la red y a la necesidad de los usuarios de averiguar si un texto es de su interés sin tener que leer el documento completo. Dada la enorme cantidad de documentos en la red, es inconcebible la realización de un resumen manual para cada uno de ellos. Gracias a que la tecnología está avanzando en los últimos años y que esta nos puede ayudar en la elaboración de los resúmenes, surgió la idea de aplicar la generación automática de resúmenes a los documentos de Internet. A pesar de que pueda parecer que la idea de los resúmenes automáticos sea parcialmente novedosa, ya aparecieron muestras de interés sobre este tema en algunas bibliotecas de EEUU en los años 60. En estas bibliotecas, se quería indexar digitalmente el contenido de los libros para así facilitar su búsqueda; pero como existían restricciones de almacenamiento, se requerían pequeños resúmenes de los textos.

Una vez explicado el concepto de resumen y a que se debe el interés general de los resúmenes automáticos, se pasarán a comentar las distintas variantes existentes de resumen, mostradas en la *Figura 1.1* y extraídas de [13], según el tipo de documentos de entrada, documentos de salida y el propósito por el cual está creado el resumen.

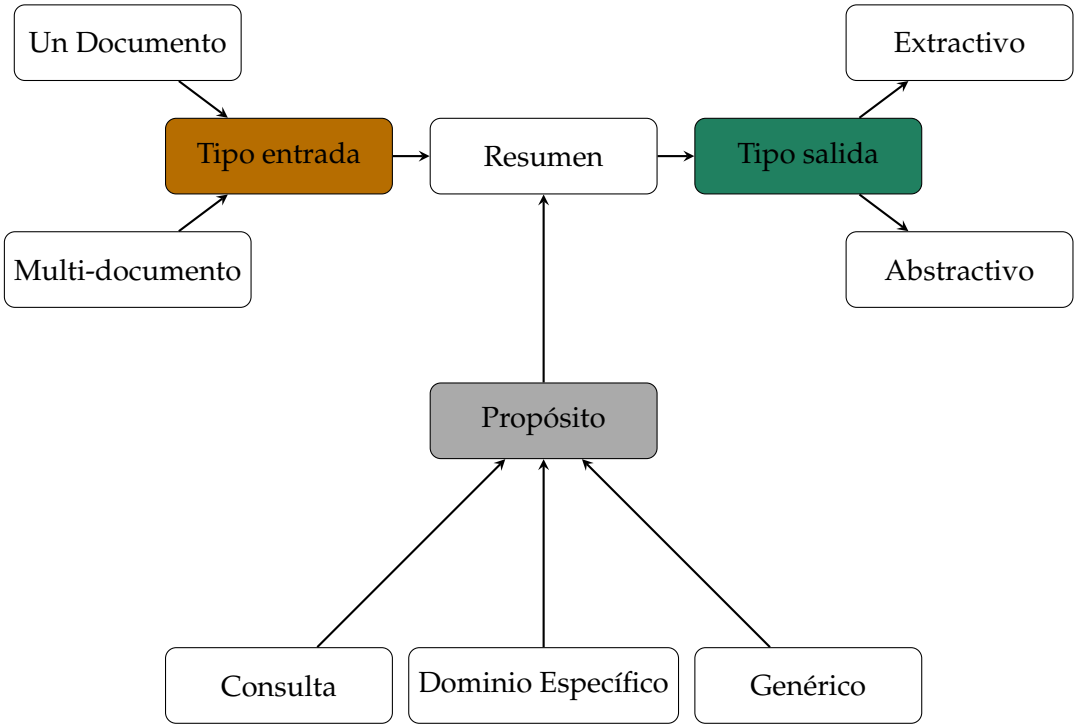


Figura 1.1: Tipos de métodos de resumen

Basados en el tipo de entrada:

- **Un documento**: El resumen generado utilizando está técnica parte de un solo documento origen.
- **Multi-documento**: El resumen es generado a partir de un conjunto de documentos no necesariamente homogéneo; es decir, no es necesario que los documentos tengan unas características comunes. Este tipo de resumen se utiliza para sintetizar varios textos sobre un mismo tema.

Basados en el tipo de salida:

- **Extractivo**: El resumen se crea a partir de fragmentos extraídos directamente del texto original sin añadir vocabulario nuevo. Estos fragmentos pueden ser frases enteras o fragmentos de frases.
- **Abstractivo**: El resumen se crea a partir de una secuencia de oraciones que intentan recoger la información mas importante del documento; para este fin no utiliza las frases del texto sino otras generadas por el sistema. Gracias a esto, se genera un resumen más similar al que realizaría un humano.

Basados en el propósito:

- **Genérico**: En un resumen con esta finalidad, no se hace ninguna asunción previa del tipo de documento que se va a resumir. El resumen, solamente proporciona la información que más representa el texto sin importar la temática de este.
- **Dominio específico**: Se crea un resumen para un dominio previamente conocido; para este fin, se le añaden una serie de conocimientos sobre el tema que va a tratar.

- **Consulta**: El resumen creado solo incluye información acerca de una pregunta previamente generada.

La tipología de resumen en la cual se basará el presente trabajo será la generación de resúmenes con un tipo de salida extractiva a partir de un solo documento de entrada y con un propósito genérico. Para este objetivo, se utilizarán diferentes técnicas y algoritmos que se comentarán en el *Capítulo 3*. Además, se presentará una base de datos no relacional –Mongodb– donde se guardarán los datos que utilizaremos para la generación automática de resúmenes; es decir, los corpus creados. Finalmente, se evaluaran los distintos modelos a comparar en base a unas métricas. Para este fin, se utilizará tanto el *Rouge* como un evaluador semántico de resúmenes implementado en el presente trabajo. Ambos, nos permitirán comparar el resumen original del texto con el generado y, a partir de los resultados, nos permitirá conocer qué técnica de las utilizadas para la generación de resúmenes ha obtenido mejores prestaciones para cada corpus.

1.1 Motivación

La principal motivación que ha llevado al autor del presente trabajo en la elección del tema de los resúmenes automáticos, ha sido la capacidad de aprendizaje y superación que se podía adquirir con un trabajo de semejante naturaleza. Desde el día que los tutores propusieron el trabajo, se percibió una excelente temática para la realización de un trabajo final de grado; influenciado principalmente por la mejora que aportan este tipo de técnicas a la sociedad. Observamos día tras día cómo el campo de la inteligencia artificial y el aprendizaje automático tienen una importancia mayúscula en la vida de las personas. Además, mediante la generación automática de resúmenes podemos ayudar a las usuarios que buscan información en la red permitiéndoles conocer las ideas fundamentales de un texto sin la necesidad de leérselo completo. Otro ejemplo donde la generación automática de resúmenes es de gran ayuda sería en el mundo de la escritura; ya que los autores podrían generar distintos tipos de resúmenes dependiendo de ciertos criterios para intentar agradar a más gente.

La generación automática de resúmenes es una área que está creciendo en los últimos años debido mayormente a la continua incorporación de contenido en la red. Además, cada vez son más las personas que tienen acceso a esta red gracias a los grandes avances que están sufriendo las infraestructuras relacionadas con la misma. Asimismo, cuando se quiere buscar documentación de cualquier tipo, se busca en Internet y ¿que mejor que la generación automática de resúmenes para hacer más factible esta tarea?

Claro está que para poder realizar esta investigación se necesita de algunos conceptos adicionales a los que se han estudiado a lo largo de la carrera; como por ejemplo solucionar los problemas que pueden surgir en la automatización de sistemas sobre datos no homogéneos, la utilización de bases de datos no relacionales o la generación de código en lenguajes de programación no estudiados en clase. No obstante, se han conseguido revertir todos los problemas del trabajo y convertirlos en oportunidades de aprendizaje aumentando así la motivación de este, más si cabe, conforme va avanzando el proyecto. Algunas de las asignaturas que sí que han ayudado en la elaboración de este trabajo han sido *Aprendizaje Automático* –para entender los conceptos relacionados con las aproximaciones utilizadas en la generación de resúmenes–, *Bases de Datos y sistemas de información* –con los conceptos de la base de datos– y *Sistemas de almacenamiento y recuperación de información* –para entender las métricas utilizadas en la evaluación de los resúmenes–.

Además de lo comentado en los párrafos precedentes, otra motivación que ha llevado a la elección de este trabajo ha sido las habilidades personales que podía proporcionar el proyecto. Sí, estamos en un momento de cambio, ya se terminan los estudios y tenemos que plantearnos qué habilidades deseamos tener y qué campos de la informática nos gustaría manejar cuando nos incorporemos al mercado laboral.

1.2 Objectivos

El objetivo principal que aborda el presente trabajo es el de analizar sistemas de generación automática de resúmenes basados en distintas técnicas y posteriormente seleccionar el sistema que mejor resultados obtenga para cada corpus. Claro está, que para poder hacer este análisis, se necesita un corpus previamente generado. Es por ello que este trabajo se dividirá en dos partes; primero una parte donde se abordará el proceso para la creación del corpus y posteriormente, otra parte donde se realizará el análisis y evaluación de los sistemas de resumen automático. Por consiguiente, se ha decidido dividir los objetivos en las dos partes comentadas anteriormente.

1.2.1. Creación del corpus

Como paso previo a la generación del corpus, se tienen que seleccionar una serie de características que deberá cumplir el mismo. En primer lugar se tiene que elegir la temática de los datos que se analizarán; en este trabajo se ha decidido estudiar noticias publicadas en prensa digital sobre diferentes temas como política, nacional, internacional, deporte, economía, cultura, sociedad, ciencia, opinión, etc. Una vez seleccionada la temática del corpus, se seleccionarán las características que deberá tener el corpus a partir de las decisiones que se tomen respecto a los campos necesarios, a los periódicos que contendrá, al volumen de datos requeridos para la correcta realización del estudio, etc. Además, dado que el autor del trabajo habla los dos idiomas reconocidos en la comunidad autónoma en la que vive –Comunidad Valenciana– y que hay pocos estudios sobre el tema que se desempeña en el presente trabajo respecto al catalán, se han decidido crear dos corpus –uno para castellano y otro para catalán–.

Por lo tanto, el objetivo principal de la primera parte del trabajo es la generación de dos corpus que cumplan las siguientes características:

- **Corpus ampliable:** Los corpus implementados tienen que ser fácilmente adaptables a nuevas incorporaciones tanto de noticias como de periódicos. Es por ello que la implementación de la base de datos ha de ser independiente de cualquier fuente.
- **Corpus con variedad:** Los corpus deben contener noticias de distintas fuentes, distintas temáticas, distintos autores, etc. Esta variedad le proporcionará a los corpus una sensación de completitud.
- **Corpus que se adapte a las necesidades:** Estos corpus deben incorporar información adicional a la requerida por los dos trabajos en los que se utilizará; permitiendo así usarlos en posteriores estudios relacionados con el Procesamiento del Lenguaje Natural –PLN–. Además, todos los campos incluidos en los corpus tienen que estar normalizados. Esta normalización facilita la automatización de las tareas en las posteriores etapas.
- **Código genérico:** El código implementado para la creación del corpus tiene que ser lo más genérico posible. Con ello, se permitirá añadir nuevo contenido a los

corpus de una forma universal e independiente de las características que presente la fuente.

Además, finalmente, se tienen que valorar los dos corpus obtenidos. Esta valoración, permitirá conocer si se han cumplido los requisitos planteados para la tarea de la creación del corpus.

1.2.2. Generación automática de resúmenes

Dado que existen multitud de técnicas diferentes, se debe elegir una pequeña porción de estas para el desempeño del trabajo. Esta elección incluirá tanto técnicas basadas en redes neuronales como basadas en algoritmos clásicos de resumen.

La generación de resúmenes es un objetivo por si solo así que se ha decidido dividirlo en unos sub-objetivos más fácilmente alcanzables, a sabiendas que una vez cumplidos todos los sub-objetivos, se habrá cumplido el objetivo inicial.

- **Análisis de técnicas:** Se tienen que analizar y posteriormente seleccionar las técnicas que se utilizarán para la generación de resúmenes.
- **Preparación de los modelos:** Dependiendo de la técnica seleccionada, se requerirá o no el aprendizaje de modelos. Es por ello, que se deben seleccionar y ajustar los modelos según las características de los corpus que se han elegido.
- **Analizar resultados:** Se tienen que analizar los resultados obtenidos para todas las técnicas y concluir a qué son debidos.
- **Comparación:** Analizar las diferencias obtenidas teniendo en cuenta el idioma de la noticia, la relación que tenga el resumen con el texto –extractiva o abstractiva– y la fuente de la cual proviene.

1.3 Estructura de la memória

En este apartado se explicará la estructura que tendrá el presente trabajo acompañada por un escueta descripción que intentará plasmar las ideas principales que tendrá cada capítulo.

1. **Introducción**: En este apartado se expone el tema que se aborda a lo largo de todo el proyecto. Además, se explican cuáles han sido las motivaciones que han llevado al autor del mismo en la elección de está temática, los objetivos planteados para el total cumplimiento del proyecto y finalmente la estructura que tiene la memoria.
2. **Creación de los corpus**: En este capítulo se expondrán en primer lugar distintos estudios acerca de los corpus existentes que contienen documentos de noticias en su interior; tanto para el caso de castellano como el caso del inglés. En segundo lugar se explicará la parte del diseño de la solución donde se aclararán conceptos tales como la arquitectura empleada para la generación del corpus, la distribución de carpetas creada para el mismo y la tecnología usada para la elaboración de los corpus. Posteriormente, se describirán los distintos detalles de implementación llevados a cabo y finalmente se comentarán algunas de las estadísticas referentes a los corpus generados.

3. **Generación de resúmenes**: En este apartado se expondrán en primer lugar distintos estudios relacionados con la generación de resúmenes tanto para la aproximación utilizando redes neuronales como para la basada en algoritmos clásicos. Además, se explicarán los distintos modelos utilizados; tanto para la representación de las palabras como para la generación de resúmenes y posterior evaluación. Posteriormente, se explicarán tanto el diseño empleado para la tarea de la generación de los resúmenes como el desarrollo de esta tarea. Seguidamente, aparecerá un apartado donde se comentarán detalles de la experimentación de los resúmenes y finalmente se plantearán los distintos resultados obtenidos mediante el uso de las métricas empleadas por el *Rouge* y por las métricas de evaluación semánticas propuestas en este trabajo.

4. **Conclusiones y trabajo futuro**: En esta sección, se analizarán qué objetivos de los planteados en el primer capítulo se han cumplido y cuáles no. Además, se intentará remarcar cuál ha sido el grado de cumplimiento de estos. En este mismo capítulo, se expondrán los distintos estudios que no se han abordado en el presente trabajo, debido a su duración, pero que sí que hubiera sido interesante realizar.

CAPÍTULO 2
Creación del corpus

En este capítulo se expondrán algunos de los estudios referentes a los corpus –tanto para el castellano como para el inglés– que forman parte del estado del arte. Posteriormente, se explicarán los detalles tanto del análisis previo y del diseño como del desarrollo de la solución propuesta y finalmente, se mostrarán las estadísticas finales de los corpus generados en este trabajo.

2.1 Estado del arte

En la actualidad existen distintos estudios y trabajos de investigación que han concluido con varios corpus de textos periodísticos. Es necesario diferenciar los corpus que se analizan en el trabajo según el idioma de las noticias que contienen; en concreto se han analizado trabajos cuyos corpus eran en castellano y corpus donde el idioma de sus noticias era el inglés. La elección de estas lenguas para el análisis de sus corpus ha sido bastante premeditada. En primer lugar se ha elegido el inglés ya que serán los corpus de este idioma los que utilizaremos como referencia en la creación de los corpus del presente trabajo. La elección del español se debe a que es uno de los idiomas para los que se creará un corpus.

De los estudios que contienen un corpus de noticias en inglés, pasaremos a comentar los que se han considerado más relevantes para el presente trabajo.

- *Document Understanding Conference (DUC)*[1]: Este corpus contiene un número relativamente reducido de noticias. Además, está áltamente especializado en la evaluación de sistemas de resumen automático y por ello solo presenta el texto de la noticia y el conjunto de resúmenes generado manualmente por varias personas. El corpus está a libre disposición de los usuarios ya que se puede adquirir de forma gratuita en Internet.
- *Gigaword*[2]: Es un corpus con un número bastante elevado de noticias –10 millones– pero de estas, no tienen almacenado el resumen. Además, este corpus está disponible tanto para los miembros del LDC –*Linguistic Data Consortium*– como para el público en general que pague el precio requerido para el acceso.
- *New York Times Corpus* [3]: El corpus contiene 1.8 millones de pares noticia-resumen pero solo de la fuente *New York Times*. Además, como en el caso anterior, está dis-

[1] https://duc.nist.gov/
[2] https://catalog.ldc.upenn.edu/LDC2003T05
[3] https://catalog.ldc.upenn.edu/LDC2008T19

ponible para los miembros de la LDC y para el público en general que pague por el acceso.

- *CNN/Daily Mail*: Como en el caso precedente, contiene pares de noticia-resumen pero solo de los periódicos *CNN y Daily Mail*. Tiene un tamaño de más de 300 mil documentos y a pesar de no ser accesible, los autores han depositado su código fuente en *GitHub*[4] para la posible replicación del corpus.

- *Newsroom*[5]: Es un corpus especialmente pensado para entrenar modelos de generación automática de resúmenes [10]. Este corpus cuenta con alrededor de 1.3 millones de noticias que tienen su correspondiente resumen y además provienen de varias fuentes. Los autores proporcionan el corpus para su descarga a cualquier persona que rellene el formulario de petición. Este será el estudio en el que se basará la parte de la creación del corpus en el presente trabajo. Además, se utilizarán algunos de los conceptos planteados en este trabajo para el análisis de los corpus.

Análogamente al caso anterior, expondremos los trabajos que incluyen un corpus de noticias en español y que se consideran más representativos para nuestro trabajo:

- *Aracne:*[6] El corpus contiene noticias desde 1914 hasta el 2014 y está pensado para realizar estadísticas sobre la evolución del lenguaje. La colección solo contiene 5167 artículos de los cuales no tienen almacenado ni el resumen ni las categorías. Además, el corpus no está accesible públicamente para su estudio.

- *Spanish News Text* [7]: Es una colección que tiene un tamaño de 1300MB de textos noticiarios escritos en español que provienen de distintos países hispanohablantes; de estos textos no tienen almacenado ni el resumen ni las categorías y está disponible tanto para los miembros del LDC como para el público en general previo pago del precio requerido.

- *Corpus del Español NOW* [8]: Es un corpus que contiene alrededor de 25 millones de noticias hasta abril del 2019, no obstante, el acceso está limitado a la plataforma *online* donde solo se muestran fragmentos de texto en los que aparece una cierta palabra de búsqueda.

- *Molino Labs*[9]: El corpus tiene alrededor 1.7 millones de noticias. Al igual que en el *corpus del Español Now*, el acceso está limitado a su página web y tampoco se muestra el texto completo de las noticias.

Como podemos observar, tanto en los corpus de inglés como en los de español, existen diferentes tamaños de corpus dependiendo del uso que se le vaya a dar a los mismos; es distinto un corpus de entrenamiento y prueba de un sistema de resumen o clasificación que uno para recoger datos estadísticos sobre el lenguaje. Así mismo, los corpus contendrán solo aquella información necesaria para las tareas que quieren llevar a cabo los creadores de la colección. Además, existen corpus que pueden tener una o varias fuentes dependiendo de las necesidades requeridas por sus creadores.

[4] https://github.com/deepmind/rc-data
[5] https://summari.es
[6] http://www.fundeu.es/aracne/
[7] https://catalog.ldc.upenn.edu/LDC95T9
[8] https://www.corpusdelespanol.org/now/
[9] http://www.molinolabs.com/corpus.html

Para el presente trabajo, se necesita crear un corpus de noticias de catalán y otro de castellano que provengan de varias fuentes, que tengan un tamaño mínimo de noticias y que como mínimo cada documento almacenado contenga el texto de la noticia, el resumen y la categoría o categorías en las que se ha clasificado. Estos requerimientos mínimos son debidos a las necesidades requeridas para los dos trabajos que utilizarán estos corpus.

2.2 Análisis del problema

Antes de cualquier implementación de código referente a la creación del corpus, se deben clarificar una serie de características que debe presentar el mismo. Inicialmente, se debe fijar la finalidad que tendrá el corpus ya que sin esta, el resto del trabajo no tendría sentido. La finalidad del corpus es la de comparar el resumen generado automáticamente para un texto periodístico con su resumen original; utilizando para ello diferentes técnicas de generación automática de resúmenes.

Una vez decidida la finalidad del corpus, se deben elegir algunos aspectos que tendrá el mismo. En primer lugar, se decidió estudiar noticias tanto del idioma castellano como del catalán; por esta razón, se decidió separar las noticias en dos corpus –uno para cada idioma–.

Una vez elegidos los idiomas de los corpus, se debían decidir los atributos necesarios para cada noticia. Aquí se tenía una prioridad, crear dos corpus que no solo se pudieran utilizar para la realización de ambos trabajos –solo se necesitaría el texto, las categorías y el resumen– sino que además, los corpus se pudieran utilizar en futuros estudios. Es por ello que se decidió almacenar información de cada noticia referente al contenido de esta, a las categorías, el resumen, el autor, la fecha de publicación, el título del *Twitter*, la descripción del *Twitter*, las palabras clave utilizadas y el enlace que referenciara a la noticia.

Como se quería tener diversidad tanto en la redacción de las noticias como en la temática de estas, se decidió incorporar distintas fuentes –periódicos– que aportaran las noticias. Para poder elegir qué periódicos se incluirían, se necesitaba hacer un análisis de los más utilizados en ambos idiomas y posteriormente contrastar las características comunes entre estos. De este análisis surgieron tanto periódicos que se podían utilizar como periódicos que no. Un ejemplo de periódico que no se pudo utilizar en el presente trabajo fue el *abc* el cual tenía todas las noticias en PDF y resultaba muy complicado poder diferenciar las distintas partes de la noticia; además, al no tener metadatos, no se podían rellenar el resto de campos. Una vez descartados los periódicos en los que no era viable la recogida de noticias, se observaron dos tipos de noticiarios; los que se podía extraer la información desde la hemeroteca y los que se tenían que extraer las noticias indexadas por categorías –internacional, economía, sociedad, deportes, cultura, ciencia, etc–. Es por ello que la implementación del código que se realizase, debería contemplar ambos casos. Un requisito primordial de nuestros corpus era que tenían que tener un número elevado de noticias almacenadas ya que se necesitan muchas muestras para el entrenamiento del sistema basado en redes neuronales; por eso, se propuso obtener un mínimo de 200 mil noticias para el corpus en catalán y un mínimo de 700 mil noticias en castellano –en catalán es menor debido al menor número de diarios que escriben en este idioma–.

2.3 Diseño de la solución

En esta sección se detallará en primer lugar la arquitectura generada para la creación del corpus donde se expondrán los distintos módulos empleados. Posteriormente, se presentará la estructura de carpetas empleada para el sistema de creación del corpus y finalmente, se explicará la tecnología utilizada en este mismo sistema y el porqué de esta decisión.

2.3.1. Arquitectura del sistema

Para el cometido referente a la creación del corpus, se ha definido un sistema que realiza cuatro tareas bien diferenciadas. La primera tarea es la recolección de las URLs de las noticias para cada una de las paginas web de un periódico referentes a un día en concreto o una categoría en concreto; ya que como se había comentado en la *Sección 2.2*, el sistema ha de ser capaz de descargar noticias tanto de la hemeroteca como de una categoría. La segunda tarea que realiza el sistema es la de recoger de cada noticia –enlace– los campos requeridos, comentados también en la *Sección 2.2*, para la creación del corpus. La tercera tarea es la de procesar las *URLs* obtenidas en la primera tarea utilizando para ello la segunda tarea. La última tarea es la encargada del almacenamiento y acceso de los documentos generados anteriormente. Es por ello que se han definido cuatro módulos distintos y que se comentarán detalladamente.

- `Recogida de URLs`: Mediante configuraciones previamente definidas para cada uno de los periódicos que se pretenden usar, el módulo es el encargado de obtener las *URLs* de las noticias de cada uno de ellos y posteriormente las almacena en distintos archivos.
- `URL a JSON`: Utilizando selectores css –Cascading Style Sheet– recoge los campos de las noticias previamente elegidos y los agrupa dentro de una estructura JSON.
- `Procesador de URLs`: A partir de los enlaces obtenidos por el módulo de `Recogida de URLs` y utilizando el módulo `URL a JSON`, se genera un documento –estructura de almacenamiento de la base de datos MongoDB– para cada noticia; en este, se guardará la información obtenida de la noticia en cuestión.
- `Base de Datos`: Es el módulo encargado de almacenar localmente los documentos previamente generados.

En la *Figura 2.1*, se mostrarán las relaciones entre los distintos módulos comentados anteriormente.

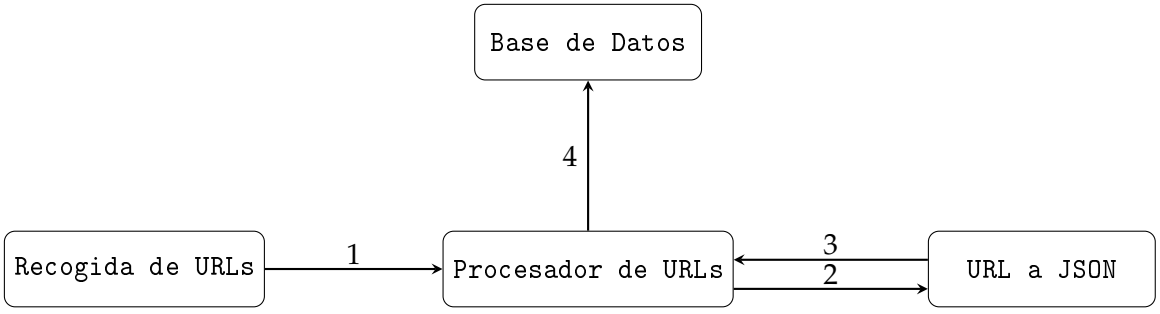


Figura 2.1: Arquitectura para la generación del corpus

Tal y como se puede observar en la *Figura 2.1*, los pasos necesarios para procesar una noticia serían los siguientes: en primer lugar, el módulo de `Recogida de URLs` se encarga de extraer los enlaces de las noticias de la página web del periódico y los escribe en un fichero. Seguidamente, el `Procesador de URLs` lee este fichero y hace una petición a `URL a JSON` con el enlace a procesar y el selector propicio para esta URL. En tercer lugar, el módulo `URL a JSON` recoge todos los campos necesarios de las noticias, los almacena en un JSON y envía la respuesta al `Procesador de URLs`. Finalmente, se envía el resultado al módulo `Base de Datos` el cual lo inserta en la colección adecuada.

2.3.2. Diseño detallado

En esta sección, se concretará el diseño elegido a partir de la estructura de directorios escogida, *Figura 2.2*, para albergar el código fuente relacionado con la creación del corpus.

Figura 2.2: Estructura de directorios creada para la generación de corpus

Tal y como se puede observar en la *Figura 2.2*, existen tres carpetas en la raíz las cuales albergan los distintos módulos comentados anteriormente y que se pasarán a detallar pormenorizadamente. Existen tres carpetas y no cuatro ya que se ha decidido implementar conjuntamente los módulos `Procesador de URLs` y `Base de Datos` dada la estrecha relación que mantienen ambos.

En primer lugar encontramos la carpeta `news_data_capture`, la cual hace referencia al módulo `URLs a JSON` y contiene la implementación de este. Además, se ubica una carpeta

auxiliar llamada `selectors` la cual contendrá las configuraciones de los selectores css para los distintos tipos de páginas de las noticias; puede existir más de una configuración para un mismo periódico ya que puede que a lo largo del tiempo, este, haya cambiado la estructura de sus noticias.

Seguidamente, encontramos la carpeta `news_urls_capture`; esta hace referencia al modulo de `Recogida de URLs` el cual es el encargado de capturar los enlaces de las noticias. En él, podemos encontrar las implementaciones para la captura de enlaces así como la carpeta `news_sites` en donde encontraremos una configuración para cada uno de los periódicos y que será la encargada de extraer una lista de enlaces a partir de una página web.

En tercer lugar, aparece la carpeta `news_urls_process` que engloba el código relacionado con la implementación del módulo encargado de procesar las URLs de las noticias y el módulo encargado de guardarlas en la BDA. En este directorio también se encuentran las carpetas `urls_to_process` y `urls_processed`; en la primera, se almacenan los ficheros de las noticias a procesar y en la segunda los archivos procesados junto con otro fichero de resultados.

Después de haber detallado la estructura de directorios para la organización del código fuente relacionado con la generación del corpus, pasaremos a explicar la estructura de la base de datos.

Como se ha utilizado una base de datos NoSQL orientada a documentos, se hablará de colecciones y de la estructura que va a tener cada documento. Inicialmente, se tiene una colección por cada una de las fuentes de las que se ha decidido extraer noticias; es decir una colección por periódico. Así mismo, existe una colección llamada `errors` donde se almacenan los errores encontrados durante el procesamiento de los enlaces de las noticias. Por último, se dispone de la colección `nocontent` donde almacenamos aquellas *URLs* y configuraciones utilizadas mediante las cuales no se ha podido extraer el cuerpo de la noticia. Estas dos colecciones nos permitirán detectar errores tanto en la creación de las URLs como en la elaboración de los selectores. La estructura que sigue el documento de cada noticia se puede observar en el *Código 2.1*.

```
{
  _id: ...,
  content: ...,
  url: ...,
  title: ...,
  keywords: ...,
  description: ...,
  publi_date: ...,
  category: ...,
  author: ...,
  twitter_title: ...,
  twitter_descrip: ...,
  twitter_image: ...,
  language: ...,
  config: ...
}
```

Código 2.1: Estructura del documento de una noticia

- **_id:** Identificador único usado por MongoDB, diferencia cada documento de una colección.
- **content:** Campo donde se almacena el cuerpo de la noticia.
- **url:** Enlace de la noticia.

- **title:** El titular de la noticia.
- **keywords:** Palabras clave asignadas por el periódico a la noticia.
- **description:** Resumen de la noticia.
- **publi_date:** Fecha de publicación de la noticia.
- **category:** Categoría/ Categorías asignadas por el periódico a la noticia.
- **author:** Autor de la noticia.
- **twitter_title:** Título que se muestra en Twitter al compartir la noticia.
- **twitter_descrip:** Descripción mostrada en el *tweet* al compartir.
- **twitter_image:** Enlace de la imagen que se mostrará en el *tweet*.
- **language:** Idioma de la noticia.
- **config:** Selector usado para extraer todos los campos nombrados anteriormente. Este campo se usa para comprobar y corregir posibles errores en los selectores.

2.3.3. Tecnología utilizada

Para el almacenamiento de los documentos se ha utilizado MongoDB; el cual es un sistema de base de datos que se ajusta completamente a las necesidades requeridas para el propósito del presente estudio. MongoDB es un sistema NoSQL –Sistema De Gestión de Base de Datos No Relacional– orientado a documentos los cuales se almacenan en BSON –Binary JSON–. Las características mas relevantes que nos han llevado a la elección de este sistema de gestión de bases de datos respecto a otros candidatos han sido:

- **Alta disponibilidad:** A través de replicación y recuperación automática.
- **Escalabilidad horizontal:** Permite la partición de la base de datos de manera transparente y en consecuencia aumenta el rendimiento del procesado de datos.
- **Flexibilidad:** No tiene esquemas rígidos de datos.
- **Auto balanceo de carga:** El balanceador de carga distribuye automáticamente los datos entre los diferentes servidores de manera uniforme.
- **Replicación nativa:** Los servidores se sincronizan automáticamente.
- ***Automatic failover*:** Elección automática de un nuevo servidor primario cuando el anterior ha caído.
- ***Opensource*:** Código abierto.

Todas estas características nos han ayudado en la replicación de la base de datos en distintas máquinas aumentando así la disponibilidad de la misma.

Para el proceso de creación del sistema de descarga se ha utilizado *Node.js* –entorno de ejecución *opensource* multiplataforma que ejecuta código escrito en *JavaScript* fuera de los navegadores– el cual está orientado a eventos asíncronos. Se ha seleccionado este entorno de ejecución debido a las siguientes características:

- ***Opensource*:** Código abierto.
- **Fácil implementación:** Como *JavaScript* es un lenguaje de *scripting* podemos escribir código de manera muy rápida e intuitiva.
- **JSON:** Trabaja de manera nativa y transparente con JSON.
- **Módulos:** Existen gran cantidad de librerías que agilizan la implementación de muchas funcionalidades.
- **Comunidad activa:** Debido al auge de *Node.js*, aparecen y se actualizan diariamente una gran cantidad de módulos para este entorno.
- **Rendimiento:** Gracias al motor de *JavaScript* V8 implementado por *Google* y usado por *Node.js* nos garantiza un alto rendimiento en la ejecución del código.

2.4 Desarrollo de la solución

En este apartado se explicarán los pasos que realizará el código de cada uno de los ficheros comentados en el apartado anterior, algunos de los problemas que se han tenido para desarrollar las tareas y las decisiones tomadas para poder solucionarlos.

Inicialmente, se buscó información en la red acerca de un sistema que estuviera implementado en *Node.js*. A partir del estudio [10] en el cual se creaba el corpus de **Summari.es** se encontró **Readability**[10] que era una *API* para extraer automáticamente una cierta información de un determinado enlace. Esta aproximación nos pareció interesante pero ya no tenía soporte; así que nos descargamos **Mercury Tools**[11] que es un *toolkit* mantenido por los mismos creadores que **Readability**. Una vez descargada la herramienta se empezó a analizar su código pero al poco tiempo nos dimos cuenta de que era mejor implementar una aproximación propia, basada en el *toolkit* comentado anteriormente, que modificarlo. Esta decisión estuvo fundamentada principalmente en que el **Mercury Tools** utilizaba una serie de implementaciones, imprescindibles para su código, no necesarias para nuestro problema.

Para un mayor entendimiento del código, se pasará a explicar la implementación utilizada para cada uno de los módulos comentados en la *Sección 2.3*.

2.4.1. Recogida de URLs

El módulo de `Recogida de URLs` es el encargado de extraer todos los enlaces a partir de un fichero de configuración de un periódico en concreto y posteriormente almacenar estos enlaces en distintos ficheros. En primer lugar se explicarán los distintos paquetes que utiliza este módulo.

- **request:** Es el paquete utilizado para hacer las peticiones *HTTP* y es necesario para la extracción de los enlaces de las noticias a partir de una *URL*.
- **cheerio:** Es un paquete que proporciona los métodos para la manipulación de la información descargada con el formato *html*.
- **fs:** Paquete necesario para el acceso a ficheros.

[10] https://www.readability.com/
[11] https://mercury.postlight.com/web-parser/

- **date-utils:** Paquete utilizado para la búsqueda de noticias en la hemeroteca; gracias a este, podemos crear las fechas para el acceso a la hemeroteca.

- **crypto:** Se utiliza para la generación del nombre del fichero; así evitamos tener solapamiento de nombres en el almacenamiento de estos.

La idea principal del código es la de extraer de manera ordenada los enlaces de una determinada página web para posteriormente almacenarlos en ficheros. Como ya se comentó en la *Sección 2.2*, no todos los noticiarios tienen hemeroteca; es por ello que se implementó un módulo capaz de trabajar dos tipos de periódicos: los accesibles mediante hemeroteca y los accesibles mediante categorías.

- **Periódicos accesibles mediante hemeroteca:** En estos, inicialmente se definía un fichero de configuración *JSON* el cual tenía un campo llamado *url_list* en el que se almacenaba el enlace de la hemeroteca y un marcador fecha. A partir de este fichero de configuración y la elección del día de inicio, se generaban los distintos enlaces de acceso a la hemeroteca hasta el día de la recolección de la información.

  Un ejemplo de fichero *JSON* de configuración para el listado de noticias es el del diario **20minutos**, *Código 2.2*.

```
1  {
2    "site" : "https://www.20minutos.es",
3   "url_list" : ["https://www.20minutos.es/archivo/{fecha}"],
4   "default_conf" : "20minutos_1",
5   "collection" : "20minutos",
6   "default_selector_urls" : "li.item a"
7  }
```

Código 2.2: Ejemplo *Json* para el listado del **20minutos** el cual es accesible desde la hemeroteca

  - **site:** Sitio web del periódico.
  - **url_list:** Listado de enlaces, en este caso como el acceso es mediante la hemeroteca, solo contiene uno.
  - **default_conf:** Configuración por defecto que se utilizará para la búsqueda de campos de una noticia de este periódico.
  - **collection:** La colección de la base de datos a la que pertenece.
  - **default_selector_urls:** El selector *css* que se utiliza para el acceso y posterior almacenamiento del enlace de la noticia.

- **Periódicos accesibles mediante categorías:** Este tipo de periódicos tienen en el fichero de configuración el campo llamado *url_list* en el cual se guardan todos los enlaces genéricos de las categorías. Además, en este mismo fichero, se almacenan tanto el número de páginas a recorrer por categoría como el selector *css* pertinente para la navegación entre las distintas páginas. A partir de los campos comentados, se generan los enlaces de cada categoría y para todas las páginas que se habían indicado.
  Un ejemplo de fichero de configuración *css* utilizado para el listado de las noticias del **abc** se muestra en el *Código 2.3*:

```
{
"site" : "https://www.abc.es",
 "url_list" : [
  "https://www.abc.es/internacional/",
  "https://www.abc.es/economia/",
  "https://www.abc.es/sociedad/",
  "https://www.abc.es/deportes/",
  "https://www.abc.es/cultura/",
  "https://www.abc.es/ciencia/"
 ],
 "default_conf" : "abc_1",
 "default_selector_urls" : "article.articulo-portada .titular a",
 "collection" : "abc",
 "num_max_pages" : 1000,
 "next_page" : "a[title=\"Siguiente\"]"

}
```

Código 2.3: Ejemplo *JSON* para el listado del **abc** el cual es accesible desde las categorías

- **num_max_pages:** Número máximo de paginas a recorrer por categoría.
- **next_page:** Selector *css* que permite recoger la *URL* de la siguiente página de una categoría.
- **site, url_list, default_conf, collection y default_selector_urls:** Tienen el mismo uso que en el caso de los periódicos accesibles utilizando la hemeroteca.

Una vez generados los enlaces del listado de noticias, se descarga para cada uno de ellos el *html* pertinente y a partir de los selectores definidos en el fichero de configuración se consiguen extraer los enlaces ya pertenecientes a las noticias. Finalmente, almacena los enlaces obtenidos en distintos ficheros dependiendo del periódico y la categoría o día de la noticia.

El caso del diario **ara** es diferente, es un diario en el que sus noticias están organizadas por categorías; como el segundo caso comentado anteriormente. Pero a diferencia del caso anterior, el selector *css* de navegación entre páginas está implementado utilizando una consulta *JSON* y como era diferente al resto de periódicos analizados, se diseñó un fichero de extracción de enlaces ad-hoc para este. Es el único periódico en el que se ha tenido que desarrollar una parte a medida.

2.4.2. URL a JSON

Es el módulo encargado de la extracción de la información de cada noticia y posterior agrupación de esta en una estructura *JSON*. Como en el caso anterior, inicialmente, se expondrán los paquetes utilizados y posteriormente se explicará el código implementado.

- **express:** Paquete utilizado para convertir el módulo en un servicio web.
- **iconv y charset-parser:** Utilizados para la modificación de la codificación del texto de las noticias.
- **cheerio, fs, request y date-utils:** Comentados ya en la explicación del módulo anterior.

Este módulo proporciona un servicio mediante el cual dada una consulta sobre un enlace, devuelve la información requerida en los corpus para cada noticia almacenándola en un *JSON*. Para esta tarea, se deben definir unas configuraciones *JSON* para cada uno de los periódicos y gracias a estas, es capaz de extraer los campos de cada noticia.

Un ejemplo de configuración *JSON* para la extracción de los campos de una noticia es la del periódico **dbalears** *Código 2.4*.

```
{
  "content" : ".cuerpo",
  "title" : "title",
  "keywords" : "meta[name=\"keywords\"]",
  "description" : {
    "selector" : "meta[name=\"description\"]",
    "attr" : "content"
  },
  "publi_date" : {
    "selector" : "meta[name=\"dc.date.issued\"]",
    "format" : "yyyy-mm-dd"
  },
  "category" : "meta[property=\"article:section\"]",
  "author" : {
    "selector" : "span[class=\"related-art-author\"]",
    "attr" : ""
  },
  "url" : "meta[property=\"og:url\"]",
  "twitter_title" : "meta[name=\"twitter:title\"]",
  "twitter_descrip" : "meta[name=\"twitter:description\"]",
  "twitter_image" : "meta[name= \"twitter:image\"]",
  "language" : "catala"
}
```

Código 2.4: Ejemplo de configuración *JSON* para la recogida de los campos del periódico **dbalears** en una determinada noticia

- **content:** Selector *css* para la extracción del cuerpo de la noticia.
- **title:** Selector *css* para la recuperación del título de la noticia.
- **keywords:** Selector *css* para la extracción de las palabras clave de la noticia.
- **description:** Selector para la recuperación del resumen de la noticia donde ''`attr`'' es el atributo a extraer.
- **publi_date:** Es el selector que permite extraer la fecha de publicación y ''`format`'' es el formato que tiene la fecha de la noticia.
- **category:** Selector para extraer las categorías de la noticia.
- **author:** Selector para extraer el autor de la noticia.
- **url:** Selector que permite recuperar el enlace que referencia a la noticia.
- **twitter_title, twitter_descrip, twitter_image:** Selectores para extraer la información relacionada con el *Twitter*.
- **language:** Idioma de la noticia.

Cuando se analizaron los periódicos, se observó que existían distintos tipos de fechas de publicación en las noticias; por ello, se implementó un método, en este módulo, que fuera capaz de normalizarlas. Recordemos que es muy importante la normalización de los campos y que sin esta, la automatización de los sistemas sería muy complicada.

2.4.3. Procesador de URLs y Base de datos

El módulo `Procesador de URLs` es el encargado de procesar las noticias identificadas en el módulo de `Recogida de URLs` utilizando para ello el módulo de `URL a JSON`. El módulo `Base de datos`, es el encargado de almacenar los *JSON* obtenidos en el módulo `Procesador de URLs` en la base de datos. Dada la relación que mantienen ambos, se decidió implementarlos en un mismo fichero. Como en los casos precedentes, inicialmente se presentarán las librerías utilizadas y posteriormente, se explicará el código implementado.

- **file-state-monitor:** Paquete utilizado para monitorizar el estado de la carpeta donde se almacenaban los enlaces de las noticias pendientes de procesar. Gracias a este, se consigue detectar un nuevo fichero en la carpeta para así poder procesarlo.
- **mongodb:** Utilizado para el acceso a la base de datos.
- **fs y request**: Comentados en la explicación de los dos módulos precedentes.

El módulo `Procesador de URLs` es el encargado de monitorizar la carpeta `urls_to_process` en la cual el módulo de `Recogida de URLs` ha ido almacenando los ficheros que contienen los enlaces de las noticias. Cuando se inserta un nuevo fichero en la carpeta monitorizada, el `Procesador de URLs` procesa un enlace de su interior utilizando para ello el servicio de `URL a JSON` previamente explicado. Una vez obtiene el *JSON* del servicio, lo almacena en la colección pertinente de la base de datos. Esta acción se repite hasta que procesa todos los enlaces de todos los ficheros.

2.4.4. Limpieza final del corpus

Una vez creados ambos corpus, nos percatamos de que había una serie de periódicos en los que se había descargado en la parte del texto contenido no perteneciente al mismo. Es por ello que se decidió implementar dos ficheros que permitieran la eliminación del contenido no deseado. Concretamente los periódicos en los que hizo falta esta limpieza fueron **naciodigital** y **elpais**. En ambos casos, se incluía al principio del texto de algunas noticias una frase referente a la fecha de publicación; por esta razón, se eliminó esta oración.

2.5 Estadísticas finales de los corpus

En esta sección se presentarán las estadísticas referentes a los corpus –tanto de castellano como de catalán– obtenidos mediante el procedimiento descrito anteriormente.

En primer lugar se exponen las estadísticas referentes al número de noticias obtenidas para ambos corpus, separándolas según el corpus del periódico al que pertenecen. Seguidamente, se presentará un gráfico referente a la distribución de noticias que tiene cada uno de los corpus.

Las estadísticas referentes al número de noticias por cada uno de los periódicos del corpus de catalán se reflejan en la *Tabla 2.1*.

Periódico	Noticias
ara	64 024
dbalears	1567
diaridegirona	13 104
diarilaveu	39 673
elpuntavui	6956
eltemps	4659
naciodigital	56 240
regio7	11 806
vilaweb	35 817
Total	233 846

Tabla 2.1: Número de noticias del corpus en catalán

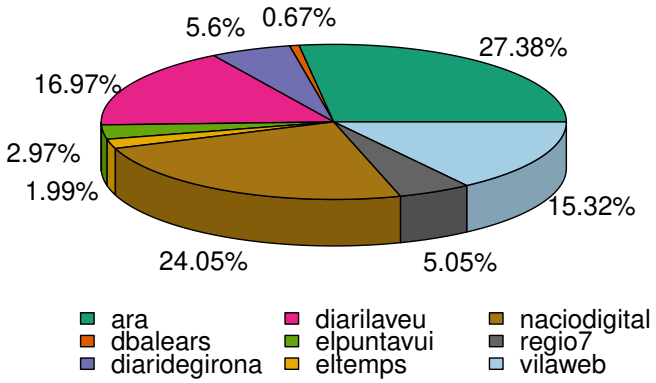
En la *Tabla 2.1* se puede apreciar que el periódico del que mayor número de noticias se han conseguido obtener ha sido el **ara** seguido de **naciodigital**; en contraposición, el periódico del que menos noticias se han conseguido extraer ha sido **dbalears**.

Periódico	Noticias
20minutos	302 486
abc	126 317
diariodemallorca	12 247
elconfidencial	38 098
eldiario	9341
elespañol	49 920
elindependiente	25 811
elpais	85 271
expansion	39 728
laser	15 355
lasprovincias	6912
levante	15 645
publico	13 039
ultimahora	32 131
Total	772 301

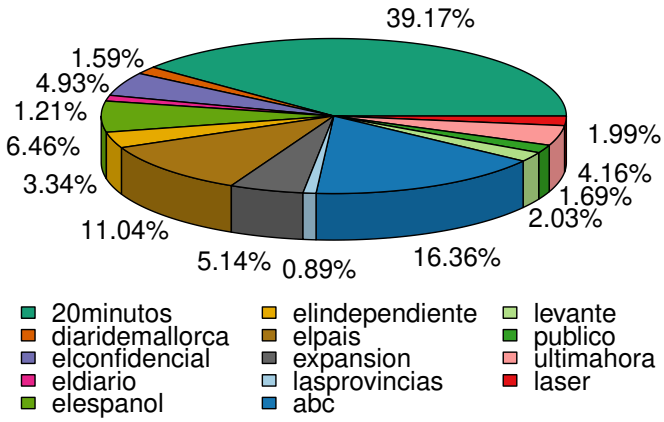
Tabla 2.2: Número de noticias del corpus en castellano

En cuanto a las noticias obtenidas para los periódicos de castellano mostradas en la *Tabla 2.2*, podemos observar que el periódico del cual se han extraído más noticias ha sido el **20minutos** seguido, aunque con menos de la mitad de noticias que este, por el **abc**. Finalmente, el periódico del cual se han conseguido extraer menos muestras ha sido **lasprovincias**.

Para poder analizar mejor la distribución de noticias que tiene cada corpus, se propone un gráfico de tarta por cada corpus, *Figura 2.3*, donde se muestran el porcentaje de noticias que aporta cada periódico sobre el total de noticias de su corpus.



(a) Corpus en catalán



(b) Corpus en castellano

Figura 2.3: Porcentaje de noticias en los corpus

Como podemos observar en la *Figura 2.3a*, el **ara** es el que tiene un mayor número de noticias en el corpus del catalán con más del 27 % de las noticias. En contraposición, **dbalears** solo aporta un 0,67 % de noticias a este mismo corpus.
En el caso del corpus en castellano *Figura 2.3b*, el periódico que mayor porcentaje de noticias aporta es el **20minutos** con casi un 40 % sobre el total de noticias y el que menos **lasprovincias** con solo un 0,89 %.

En primer lugar, si comparamos las estadísticas del corpus de catalán obtenido con los corpus de inglés presentados en la *Sección 2.1* vemos que este tiene un mayor número de noticias que el *DUC* pero menor que el resto de corpus. Si hacemos lo propio pero con los corpus presentados para el castellano, observamos que tiene más noticias que el de *Aracne* pero inferior al resto.

En segundo lugar, el corpus de castellano supera en tamaño tanto al *DUC* como al *CNN/Daily Mail* –en el caso de los corpus presentados para el inglés–. Si hablamos de la comparación frente a los corpus de castellano presentados solo contiene más noticias, al igual que el corpus en catalán, que el *Aracne*.

2.6 Aspectos legales de los corpus

Los corpus obtenidos en el presente trabajo, no se pueden liberar; esto es debido a que el contenido de las noticias proviene de los autores de los distintos periódicos. En contraposición, el sistema para la generación del corpus sí que se puede liberar ya que es una creación propia. El código para la replicación del corpus está disponible en el enlace al *GitHub* `https://github.com/vicfer-tfg/corpus-creation` bajo la licencia GPL 3 (*General Public License*).

CAPÍTULO 3
Generación de resúmenes

En el presente capítulo se expondrán inicialmente algunos de los estudios que forman parte del estado del arte en la generación automática de resúmenes. Además, se presentarán los distintos modelos utilizados para esta tarea tales como una aproximación para la generación de resúmenes basada en redes neuronales, distintas aproximaciones para el mismo fin pero basadas en algoritmos clásicos, un modelo para la representación vectorial de las palabras, un evaluador sintáctico para los resúmenes –*ROUGE*– y un modelo para la evaluación semántica de los resúmenes. Una vez explicados los modelos utilizados, se expondrán los detalles tanto del diseño como del desarrollo del sistema utilizado. Finalmente, se presentará la experimentación realizada y se comentarán los resultados obtenidos comparando para ello los resúmenes generados por los modelos utilizados con los de referencia proporcionados en el corpus.

3.1 Estado del arte

El resumen automático es una aplicación del área del PLN la cual consiste en la sintetización automática de un texto. Como se ha comentado anteriormente, según como sea la relación entre el resumen generado y el texto, el resumen podrá ser extractivo o abstractivo.

Los avances del aprendizaje automático –*Machine Learning*– en los últimos años han permitido el desarrollo de modelos de aprendizaje profundo –*Deep Learning*– para la tarea de generación automática de resúmenes. El *Deep Learning* es un subcampo del *Machine Learning* que a su vez, es un disciplina científica del ámbito de la Inteligencia Artificial donde se crean sistemas que son capaces, automáticamente, de identificar patrones entre un gran número de datos; estos sistemas se utilizan para predecir comportamientos futuros. El *Deep Learning* se utiliza para imitar el comportamiento de la mente humana en el procesamiento de la información creando para ello distintos patrones que permiten tomar decisiones. Los sistemas de *Deep Learning*, se caracterizan por tener una red neuronal con múltiples capas ocultas –capas que no son ni de entrada ni de salida–.

Además de los sistemas basados en redes neuronales, existen otros que se basan en algoritmos clásicos –no supervisados, heurísticos, ...– fundamentados principalmente en las estadísticas obtenidas por las palabras en una frase o documento.

Algunos de los estudios que se han considerado relevantes para este trabajo en los que se utilizan tanto redes neuronales como algoritmos clásicos para la generación automática de resúmenes han sido:

- *SHA-NN –Siamese Hierarchical Attention for Neural Networks–* : Es un sistema extractivo basado en técnicas de aprendizaje supervisado. Utilizando una red neuronal, es capaz de aprender a partir de ejemplos tanto positivos –resumen correcto del documento que está analizando– como negativos –resumen no apropiado para el documento–. Este modelo se utilizará en el presente trabajo y se explicará más detalladamente en la *Sección 3.2*. Toda la información referente al modelo de *SHA-NN* ha sido extraída de [8].

- *Get To The Point*: Es un generador automático de resúmenes abstractivo evaluado sobre el corpus del CNN/Daily Mail [27]. Utilizando redes neuronales, es capaz de seleccionar palabras del texto y generar nuevas frases dando así una sensación de mayor naturalidad al resumen. Además, soluciona el problema de la repetición de palabras en el resumen, sobre todo si este está formado por múltiples oraciones, utilizando para ello el *coverage*. El *coverage* es una medida que permite conocer qué parte del texto sintetiza cada una de las oraciones que forman parte del resumen.

- *Resumen con una arquitectura híbrida*: Es un generador de resúmenes abstractivos basado en *Deep Learning* [4]. En primer lugar se seleccionan los fragmentos extractivos que se han considerado más representativos para el texto y posteriormente se reescriben estos fragmentos creando así un resumen abstractivo.

- *Lexrank: Graph-based Lexical Centrality as Salience in Text Summarization*: Es un trabajo basado en la generación de resúmenes extractivos utilizando para ello el algoritmo clásico *LexRank* el cual se explicará más detalladamente en la *Sección 3.2*. En este trabajo se evalúan las distintas aproximaciones del *LexRank* para la selección de las frases más importantes de un documento en el corpus *DUC*. La información sobre este se ha obtenido de [6].

- *Text summarization using Latent Semantic Analysis*: Como en el caso anterior, el trabajo está basado en la generación automática de resúmenes de carácter extractivo pero utilizando para ello el modelo *LSA*. Además de explicar el sistema, los autores del mismo lo evalúan utilizando para ello documentos en los idiomas Turco e Inglés [11].

- *TextRank: Bringing Order into Texts [19]*: Es un estudio en el cual se presenta el modelo del *TextRank*. Asimismo, se proponen dos nuevas modificaciones no supervisadas tanto para la utilización del *Textrank* en la extracción de *keywords* como para su utilización en la extracción de oraciones.

- *Luhn:* Es un método heurístico presentado en [17] donde, inicialmente, se escriben las principales ventajas de la generación automática de resúmenes y posteriormente se explica el modelo del *Luhn*.

Como podemos observar, existe una gran diversidad de estudios que plantean el problema de la generación automática de resúmenes tanto basados en redes neuronales como basados en algoritmos clásicos. Además, como se ha podido comprobar existen tanto estudios donde se aborda la generación de resúmenes desde una perspectiva extractiva como estudios donde se aborda la generación de resúmenes desde un punto de vista abstractivo. En nuestro planteamiento inicial se ha decidido apostar por unos resúmenes de carácter extractivo, para ello se utilizarán técnicas basadas tanto en redes neuronales como técnicas basadas en algoritmos clásicos. Se ha decidido elegir *SHA-NN* como sistema basado en redes neuronales; las implementaciones de *Deep Learning* se caracterizan por tener una red neuronal con múltiples capas ocultas, el *SHA-NN* no tiene muchas capas ocultas y por ello no sería una técnica extrictamente basada en *Deep Learning*. Nuestra

elección está fundamentada principalmente en el poco tiempo requerido para el entrenamiento del mismo y por los prometedores resultados presentados en el artículo [8]. Para el caso de los algoritmos clásicos se utilizará el *Lexrank, LSA, TextRank y el Luhn.*

3.2 Modelos

En este apartado se explicarán los distintos modelos utilizados en el trabajo. Inicialmente se proporcionará una descripción introductoria de los modelos de redes neuronales; además, se presentará el modelado de los *embeddings*, el sistema *SHA-NN* que se utilizará para la generación automática de los resúmenes basados en redes neuronales, los algoritmos clásicos de resumen así como las métricas utilizadas para la evaluación de estos resúmenes.

3.2.1. Redes neuronales

La red neuronal es un modelo computacional basado en capas en el interior de las cuales se ubican unos nodos –neuronas– conectados entre sí que son capaces de transmitir información entre ellos [28]. Este modelo está basado en el comportamiento biológico de las neuronas; es por ello que se les conoce como redes neuronales o también como sistema conexionista. La idea inicial es que dada una entrada, esta se procesa a través de la red produciendo unos valores de salida. Cada uno de los enlaces que conectan los distintos nodos tienen un cierto peso y que multiplicado por la salida del nodo anterior produce la entrada del propio nodo; estos pesos se actualizan a lo largo del periodo de entrenamiento de la red. En la salida de cada nodo puede hallarse una función umbral, también llamada función de activación, que modifica su valor de salida dependiendo de la función utilizada.

Los sistemas basados en redes neuronales son capaces de aprender por sí mismo de forma similar a como lo haría un ser humano. La actualización de los pesos comentada anteriormente junto con un cierto factor de aprendizaje, permite el entrenamiento de la red minimizando para ello la función de error del sistema. Este proceso de actualización de los pesos se le conoce como *Backpropagation* –retropropagación del error– y mediante la técnica del descenso por gradiente es capaz de calcular la variación que tendrá cada uno de los pesos. Para expresar mejor esta idea de la red neuronal, se presentará un ejemplo de red con una sola capa oculta –capa que no es de entrada ni de salida– *Figura 3.1*.

Como se observa en la *Figura 3.1*, la red neuronal presentada como ejemplo tiene solamente cuatro características de entrada –X1, X2, X3, X4–; es por ello que la capa de entrada tiene solo cuatro nodos. En el caso de la capa oculta se tendrán tantos nodos como se considere oportuno. Finalmente, la capa de salida contendrá también el número de nodos que se requieran para la salida de la red; en nuestro ejemplo solamente uno. Además, todos los enlaces que conectan los nodos tendrán un peso predefinido inicialmente y este se actualizará conforme avance el entrenamiento de la red.

En nuestro trabajo, se utilizarán las redes neuronales para la implementación del modelo *SHA-NN*, el cual se explicará en este mismo capítulo. Además, internamente, este incorpora redes neuronales recurrentes que son un tipo de redes neuronales donde se permiten conexiones arbitrarias entre neuronas; pudiendo incluso crear ciclos dando la sensación de pérdida de capas; sobre todo si existen muchas capas ocultas. Estas conexio-

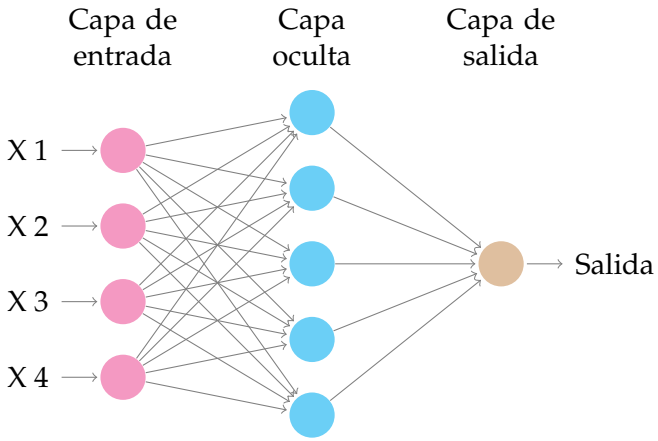


Figura 3.1: Red neuronal con una capa oculta

nes arbitrarias proporcionan a la red un comportamiento dinámico permitiendo que esta tenga memoria. Esta memorización implica una acumulación creciente de la información conforme avanza el entrenamiento de la red.

Existen diferentes arquitecturas de redes recurrentes, concretamente la que utiliza *SHA-NN* es la *LSTM –Long short-term memory–* [12] la cual resuelve el problema de la creciente acumulación de información incorporando para este fin distintos criterios de selección de la información a almacenar.

Como se ha comentado anteriormente, los nodos de las redes neuronales tienen en la salida una función de activación que modificará el valor calculado en el nodo obedeciendo a la función pertinente. A pesar de que existen diversas funciones de activación [3], se ha decidido comentar solo las que utiliza la implementación de *SHA-NN*.

- **Tangente hiperbólica**: Trasforma los valores de entrada en valores en una escala de -1 a 1. Su función es $f(z) = \frac{exp(z) - exp(-z)}{exp(z) + exp(-z)}$.
- **Softmax**: Transforma la entrada en probabilidades donde la suma de todas da como resultado 1. Su función es: $f(z_i) = \frac{exp(z_i)}{\sum_j exp(z_j)}$.

3.2.2. Modelo para la creación de los *embeddings*

Se llama *embeddings* a la representación vectorial continua de las palabras. Los *embeddings* se utilizan en el PLN para convertir las frases o palabras de un vocabulario en vectores numéricos. Estos vectores se utilizan para la representación vectorial de las palabras tanto en el entrenamiento como en la generación de resúmenes de *SHA-NN*; además, se utilizarán para la evaluación semántica de los resúmenes. La herramienta utilizada para la creación de los *embeddings* ha sido *Word2Vec*.

3.2.2.1. Word2Vec

Es una herramienta que permite la implementación de modelos mediante los cuales se crean representaciones vectoriales, en un espacio multidimensional, de las palabras. Trabajan de forma que la representación de palabras semánticamente similares corresponden a zonas cercanas en el espacio vectorial de representación. Por ejemplo la repre-

sentación de la palabra "verde" está muy cerca de la palabra "rojo". Esta herramienta permite la implementación de la representación vectorial utilizando dos modelos; el CBOW – *Continuous Bag of Words*– y el *Continuous Skip-Gram.*

- **CBOW:** Es una arquitectura propuesta por T.Mikolov en [20]; está implementada utilizando una red neuronal y la distribución de capas que tiene es la siguiente: una capa de entrada, una capa de proyección compartida para todas las palabras y finalmente la capa de salida. En la capa de entrada, todas las palabras son codificadas en *one-hot vector*; el cual es un vector con tantas posiciones como palabras tiene el vocabulario donde todas las celdas contienen un 0 excepto la celda que representa la palabra de entrada que tiene un 1. El resultado de la capa de entrada es proyectado en la segunda, la cual tiene una dimensión de D que es la dimensión de los *embeddings*. La capa de proyección obtiene los *embeddings*. Finalmente se pasa la proyección a la capa de salida y esta, obtendría como resultado un vector del mismo tamaño que el vocabulario donde cada celda representa la probabilidad de que esta palabra sea vecina de la palabra de entrada.

  Como la capa de proyección es la misma para todas las palabras, estas siempre se proyectarán en la misma posición y el orden de las palabras no afectará a la misma. Además, esta arquitectura utiliza una representación continua de la distribución del contexto; es por ello, que le pusieron el nombre de *Continuous Bag of Words*.

  Esta aproximación intenta predecir una palabra objetivo a partir de un cierto contexto. Para aclarar el concepto, se presenta la siguiente imagen extraída de [20].

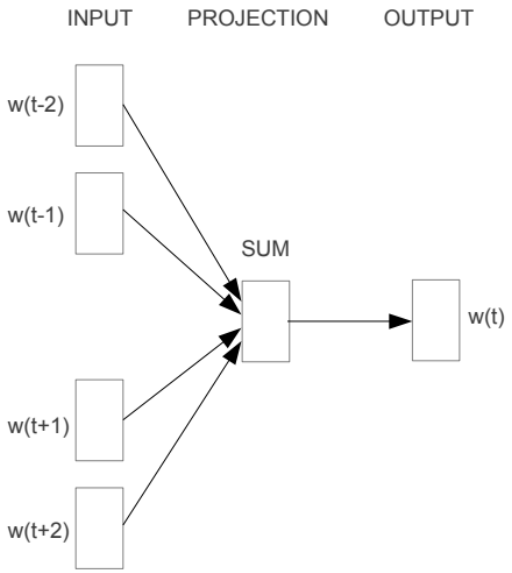


Figura 3.2: Arquitectura CBOW

  Como se observa en la *Figura 3.2*, la entrada de la red neuronal es un conjunto de palabras:
  $w_{(t-2)}, w_{(t-1)}, w_{(t+1)}, w_{(t+2)}$ donde 2 es el tamaño de la ventana de análisis. Posteriormente, en la segunda capa se proyecta la entrada y finalmente, la red neuronal nos proporciona como salida la $w_{(t)}$ que es la palabra que se buscaba. El 2 que aparece en la figura se substituiría por el tamaño del contexto a analizar.

El coste computacional del algoritmo que acabamos de comentar es de $N * D + D * log_2 V$ donde N es la dimensión de la capa de entrada. Además, tal y como comentan los autores de [22] el propósito de CBOW es maximizar la siguiente función:

$$\frac{1}{V}\sum_{t=1}^{V} log\ p(w_t \mid w_{t-2}...w_{t+2})$$

Donde V representaría el tamaño del vocabulario y $p(w_t \mid w_{t-2}...w_{t+2})$ es la probabilidad de una palabra dado un contexto.

- **Continuous Skip-gram:[21]** Es una arquitectura similar a la *CBOW* pero en vez de predecir la palabra a partir del contexto, se quiere encontrar una representación vectorial la cual permita predecir el contexto a partir de una palabra en una frase. En un estudio realizado en el documento [20] se observó un incremento notable en la calidad del resultado si se aumentaba el rango de búsqueda; a costa de un incremento en la complejidad computacional. Como en el caso anterior, también presentaremos gráficamente esta arquitectura *Figura 3.3*.

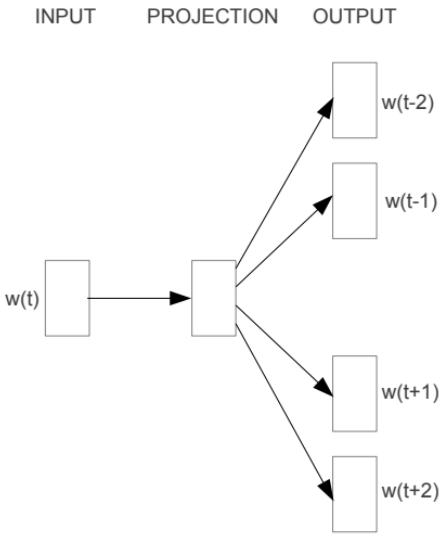


Figura 3.3: Arquitectura Continuous Skip-gram

El coste computacional de esta aproximación es $C * (D + D * log_2(V))$ donde C es la máxima distancia entre palabras. La función que busca optimizar esta aproximación según comentan los autores de [22] es:

$$\frac{1}{V}\sum_{t=1}^{V}\sum_{j=t-C, j\neq t}^{t+C} log\ p(w_j \mid w_t)$$

Donde V, como en el caso anterior, hace referencia al tamaño del vocabulario y $\sum_{j=t-C, j\neq t}^{t+C} log\ p(w_j \mid w_t)$ es la probabilidad de todo el contexto a partir de la palabra de entrada.

Tal y como comentan los autores en [20], cada una de estas aproximaciones tiene sus propias ventajas y desventajas. Si tenemos un corpus de entrenamiento pequeño, la mejor

opción será la de *Skip-gram* ya que en esta se representan mejor las palabras con escasa frecuencia. En contraposición, el *CBOW* es mejor utilizarlo en corpus más grandes ya que es más rápido que el *Skip-gramm* y además trabaja bastante bien con palabras muy frecuentes.

En el *Word2Vec* existen dos modelos de entrenamiento:

- ***softmax jerárquico*:** Permite una reducción en el número de evaluaciones de la capa de salida. En lugar de valorar V nodos, se calculan solo $log_2 V$. Está reducción, más que notable, de la evaluación de los nodos es debida a la representación en forma de árbol binario que se consigue en la capa de salida utilizando esta aproximación de entrenamiento. Es una representación donde las palabras son las hojas del árbol y las probabilidades de cada palabra se calculan utilizando el camino hasta llegar a ellas. Es interesante comentar que es necesario conocer las probabilidades de transición entre un nodo y sus hijos. Un ejemplo de esta aproximación se puede ver en la *Figura 3.4*:

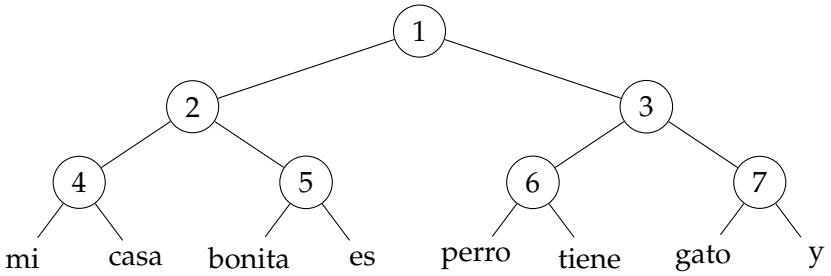


Figura 3.4: Ejemplo de árbol binario en el modelo de *softmax jerárquico*

Donde si se quiere calcular : $p(bonita \mid contexto) = p(\delta(1,2) \mid contexto) * p(\delta(2,5) \mid contexto) * p(\delta(5,bonito) \mid contexto)$.

- ***negative sampling*:** La principal idea de este algoritmo es limitar el número de vectores que necesitan ser actualizados en el entrenamiento. La elección de que vectores se deben actualizar depende de una distribución de ruido la cual nos permitirá distinguir los datos reales de los falsos.

En conclusión, tal y como comentan los autores en [20], cada modelo de entrenamiento comentado funciona mejor en unas situaciones u otras. *Negative sampling* es más eficiente en pocas dimensiones y además trabaja mejor con palabras frecuentes; no obstante *softmax jerárquico* funciona mejor con palabras que son poco frecuentes.

Una de las características más importantes del *Word2Vec* es que este, al igual que el *SHA-NN*, no se basa en *deep learning*. Esto es debido a que las arquitecturas que se han comentado anteriormente solo tienen una capa oculta y esto permite la reducción de la complejidad computacional posibilitando el entrenamiento de modelos más eficientemente.

Además de la representación de las palabras de los documentos, el resultado del *Word2Vec* nos permite determinar las relaciones sintácticas y semánticas realizando operaciones con los vectores obtenidos. Así por ejemplo, tal y como explican los autores en [1], si una vez entrenados los modelos de *embeddings* realizáramos la operación vec-

[1] https://code.google.com/archive/p/word2vec/

tor ('Paris') - vector('France') + vector('Italy') nos debería dar un vector muy cercano al vector('Rome').

3.2.3. Generación de resúmenes basados en redes neuronales

En esta sección se explicará la aproximación que se ha elegido para la generación automática de resúmenes basados en redes neuronales.

3.2.3.1. SHA-NN –Siamese Hierarchical Attention for Neural Networs–

Esta aproximación, que fue presentada en [8], se enmarca dentro de los algoritmos supervisados; esto es debido a que para el entrenamiento del sistema se requiere tanto el texto que queremos resumir como un resumen adecuado del mismo –lo que se quiere conseguir–.

El *SHAN-NN* plantea la tarea de generación de resúmenes extractivos como un problema de clasificación. Concretamente el modelo aprende si un resumen es correcto o incorrecto para un texto dado utilizando para ello la diferencia semántica que tienen ambos. Para la representación semántica de los documentos en esta aproximación se utiliza HAN –Hierarchical Attention Networks– que les permite representar el documento y el resumen a partir de la representación de las frases que lo componen. Estas representaciones de las oraciones a su vez provienen de la representación de sus palabras obtenidas utilizando *embeddings* estimadas mediante la herramienta *Word2Vec* que se ha explicado anteriormente.

Para entrenar el modelo, se construye para cada documento una muestra positiva –utilizando el texto original con el resumen original– y una muestra negativa –con el texto original y con un resumen de otro texto–. El modelo realiza la clasificación utilizando la función de activación *softmax* a partir de la concatenación del vector que representa el documento, el vector que representa el resumen y el vector que tiene la diferencia entre el documento y el resumen. El sistema consta de dos redes siamesas, en una de ellas se procesa el documento y en la otra se procesa el resumen.

En el proceso de entrenamiento del sistema, se calculan parámetros y algunos de estos se utilizan para la selección de las oraciones que formarán parte del resumen; en la fase de la generación del resumen, únicamente entra en juego la red que procesa el documento. El mecanismo de atención que utiliza para la generación de resúmenes pondera las frases y palabras utilizando para ello una de las dos redes neuronales que contiene. Este proceso le permite seleccionar las frases que mejor representan al texto en cuestión para la generación del resumen. Como podemos observar en el artículo [8], los resultados obtenidos por esta aproximación son muy prometedores y este es el motivo de la elección de este sistema para la generación de resúmenes basados en redes neuronales del presente trabajo.

3.2.4. Generación de resúmenes basados en algoritmos clásicos

Se pasarán a comentar los algoritmos clásicos que se han seleccionado para la generación automática de resúmenes. Los algoritmos elegidos se fundamentan en una aproximación no supervisada para la generación de los resúmenes; es decir, no requieren de un entrenamiento previo del modelo sino que a partir de un texto son capaces de generar un resumen basándose en las estadísticas que tienen las palabras en el texto.

3.2.4.1. LexRank:

Es un sistema de resúmenes automático no supervisado basado en grafos y el cual está inspirado tanto en técnicas de PageRank como de Hits. Se basa en la idea de que la importancia de una frase depende de la similitud que esta tenga con el resto de oraciones del texto. Además, utiliza un idf modificado basado en el coseno para calcular la similitud entre dos frases. Dos frases están conectadas si el coseno de la similitud entre ambas es superior a cierto umbral.
Para la selección de las frases que forman parte del resumen, busca cuáles son los nodos –oraciones– más centrales del texto; es por ello que existen diversas medidas de centralidad las cuales pasaremos a comentar:

- **Centralidad basada en el grado**: Donde la importancia de una frase depende del número de conexiones que esta tenga con el resto de nodos.
- **Centralidad basada en vectores propios**: Pondera cada una de las conexiones con la importancia del nodo con el que se conecta; por ello un nodo será más importante cuanto mayor sea el valor de los nodos a los que se conecta.

Este sistema finalmente incorpora una tarea de post-procesado para asegurar que el resumen proporcionado no tiene distintas oraciones con contenido similar.

En el caso de este trabajo, se ha utilizado la centralidad basada en el grado ya que es la que implementa el paquete utilizado para la generación automática de resúmenes basados en *Lexrank*.

3.2.4.2. LSA –Latent Semantic Analysis–:

Es un método de generación de resúmenes no supervisado el cual combina técnicas de frecuencia de término con descomposición de términos singulares. Es una de las técnicas más recientes que no utiliza redes neuronales para el resumen. Inicialmente se crea una matriz donde las columnas representan las diferentes frases del documento y la filas representan los términos. Cada una de las celdas de la matriz representa la importancia de una palabra en una oración y para rellenar esta matriz existen diferentes aproximaciones; se pasará a comentar algunas de ellas:

- **Número de ocurrencias**: Cada una de las celdas representa la frecuencia de una palabra en una frase.
- **Representación binaria del número de ocurrencias**: Si una palabra aparece en una frase, la celda pertinente estará rellena con un 1 si por el contrario no aparece la celda contendrá un 0.
- **TF-IDF –*Term Frequency-Inverse Document Frequency*–:** La celda está rellena con el TF-IDF. La importancia de una palabra depende de que si esta es muy frecuente en una frase y poco frecuente en el documento. Esta función se calcula como TF*IDF donde el TF –*Term Frecuency*– mide la frecuencia relativa de un término en un documento o frase y se calcula como: $tf(i,j) = \frac{n(i,j)}{\sum_k n(k,j)}$ (i es la palabra, j es la frase, $n(i,j)$ es el número de ocurrencias de la palabra en la frase y el sumatorio es el número de ocurrencias de todas las palabras en la oración). El término IDF –*Inverse Document Frecuency*– mide la importancia de un término en toda la colección de documentos

o en un documento y se calcula como: $idf(i) = log(\frac{|D|}{d_i})$($|D|$ es el número de frases del documento y d es el número de oraciones donde aparece el término i).

- **Modificación del TF-IDF**: Inicialmente se rellenan las celdas con los valores del TF-IDF, análogamente al caso anterior, posteriormente se calcula la media del TF-IDF para cada una de las filas y finalmente si el valor de cada celda es menor que la media, se modifica por un 0 y si por el contrario es mayor, se modifica por el valor de la media.

En la implementación utilizada en este trabajo para la generación de resúmenes basados en *LSA*, se utiliza como matriz de entrada una modificación de la aproximación basada en el número de ocurrencias; esta modificación es una normalización de la matriz sobre la mayor frecuencia obtenida.

El siguiente paso es el de aplicar el método de la descomposición del valor singular –SVD– en el cual se refleja la relación entre los términos y las oraciones. Además, utilizando este método se reduce el ruido de los datos mejorando el resultado final. El SVD descompone la matriz anterior como:

$$M = U\Sigma V^T$$

Donde M es una matriz *(m * n)*, U es una matriz *(m * n)* la cual representa palabras*conceptos extraídos, Σ es una matriz *(n * n)* que simboliza los valores escalares y finalmente V es una matriz *(n * n)* y representa las oraciones * conceptos extraídos. La matriz V^T representa la presencia de los conceptos en las oraciones y es la que se utilizará en todas las aproximaciones que se explicarán.

Finalmente existen diferentes algoritmos para la selección de las frases que formarán parte del resumen; pasaremos a comentar algunos de ellos.

- **Gong y Liu, 2001:[7]** Las columnas del V^T representan las frases de la matriz de entrada y las filas los conceptos extraídos del método SVD. Los conceptos más importantes están ordenados en la matriz; es decir, el concepto más importante está en la primera linea. Todo esto se observa mejor si se plantea un ejemplo.

	frase1	**frase2**	**frase3**
con1	0.45	0.55	0.22
con2	0.68	0.12	0.15
con3	0.1	0.369	0.489

  A partir de esta tabla se seleccionaría para cada uno de los conceptos más importantes la frase que mayor valor tiene. Por ejemplo para el primer concepto se seleccionaría la segunda frase, para el segundo concepto la primera frase y este paso se haría tantas veces como frases queramos en el resumen.

- ***Topic method:[23]*** En esta aproximación existe un paso previo a la selección de las oraciones. La principal idea de la aproximación es que existen tanto conceptos como sub-conceptos. Es por ello que en el resultado del método SVD se pueden agrupar distintos sub-conceptos que formen parte de otro concepto; de esta forma, es más fácil seleccionar las frases que representan al texto.

- **Mejora de la aproximación *Gong y Liu:*** Tal y como comentan los autores en [29] la aproximación de *Gong y Liu, 2001* tiene problemas; debidos principalmente a que no se pueden seleccionar más de una frase por concepto a pesar de que este sea el más importante del texto. Por esta razón, ellos plantean una variación en la parte de la selección de las frases que formarán parte del resumen.

  A partir de las matrices provenientes del proceso de SVD, calculan la longitud para cada una de las frases. Esta longitud la calculan multiplicando cada uno de los conceptos de la frase por su valor singular correspondiente; así los conceptos más importantes obtendrán mayor longitud. La fórmula que utilizan es:

$$S_k = \sqrt{\sum_{i=1}^{n} v_{k,i}^2 \sigma_i^2}$$

  Donde s_k es la longitud del vector de la k-ésima frase del texto, $v_{k,i}$ es la i-ésima palabra de la oración k y σ_i el valor propio de la palabra i-ésima. Además esta longitud será la métrica utilizada para la selección de las oraciones mas importantes del texto.

La aproximación utilizada en este trabajo para la selección de las frases que formarán parte del resumen en el modelo *LSA* es la última que se ha comentado.

3.2.4.3. LUHN:

Tal y como se comenta en [17], es un método heurístico basado en la puntuación de las frases dependiendo de la frecuencia de las palabras más importantes que contiene. Normalmente las palabras que aparecen muy frecuentemente en el texto están asociadas al tópico principal del texto; siempre y cuando se eliminen las *stopwords*. El algoritmo inicialmente elimina las *stopwords* del texto. Posteriormente, agrupa los términos viendo para ello la ortografía –por ejemplo "similar" y "similitud" irían en el mismo grupo– a esto se le llama un proceso de lematización. Una vez agrupados los términos se calcula la frecuencia de estos y se eliminan los grupos de términos que tienen una frecuencia menor que cierto umbral. A partir de aquí, se dividen las oraciones en fragmentos y se ponderan calculando el número de términos que contiene el fragmento y que dichos fragmentos estén dentro de algún grupo. Finalmente el peso de la frase depende del mayor valor que tenga un fragmento de esta. A partir de este resultado se seleccionan las oraciones con mayor valor para la elaboración del resumen pertinente.

3.2.4.4. TextRank:

Es una aproximación no supervisada basada en grafos al igual que *LexRank*. Utiliza el PageRank para extraer la palabras clave del documento. Los nodos de los grafos son las frases y las aristas miden la similitud entre los diferentes nodos basándose en el número de palabras que tienen en común; está es la principal diferencia que tiene el algoritmo *Textrank* frente al *Lexrank*. A diferencia del PageRank, las aristas no son dirigidas. Además, utiliza la misma técnica que el *LexRank* para seleccionar las frases más centrales.

3.2.5. Evaluación de resúmenes automáticos

En este apartado se expone la métrica de evaluación de resúmenes automáticos *Rouge –Recall-Oriented Understudy for Gisting Evaluation–* [16] que se utiliza en el presente tra-

bajo. Este, está inspirado en el Bleu –*Bilingual Evaluation Understudy*– que se utiliza para la evaluación de traducciones automáticas [24]. Realmente *Rouge* comprende un conjunto de métricas que permiten determinar la calidad sintáctica de un resumen generado comparándolo para ello con el resumen de referencia. Claro está que este sistema ayuda muy notablemente en la evaluación del presente trabajo y que sin esta, sería imposible la evaluación manual de los resultados; dada la gran cantidad de noticias que se han conseguido extraer en el corpus.

Además de este, también se presenta otro evaluador creado expresamente para el trabajo el cual determina la calidad semántica entre los resúmenes generados y los de referencia.

3.2.5.1. *Rouge*

Para la evaluación de los resúmenes utilizando *Rouge*, se ha empleado un paquete de *Python* con el mismo nombre que el sistema y que nos facilitará la utilización del mismo.

El paquete del *Rouge* utilizado incluye diversas variantes que pasaremos a comentar pormenorizadamente:

- Rouge-N: Se basa en el cálculo del *n-gram* donde n es la longitud de los de gramas que queremos comparar entre ambos resúmenes. Para calcularlo divide el número de n-gramas coincidentes en ambos resúmenes entre el número de n-gramas totales del resumen de referencia o del resumen generado según si estamos calculando el *recall* o la *precision*.

- Rouge-L: Se basa en el cálculo de la secuencia de longitud más larga coincidente en ambos resúmenes y que posteriormente se dividirá con el tamaño del resumen de referencia o del resumen generado según si estamos calculando el *recall* o la *precision*.

- Rouge-W: Es muy parecido al Rouge-L pero utiliza una técnica para la ponderación de las palabras. Esta ponderación intenta asignar mayor puntuación a palabras consecutivas que a palabras que a pesar de estar en secuencia tienen texto adicional. Esto que se ha comentado se entenderá de mejor forma si se plantea un ejemplo donde el texto de referencia es "Las casas son grandes".

ID	TEXTO	ROUGE-L
A	"Todas las casas son grandes"	1
B	"Las casas son grandes"	1

Tabla 3.1: Ejemplo de problema con Rouge-L

Como podemos observar en la *Tabla 3.1* tanto el texto A como el B tienen la misma valoración por parte del Rouge-L –siempre y cuando se calcule el *recall*– siendo el texto B idéntico al de referencia. El *Rouge-W* intenta resolver el problema planteado, asignándole una mayor puntuación al texto B.

- Rouge-S: Es muy similar al caso del Rouge-N pero pueden haber palabras no coincidentes entre medias de los fragmentos a evaluar. Estas ocurrencias no las había detectado el Rouge-N. Estas palabras no coincidentes son controladas mediante un parámetro que indica el contexto que debe considerarse.

Para el presente trabajo se han utilizado Rouge-L, Rouge-1 –unigramas– y el Rouge-2 –con bigramas–, que son las medidas más utilizadas en los trabajos recientes de resumen automático.

Para poder cuantificar el valor de solapamiento, el *Rouge* proporciona una serie de resultados referentes a este tales como Average_R que hace referencia al *recall*, Average_P que hace alusión a la *precision* y finalmente el Average_F que guarda relación con el *F-measure*.

El *recall* se traduce como cobertura y en el caso de la evaluación de los resúmenes en unigramas se calcula como:

$$Recall = \frac{número\ de\ solapamientos}{número\ de\ palabras\ del\ resumen\ de\ referencia}$$

La *precision* que se traduce como precisión y se calcula para los unigramas como:

$$Precision = \frac{número\ de\ solapamientos}{número\ de\ palabras\ del\ resumen\ generado}$$

El *F-measure* relaciona en una misma fórmula los términos *recall* y *precision*:

$$F_\beta = \frac{(\beta^2 + 1) * P * R}{\beta^2 * P + R}$$

Donde β es un factor que si es superior a 1 da más valor a la *precision* y si por el contrario es menor que 1 favorece al *recall*. El caso más habitual es cuando $\beta = 1$ y gracias a esto, se simplifica el cálculo:

$$F_1 = \frac{2 * P * R}{P + R}$$

3.2.5.2. Evaluador semántico

Para poder evaluar semánticamente los resúmenes, se ha propuesto una medida de evaluación que mide las diferencias de significado entre el resumen generado y el de referencia. Este sistema se basa fundamentalmente en las representaciones de las palabras –*embeddings*– comentadas anteriormente ya que a partir de estas, podemos ver las diferencias semánticas que tienen las palabras. Para la evaluación semántica, se han propuesto dos métricas que pasaremos a comentar pormenorizadamente.

- Total del resumen: En esta aproximación, se obtiene el *embedding* que identifica a un resumen. Para este fin, se obtiene el vector de representación de cada palabra, se suman todos los vectores de cada una de las palabras del resumen y finalmente se divide por el número de palabras que tiene el mismo. Una vez obtenido el vector que representa al resumen, se obtiene la palabra cuyo vector es más parecido al vector obtenido –centroide–. Una vez obtenidas las palabras que representan tanto al resumen generado como al resumen de referencia, se compara el *embedding* que representa cada resumen utilizando para ello la similitud coseno. La similitud del coseno mide el ángulo que forman dos vectores n-dimensionales en un espacio de representación; cuando menor sea este valor, menor ángulo formarán y mas similares serán los vectores. La fórmula para calcular la similitud del coseno es [26]:

$$Similarity(A, B) = \frac{A \cdot B}{|| A || \times || B ||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Donde A y B son el *embedding* del resumen generado y del resumen de referencia respectivamente. El resultado de la similitud coseno es un valor entre el rango de -1 y 1; cuanto mayor sea el valor, mas similares serán los vectores.

- Frases del resumen: En esta métrica, la idea principal es la de buscar la frase que más similitud tiene entre el resumen de referencia y el resumen generado. Para obtener el vector que representa una frase, se suman los *embeddings* de cada una de las palabras y se divide por el número de palabras que tiene la oración. Una vez obtenidos los vectores que representan las frases, se obtiene la palabra que tiene un vector más parecido al vector obtenido. A partir de los *embeddings* de las palabras que representan cada una de las frases, se comparan cada una de las frases del resumen generado con cada una de las oraciones del resumen de referencia obteniendo la que mayor similitud tiene para cada una de ellas. Posteriormente, se hacen las mismas comparaciones pero de modo simétrico; es decir, cada una de las frases del resumen de referencia contra las oraciones del resumen generado. Finalmente se suman todas las mejores similitudes y se divide por el número de frases que tienen ambos resúmenes. Para obtener la similitud entre dos frases –dos vectores– se utiliza, como en el caso precedente, la similitud coseno.

A pesar de haber hecho más pruebas como por ejemplo: incluir las *stopwords*, ponderar las palabras por frecuencia, ponderar las palabras por el valor TF-IDF, etc; no se han conseguido obtener criterios más discriminativos que con las métricas propuestas. Por esa razón, se ha decidido no incluirlas en el presente trabajo.

3.3 Diseño de la solución

En este apartado se detallará la arquitectura modular empleada, el diseño y la tecnología utilizada en la generación automática de los resúmenes así como para su posterior evaluación.

3.3.1. Arquitectura del sistema

El sistema que se presenta realiza cinco tareas bien diferenciadas. La primera tarea es la de almacenar las noticias en ficheros; gracias a esto, se separarán las noticias en distintos archivos según la función de estas –entrenamiento, test y desarrollo–. Además, esta tarea también es la encargada de normalizar el texto. La segunda tarea es la de entrenar dos modelos basados en redes neuronales –uno para el corpus en castellano y otro para el corpus en catalán– utilizando para ello las noticias separadas para este fin. La tercera tarea es la generación de los resúmenes basados en redes neuronales utilizando los modelos entrenados previamente. En cuarto lugar la generación de resúmenes utilizando los algoritmos clásicos que no necesitan un entrenamiento previo. Finalmente, tenemos el módulo que compara los resúmenes almacenados inicialmente para cada noticia con los que se han generado utilizando este proceso; es por ello que se ha decidido dividir el sistema en cinco módulos:

- `BDA a Fichero`: Es el módulo encargado de almacenar en ficheros las noticias de la base de datos separándolas para ello en distintas proporciones según sea la finalidad de estas –entrenamiento, test o desarrollo–. Esta separación se realizará tanto para el corpus de castellano como para el corpus de catalán. Además, es el módulo encargado de la *tokenización* del texto permitiendo así almacenar las noticias ya *tokenizadas* y que sean más fácilmente manejables en tareas posteriores.

- `Entrenamiento SHA-NN`: Es el módulo encargado de la generación de los modelos del sistema *SHA-NN* utilizando para ello las muestras de entrenamiento. En este módulo también está incluida la parte del ajuste de los parámetros utilizando el conjunto de noticias separado para desarrollo.

- `Generación SHA-NN`: Este módulo es el encargado de generar los resúmenes basados en redes neuronales a partir de los modelos entrenados en el apartado anterior y del texto de las noticias separadas para test.

- `Generación Algoritmos Clásicos`: Este es el módulo encargado de generar automáticamente resúmenes no supervisados utilizando para ello distintos métodos y algoritmos clásicos; estos algoritmos, como se ha comentado en apartados anteriores, no necesitan un entrenamiento previo.

- `Evaluador`: Este es el módulo capaz de proporcionar un veredicto, utilizando distintas métricas, sobre la similitud entre el resumen original, extraído de la noticia, y el generado utilizando los distintos métodos comentados anteriormente.

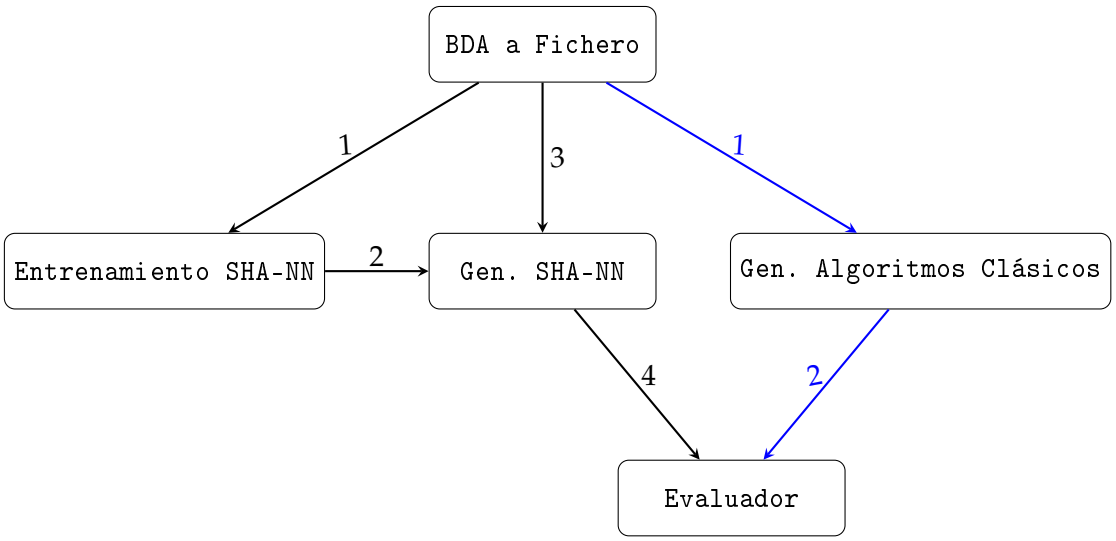


Figura 3.5: Arquitectura para la generación de resúmenes

Tal y como podemos observar en la *Figura 3.5*, existen dos maneras para poder generar el resumen de una noticia: utilizando el *SHA-NN* o utilizando algoritmos clásicos.

1. **Utilizando el *SHA-NN*:** En primer lugar se entrenaría el modelo a partir de las noticias separadas para este fin. Además, se ajustarían los parámetros del modelo utilizando las noticias separadas para el desarrollo. Posteriormente, el generador de resúmenes utilizaría los ficheros de test –donde están almacenadas las noticias de las cuales queremos generar el resumen para validar el sistema– y el modelo previamente entrenado y generaría los resúmenes. Finalmente, este módulo almacenaría un fichero donde en cada linea estarían tanto el resumen generado como el resumen original separados por una tabulación y el módulo `Evaluador` sería el encargado de, utilizando el fichero creado, comparar los resúmenes y proporcionar un veredicto.

2. **Utilizando Algoritmos Clásicos:** El generador de resúmenes con algoritmos clásicos solo necesita el texto original de las noticias. A partir de este, generaría los resúmenes y almacenaría en un fichero, de igual forma al caso anterior, el resumen generado y el resumen original separándolos con una tabulación. Finalmente, proporcionaría el documento al `Evaluador` para que los comparase y proporcionase el resultado.

3.3.2. Diseño detallado

En esta sección se concretará el diseño elegido a partir de la estructura de directorios escogida para albergar el código fuente relacionado con la generación de resúmenes.

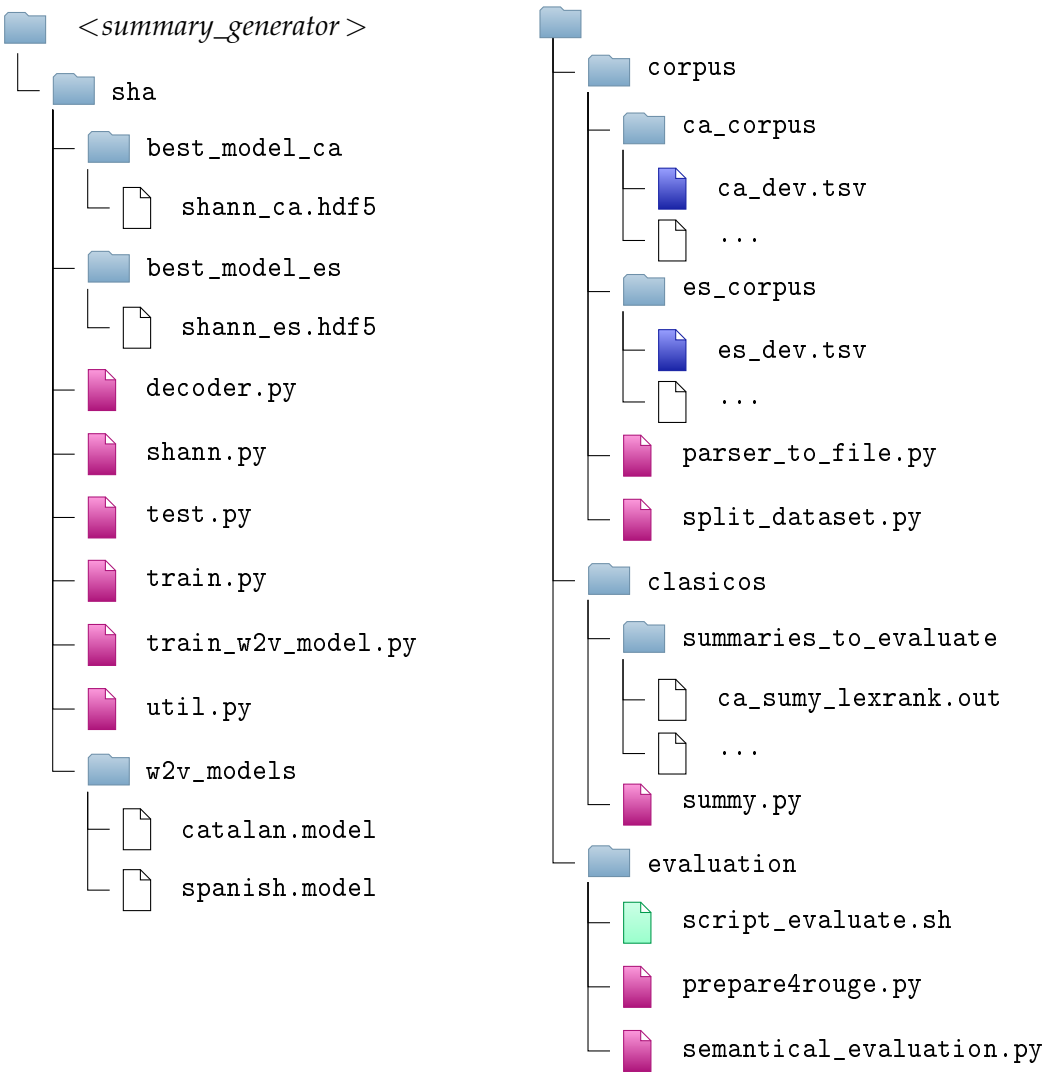


Figura 3.6: Estructura de directorios creada para la generación el corpus

Tal y como se observa en la *Figura 3.6*, inicialmente, existen cuatro carpetas en la raíz que se pasarán a comentar pormenorizadamente. En primer lugar se ubica la carpeta `sha` en la cual se almacena tanto el fichero utilizado para el entrenamiento del modelo *train.py* como el fichero empleado para la generación de los resúmenes *test.py*. Además, en esta misma carpeta también se almacenan los modelos de *embeddings*, generados con *Word2Vec*, necesarios para el entrenamiento del *SHA-NN*. Finalmente, en este direc-

torio se guardan los ficheros relacionados con la implementación interna del *SHA-NN*: *shann.py* y *util.py*.

En la carpeta de `corpus`, se almacenan todos los ficheros necesarios tanto para el proceso de descarga de noticias de la base de datos y posterior almacenamiento de estos en ficheros como los corpus propiamente dichos. El fichero *parser_to_file.py* es el encargado de descargar las noticias de la base de datos y almacenarlas en ficheros creando para ello archivos con una extensión CSV –Comma-separated values–. Este fichero también es el encargado de *tokenizar* tanto el texto como el resumen de las noticias. Asimismo, en esta misma carpeta, también se ubica el fichero *split_dataset.py* que es el encargado de separar inicialmente las noticias por idioma y posteriormente entre ficheros de entrenamiento, desarrollo y test. Finalmente se almacenan los ficheros resultantes de la ejecución del *split_dataset.py*.

La carpeta `clasicos`, es la encargada de almacenar todos los ficheros necesarios para la generación automática de resúmenes basados en algoritmos clásicos. En esta, se ubica el fichero *sumy.py* el cual se encarga de generar un resumen para cada uno de los algoritmos clásicos elegidos. En esta carpeta también se encuentra el directorio *summaries_to_evaluate* donde se almacenan los ficheros en los cuales cada linea contiene el resumen generado junto al resumen original para cada una de las noticias. Este fichero se crea para cada uno de los algoritmos escogidos y son los ficheros que utiliza el módulo `evaluador` para compararlos.

Finalmente en la carpeta `evaluation` tenemos el fichero *script_evaluate.sh* el cual es un script que ejecuta el *Rouge*. Además, encontramos el fichero *prepare4rouge.py* mediante el cual se crean las carpetas y archivos necesarios para la ejecución del *Rouge*. Finalmente, tenemos el archivo *semantical_evaluation.py* mediante el cual se consigue comparar semánticamente los resúmenes.

3.3.3. Tecnología utilizada

Tanto para la parte de generación de resúmenes de todos los sistemas como para la fase de importación de las noticias desde la base de datos a los ficheros, se ha utilizado *Python*. Este, es un lenguaje de programación multiparadigma, ya que soporta orientación a objetos y de programación imperativa. Es un lenguaje interpretado, con tipado dinámico y multiplataforma. Las características más relevantes que nos han llevado a la elección de este lenguaje de programación han sido:

- ***Opensource***: Código abierto.
- **Simplificado y rápido**: Es un lenguaje de *scripting* –si se pretende conseguir algo sencillo, con pocas lineas se puede conseguir–.
- **Flexible**: No se declara el tipo de las variables.
- **Portable**: Entre distintos sistemas operativos.
- **Comunidad**: Tiene una comunidad muy amplia y activa que permite resolver cualquier problema muy rápidamente.
- **Paquetes**: Proporciona un conjunto elevado de paquetes que agilizan la implementación de muchas funcionalidades.

Aunque no se ha implementado el código del *Rouge* como tal, sí que se ha utilizado. Se disponía de un fichero en *Perl* para su cálculo. Para poder usarlo más fácilmente, se ha tenido que crear un *script* de *bash* con el que se ejecutan secuencialmente varias evaluaciones.

Además, para la generación de gráficas presentadas en los apartados de experimentación y resultados, se ha utilizado el lenguaje de programación *R*. Algunas de las características más relevantes que nos han llevado a la elección de este lenguaje de programación para la generación de gráficas han sido:

- **Proyecto colaborativo y abierto**: Se puede descargar el código y modificarlo para incluir mejoras.
- **Amplia gama de herramientas estadísticas**: Para el análisis de datos y también la generación de gráficas de alta calidad.
- **Gran volumen de datos**: Permite el manejo de grandes volúmenes de datos.
- **Portable**: Entre distintos sistemas operativos.
- **Comunidad**: Tiene una comunidad muy amplia y activa que permite resolver cualquier problema muy rápidamente.
- **Paquetes**: Proporciona un conjunto elevado de paquetes que agilizan la implementación de muchas funcionalidades.

3.4 Desarrollo de la solución

En este apartado se detallará el desarrollo seguido en cada uno de los módulos explicados en la *Sección 3.3*.

3.4.1. BDA a Fichero

Como se ha comentado en la parte del diseño de la solución, se han utilizado dos ficheros para el proceso de importación de las noticias desde la base de datos a los archivos. En primer lugar, se explicarán los paquetes utilizados en estos ficheros.

- **html:** Se utiliza para la eliminación de los *tags html* que tienen los datos extraídos de la noticia tanto para el caso del resumen como para el caso del texto.
- **pymongo:** Se utiliza en el acceso a la base de datos almacenada para descargar todas las noticias guardadas en esta permitiendo así poder almacenarlas en ficheros.
- **nltk.tokenize:** Este paquete se utiliza para la tokenización del texto. La tokenización utilizada se explicará mas adelante.
- **pandas:** Este módulo se utiliza para el almacenamiento de las noticias en ficheros con la extensión *csv*.
- **sklearn.utils:** Se utiliza la función *shuffle* de este paquete para barajar las distintas noticias. Esta función se utiliza en la separación de noticias entre ficheros de entrenamiento, desarrollo y test.

Después de explicar los paquetes utilizados para la implementación del código, se pasará a comentar la funcionalidad de cada fichero. El archivo *parser_to_file.py* es el encargado de descargar las noticias de la base de datos y almacenarlas en un fichero. Antes de guardarlo, normaliza los textos tanto del resumen como del cuerpo de la noticia utilizando para ello la llamada a la siguiente función:

```
1    content = " ".join(word_tokenize(html.unescape(content.lower().replace("\n"
         ," ").replace("\t"," "))))
2    summary = " ".join(word_tokenize(html.unescape(summary.lower().replace("\n"
         ," ").replace("\t"," "))))
```

Código 3.1: Llamada a la función que tokeniza el texto

En estas llamadas, se observa como se pasa el texto a minúscula –content.lower()–, posteriormente, se reemplazan tanto los saltos de linea como las tabulaciones en espacios –con el replace–, seguidamente, se eliminan los *tags html* –con el html.unescape– y este resultado se pasa a la función *word_tokenize* la cual separa un texto por sus signos de puntuación y por sus espacios y lo convierte en un vector. Finalmente, el resultado de la función *word_tokenize* se concatena separando los distintos elementos del vector mediante espacios en blanco.
Además del fichero anteriormente comentado, se ha implementado otro fichero encargado de la separación de las noticias en tres partes: entrenamiento, desarrollo y test. En este, inicialmente, se carga el resultado del fichero anterior y se separan las noticias por idioma. Una vez separadas las noticias por idiomas, se agrupan las noticias de cada uno de los periódicos y se mezclan las noticias. Posteriormente, se separan para cada uno de los periódicos el 90 % de las noticias para entrenamiento, el 5,5 % para desarrollo y el resto para test. Este proceso se hace de esta manera para mantener las mismas proporciones entre los distintos periódicos de estudio escogidos. Finalmente, se guardan las noticias ya separadas por idioma y por el uso que se les va a dar a estas.

3.4.2. Entrenamiento SHA-NN

Para poder entrenar la red neuronal utilizada para la generación de resúmenes basados en *SHA-NN*, se requiere de un modelo de *embeddings*. Para poder crear los *embeddings* –tanto para el caso del corpus en castellano como para el de catalán–, se ha utilizado el paquete "gensim" [2]. De este paquete se ha utilizado Word2vec que es un herramienta que crea modelos de *embeddings* a partir de un conjunto de parámetros como ya se comento en la *Sección 3.2*. La llamada a la función utilizada para la creación de los *embeddings* ha sido la siguiente.

```
1  Word2Vec(data, size=300, min_count=5, window=5, workers=12, sg=1, hs=0,
       negative=5)
```

Código 3.2: Llamada a la función que crea el modelo de *embedding*

- **data**: Son todos los textos de las noticias y de los resúmenes almacenados dentro de un vector.
- **size**: Número de dimensiones o caracteríticas que tendrán los *embeddings*.
- **min_count**: Número de veces que ha de aparecer una palabra como mínimo para que sea considerada en los *embeddings*.

[2] https://pypi.org/project/gensim/

- **window**: Número de palabras que forman parte de un contexto.
- **workers**: Número de hilos que se utilizarán mientras se entrena el modelo.
- **sg**: Es la arquitectura utilizada en el modelo. El 1 significa CBOW o Continuous Bag of Words –predecir una palabra a partir del contexto– y 0 significa Continuous Skip gram –predecir el contexto a partir de la palabra–; ambas arquitecturas están explicadas en la *Sección 3.2*
- **hs**: Son los modelos de entrenamiento utilizados; 1 significa que se utiliza la *softmax jerárquico* para el entrenamiento y 0 si se utiliza el *negative sampling* para el entrenamiento. Ambos modelos también están explicados en la *Sección 3.2*
- **negative**: Cuantas palabras de ruido se deben incluir.

A partir de los modelos de *embeddings* generados, se ha de entrenar el generador automático de resúmenes. Este generador es el *SHA-NN* que fue presentado en [8] y está explicado en la *Sección 3.2*.

Para el entrenamiento de *SHA-NN* se han utilizado distintos hiper-parámetros que pasaremos a comentar:

- **steps_per_epoch**: Son el número de pasos (lotes de muestras) que se deben utilizar para completar un *epoch*. En nuestro entrenamiento se ha utilizado 5000.
- **epochs**: Número de iteraciones de todos los datos que se han de realizar para entrenar el modelo. En nuestro caso, el sistema se ha entrenado durante 20 *epochs*.
- **funciones de activación**: Son funciones para introducir no linealidad en los modelos –si usamos funciones de activación no lineales buscamos tener un comportamiento parecido a los *kernels* de las máquinas de vector soporte para conseguir un comportamiento no lineal–. En el sistema se han utilizado diferentes funciones de activación las cuales se han explicado en la *Sección 3.2*:
  - **Tangente hiperbólica**: Se utiliza internamente en la capa *LSTM*.
  - **Softmax**: Se utiliza en la capa de salida para calcular la distribución de probabilidad pertinente.
- **dropout**: Es una técnica de regularización donde se especifica el porcentaje de nodos que no funcionarán intencionalmente en una capa para cada iteración. En nuestro caso solo se ha utilizado para la capa de entrada y con un valor de 0.3.
- **arquitectura**: Se ha utilizado *LSTM* que es una arquitectura para una *recurrent neural network* como se ha comentado en la *Sección 3.2*.
- **dimensión de la LSTM**: Tamaño de los vectores de salida de la red *LTSM*. En nuestro caso se ha utilizado 512.
- **función *loss***: Función objetivo; en nuestro entrenamiento se ha utilizado la *categorical_crossentropy*.
- **optimizer**: Optimizador a utilizar. En nuestro entrenamiento se ha utilizado *Adam*.
- **factor de aprendizaje**: Respecto que factor varían los pesos en cada iteración. Se ha utilizado el valor por defecto del *Adam* que es 0,001.

Los valores de los hiper-parámetros comentados anteriormente han sido heredados de la experimentación del *SHA-NN* en [8] sobre el corpus del *CNN/Daily Mail* el cual es similar al corpus que se ha obtenido en el presente trabajo –el corpus es sobre el mismo dominio y se utilizan las mismas técnicas–. Debido a que no se ha hecho una experimentación extensa sobre los hiper-parámetros, a causa del escaso tiempo que tenemos para la realización del TFG, se deja como trabajo futuro una puerta abierta a obtener mejoras en los resultados.

3.4.3. Generación SHA-NN

Este es el módulo del sistema encargado de generar los resúmenes a partir de los modelos de *embeddings* y *SHA-NN* previamente entrenados. Después de haber instanciado ambos objetos, llamaremos a la función de generar resúmenes pasándole parte de los parámetros utilizados; como por ejemplo el numero de frases que ha de tener el resumen, número de muestras positivas y negativas, etc.
A partir de los resúmenes generados y de los resúmenes originales, se creará un fichero específico para que el módulo `Evaluador` sea capaz de compararlos y proporcionar los resultados pertinentes.

3.4.4. Generación Algoritmos Clásicos

Para la generación de resúmenes utilizando algoritmos clásicos, se han utilizado distintos paquetes de *Python*:

- **pandas**: Para poder leer los ficheros en los que estaban ubicados las noticias.
- **sumy**: Paquete que se ha utilizado para generar todos los resúmenes utilizando los algoritmos clásicos.
- **os**: Para poder leer todos los archivos de una determinada carpeta.
- **stop_words**: Que contiene las stop_words tanto para castellano como para catalán.

En primer lugar se instancian los distintos sistemas de resumen; aquí se tuvo un problema ya que al tener un corpus de noticias en catalán, se necesitaba tener un *stemmer* del mismo idioma pero la librería *sumy* [3] no lo incluía en sus repositorios. Después de una exhaustiva búsqueda, se concluyó que tenían un *stemmer* en catalán que aunque no lo habían elaborado ellos, si que lo habían verificado y lo tenían incluido en su página web. Es por ello que se consiguió descargar e incorporar a la librería para así poder continuar con el trabajo. Además, para la generación del resumen, se necesitaba un *tokenizer* del catalán así que como no estaba creado se tuvo que entrenar uno utilizando para ello la librería *nltk.tokenize.punkt* [4] junto con las noticias almacenadas del catalán.

[3] https://pypi.org/project/sumy/
[4] https://www.nltk.org/_modules/nltk/tokenize/punkt.html

3.4.5. Evaluador

Para el módulo de evaluación sintáctica se utilizó *Rouge* que es un paquete implementado en *perl* y que incluye varios criterios para calcular la similitud entre resúmenes. La llamada que se ha hecho a *Rouge* ha sido:

```
perl /home/nandotorre/Escritorio/SummarizationTFG/evaluation/rouge/ROUGE
    -1.5.5.pl -e /home/nandotorre/Escritorio/SummarizationTFG/evaluation/
    rouge/data -a    -n 2 -c 95 -r 1000 ./file/settings.xml
```

Código 3.3: Llamada a la función *Rouge*

- **-n**: El número de ngramas máximo a utilizar.
- **-c**: Intervalo de confianza.
- **-r**: Número de muestras en cada *resampling* para calcular el intervalo de confianza.
- **-a**: Evaluar todos los resúmenes.
- **file**: Nombre de la carpeta donde se han creado los archivos y directorios necesarias para la evaluación del *Rouge*.

Este sistema proporciona un resultado a modo de tabla donde para cada uno de los criterios utilizados para la evaluación –en nuestro caso Rouge-1, Rouge-2, Rouge-L– presenta tanto los resultados para las distintas medidas –*Average_R, Average_P y Average_F*– como su intervalo de confianza.

Para el módulo de evaluación semántica, se utilizaron distintos paquetes del *Python*:

- **csv**: Paquete utilizado para la importación de los ficheros a utilizar.
- **numpy**: Utilizado para dividir los elementos de un vector entre un determinado valor; en nuestro caso el número de palabras.

Para esta evaluación semántica, se desarrollaron varios métodos en los cuales se calculaban las distintas métricas. La idea inicial de estos métodos era comparar ambos resúmenes utilizando para ello las similitudes semánticas de sus palabras.

3.5 Marco Experimental

En esta sección, se explicarán los distintos datos obtenidos en el proceso de experimentación realizado para los corpus de las noticias. Se presentarán tanto datos referentes a las noticias de cada uno de los corpus como comparaciones entre los corpus obtenidos y otros corpus del estado del arte.

3.5.1. Estadísticas de los corpus

Inicialmente, se ha considerado una noticia válida para el presente trabajo si tanto el tamaño del resumen como el tamaño del texto era de al menos 10 palabras. Las estadísticas del corpus de catalán referentes al número de noticias de las que no se han conseguido extraer un resumen o un texto válido y que por lo tanto no se pueden utilizar en el presente estudio se muestran en la *Tabla 3.2* donde: **E.Texto** hace referencia a las noticias con un texto no válido, **E.Resumen** son noticias con un resumen no válido y **N.Utilizables** son las noticias finales después de la eliminación de las erróneas.

Periódico	Noticias	E.Texto	E.Resumen	N.Utilizables
ara	64 024	13	2910	61 101
dbalears	1567	0	108	1459
diaridegirona	13 104	0	369	12 735
diarilaveu	39 673	53	11 222	28 398
elpuntavui	6956	2	94	6860
eltemps	4659	9	100	4550
naciodigital	56 240	3 211	3496	49 533
regio7	11 806	4	548	11 254
vilaweb	35 817	1	50	35 766
Total	233 846	3293	18 897	211 656

Tabla 3.2: Estadísticas del corpus final de catalán

En la columna de noticias con error por texto, contabilizamos todas aquellas noticias que como mínimo no cumplen el requisito comentado para el texto. Por otro lado, contabilizamos un error por resumen, solo cuando el texto cumple los requisitos establecidos pero el resumen no. Esta afirmación se aplicará tanto para la tabla que acabamos de presentar como para la *Tabla 3.3*.

Como podemos observar en la *Tabla 3.2*, son más las noticias que se han descartado por el resumen que las que se han descartado por el texto; en concreto casi seis veces más. El periódico en el que más noticias se han descartado a causa de la longitud de su texto ha sido **naciodigital** con 3211 noticias. Por otro lado, el noticiario en el que más documentos se han considerado no válidos a causa de la longitud de su resumen ha sido **diarilaveu** con 11 222 noticias; esto es debido a que muchas noticias de este periódico no contenían resumen. Finalmente, el periódico en el cual se han descartado menos noticias en total ha sido **vilaweb**; donde de las 51 noticias descartadas, 50 son debidas a fallos en la longitud del resumen.

Después de la eliminación de las noticias en las que no se ha conseguido un resumen o un texto con más de 9 palabras y por tanto no son utilizables, el corpus de catalán se ha quedado con 211 656 noticias que supone el 90,5 % de las noticias iniciales.

En la *Tabla 3.3*, se presenta una tabla análoga a la anterior pero referente al corpus de castellano.

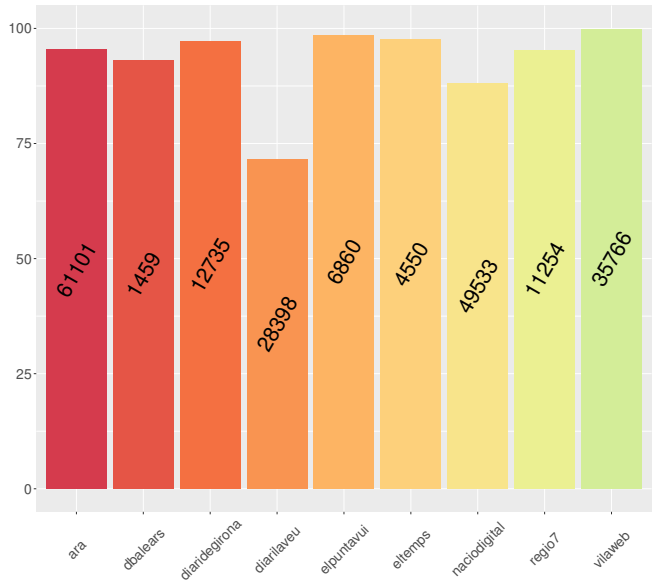
Periódico	Noticias	E.Texto	E.Resumen	N.Utilizables
20minutos	302 486	279	6	302 201
abc	126 317	111	2904	123 302
diariodemallorca	12 247	1	145	12 101
elconfidencial	38 098	0	141	37 957
eldiario	9341	3	11	9327
elespañol	49 920	47	234	49 639
elindependiente	25 811	7	15 667	10 137
elpais	85 271	2	4436	80 833
expansion	39 728	5	176	39 547
laser	15 355	13	2233	13 109
lasprovincias	6912	4	397	6511
levante	15 645	7	174	15 464
publico	13 039	1	32	13 006
ultimahora	32 131	0	419	31 712
Total	772 301	480	26 975	744 846

Tabla 3.3: Estadísticas del corpus final de castellano

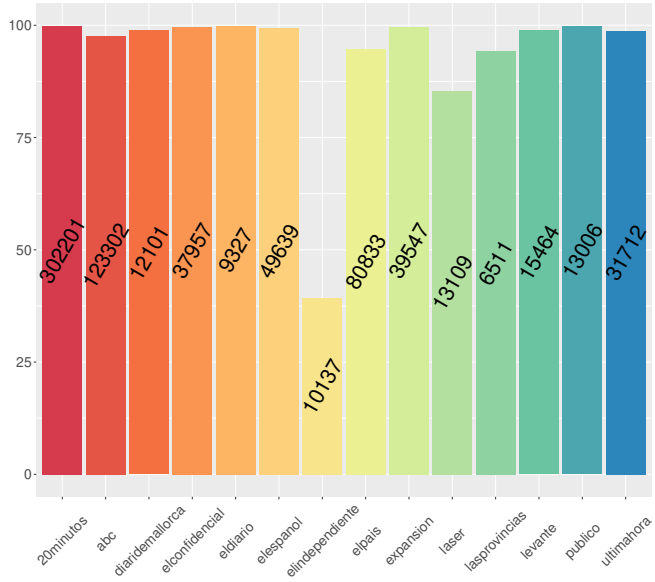
Como podemos ver en la *Tabla 3.3*, al igual que en el caso de catalán, son más las noticias que se han descartado por errores en el resumen que por errores en el texto; en esta ocasión 56 veces más. El periódico en el que se han descartado más noticias a causa del tamaño de su texto ha sido **20minutos** con 279 noticias. Por otra parte, el periódico que más noticias se han descartado por la longitud de su resumen ha sido **elindependiente** con 15 667 noticias; debido principalmente a la no existencia de resumen en esas noticias. Finalmente, el diario que menos noticias tiene como no válidas ha sido **eldiario** con solo 14.

Después de la eliminación de las noticias que no cumplían los requisitos anteriormente comentados nos quedamos con 744 846 noticias, lo que supone el 96,4 % de las noticias iniciales del corpus de castellano.

Para poder visualizar mejor estos datos, se han propuesto unas gráficas donde se observa para cada uno de los periódicos de cada corpus, el porcentaje de noticias utilizables.



(a) Noticias utilizables para cada periódico del corpus de catalán



(b) Noticias utilizables para cada periódico del corpus de castellano

Figura 3.7: Comparativa de noticias utilizables para cada periódico del corpus

Como podemos observar, la *Figura 3.7a* y la *Figura 3.7b* muestran como la mayoría de los periódicos tienen un tamaño de resumen y de texto que se ha considerado válido. Algunos de los periódicos donde se han tenido bastantes noticias con fallos ha sido por ejemplo el caso del **diarilaveu** –corpus de catalán– solo el 71,6 % de las noticias se pueden utilizar; esto es debido principalmente a que no se ha conseguido obtener el resumen de 11 222 noticias. Otros casos –también en el corpus de catalán– donde no se ha conseguido superar el 95 % de las noticias utilizables han sido en **naciodigital y dbalears**.
En el caso del corpus de castellano, se pueden observar unos porcentajes bastante altos a excepción de **elindependiente, laser, lasprovincias y elpais**. En el primero, solo el 40 % de las noticias tenían resumen y por ello el 60 % restante no se puede utilizar para el presente estudio. En el caso de **laser, lasprovincias y elpais**, a pesar de que tienen unos porcentajes bastantes altos, no se ha conseguido superar el 95 % de las noticias utilizables.

3.5.2. Estadísticas del contenido de las noticias

Una vez eliminadas las noticias que han presentado problemas en el corpus, se pasará a comentar cuales son las estadísticas referentes a los contenidos de las noticias. Las estadísticas iniciales que se proporcionan están relacionadas con el número de *tokens* obtenidos en el total de noticias para cada uno de los periódicos. La *Tabla 3.4* muestra las estadísticas del corpus en catalán y la *Tabla 3.5* las de castellano.

Periódico	***Tokens* Texto**	***Tokens* Resumen**
ara	35 013 009	1 194 828
dbalears	644 356	24 083
diaridegirona	6 066 182	312 240
diarilaveu	13 846 094	669 043
elpuntavui	3 164 767	203 143
eltemps	6 764 245	285 573
naciodigital	23 713 705	1 250 787
regio7	5 000 776	237 817
vilaweb	22 268 474	2 813 013
Total	116 481 608	6 990 527

Tabla 3.4: Estadísticas de *tokens* del corpus de catalán

Como podemos observar en la *Tabla 3.4*, el periódico que contiene un mayor número de *tokens* en el texto es el **ara** con casi el 30 % del total de *tokens* del corpus de catalán. Esto es debido a que es el periódico que más noticias aporta a su corpus. En contraposición, el periódico que menos *tokens* aporta al texto es **dbalears**. En el caso de *tokens* en el resumen, se observa como a pesar de que **vilaweb** aporta menos noticias que el **ara**, tiene un mayor número de *tokens* para el resumen. Este resultado es debido, como se comentará más adelante, al mayor tamaño de media que tienen los resúmenes del primero frente al segundo.

Periódico	***Tokens* Texto**	***Tokens* Resumen**
20minutos	128 224 770	8 571 376
abc	62 516 023	2 767 377
diariodemallorca	5 895 082	258 596
elconfidencial	35 304 597	1 130 674
eldiario	4 965 132	328 569
elespañol	33 051 996	1 833 913
elindependiente	8 649 862	258 737
elpais	66 799 974	1 727 171
expansion	21 748 886	1 907 459
laser	4 846 115	312 038
lasprovincias	3 495 526	152 623
levante	7 739 606	315 642
publico	9 388 722	490 919
ultimahora	11 340 186	1 624 656
Total	403 966 477	21 679 750

Tabla 3.5: Estadísticas de *tokens* del corpus de castellano

En cuanto a los resultados reflejados en la *Tabla 3.5*, observamos que el periódico que más *tokens* tiene es **20minutos**; esto es debido a que es el periódico que más noticias

aporta al corpus del castellano. Además, a pesar de que **eldiario** tiene 3782 noticias menos que el **laser**, tiene más *tokens* en el texto y en los resúmenes. Esto es debido, como se verá más adelante, al mayor tamaño de media que tienen tanto en el resumen como en el texto las noticias del primero frente al segundo.

Una vez eliminados los símbolos de puntuación y otros símbolos, se pasará de hablar de *tokens* a hablar de palabras. A continuación se presentan unas tablas donde se muestra el número medio de palabras por noticia, el número medio de palabras por resumen y la talla del vocabulario tanto para el caso del texto como para el caso del resumen.

Periódico	AVG Texto	AVG Resumen	Voc.Texto	Voc.Resumen
ara	508,91	18,48	346 605	60 306
dbalears	391,96	15,30	41 615	4455
diaridegirona	422,51	23,23	136 555	28 041
diarilaveu	432,11	22,19	286 480	38 016
elpuntavui	415,34	27,87	76 153	21 422
eltemps	1319,17	56,54	158 477	27 450
naciodigital	425,48	23,55	293 312	74 013
regio7	393,29	20,01	116 933	26 969
vilaweb	561,49	72,93	232 875	66 399
Total	541,14	31,12	778 974	152 912

Tabla 3.6: Estadísticas generales a nivel de palabra del corpus de catalán

Como podemos observar en la *Tabla 3.6*, la media de palabras en el texto de cada periódico es muy parecida a excepción del periódico **eltemps**, el cual tiene de media 1319,17 palabras; que es más del doble que el resto de periódicos. Esto es debido a que **eltemps** es un semanario y por norma general, estos, tienen un tamaño superior en el texto respecto a los diarios. Además, el periódico que presenta una media en el texto más bajo es **dbalears**. En el caso de los resúmenes, **dbalears** tiene 15,3 palabras de media y es el menor de todos los periódicos que se han analizado para este corpus. Además, en los resúmenes, también tenemos el periódico **eltemps** que como en el caso anterior tiene un número medio de palabras alto y **vilaweb** que aunque no era el que mayor media tenía en el texto, sí que tiene la mayor media de palabras en los resúmenes; superando incluso el periódico **eltemps** por 16,39 palabras de media.

En cuanto al vocabulario del texto, se observa como el del **ara** es superior al del resto de periódicos; esto era de esperar ya que como se ha comentado anteriormente, es el que mayor número de noticias aporta al corpus en catalán. Además, se observa como a pesar de que **diarilaveu** tiene 7368 noticias menos que **vilaweb**, tiene un vocabulario con 53 605 palabras más; esto puede ser debido a que el primero tiene una mayor diversidad de palabras que hacen que su vocabulario sea mas extenso. Finalmente se puede notar que aunque **eltemps** tiene 8185 noticias menos que **diaridegirona** su vocabulario es superior; esto es debido a que el número de palabras de media del primero en el texto es tres veces superior a las del segundo.

El caso del vocabulario del resumen es diferente, el periódico **naciodigital** a pesar de no aportar el mayor número de noticias al corpus, inferior al del **ara**, si que tiene el mayor vocabulario en los resúmenes. Esto sucede a causa de que tiene casi 5 palabras más de media en el resumen de la noticia que el periódico **ara**. Además, aunque el **ara** tiene un mayor número de noticias que el periódico **vilaweb**, tiene un menor número de palabras en el vocabulario del resumen; esto es debido a que el primero tiene de media 54.45 pala-

bras menos en cada resumen que el segundo. Finalmente se encuentra **dbalears** que tiene el menor vocabulario para el resumen ya que es el que menos noticias aporta al corpus y menor número de palabras medio para el resumen tiene.

Periódico	AVG Texto	AVG Resumen	Voc. Texto	Voc. Resumen
20minutos	374,43	26,04	712 694	161 636
abc	447,02	21,00	511 475	93 292
diariodemallorca	433,28	20,41	136 922	25 295
elconfidencial	826,75	28,23	363 723	56 451
eldiario	469,88	32,18	104 845	26 390
elespañol	590,57	33,42	362 994	70 253
elindependiente	766,45	23,93	197 838	22 586
elpais	733,62	20,48	430 969	65 557
expansion	490,12	44,52	272 847	47 950
laser	328,84	22,48	126 026	28 443
lasprovincias	480,68	22,02	98 892	15 133
levante	444,48	19,55	154 516	27 989
publico	639,95	34,28	174 896	35 438
ultimahora	318,75	46,34	183 651	61 243
Total	524,63	28,21	1 578 653	271 979

Tabla 3.7: Estadísticas generales a nivel de palabra del corpus de castellano

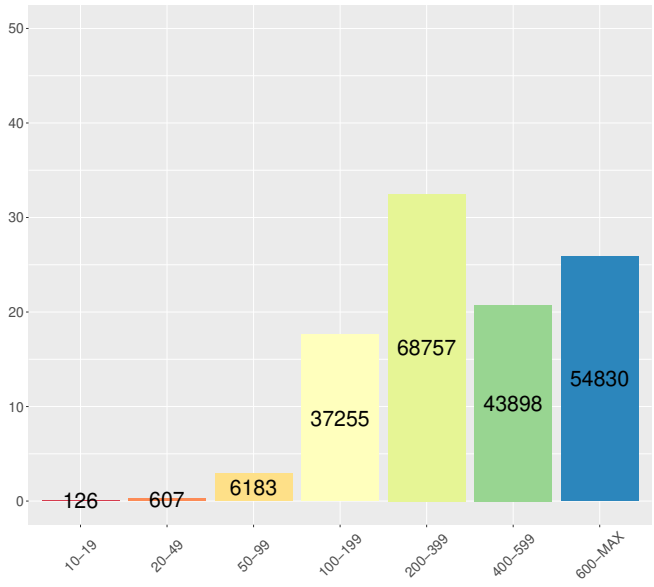
Si analizamos las estadísticas de contenido de castellano, podemos observar en la *Tabla 3.7* que el periódico que tiene una menor media de palabras en el texto es **ultimahora**, el que tiene una media mayor es **elconfiencial** y en el resto de periódicos se observa que la media está entre los dos rangos comentados anteriormente pero destacando los que están más cerca del rango inferior. En el caso de los resúmenes, notamos que el periódico que tenía menor tamaño en el texto –**ultimahora**– es el que mayor tamaño medio tiene en el resumen y el que tenía mayor tamaño en el texto – **elconfidencial**– tiene 18.11 palabras menos de media en el resumen que el mayor.

En cuanto al vocabulario del texto se puede notar como **20minutos** es el que tiene un vocabulario mayor, esto es debido principalmente a que las noticias de este periódico ocupan casi el 40 % del corpus total para castellano. En el caso de **elconfidencial**, a pesar de solo tener 6245 noticias más que **ultimahora**, tiene casi el doble de palabras en el vocabulario; esto es debido, tal y como se ha comentado anteriormente, a la diferencia de palabras de media que tienen las noticias de uno y de otro periódico. En el caso del vocabulario del resumen, se observa que **ultimahora** como es el que tiene la media en el resumen más alta, tiene mayor vocabulario que **elconfidencial**.

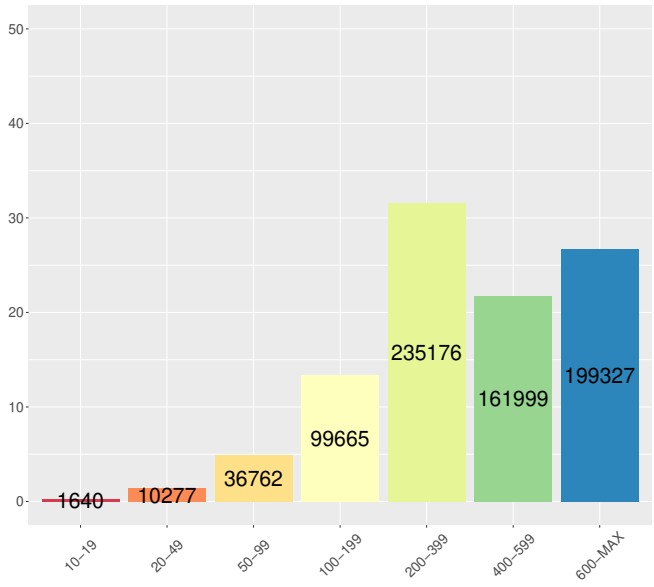
Si analizamos las diferencias entre los dos corpus, percibimos que el periódico que mayor media de palabras en el texto tiene es **eltemps** –pertenece al corpus de catalán– y el que tiene un menor tamaño de media es **ultimahora** –corpus de castellano–. Además, a pesar de que parezca que la diferencia de media de palabras en el texto no sea muy grande –16,51 palabras a favor del corpus de catalán–, observamos como la media del corpus en catalán está muy influenciada por el periódico **eltemps** y que si se eliminase del calculo, la diferencia entre ambos corpus sería de 80,73 palabras a favor del corpus de castellano. Si observamos las diferencias de tamaño de media en el resumen vemos que no es muy grande solo 2,91 palabras a favor del corpus de catalán. El corpus de castellano

solo tiene más o menos el doble de tamaño de vocabulario –tanto para el resumen como para el texto– que el corpus de catalán; a pesar de que el número de noticias en castellano es tres veces superior al corpus de catalán.

Una vez expuestas las tablas de las estadísticas internas de los corpus, se presentarán las gráficas que muestran el número de noticias que tiene cada uno de los rangos definidos de palabras tanto para el caso del texto como para el caso del resumen. En primer lugar se mostrarán las gráficas referentes al texto y en segundo lugar las referentes al resumen.



(a) Noticias para el corpus de catalán

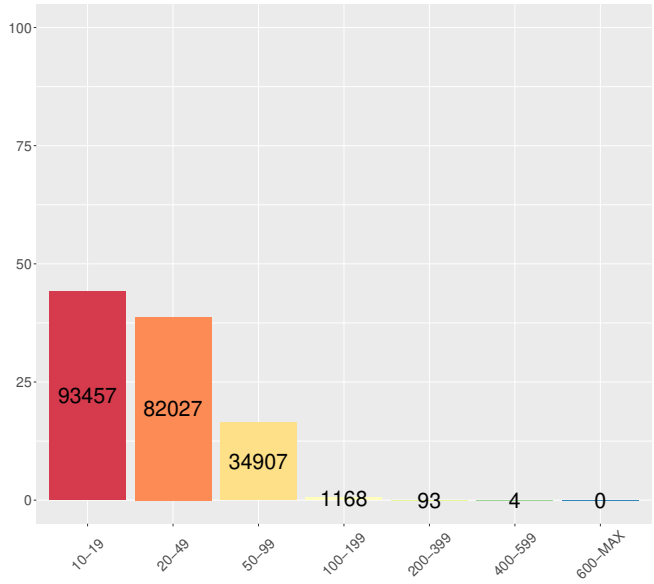


(b) Noticias para el corpus de castellano

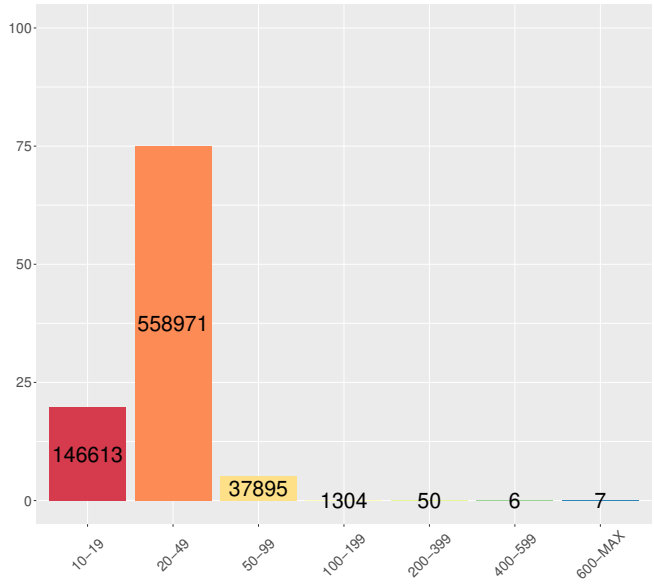
Figura 3.8: Comparativa de noticias cuyo tamaño de texto está entre un determinado rango

En la *Figura 3.8* podemos observar cómo en ambos corpus la mayor cantidad de noticias tienen un texto entre 200 y 399 palabras. Además, en ambos corpus, en todos los

rangos que se han definido se mantiene un orden proporcional; a pesar que tienen ligeras diferencias el cuanto al porcentaje de noticias que están en cada uno de ellos. Por ejemplo, el porcentaje de las noticias que tienen un texto entre 100 y 199 palabras es superior en el corpus de catalán; en contra posición, el porcentaje de noticias que tienen un texto entre 50 y 99 palabras es ligeramente superior en el corpus de castellano.



(a) Noticias para el corpus de catalán



(b) Noticias para el corpus de castellano

Figura 3.9: Comparativa de noticias cuyo tamaño de resumen está entre un determinado rango

En la *Figura 3.9* observamos que, a diferencia del caso anterior, no se mantiene el orden en ambas gráficas. En el caso del corpus de catalán, el 44 % de las noticias se encuentran en el rango entre 10 y 19 palabras y es el máximo porcentaje de noticias en todos los rangos de este corpus. Además, el 16,5 % de las noticias de este mismo corpus tienen entre 50 y 99 palabras. En el caso del corpus de castellano, el 75 % de las noticias tienen un resumen con un número de entre 20 y 49 palabras. Finalmente, en este mismo corpus, tanto el porcentaje de noticias que tienen entre 10-19 palabras como el de 50-99 palabras

es significativamente inferior al del corpus de catalán; esto es debido a que tres cuartas partes de las noticias del corpus de castellano están en un solo rango.

3.5.3. Estadísticas comparativas entre el texto y el resumen

En primer lugar y para el total entendimiento de las tablas y gráficas que presentaremos en el presente apartado, se deben introducir los conceptos de *density* y *coverage*.

Como ya se comentó en la introducción de este trabajo, según como sea el resumen referente al texto, este podrá ser extractivo o abstractivo. Para poder diferenciar la extractividad o abstractividad de las noticias de cada periódico, se utilizarán los conceptos expuestos en [10] para este mismo fin.

En primer lugar se presenta la función que se utiliza en [10] para poder extraer los fragmentos extractivos del resumen.

```
1  function F(A, S) # A = Texto, S = Resumen
2    F ← ∅, ⟨i, j⟩ ← ⟨1, 1⟩
3    while i ≤ | S | do
4      f ← ⟨⟩
5      while j ≤| A | do
6        if s_i = a_j then
7          ⟨i', j'⟩ ← ⟨i, j⟩
8          while s_i' = a_j' do
9            ⟨i', j'⟩ ← ⟨i' + 1, j' + 1⟩
10         if | f |< (i' − i − 1) then
11           f ← ⟨s_i ··· s_i'−1⟩
12         j ← j'
13       else
14         j ← j + 1
15     ⟨i, j⟩ ← ⟨i + max{| f |, 1}, 1⟩
16     F ← F ∪ {f}
17   return F
```

Código 3.4: Función para extraer la frases extractivas de una noticia

Además, en el mismo documento se explica como se calcula el *coverage* y la *density* a partir de la función anteriormente expuesta.

$$Coverage(A,S) = \frac{1}{\mid S \mid} \sum_{f \in F(A,S)} \mid f \mid$$

Es decir, la suma de las longitudes de todos los fragmentos extractivos –calculados en la función expuesta anteriormente– y dividido entre el tamaño del resumen. Cuanto mayor sea el *coverage* de una noticia, más fragmentos del texto y de mayor tamaño se han encontrado en el resumen de este.

$$Density(A,S) = \frac{1}{\mid S \mid} \sum_{f \in F(A,S)} \mid f \mid^2$$

Es la misma operación que el *coverage* pero se suma el cuadrado de las longitudes. La conclusión es análoga a la del *coverage* pero en la *density* se valoran más positivamente los fragmentos que son más largos.

Una vez presentados los conceptos de *density* y *coverage*, se mostrarán las estadísticas donde se comparan diferentes conceptos relacionados con el texto y el resumen de las noticias. Algunas de las estadísticas que se incluyen son las referentes al *coverage* –cuantifica el porcentaje de palabras en el resumen que forman parte de un fragmento extractivo del artículo–, *density* –la longitud media del fragmento extractivo que pertenece al resumen–, el *Compression Ratio* –el ratio entre el número de palabras del resumen y las del texto– y el rango que es el máximo y mínimo número de palabras de cada periódico.
Como en el caso anterior, en primer lugar mostraremos las estadísticas del corpus en catalán; para no extender demasiado el tamaño de la tabla, se ha substituido rango por *R.*.

Periódico	**Density**	**Coverage**	**Compression**	**R.Texto**	**R. Resumen**
ara	4,92	0,61	27,54	10-14 841	10-134
dbalears	12,50	0,92	25,62	42-2688	10-64
diaridegirona	5,20	0,60	18,19	14-6478	10-125
diarilaveu	4,46	0,62	19,47	10-4261	10-251
elpuntavui	6,86	0,64	14,90	22-5594	10-53
eltemps	9,67	0,51	23,33	10-10 601	10-252
naciodigital	7,84	0,69	18,07	10-18 690	10-360
regio7	2,96	0,56	19,65	12-4149	10-569
vilaweb	43,61	0,88	7,70	10-11 002	10-115
Total	10,89	0,67	19,39	10-18 690	10-569

Tabla 3.8: Estadísticas de resumen en el corpus de catalán

Como podemos observar en la *Tabla 3.8*, el periódico que tiene una mayor densidad media es **vilaweb**; este resultado es debido a que la mayor parte del resumen que proporciona es una copia del primer fragmento del texto. El que menor densidad presenta es el de **regio7**; esto es debido a que los resúmenes del periódico no son extractivos. Tanto **dbalears** como **vilaweb** son los que presentan un mayor *coverage* debido a que el resumen que presentan ambos periódicos está creado a partir de fragmentos literales del texto. El periódico que menor *coverage* tiene es **eltemps** a causa de que tiene el resumen más abstractivo de los que se han analizado para este corpus.

Si observamos los resultados de la compresión, vemos como el periódico que mayor valor tiene es el **ara** donde el resumen ocupa 27 veces de media menos de lo que es la noticia. En contraposición, el periódico que menos comprime las noticias es **vilaweb** que como se había comentado anteriormente no tiene de media el texto más largo pero sí del resumen.

Las estadísticas referentes a la comparación entre el texto y el resumen del corpus en castellano se muestran en *Tabla 3.9*.

Periódico	Density	Coverage	Compression	R. Texto	R. Resumen
20minutos	22,71	0,94	14,38	11-26 571	10-66
abc	7,44	0,68	21,29	10-19 467	10-152
diariodemallorca	3,94	0,61	21,22	18-6332	10-262
elconfidencial	5,62	0,60	29,29	17-19 441	10-51
eldiario	18,82	0,87	14,60	11-8690	10-159
elespañol	5,29	0,54	17,67	10-16 869	10-403
elindependiente	4,26	0,65	32,02	62-8313	10-65
elpais	3,25	0,58	35,83	10-32 151	10-101
expansion	14,05	0,58	11,00	10-6749	12-152
laser	3,31	0,53	14,63	10-6583	10-121
lasprovincias	6,54	0,62	21,83	25-3797	10-39
levante	2,97	0,57	22,73	13-6046	10-93
publico	7,96	0,70	18,67	30-16 227	10-154
ultimahora	44,03	0,96	6,88	11-4328	10-739
Total	10,73	0,67	20,15	10-32 151	10-739

Tabla 3.9: Estadísticas de resumen en el corpus de castellano

Como podemos observar en la *Tabla 3.9*, el periódico que tiene una mayor *density* para el corpus en castellano es **ultimahora**; esto es debido principalmente al gran tamaño que tiene de media su resumen y a que este es un fragmento extraído literalmente de la noticia. Además, el que tiene una menor densidad es el **levante**; a causa de ser el periódico menos extractivo de los que se han analizado para este corpus. En cuanto al *coverage*, observamos que tanto **ultimahora** como **20minutos** son los que tienen el mayor valor –por encima de 0,9– y esto significa que son los periódicos que más palabras tienen en el resumen que forman parte de un fragmento extractivo de la noticia. A pesar de que el periódico **ultimahora** es el que predominaba en las dos métricas comentadas anteriormente, vemos como es el que menor compresión aplica del texto al resumen –el resumen de media es de solo 6,88 veces inferior que el texto original–; en contraposición tenemos el caso de **elpaís** donde cada resumen es 35,83 veces inferior de media que el texto original.

Si comparamos las estadísticas de ambas tablas, observamos que no hay mucha diferencia ni en cuanto a la *density* ni en cuanto al *coverage*; pero si analizamos la diferencia de compresión entre ambos corpus, se percibe como la del corpus en castellano es 0,76 veces superior que la del catalán. La afirmación anterior significa que los resúmenes en castellano son de media 0,76 veces más pequeños, en relación a sus noticias, que los del corpus de catalán.

Una vez presentadas las estadísticas del corpus, pasaremos a comentar un modelo para la representación de las estadísticas anteriormente comentadas donde se observará de manera gráfica la extractividad o abstractividad de cada uno de los periódicos.

Un periódico tiene unos resúmenes más extractivos cuanto mayor sea su *coverage* –mayor número y/o longitud de los fragmentos extractivos– y mayor *density* –mayor longitud de los fragmentos extractivos–; es decir cuanto más a la derecha y arriba estén los puntos en las gráficas que os proponemos, mayor grado de extractividad tendrá el resumen.

Las gráficas que se pasan a comentar, representan cada documento como un punto en un espacio bidimensional donde el eje x representa el *coverage* y el eje y la *density*. La *density* está normalizada y se define como: $\frac{x_i - \bar{X}}{\sigma}$; donde x_i es el valor de la *density* calculado con la fórmula anteriormente comentada, $\bar{X}$ es la media de *density* del diario y σ es la desviación típica de la *density* del diario.

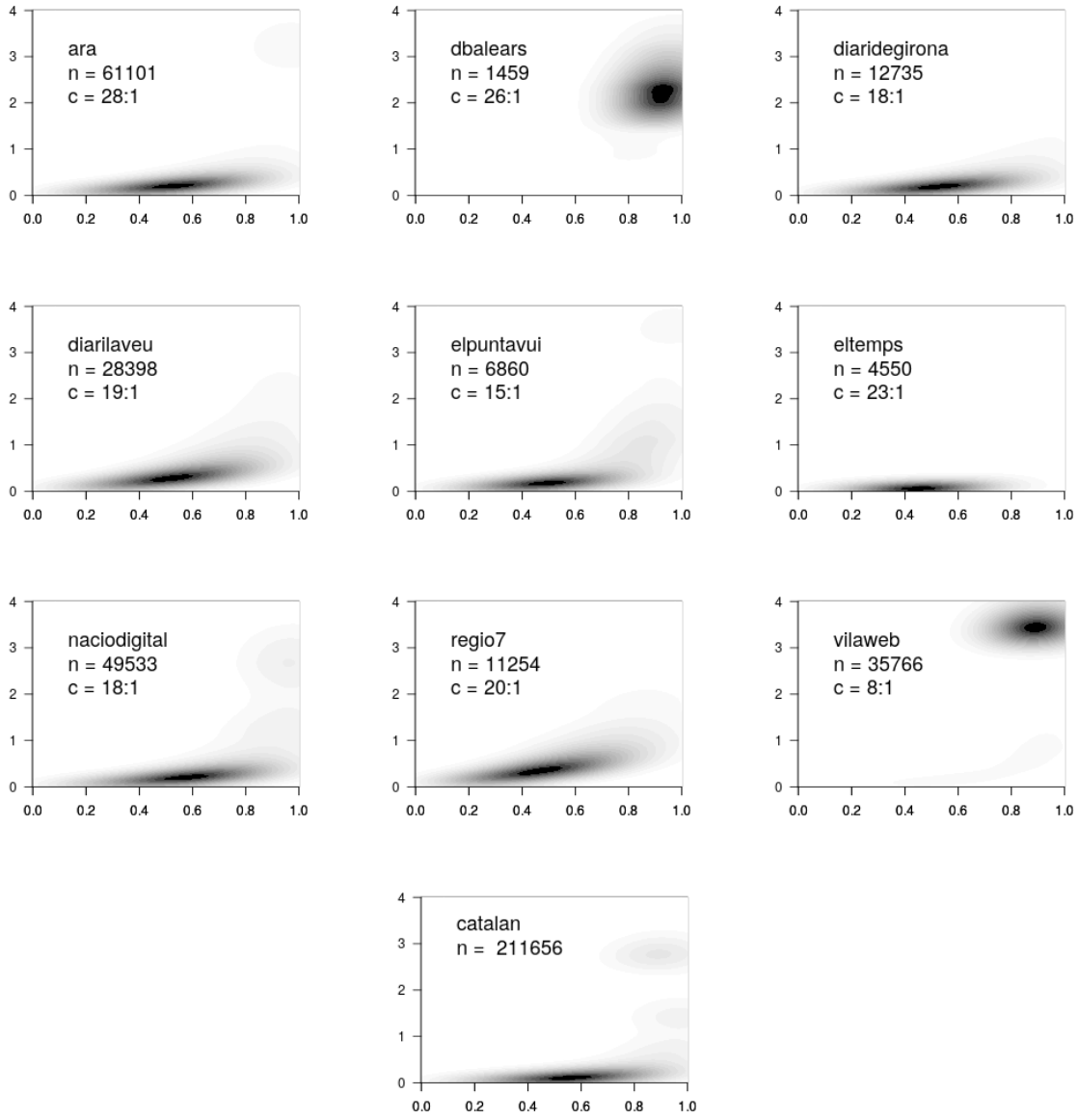


Figura 3.10: Gráficas de *Densidad/Coverage* para los periódicos del corpus de catalán

Como podemos observar en las gráficas de la *Figura 3.10* y a partir del razonamiento anterior, los periódicos donde se observa un resumen más extractivo son tanto **dbalears** como **vilaweb**. Además, a la vista de los gráficos presentados, observamos como el periódico **eltemps** tiene el resumen más abstractivo de los que estamos analizando para el corpus de catalán.

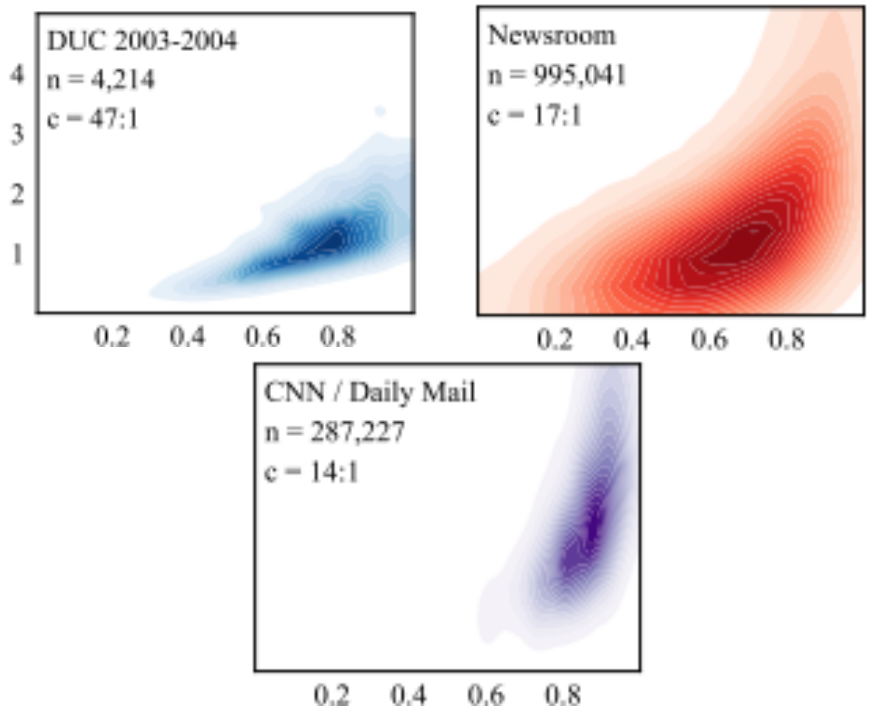


Figura 3.11: Gráficas de los corpus extraídos de [10]

Si comparamos la gráfica obtenida para el catalán en el presente estudio con las presentadas para el *DUC 2003-2004*, *CNN/Daily Mail* y *Newsroom* en [10] *Figura 3.11*, observamos como el grueso de las noticias del catalán tienen un *coverage* de cerca del 0,6 y que este es parecido al del *Newsroom* aunque ligeramente inferior a los otros dos. Si observamos las diferencias referentes al *density*, percibimos como los valores para el catalán son inferiores a los de los demás corpus; esto puede ser debido a que por lo general los fragmentos extractivos obtenidos en el corpus en catalán son inferiores a los mostrados en [10]. Además, si comparamos los periódicos del estudio citado anteriormente con los que se han obtenido para el presente trabajo, observamos que como norma general el carácter de los periódicos de este trabajo son más abstractivos que los presentados en [10].

Las gráficas referentes a la comparación entre *density* y *coverage* del fragmento extractivo para el corpus en castellano se muestran en la *Figura 3.12*.

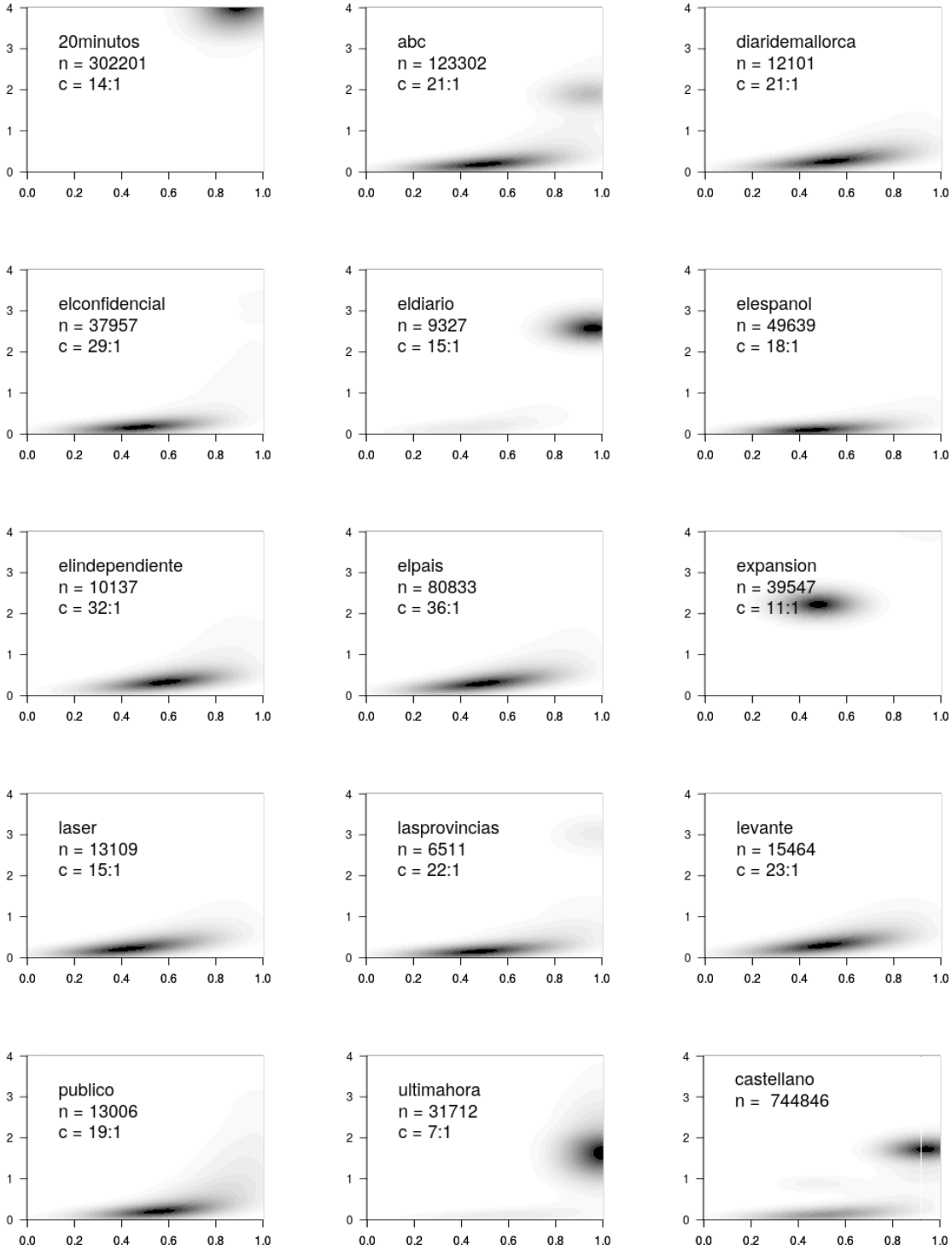


Figura 3.12: Gráficas de *Densidad/Coverage* para los periódicos del corpus de castellano

Para el corpus de castellano, los periódicos en los que observamos una mayor extactividad en sus resúmenes son: **20minutos, eldiario y ultimahora**. Aunque el periódico **expansion** tenga una alta *density*, tiene un muy bajo *coverage* y por esa razón no tiene un resumen tan extractivo como los periódicos que se han expuesto anteriormente.
Si comparamos la gráfica obtenida en el presente trabajo para el corpus de castellano con las obtenidas en el [10], observamos como el grueso de las noticias en el corpus de castellano tiene un *coverage* de alrededor de 0,9 y una *density* de 2. Esta gráfica es muy

parecida a la del *CNN/Daily Mail* que se presenta en el estudio anteriormente citado. Además, como en el caso del corpus de catalán, los periódicos por norma general son más abstractivos que los que se presentan en [10].

Si comparamos las gráficas obtenidas para el corpus de castellano y para el corpus de catalán, observamos como el corpus de castellano tiene un resumen más extractivo que el corpus de catalán; esto puede ser debido en gran parte al diario **20minutos** que tiene un resumen muy extractivo y tiene más noticias, el solo, que todo el corpus de catalán.

3.6 Resultados del resumen automático

3.6.1. Evaluación sintáctica

La evaluación sintáctica de los resúmenes generados –tanto en algoritmos clásicos como basados en redes neuronales– se ha hecho utilizando un paquete llamado *Rouge*. Este, como ya se explicó en el apartado de los modelos, es un software para la evaluación automática de resúmenes basándose para ello en distintas métricas. Este paquete será el que permita evaluar y comparar sintácticamente los resultados obtenidos utilizando los distintos algoritmos.

En este trabajo, se ha evaluado sintácticamente cada resumen respecto al Rouge-1 –donde solo se contemplan unigramas–, Rouge-2 –donde se contemplan los bigramas– y el Rouge-L –donde se busca la secuencia de longitud más larga coincidente en el resumen generado automáticamente y el de referencia–. Para cada una de estas evaluaciones, el sistema nos provee de tres resultados: **Average_R** el cual nos indica el *Recall*, **Average_P** el cual se refiere a la *Precision* y el **Average_F** el cual nos da cuenta del F_1.

En las tablas de resultados que presentamos en esta sección se resaltan en negrita el o los mejores resultados para cada medida.

En primer lugar se presentan los resultados del para el corpus de catalán *Tabla 3.10*.

		Rouge-1	Rouge-2	Rouge-L
Recall	LexRank	0,66	**0,42**	**0,60**
	LSA	0,58	0,32	0,51
	LUHN	0,66	**0,42**	0,59
	TextRank	0,61	0,37	0,55
	SHA-NN	**0,67**	0,41	**0,60**
Precision	LexRank	0,23	0,17	**0,22**
	LSA	0,19	0,12	0,17
	LUHN	**0,24**	**0,18**	**0,22**
	TextRank	0,22	0,15	0,20
	SHA-NN	0,21	0,15	0,20
F_1	LexRank	**0,31**	0,21	**0,28**
	LSA	0,26	0,15	0,23
	LUHN	**0,31**	**0,22**	**0,28**
	TextRank	0,28	0,19	0,26
	SHA-NN	0,29	0,20	0,26

Tabla 3.10: Resultados del *Rouge* para el corpus de catalán

Como se puede observar en la *Tabla 3.10*, para la evaluación del resumen considerando solo los unigramas –primera columna–, el generador de resúmenes que mejor resultados obtiene para el *Recall* es el SHA-NN, para la *Precision* el Luhn y finalmente para la F_1 tanto el LexRank como el Luhn; es por ello que el mejor sistema que se elegiría sería el Luhn ya que obtiene mejores resultados en dos de las tres medidas. Si se contempla la evaluación para el caso de los bigramas –segunda columna–, se observa como el Luhn también obtiene mejores resultados para todas las medidas. Finalmente, considerando los resultados obtenidos usando la métrica *Rouge-L* –tercera columna– que considera los segmentos de longitud más larga coincidentes en ambos resúmenes, el LexRank ha obtenido unos mejores resultados; a pesar de que obtiene unos resultados muy parecidos al Luhn. Por todo lo comentado anteriormente, se ha elegido el sistema Luhn como el mejor para la generación de resúmenes en el corpus de catalán.

Pasaremos ahora a comentar los resultado del *Rouge* para el corpus de castellano.

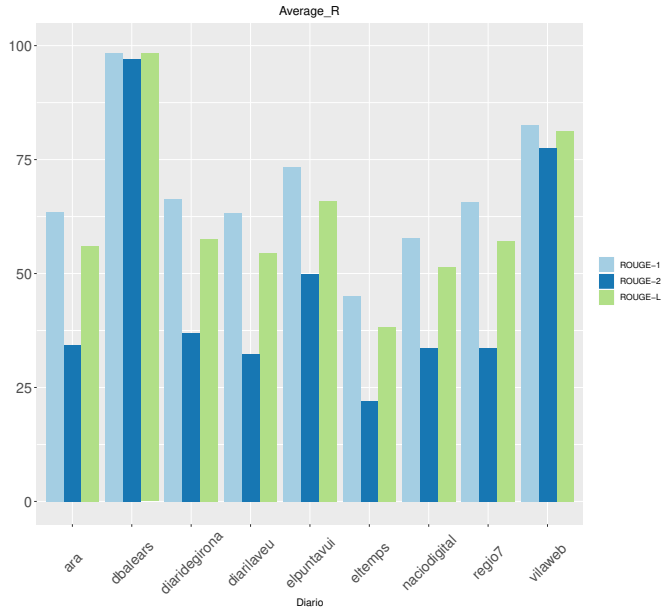
		Rouge-1	Rouge-2	Rouge-L
Recall	LexRank	0,71	0,51	0,65
	LSA	0,67	0,42	0,61
	LUHN	0,68	0,44	0,62
	TextRank	0,69	0,46	0,63
	SHA-NN	**0,79**	**0,63**	**0,75**
Precision	LexRank	0,19	0,13	0,17
	LSA	0,16	0,10	0,14
	LUHN	0,16	0,11	0,15
	TextRank	0,16	0,10	0,14
	SHA-NN	**0,21**	**0,17**	**0,2**
F_1	LexRank	0,29	0,20	0,26
	LSA	0,24	0,15	0,22
	LUHN	0,25	0,17	0,23
	TextRank	0,24	0,16	0,22
	SHA-NN	**0,32**	**0,25**	**0,30**

Tabla 3.11: Resultados del *Rouge* para el corpus de castellano

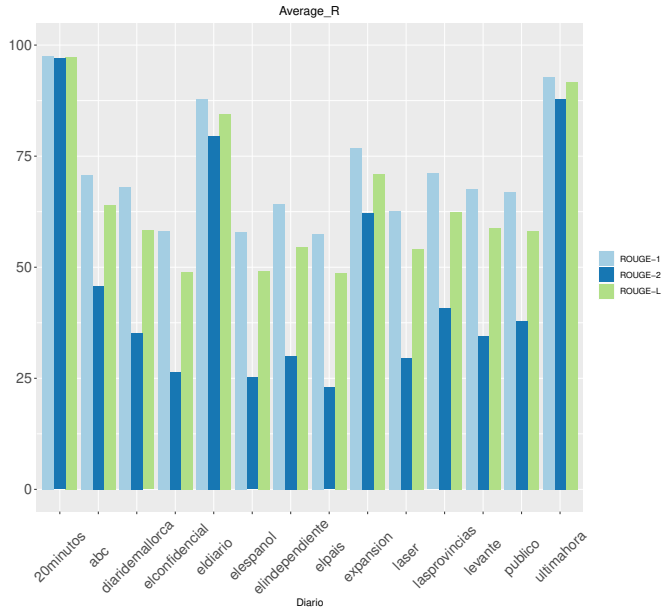
En el caso del corpus de castellano, se observa como el sistema que nos ofrece unos mejores resultados y además con bastante diferencia a los demás en todas las medidas y para todos los criterios ha sido SHA-NN.

Si se comparan los resultados obtenidos en ambos corpus, en términos absolutos, se percibe que los mejores resultados los obtiene el corpus de castellano. De este resultado se pueden concluir dos ideas: la primera es que en los algoritmos clásicos era de esperar que se obtuvieran mejores resultados en el corpus de castellano que en el corpus de catalán; esto es debido principalmente a que el corpus de castellano es más extractivo que el corpus de catalán y que tanto la generación de resúmenes como su posterior evaluación sintáctica tienen un carácter extractivo. La segunda idea es que con el algoritmo basado en redes neuronales, no sabemos si la mejor calificación del castellano es debido al mayor número de muestras de este o a su extractividad; es por ello que se ha propuesto un apartado donde se analizará esta idea.

Como los mejores resultados que se han obtenido han sido para el corpus de castellano el SHA-NN y para el corpus de catalán el Luhn, se ha decidido hacer un estudio pormenorizado donde se evalúe cada periódico con el mejor sistema encontrado para su corpus. Primero analizaremos el *Recall*, seguidamente la *Precision* y finalmente el F_1



(a) Evaluación Luhn para el corpus de catalán



(b) Evaluación SHA-NN para el corpus de castellano

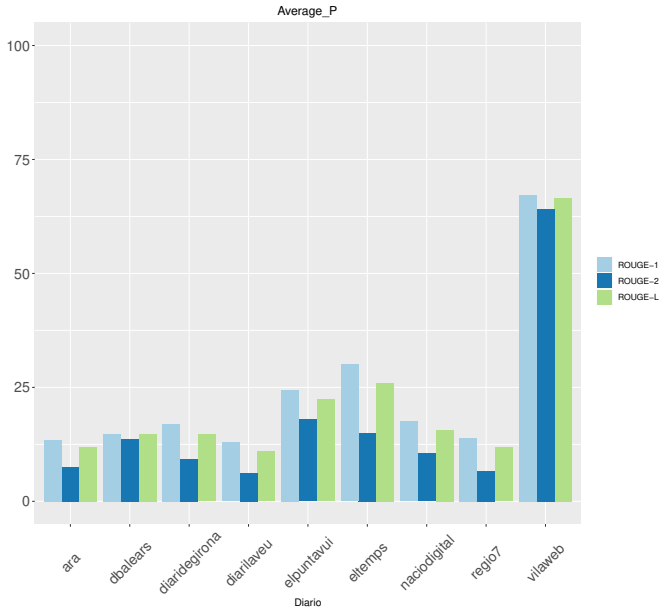
Figura 3.13: Evaluación del *Recall* de los periódicos

En el caso del corpus de catalán, *Figura 3.13a*, se observa que el periódico que mejores resultados obtiene es **dbalears**; esto es debido a que el resumen extraído de las noticias de este periódico contiene la primera frase del texto; por consiguiente el resumen es muy extractivo. El periódico que peor resultado ha obtenido –también en el corpus de catalán– es **eltemps** y esto es debido a que el resumen de este es abstractivo. Por norma general, se obtienen mejores resultados en corpus extractivos; esta es una conclusión clara ya que las valoraciones del *Rouge* se basan en las similitudes sintácticas y no semánticas de los

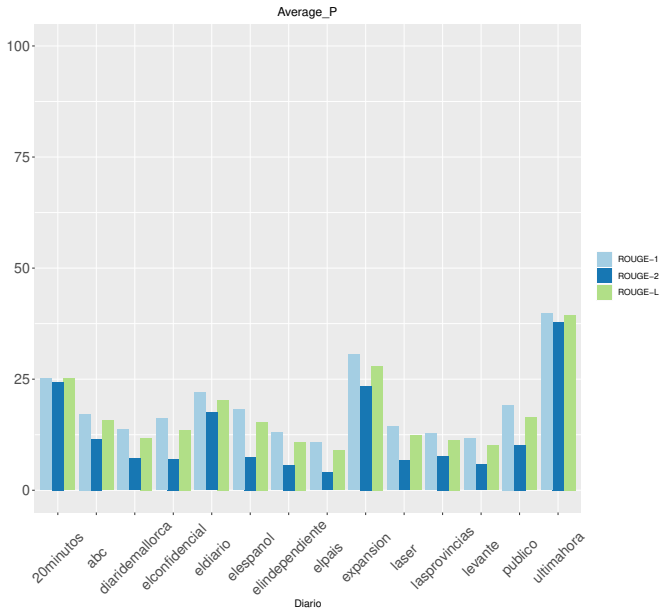
resúmenes.

En el caso del corpus de castellano, *Figura 3.13b*, los periódicos que han obtenido mejores resultados han sido el **20minutos** y **ultimahora**; esto es debido a la naturaleza extractiva de sus resúmenes. En contraposición, el que peor resultados ha conseguido ha sido **elpaís** ya que tiene uno de los resúmenes más abstractivos del corpus de castellano.

Si comparamos las gráficas de ambos corpus, se observa que como hemos comentado anteriormente los resultados obtenidos para el corpus de castellano son superiores a los de catalán. Además, en todos los periódicos analizados, la mejor puntuación se obtiene con unigramas y la peor con bigramas.



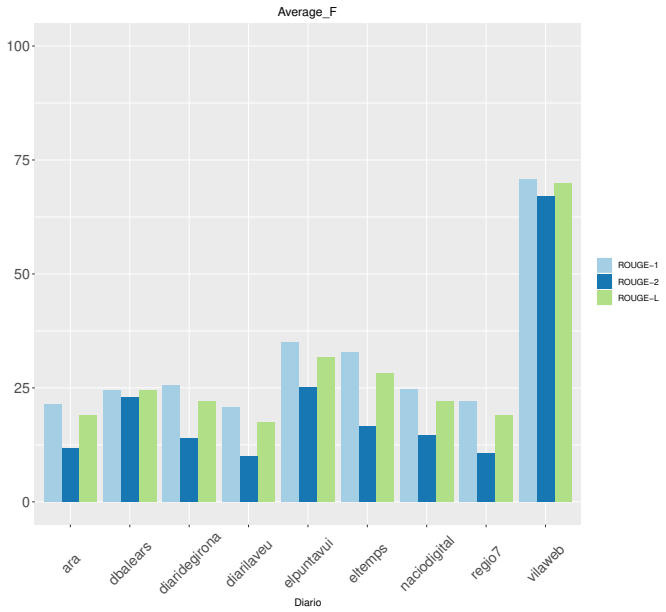
(a) Evaluación Luhn para el corpus de catalán



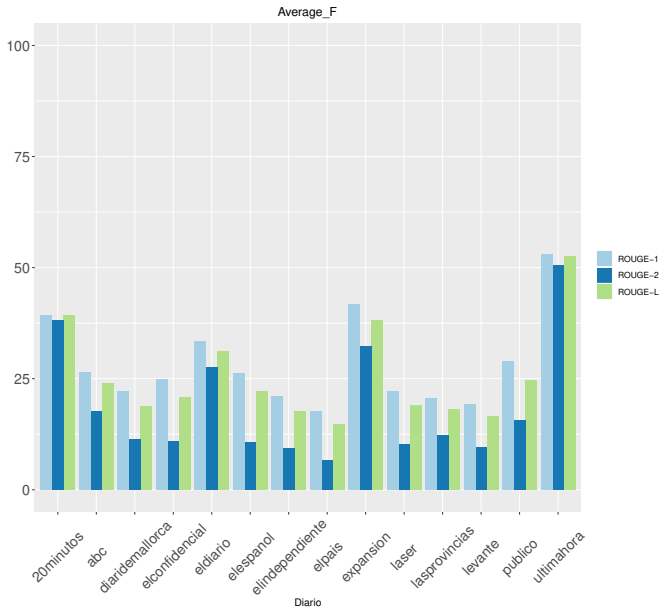
(b) Evaluación SHA-NN para el corpus de castellano

Figura 3.14: Evaluación del *Precision* de los periódicos

Si observamos la *precision* en la *Figura 3.14*, en los periódicos del corpus de catalán se percibe un decremento notable en el periódico **dbalears** esto es debido a la alta compresión que hace ese mismo periódico; ya que al ser la compresión que realiza muy alta, la diferencia entre el texto y el resumen es mayor. Por esa misma razón, **vilaweb** en *Recall* tenía un buen resultado y como en este la compresión es mucho menor, sigue teniendo un buen resultado. Esto mismo ocurre, aunque no de manera tan clara, con los periódicos **20minutos** y **ultimahora**.



(a) Evaluación Luhn para el corpus de catalán



(b) Evaluación SHA-NN para el corpus de castellano

Figura 3.15: Evaluación del F_1 de los periódicos

Mediante la métrica F_1, *Figura 3.15a* y la *Figura 3.15b*, se evalúa tanto el *recall* como la *precision* y por ello los periódicos mejor valorados en cómputo general han sido **vilaweb, ultimahora y 20minutos** debido a que son los periódicos que se han catalogado como más extractivos para ambos corpus.

3.6.1.1. Diferencias entre los corpus con algoritmos de redes neuronales

Como se explicó anteriormente, los mejores resultados utilizando el algoritmo de redes neuronales –SHA-NN– se obtuvieron en el corpus de castellano. Estos resultados podrían tener dos motivaciones diferentes; o bien el mayor número de muestras del corpus de castellano permiten entrenar un mejor modelo y esto produce una mejora en los resultados o bien el carácter más extractivo de las noticias del corpus de castellano provocan este resultado.

Es por ello que se realizó un análisis exhaustivo para concluir cual de las dos hipótesis anteriores era la correcta. Para este análisis, se entrenó un modelo de castellano utilizando el mismo número de noticias –tanto en entrenamiento, en desarrollo y en test– que el corpus de catalán. Además, el resto de parámetros utilizados son los mismos con los que se ha realizado el trabajo. Para una mejor visualización de las noticias seleccionadas, se ha decidido generar una gráfica, análoga a las presentadas en *Sección 3.5*, donde se observe tanto el *coverage* como la *density* de las noticias seleccionadas.

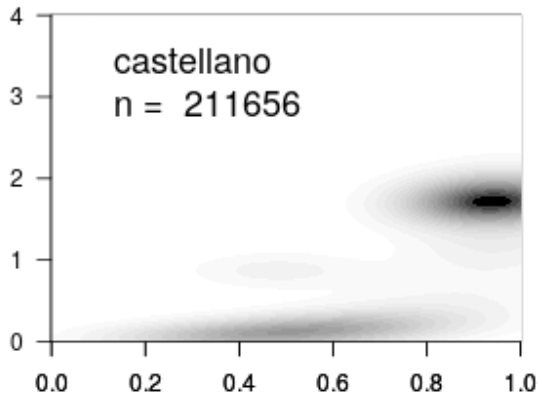


Figura 3.16: Gráficas *Densidad/Coverage* para el corpus reducido de castellano

Como podemos observar, la *Figura 3.16* es prácticamente la misma que la presentada en la *Sección 3.5* para el corpus total de castellano. Es por ello que tiene las mismas características de extractividad que las allí comentadas. Una vez presentada la gráfica de extractividad del corpus, en la *Tabla 3.12* se mostrarán los resultados obtenidos para la evaluación de este nuevo corpus.

		Rouge-1	Rouge-2	Rouge-L
Recall	SHA-NN	0,90	0,63	0,75
Precision	SHA-NN	0,20	0,16	0,19
F_1	SHA-NN	0,31	0,24	0,29

Tabla 3.12: Resultados del *Rouge* para el corpus reducido de castellano

Si comparamos los resultados obtenidos en la *Tabla 3.12* con los obtenidos para el *SHA-NN* en la *Tabla 3.10*, observamos que a pesar de que entrenemos el modelo de castellano con el mismo número de muestras que el modelo de catalán, obtenemos mejores resultados en el modelo de castellano. Gracias a este análisis, podemos concluir que lo

que afecta a los mejores resultados utilizando algoritmos basados en redes neuronales es la extractividad de las noticias que contienen los diarios más que el número de muestras que utilicemos; siempre y cuando se tengan bastantes muestras como para entrenar un modelo que obtenga unos buenos resultados. No está de más comentar que a pesar de utilizar en este experimento un tercio de las noticias de entrenamiento de castellano, obtenemos unos resultados solo ligeramente inferiores a los mostrados en la *Tabla 3.11*.

3.6.2. Evaluación semántica

La evaluación semántica de los resúmenes generados en el presente trabajo se ha hecho utilizando la implementación ya comentada en la *Sección 3.2*. Es por ello que inicialmente presentaremos los resultados obtenidos y posteriormente se mostrarán distintas figuras donde se puedan observar gráficamente las diferencias entre ellos.

Inicialmente presentaremos los resultados referentes al corpus de catalán:

	Vector Resumen	Frases del Resumen
SHA-NN	**0,74**	**0,70**
LexRank	**0,74**	0,67
LSA	0,73	0,66
LUHN	**0,74**	0,67
TextRank	0,73	0,65

Tabla 3.13: Resultados de la evaluación semántica para el corpus de catalán

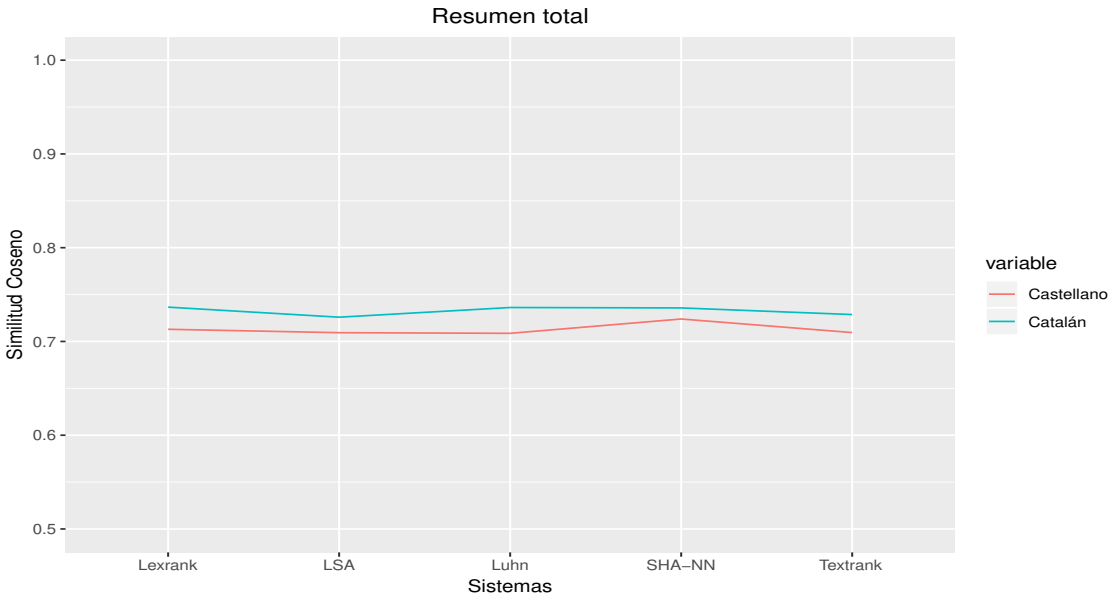
Como podemos observar en la *Tabla 3.13*, los mejores resultados se obtienen con los sistemas *SHA-NN, LexRank y Luhn*.

Para el caso del corpus de castellano, se han obtenido los siguientes resultados:

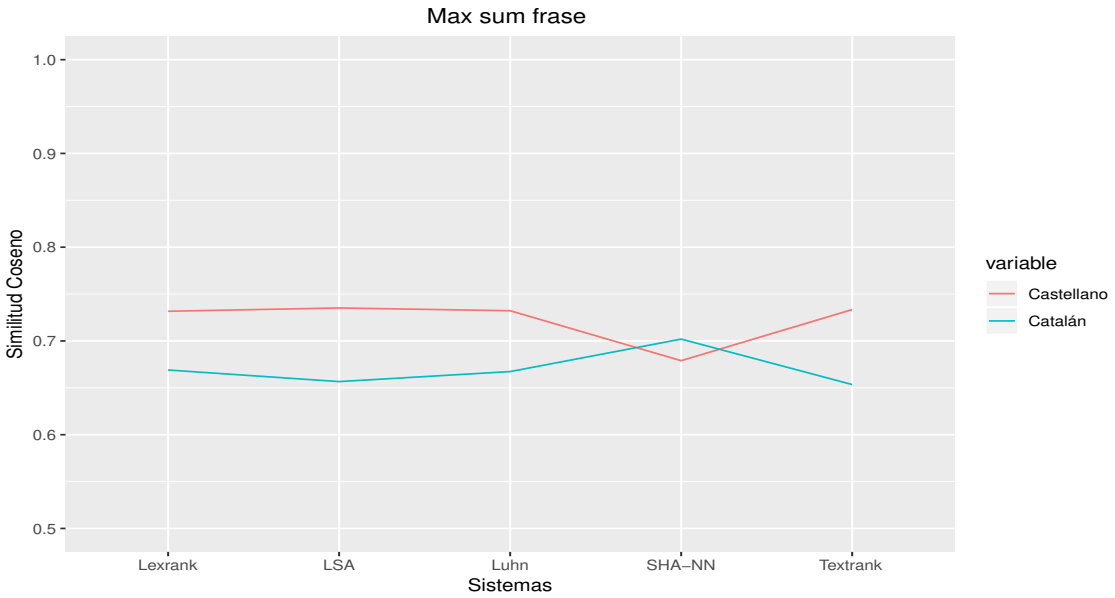
	Vector Resumen	Frases del Resumen
SHA-NN	**0,72**	0,68
LexRank	0,71	**0,74**
LSA	0,69	0,73
LUHN	0,71	0,73
TextRank	0,70	0,72

Tabla 3.14: Resultados de la evaluación semántica para el corpus de castellano

Análogamente al caso anterior, los mejores resultados se han obtenido para el *SHA-NN* y el *LexRank*. Para poder observar más gráficamente las diferencias entre el corpus de catalán y el de castellano para los dos métricas propuestas, se ha decidido mostrar la *Figura 3.17a* y la *Figura 3.17b*



(a) Análisis semántico para el Total del Resumen

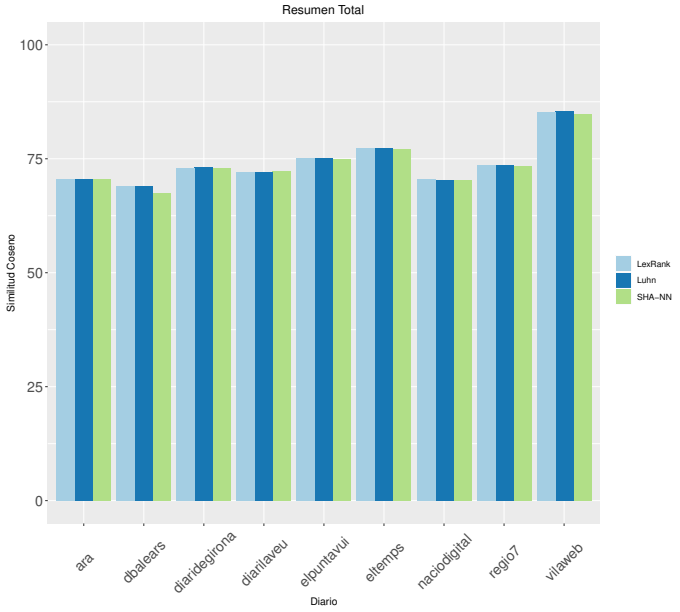


(b) Análisis semántico para Frases del Resumen

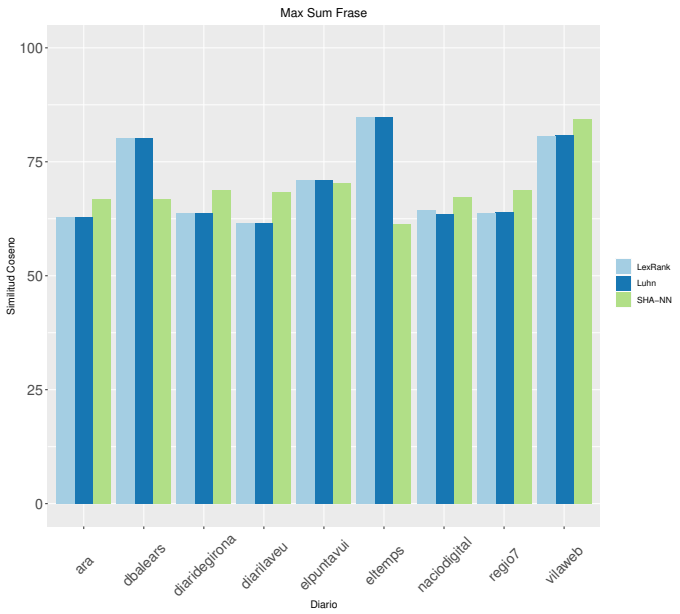
Figura 3.17: Comparativa semántica en los corpus de castellano y catalán

En el caso de la *Figura 3.17a* observamos como los resultados obtenidos por el corpus de catalán, son ligeramente superiores a los de castellano; es por ello que podríamos concluir que en el análisis semántico, cuando se está comparando las palabras que representan el resumen de referencia con el generado, el catalán obtiene mejores resultados. Si ahora observamos la *Figura 3.17b*, vemos como, si comparamos las palabras que representan cada una de las frases del resumen, el castellano obtiene mejores resultados en todos los sistemas excepto en el SHA-NN.

Por todo lo comentado anteriormente, elegimos los sistemas *SHA-NN, LexRank y Luhn*, como los sistemas que mejores resultados obtienen para el análisis semántico empleado; es por ello que se ha decidido realizar un análisis pormenorizado para poder observar como se comportan los resultados para cada uno de los periódicos que forman los dos corpus.



(a) Análisis semántico para el Total del Resumen de catalán



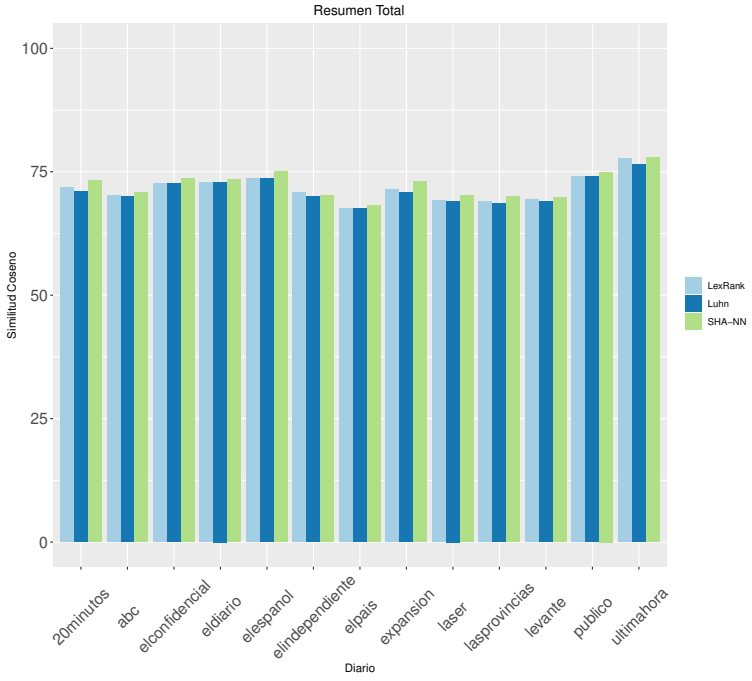
(b) Análisis semántico para Frases del Resumen de catalán

Figura 3.18: Análisis semántico para el corpus de catalán

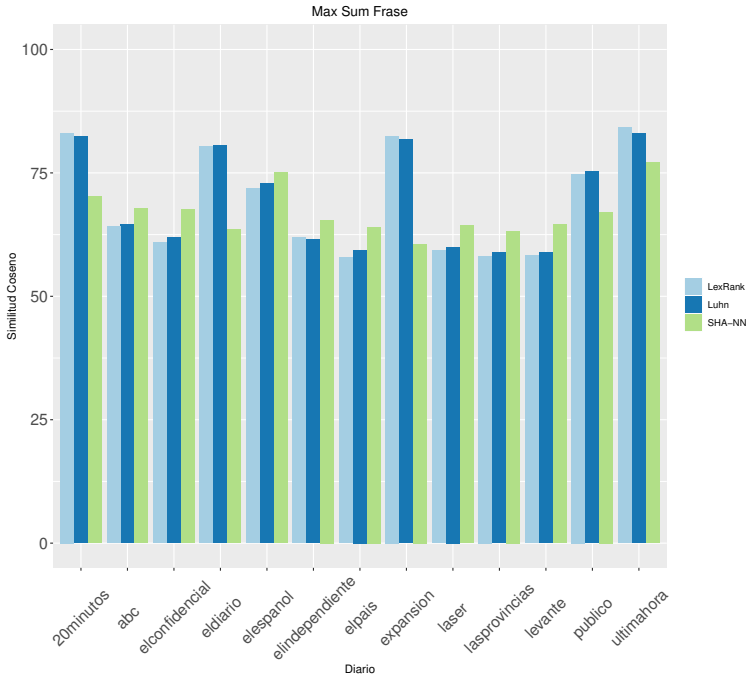
Cuando observamos el análisis semántico de los *embeddings* que representan todo el resumen, *Figura 3.18a*, vemos como los periódicos que mejor resultado obtienen son **vilaweb** y **eltemps**. El primero es el más extractivo del corpus de catalán y el segundo el más abstractivo del mismo corpus. Con este análisis podemos comprobar como han variado los resultados obtenidos respecto el análisis sintáctico explicado anteriormente. Además, los peores resultados los obtienen **dbalears** y **naciodigital**. También es interesante comentar, como los resultados obtenidos por *LexRank y Luhn* son bastante similares en la mayoría de periódicos y difieren de los valores obtenidos por el *SHA-NN*. Si observamos los resultados utilizando la medida de comparación de frases, vemos como los mejores resultados los obtienen **dbalears**, **eltemps** y **vilaweb**. El caso de **dbalears** es curioso ya que este buen resultado es debido a que el resumen de referencia contiene la primera frase del texto y por eso razón, cuando comparamos frase a frase, obtiene buenos resultados.

Es interesante remarcar como a pesar de ser el periódico **eltemps** el que tiene el resumen más abstractivo, es uno de los que mejores resultados obtiene para el análisis semántico.

En el caso del corpus de castellano:



(a) Análisis semántico para el Total del Resumen de castellano



(b) Análisis semántico para Frases del Resumen de castellano

Figura 3.19: Análisis semántico para el corpus de castellano

Si observamos las gráficas referentes al análisis semántico del corpus de castellano, vemos como **ultimahora** obtiene los mejores resultados; esto mismo ocurría en el análisis sintáctico. El segundo periódico que ha obtenido mejores resultados ha sido **elespañol**; que a pesar de ser uno de los mejores en el análisis semántico, era uno de los

peores valorados en el análisis sintáctico. Si observamos los resultados obtenidos en las comparaciones de frases contra frases –*Figura 3.19b*–, los mejores resultados los obtienen los periódicos **ultimahora**, **20minutos**, **expansion** y **eldiario**. En todos ellos, los mejores resultados se obtienen con el *LexRank* y con el *Luhn* y son bastante superiores a los resultados obtenidos por el *SHA-NN*. Además, es interesante comentar como en el resto de periódicos, los resultados obtenidos por el sistema *SHA-NN* son superiores a los del resto de sistemas.

Finalmente, si comparamos ambos corpus, observamos como el periódico mejor valorado en cuanto a las comparaciones de *embeddings* del resumen total es **vilaweb** –corpus de catalán–. Si hablamos ahora sobre la comparación de los *embeddings* de las frases, vemos como los mejores periódicos son **vilaweb**, **eltemps** –corpus de catalán– y **ultimahora**, **expansion** y **20minutos** –corpus de castellano–; siendo **vilaweb** en el único que destaca el resultado del sistema *SHA-NN* sobre los algoritmos clásicos.

CAPÍTULO 4
Conclusiones y trabajo futuro

4.1 Conclusiones

Durante el presente trabajo, se ha elaborado un sistema capaz de descargar, normalizar y almacenar textos noticiarios para el proceso de la generación de los corpus de castellano y de catalán. Como se ha podido comprobar, se han generado dos corpus con los requisitos necesarios para la completa realización tanto del presente trabajo como el de [1]. Además, una vez generados los corpus, se han analizado los mismos para poder conocer el número de noticias de cada corpus y la distribución de noticias para cada diario dando así una primera noción del contenido de los corpus. Se han cumplido las características requeridas ya que el corpus es ampliable, proviene de distintas fuentes y temáticas y el código generado es independiente de cualquier fuente. Por todo lo comentado, se han cumplido con creces los objetivos planteados en la parte de la generación del corpus. Adicionalmente, para este trabajo, se han utilizado herramientas de software libre y el código necesario para la replicación de los corpus, ha sido liberado.

En lo referente a la parte de la generación automática de resúmenes, se han conseguido alcanzar todos los sub-objetivos y por ende el objetivo global planteado. Para este fin, se han analizado distintas técnicas de resumen, tanto basadas en redes neuronales como basadas en algoritmos clásicos. Además de sus explicaciones técnicas, en el apartado del Marco Experimental, se ha comparado el análisis realizado para el corpus de castellano con el realizado para el corpus de catalán y también el análisis de estos dos corpus con el de algunos corpus presentados en el estado del arte. Posteriormente, se han evaluado los generadores de resumen tanto sintácticamente como semánticamente. Es por ello, que se debería analizar cuales han sido los mejores sistemas de resumen según las condiciones que se dispongan. En la evaluación sintáctica, ha obtenido mejores resultados el *SHA-NN* –para el castellano– y el *Luhn* –para el catalán– aunque el primero ha obtenido unos resultados más altos. Por esta razón, si se requiere obtener el mejor resultado para el corpus de castellano, se utilizaría la aproximación basada en redes neuronales pero si por el contrario solo se requiere una estimación, los algoritmos clásicos serían la mejor opción ya que obtienen unos resultados solo ligeramente inferiores a los basados en redes neuronales pero el cómputo necesario para la generación de los resúmenes es significativamente inferior –debido a la no necesidad de entrenamiento previo–. En el corpus de catalán, los mejores resultados se han obtenido con el algoritmo clásico *Luhn*; por ello, no habría duda en la elección del sistema. Para el caso de la evaluación semántica, se han obtenido mejores resultados con el *SHA-NN, LexRank y Luhn*; aunque con unos resultados muy parejos entre todos los sistemas y para ambos corpus. Es por ello, que se tendría una conclusión análoga a la comentada para el caso de la evaluación sintáctica. Para el come-

tido de la evaluación sintáctica comentada, se ha utilizado *Rouge* y para la evaluación semántica se han propuesto dos métricas explicadas en el apartado de Modelos. Evaluando los resultados con métricas sintácticas y semánticas, se ha podido observar como existían periódicos en los que se obtenían malos resultados en el análisis sintáctico y buenos resultados en el semántico. Esto es debido a que los resúmenes generados tenían un carácter extractivo, los resúmenes almacenados para estos periódicos eran abstractivos y que la evaluación sintáctica solo funciona bien cuando ambos resúmenes tienen carácter extractivo. Gracias a la medida propuesta, se ha conseguido mejorar la evaluación final de los sistemas.

Finalmente comentar que las conclusiones respecto a los sistemas de generación automática de resúmenes dependen de las características del corpus en cuestión y que si se ampliaran o modificaran estas, podrían variar ligeramente las conclusiones aquí comentadas.

4.2 Trabajo futuro

En esta sección se presentarán algunos estudios complementarios al presente trabajo. Estos, a pesar de que eran del interés del autor del trabajo, no se han podido llevar a cabo a causa de la limitación temporal del estudio presente.

- Un posible avance en el trabajo podría estar enfocado en el análisis y variación de los hiper-parámetros del *SHA-NN*. A través de este estudio, se podría analizar como afectan los distintos parámetros en la evaluación del mismo consiguiendo incluso una mejora de los resultados mostrados en el presente trabajo.
- Otro avance, que también se considera interesante por el autor del trabajo, sería la utilización de modelos que produzcan resúmenes de carácter abstractivo. Con este estudio, se conseguiría abrir el alcance del presente trabajo obteniendo resúmenes de otros tipos de sistemas.
- Un avance, aunque más a largo plazo, sería la creación de un generador de resúmenes ya bien sea extractivo o abstractivo que implemente otras medidas de selección de contenido del texto diferentes de las que se han comentado en este trabajo. Esta tarea sería muy compleja ya que estos sistemas han de pasar verificaciones y mejoras para poder obtener buenas aproximaciones de resúmenes.
- Además, también se podría utilizar el corpus creado para otros fines; como por ejemplo el análisis de las noticias dependiendo del autor que las creara. A través de este estudio se podrían analizar los diferentes enfoques que tiene una noticia según la fuente que los haya escrito.
- Finalmente, se podría ampliar los corpus generados en el trabajo. En esta ampliación podrían incluirse nuevas noticias de los mismos periódicos, nuevas fuentes para la extracción de noticias y incluso nuevos campos o idiomas que hagan que los corpus sean utilizables en numerosos campos de estudio.

Bibliografía

[1] Vicent Ahuir Esteve. Creación de corpus de artículos de prensa y categorización de noticias. *Universitat Politécnica de València*, 2019.

[2] Federico Barrios, Federico López, Luis Argerich, and Rosa Wachenchauzer. Variations of the similarity function of textrank for automated summarization. *CoRR*, abs/1602.03606, 2016.

[3] Francisco Casacuberta Nolla and Enrique Vidal Ruiz. Tema 6:redes neuronales multicapa. 2018.

[4] Yen-Chun Chen and Mohit Bansal. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[5] Dipanjan Das and André F. T. Martins. A survey on automatic text summarization, 2007.

[6] Günes Erkan and Dragomir R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479, 2004.

[7] Yihong Gong and Xin Liu. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 19–25, New York, NY, USA, 2001. ACM.

[8] José-Ángel González, Encarna Segarra, Fernando García-Granada, Emilio Sanchis, and Lluís-F. Hurtado. Siamese hierarchical attention networks for extractive summarization. *Journal of Intelligent & Fuzzy Systems (JIFS)*, To be published, 2019.

[9] Julian S. Griggs. Tl;dr: Automatic summarization with textual annotations. 2015.

[10] Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. pages 708–719, 01 2018.

[11] Makbule Gulcin Ozsoy, Ferda Alpaslan, and Ilyas Cicekli. Text summarization using latent semantic analysis. *J. Information Science*, 37:405–417, 08 2011.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[13] Daniel Jurafsky and James H. Martin. Speech and language processing. In *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2007.

[14] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org, 2014.

[15] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[16] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. pages 74–81, July 2004.

[17] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958.

[18] Rada Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. 170-173, 01 2004.

[19] Rada Mihalcea and Paul Tarau. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[20] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[22] Marwa Naili, Anja Habacha, and Henda Ben Ghezala. Comparative study of word embedding methods in topic segmentation. *Procedia Computer Science*, 112:340–349, 12 2017.

[23] Makbule Gulcin Ozsoy, Ilyas Cicekli, and Ferda Nur Alpaslan. Text summarization of turkish texts using latent semantic analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 869–876, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[24] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. pages 311–318, 2002.

[25] Dragomir R. Radev, Eduard H. Hovy, and Kathleen McKeown. Introduction to the special issue on summarization. *Computational Linguistics*, 28:399–408, 2002.

[26] Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi. Semantic cosine similarity. 10 2012.

[27] Abigail See, Peter Liu, and Christopher Manning. Get to the point: Summarization with pointer-generator networks. In *Association for Computational Linguistics*, pages 1073–1083, 2017.

[28] Kuldeep Shiruru. An introduction to artificial neural network. *International Journal of Advance Research and Innovative Ideas in Education*, 1:27–30, 09 2016.

[29] Josef Steinberger and Karel Jezek. Using latent semantic analysis in text summarization and summary evaluation. pages 93–100, 01 2004.

[30] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394. Association for Computational Linguistics, 2010.