Trabajo de Fin de Grado

Grado en Ingeniería Informática

Extracción de Información a partir de correos electrónicos usando técnicas de PLN

Information Extraction from e-mails using NLP techniques

Sergio Jesús Rodríguez Martín

La Laguna, 25 de junio de 2017

D. **Isabel Sánchez Berriel**, con N.I.F. 42.885.838-S profesor Contratado Doctor de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Rodrigo Martín Cano**, con N.I.F. 78.692.554-D Program Manager en Datum Solutions, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Extracción de Información a partir de correos electrónicos usando técnicas de PLN"

ha sido realizada bajo su dirección por D. **Sergio Jesús Rodríguez Martín**, con N.I.F. 78.644.661-W.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 25 de junio de 2017.

Agradecimientos

Me gustaría mostrar mi agradecimiento a mi tutora Isabel Sánchez Berriel por toda la ayuda, esfuerzo e interés entregados al proyecto.

También me gustaría expresar mi agradecimiento a Beatriz García y a Rodrigo Martín de Datum Solutions por su interés y ayuda.

Licencia

Resumen

El objetivo de este trabajo de fin de grado es la determinación y evaluación de las técnicas, herramientas y tareas idóneas para la creación de un sistema que tome como entrada un conjunto de correos electrónicos y que, analizando su contenido, sea capaz de discernir quién, en una organización, debe recibirlos.

Para alcanzar este objetivo se ha hecho un análisis de diferentes tecnologías y herramientas orientadas al trabajo con el procesamiento del lenguaje natural y se ha implementado una solución utilizando dichas herramientas. Los datos obtenidos son almacenados en una base de datos para su posterior análisis.

Los destinatarios principales de este proyecto son aquellas organizaciones que deseen automatizar el proceso de lectura de los correos electrónicos que reciben y su posterior redirección a los diferentes departamentos de las mismas.

Palabras clave: *Procesamiento del lenguaje natural, Organización, Empresa, Base de datos, Python, Freeling*

Abstract

The main goal of this project is to determine and evaluate which techniques and tools would be ideal for the creation of a system that takes as an input a set of e-mails and that, after analyzing their content, can find who, within an organization, should receive them.

In order to reach this objective, several natural language processing technologies and tools have been analyzed and a solution has been implemented using them. The resulting data is stored in a database for its later analysis.

This project is mainly addressed to those organizations that want to make the process of reading their received e-mails and redirecting them to the corresponding departments automatic.

Keywords: *Natural language processing, Organization, Company, Database, Python, Freeling*

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

El uso del procesamiento del lenguaje natural en las organizaciones presenta una oportunidad de cara a agilizar muchos de sus trámites analizando los datos de sus usuarios, por ejemplo, de los correos electrónicos que intercambian con los mismos. Dada la desbordante cantidad de información que se maneja en el día a día es prácticamente indispensable disponer de un sistema que la clasifique y que se encargue de entregarla a quienes corresponda.

Este trabajo de fin de grado surge en colaboración con la empresa Datum Solutions con la idea de determinar y evaluar técnicas, herramientas y tareas idóneas para la creación de un sistema que tome como entrada un conjunto de correos electrónicos y que analizando su contenido sea capaz de discernir quién, en una organización, debe recibirlos.

1.1 Objetivos

El principal objetivo del proyecto es la creación de una herramienta que sea capaz de obtener información a partir de los correos electrónicos recibidos en una cuenta dada y que utilice dicha información para la detección de entidades y la clasificación de cada correo.

En el desarrollo del proyecto se abordará la obtención de información estructurada a partir de correos electrónicos y sus adjuntos. Se aplicarán técnicas de procesamiento del lenguaje natural sobre los textos en dichos correos. La información obtenida será almacenada utilizando una base de datos NoSQL, debido a su eficiencia y escabilidad.

1.2 Alcance

En el trabajo se llevarán a cabo las siguientes tareas:

- Integración con un servicio de correo electrónico como GMail.
- Lectura de la bandeja de entrada de una cuenta de correo electrónico de dicho servicio.
- Detección del idioma de cada correo para poder aplicar posteriormente las técnicas de procesamiento del lenguaje natural adecuadas.
- Aplicación de técnicas de procesamiento del lenguaje natural sobre el contenido de dichos correos para la obtención de datos relevantes.
- Clasificación de los correos electrónicos en función de su contenido tras haber aplicado un análisis con herramientas de procesamiento del lenguaje natural.
- Almacenamiento de los datos obtenidos utilizando una base de datos NoSQL.
- Evaluación de resultados

1.3 Antecedentes

Respecto a los objetivos de este trabajo, podemos encontrar proyectos basados en el análisis de correos electrónicos para filtrarlos en función de si son correos basura o importantes [1]. Otro ejemplo es un proyecto de 2013 que se basa en la extracción, de un grupo grande de correos electrónicos pertenecientes a un trabajador, una muestra de correos electrónicos que represente las funciones del cargo del trabajador [2].

En el proyecto se hará uso de Freeling, que provee un conjunto de herramientas de análisis lingüístico.

1.4 Destinatarios

Este proyecto se encuentra dirigido a aquellas organizaciones que deseen automatizar el proceso de lectura de los correos electrónicos que reciben y su posterior redirección a diferentes departamentos de la misma.

Capítulo 2.
Análisis del problema

En este capítulo se presentarán cada uno de los aspectos que se han analizado respecto al problema abordado en el trabajo. En primer lugar, las fuentes de información para obtener un conjunto de correos sobre los que realizar las pruebas. A continuación, se explica la opción elegida para la persistencia de los datos, en este caso las bases de datos NoSQL y por último las técnicas de procesamiento de lenguaje natural con las que se experimentó.

2.1 El correo electrónico como fuente de información de una organización

En nuestros días uno de los principales métodos de comunicación entre las organizaciones y sus clientes es el correo electrónico. Este provee a los mismos de una sencilla manera en la que comunicarse, y a la vez les permite intercambiar información en mayor cantidad y mejor formato que otros sistemas actuales, como la mensajería instantánea.

Es por esto que el análisis de los datos contenidos en los correos electrónicos de una organización facilita la recogida de información de la misma. Gracias a ellos podemos conocer datos como proyectos establecidos con los clientes, compras de productos, quejas recibidas por los clientes, etc.

Además, si la organización posee una cuenta de correo electrónico pública será susceptible a la recepción de correo basura o no relevante para el funcionamiento de la misma, por lo que su análisis también facilitaría su detección y eliminación.

En este proyecto, la principal fuente de información han sido los correos electrónicos recibidos en una cuenta que simula la de una organización. Además, con el objetivo de abarcar la mayor cantidad de organizaciones posibles, se supone que a dicha cuenta pueden llegar correos escritos en

diferentes idiomas, por lo que la herramienta implementada ha de estar preparada para ello. Por esta razón se han realizado envíos a ella desde diferentes fuentes que se pasan a describir a continuación.

2.1.1 TREC Spam Track [3]

TREC (Text REtrieval Conference) es una conferencia celebrada cada año en Estados Unidos, cuyo propósito es apoyar la investigación en el campo de la recogida de información. Para ello ponen a disposición de sus participantes (y posteriormente a todo el mundo), conjuntos de datos (tracks) sobre los cuales los participantes han de aplicar las técnicas de procesamiento exigidas por TREC.

En el proyecto se ha decidido hacer uso de un track de correos basura, que contiene 25220 correos relevantes y 50199 correos basura.

2.1.2 Enron Mail Dataset [4]

Enron fue una empresa estadounidense que se vio envuelta en una investigación por fraude financiero en 2002. Tras la conclusión de las investigaciones, las autoridades hicieron pública una recopilación de correos electrónicos pertenecientes a cerca de 150 trabajadores de la empresa.

Se ha elegido hacer uso de este conjunto de correos electrónicos ya que es una muestra real de correspondencia dentro de una organización.

2.1.3 Linguee

Linguee [5] es un diccionario multilingüe en línea centrado en la búsqueda de palabras en otros idiomas. Se caracteriza por ofrecer ejemplos de uso de las palabras buscadas en el idioma seleccionado.

Es por esta última característica por la que se ha escogido como fuente de información, ya que permite extraer pequeñas porciones de texto que giran alrededor de una palabra determinada. Con estos datos se ha podido controlar los experimentos iniciales para valorar la precisión de los resultados que se obtienen.

2.2 NoSQL

NoSQL (o no sólo SQL) es una clase de sistemas de gestión de bases de datos que difieren del modelo clásico de Sistema de Gestión de Bases de Datos Relacionales (SGBDR) en varios aspectos, estando entre sus características el que no utilizan SQL como el lenguaje principal para sus consultas, de ahí su nombre.

Además, los datos almacenados no requieren estructuras fijas, sino que se registran con diferentes formatos en una misma tabla. Los sistemas de bases de datos NoSQL suelen clasificarse en función del método de almacenamiento que sigan, existiendo diferentes tipos:

- **clave-valor:** utilizan como estructura para almacenar los datos una tabla hash o diccionario, donde a cada valor almacenado se le asigna una clave única. Algunos ejemplos son ArangoDB, InfinityDB y Oracle NoSQL Database.
- **documentales:** las bases de datos documentales asumen que los datos son encapsulados en documentos siguiendo formatos estándar como XML y JSON. Algunos ejemplos son MongoDB y CouchDB.
- **orientados a objetos:** como su nombre indica, en este tipo de bases de datos la información se almacena en objetos como los utilizados en los lenguajes de programación orientada a objetos. Es por ello que se suelen utilizar junto con alguno de estos lenguajes, almacenando sus objetos directamente. Algunos ejemplos son ObjectDatabase++ (C++) y ObjectDB (Java).
- **orientados a grafos:** las bases de datos orientadas a grafos se caracterizan, como su nombre indica, por el hecho de que los datos almacenados en las mismas se relacionan entre sí mediante una representación de grafo. Algunos ejemplos son AllegroGraph, FlockDB e InfiniteGraph, entre otros.

Este tipo de sistemas de bases de datos surgió fruto del crecimiento de la web en tiempo real, principalmente debido al auge de las redes sociales. Por esta razón se dio la necesidad de hacer uso de bases de datos ágiles y altamente escalables, ya que el volumen de datos aumentaba a una velocidad que los sistemas de bases de datos tradicionales no podían manejar. Es por esto que la mayoría de las bases de datos NoSQL están optimizadas para

acelerar las operaciones de recuperación y agregación, y no tanto para las operaciones de consistencia de datos.

La principal razón por la que se ha elegido esta tecnología para la persistencia de datos es la gran cantidad de correos electrónicos que puede llegar a recibir una organización en un solo día. Por ello, es preciso hacer uso de una base de datos capaz de manejar un crecimiento como tal.

2.3 Procesamiento del Lenguaje Natural

El procesamiento del lenguaje natural (PLN) es una herramienta muy extendida en la época actual y muchos proyectos hacen uso de ella. Es un campo de la inteligencia artificial, ciencias de la computación y lingüística que se basa en las interacciones de los ordenadores con el lenguaje humano. Su objetivo es permitir que los ordenadores puedan comprender textos redactados por humanos. Entre sus principales aplicaciones se encuentran el resumen y la traducción automática, y la utilizada en este proyecto, la extracción de información. Dependiendo del propósito con el que se use será necesario aplicar diferentes tareas, como el análisis morfológico, sintáctico-semántico y la identificación del lenguaje, entre otras.

En este proyecto en concreto se utilizan técnicas para la detección del idioma y la extracción de tópicos de los textos de los correos. Su implementación requiere llevar a cabo tareas típicas del PLN, como la tokenización, el análisis morfosintáctico, la detección de entidades, etc. En este apartado se exponen las técnicas utilizadas en el proyecto.

2.3.1 Detección del idioma

Dentro del procesamiento del lenguaje natural, la detección del idioma, como su propio nombre indica, se encarga de determinar en qué idioma se encuentra el texto a procesar.

Uno de los algoritmos más básicos en este campo consiste en comparar el texto a procesar con otros textos escritos en un conjunto de lenguajes conocidos. Si existe una alta coincidencia de palabras entre el texto a procesar y alguno de los textos conocidos se puede determinar que el texto objetivo está en el idioma del conocido. Sin embargo, para conseguir resultados fiables

con este algoritmo es necesario que los textos con los que se compare estén compuestos de un número bastante elevado de palabras del idioma.

Una aproximación más elaborada y precisa es el uso de un algoritmo clasificador Naive-Bayes, donde los diferentes idiomas son las categorías en las que clasificar. El clasificador se entrena con un conjunto de modelos de n-gramas que se obtienen a partir de textos de entrenamiento en los idiomas que se quieran detectar, tal como lo describen Cavnar y Trenkle (1994) [6]. En su estudio, describen un n-grama como un corte de n caracteres de una palabra de tamaño mayor a n. Una palabra puede ser dividida en un conjunto de n-gramas que se sobreponen entre sí. Por ejemplo, la palabra "datos" puede descomponerse en los siguientes n-gramas:

- bi-gramas: _D, DA, AT, TO, OS, S_
- tri-gramas: _DA, DAT, ATO, TOS, OS_, S___
- quad-gramas: _DAT, DATO, ATOS, TOS_, OS___, S____
- penta-gramas: _DATO, DATOS, ATOS_, TOS___, OS____, S_____

Por lo general, una palabra de tamaño k tendrá $k + 1$ n-gramas de cada tipo, hasta que n sea igual a k.

En 2010, Gottron y Lipka [7] realizaron una clasificación de métodos de detección del idioma en textos cortos, como los que se pueden encontrar en un correo electrónico. Para su estudio entrenaron varios clasificadores basados en diferentes algoritmos (Naive-Bayes, Multinomial, modelo de Markov, modelo de rango-frecuencia y modelo de espacio vectorial) con textos extraídos de noticias en diferentes idiomas.

En sus conclusiones determinaron que los clasificadores Naive-Bayes basados en un modelo de n-gramas proporcionan la identificación más fiable en muestras de texto pequeñas.

2.3.2 Modelado de tópicos

El modelado de tópicos es un campo del aprendizaje automático y del procesamiento del lenguaje natural que consiste en la identificación de temas abstractos que pueden existir dentro de una colección de documentos. Los temas o tópicos son conjuntos de términos arbitrarios que se repiten a lo largo

de los documentos y que pueden utilizarse para representar el contenido semántico de los mismos.

Un documento suele contener diferentes temas en diferentes proporciones, por ejemplo, en un documento que trate sobre el tráfico en una autopista, sería normal encontrar un porcentaje mayor de palabras relacionadas sobre coches que sobre otros temas, como peatones. Aplicando esta técnica a dicho documento uno de los tópicos que se extrajeran podría ser, por ejemplo: t = [“coche”, “velocidad”, “accidente”]. Una vez obtenido este tópico, si se quiere analizar otro documento sobre el mismo modelo de tópicos, se ha de tomar la decisión de si el nuevo documento se puede asociar t o si se ha de crear uno nuevo. Por ejemplo, si el nuevo documento a procesar trata de las ventajas a la salud de caminar, con palabras comunes como “caminar”, “andar” y “parque”, que difieren bastante de las palabras del anterior documento, el modelo asignaría a este documento un nuevo tópico, ya que se debería obtener una probabilidad de pertenecer a t muy baja.

El modelo más utilizado dentro del modelado de temas es el Latent Dirichlet Allocation o LDA (2003, Blei, Ng y Jordan) [8]. En este modelo, los documentos, definidos como secuencias de N palabras, se representan como una mezcla arbitraria sobre temas latentes, donde cada tema se compone por una distribución de palabras, definidas como elementos de un diccionario de vocabulario a los cuales se asigna un índice.

Aunque no aparece en la descripción original del modelo LDA, implementaciones posteriores han utilizado el modelo “bolsa de palabras” (bag of words) para representar los documentos con los que entrenar LDA. Este modelo se basa en la representación de las palabras únicas de un conjunto de documentos sin importar su orden. El objetivo principal de este tipo de modelo es conocer la frecuencia de las palabras que aparecen en los documentos, para así ayudar en su clasificación.

Por ejemplo, si disponemos de las siguientes frases,

- *“Pepe juega mucho con su hijo Juan”* y
- *“Juan se lo pasa bien cuando juega mucho con su cochecito y cuando juega con Pepe”*,

un modelo bolsa de palabras construiría la siguiente “bolsa”:

"Pepe, juega, mucho, con, su, hijo, Juan, se, lo, pasa, bien, cuando, cochecito, y"

Para conocer la frecuencia de cada palabra en las dos frases, se evaluarían contra el modelo obteniendo los siguientes resultados:

1. [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]
2. [1, 2, 1, 2, 1, 0, 1, 1, 1, 1, 1, 2, 1, 1]

En la segunda lista de resultados, asociada a la segunda frase, vemos un "2" en la segunda posición ya que la palabra "juega" aparece en la segunda frase dos veces. Los mismo ocurre con la palabra "con" y con la palabra "cuando".

Como se puede observar, el modelo bolsa de palabras nos puede ayudar a conocer la frecuencia de una palabra en un texto de manera sencilla, pero es incapaz de determinar si dicha palabra es relevante dentro del texto, ya que palabras con escaso valor semántico, como conectores y conjunciones, suelen aparecer con mucha frecuencia en los textos. Es por esto por lo que se hace uso de modelos como el LDA para refinar los datos que ofrece el modelo bolsa de palabras.

Capítulo 3. Herramientas a utilizar en la implementación

Una vez explicadas las tecnologías en las que se basa el proyecto, en este capítulo se hablará de diversas herramientas basadas en ellas que se han valorado para determinar si su uso resultaría útil en la implementación de una solución para resolver el problema que se está tratando de resolver.

3.1 Herramientas utilizadas

3.1.1 Python

Python [9] es un lenguaje de programación interpretado, con tipado dinámico y multiparadigma. Soporta la programación orientada a objetos, la programación imperativa y la programación funcional. Se centra alrededor de la llamada "filosofía Python", que marca unos principios de transparencia y legibilidad para el código desarrollado bajo el lenguaje.

Python surge a finales de la década de los ochenta en los Países Bajos como un sucesor al lenguaje ABC, utilizado hasta la fecha en el Centrum Wiskunde & Informatica (Centro para las Matemáticas y la Informática).

En la actualidad Python se está imponiendo como uno de los lenguajes de programación más empleados en el campo de la ciencia, ya que dispone de numerosas herramientas en la Ciencia de los Datos.

3.1.2 R

R [10] es un lenguaje de programación orientado al análisis estadístico. Es también multiparadigma, soportando la programación orientada a objetos, funcional, imperativa, vectorial y procedural.

Es uno de los lenguajes más utilizados en el campo de la matemática y estadística, aunque también es popular en otros campos, como el procesamiento del lenguaje natural y la minería de datos, debido a su soporte a librerías y paquetes externos.

3.1.3 API Oficial de Google

Google pone a disposición de los desarrolladores una extensa API con acceso a todos sus servicios, tales como Maps, GMail, Search, etc. En el caso de este proyecto se hará uso de la API de Gmail [11], que permite el acceso a cuentas de correo de Google y la posibilidad de leer y/o enviar correos desde las mismas.

Desde el punto de vista de un desarrollador, la API de GMail ofrece soporte para numerosos lenguajes de programación (como Python, Java, Ruby, JavaScript, etc.), además de la posibilidad de poder hacerle peticiones REST directamente. Dispone de métodos para realizar todo tipo de operaciones sobre una cuenta de correo electrónico de la compañía, siendo la más destacable la posibilidad de recuperar correos electrónicos que se ajusten a criterios de búsqueda fijados por el desarrollador, utilizando la misma notación que el sistema de búsqueda de la bandeja de entrada de GMail. Un ejemplo de búsqueda con filtro podría ser el siguiente:

- *from:(unacuenta@gmail.com) after:2017/5/31 before:2017/6/3*

Este criterio de búsqueda devolvería los correos recibidos desde la cuenta "unacuenta@gmail.com" después del 31 de mayo de 2017 y antes del 3 de junio del mismo año.

Para hacer uso de la API es necesario realizar autenticación OAuth la primera vez que se ejecuta, tras lo cual las credenciales son almacenadas localmente y no requeridas de nuevo hasta que caduquen o si se modifica el nivel de acceso de la aplicación (sólo lectura, lectura y escritura, etc.).

3.1.4 Librerías de Procesamiento del Lenguaje Natural

3.1.4.1 Freeling

FreeLing [12] es una librería de código abierto para el procesamiento multilingüe, que proporciona una amplia gama de funcionalidades de análisis para varios idiomas.

El proyecto FreeLing se inició desde el centro investigador TALP de la Universidad Politécnica de Catalunya para avanzar hacia la disponibilidad general de recursos y herramientas básicos de Procesamiento del Lenguaje Natural (PLN). Se estructura como una librería que puede ser llamada desde cualquier aplicación de usuario que requiera servicios de análisis del lenguaje, distribuyéndose como código abierto bajo una licencia GNU General Public License y bajo licencia dual a empresas que deseen incluirlo en sus productos comerciales.

La versión actual soporta (a diferentes niveles de completitud) los siguientes idiomas: alemán, asturiano, catalán, castellano, croata, esloveno, francés, galés, gallego, inglés, italiano, noruego, portugués y ruso.

	as	ca	cy	de	en	es	fr	gl	hr	it	nb	pt	ru	sl
Tokenization	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Sentence splitting	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Number detection		X		X	X	X	X	X		X		X	X	
Date detection		X		X	X	X	X	X				X	X	
Morphologic dictionary	X	X	X	X	X	X	X	X		X	X	X	X	X
Affix rules	X	X	X	X	X	X	X	X		X	X	X		
Multiword detection	X	X	X		X	X	X	X		X		X		
Basic named entity detecion	X	X	X		X	X	X	X		X		X	X	X
B-I-O named entity detection		X			X	X		X				X		
Named Entity Classification		X			X	X						X		
Quantity		X			X	X		X				X	X	

detection														
PoS tagging	X	X	X	X	X	X	X	X		X	X	X	X	X
WN sense annotation		X			X	X	X	X	X					X
UKB sense disambiguation		X			X	X	X	X	X					X
Semantic Role Labelling		X		X	X	X								
Coreference resolution					X	X								

Tabla 1. Idiomas soportados por Freeling (Fuente: manual de usuario de Freeling [13])

La arquitectura de la librería se basa en un enfoque de dos capas cliente-servidor: una capa básica de servicios de análisis lingüístico (morfológico, morfosintáctico, sintáctico, detección del idioma...) y una capa de aplicación que, actuando como cliente, realiza las peticiones deseadas a los analizadores y usa su respuesta según la finalidad de la aplicación.

A continuación, se muestra un ejemplo de un análisis realizado por Freeling sobre una frase en español ("La canasta de consumo") y devuelto en XML:

```
<sentences><sentence id="1">
 <token id="t1.1" form="La" lemma="el" tag="DA0FS0" ctag="DA" pos="determiner"
type="article" gen="feminine" num="singular">
 </token>
 <token id="t1.2" form="canasta" lemma="canasta" tag="NCFS000" ctag="NC" pos="noun"
type="common" gen="feminine" num="singular">
 </token>
 <token id="t1.3" form="de" lemma="de" tag="SP" ctag="SP" pos="adposition"
type="preposition">
 </token>
 <token id="t1.4" form="consumo" lemma="consumo" tag="NCMS000" ctag="NC" pos="noun"
type="common" gen="masculine" num="singular">
 </token>
</sentence>
</sentences>
```

Figura 1. Ejemplo de la salida del análisis de Freeling

El analizador realiza una tokenización de cada palabra de la frase y le asigna a cada una las etiquetas correspondientes:

- **form:** el token.

- **lemma:** la forma lematizada del token.
- **tag:** la etiqueta gramatical del token (si es un nombre común o propio, un verbo, un adjetivo…). Para indicar cada una de estas categorías se utiliza el sistema de etiquetas EAGLES [14].
- **ctag:** forma simple de la etiqueta gramatical.
- **pos:** posición de la palabra en el texto (sustantivo, verbo, determinante…).
- **type:** el tipo de palabra. Varía en función de su posición (un determinante puede ser un artículo, posesivo, demostrativo, etc.).
- **gen:** el género de la palabra, si lo tiene.
- **num:** el número de la palabra (singular, plural o invariable).

3.1.4.2 API de Freeling para Python: PyFreeling

PyFreeling [15] es una librería para Python que encapsula funciones para comunicarse con el analizador de Freeling directamente desde un programa en Python.

Permite indicar a Freeling el nivel de análisis a realizar y el idioma en el que estará el texto a procesar. Devuelve los resultados de Freeling en formato XML.

3.1.4.3 Stanford CoreNLP

Stanford CoreNLP [16] es un conjunto de herramientas de procesamiento y análisis del lenguaje humano. Provee la mayoría de las tareas comunes en el procesamiento del lenguaje natural (PLN), desde la tokenización hasta la resolución de correferencias.

Tal como su nombre indica es un proyecto iniciado por la Universidad de Stanford, en California (EEUU). Actualmente se encarga de su mantenimiento y actualización el Stanford NLP Group, un equipo de investigadores, programadores y estudiantes de la misma universidad.

CoreNLP se encuentra estructurado como una librería Java, lo cual permite su uso desde cualquier código de este lenguaje. Sin embargo, existen varias implementaciones en diferentes lenguajes de programación que proveen "wrappers" para la librería original, incluido el paquete estadístico R.

Además del inglés, CoreNLP también pone a disposición del usuario sus herramientas de PLN para otros idiomas como el árabe, chino, francés, español y alemán.

ANNOTATOR	AR	ZH	EN	FR	DE	ES
Tokenize / Segment	✓	✓	✓	✓		✓
Sentence Split	✓	✓	✓	✓	✓	✓
Part of Speech	✓	✓	✓	✓	✓	✓
Lemma			✓			
Named Entities		✓	✓		✓	✓
Constituency Parsing	✓	✓	✓	✓	✓	✓
Dependency Parsing		✓	✓	✓	✓	
Sentiment Analysis			✓			
Mention Detection		✓	✓			
Coreference		✓	✓			
Open IE			✓			

Figura 2. Idiomas soportados por Stanford CoreNLP (Fuente: sitio web de Stanford CoreNLP [17])

3.1.4.4 NLTK

NLTK [18] es una plataforma para el desarrollo de programas que trabajen con datos de lenguaje natural bajo el lenguaje de programación Python. Incorpora 50 corpus con datos para el entrenamiento de sus algoritmos o para realizar pruebas sobre los programas de los usuarios, así como un conjunto de librerías que se encargan de las principales funciones del procesamiento del lenguaje natural.

Uno de sus principales objetivos es el apoyo a la enseñanza e investigación en PLN u otras áreas relacionadas, como las ciencias cognitivas, inteligencia artificial y el machine learning, entre otras.

NLTK no posee soporte multilenguaje como base, pero puede ser entrenado a partir de diferentes corpus en idiomas diferentes al inglés.

3.1.4.5 Librería de detección del idioma: langdetect

La librería langdetect [19] es una conversión a Python de language-detection, una librería de detección de idiomas para Java [20], la cual posee un 99% de precisión en la detección para un total de 49 idiomas diferentes. Además, es capaz de realizar la detección rápidamente, ya que no necesita leer el documento en su totalidad, finaliza el análisis una vez puede asegurar con certeza el idioma en el que está redactado.

Internamente implementa un algoritmo de clasificación Naive-Bayes, el cual hace uso de una representación de palabras basada en n-gramas, entrenado a partir de textos extraídos de Wikipedia.

```
>>> from langdetect import detect
>>> detect("War doesn't show who's right, just who's left.")
'en'
>>> detect("Ein, zwei, drei, vier")
'de'
```

Figura 3. Ejemplo de funcionamiento de langdetect

3.1.4.6 Librería de modelado de tópicos: gensim

Gensim [21] es un conjunto robusto de herramientas open-source enfocado al modelado de temas (topic modelling) y al modelado de vectores de términos (vector space model[1]) implementado en Python. Está específicamente diseñado para procesar grandes cantidades de texto, usando streaming de datos y algoritmos incrementales eficientes, lo que lo diferencia de otros softwares científicos que se centran en el procesamiento en memoria y en lotes.

Enfoca el modelado de temas haciendo uso de un modelo Latent Dirichlet Allocation. Para ello también hace uso del modelo bolsa de palabras, creando

[1] Representación de las palabras de un texto como vector numérico para realizar diferentes cómputos.

primero un diccionario (la “bolsa”) en el que se almacenan las palabras de los documentos a procesar, cada una con un índice asociado. Este diccionario es utilizado como recurso por el modelo LDA, donde se utiliza para determinar el tamaño del vocabulario y para la posterior visualización de tópicos.

El modelo LDA es entrenado con los documentos especificados por el usuario, tras convertirlos también a un modelo bolsa de palabras y formar una matriz de términos con todas las “bolsas”. Una vez ha sido entrenado, se pueden procesar nuevos documentos contra el modelo LDA para conocer sus tópicos.

```
>>> lda = LdaModel(corpus, num_topics=100)  # se entrena el modelo a partir
del corpus “corpus”
>>> print(lda[doc_bow]) # Se obtiene la lista de tópicos asociada al
documento “doc_bow” (ya transformado al modelo bolsa de palabras)
>>> lda.update(corpus2) # se actualiza el modelo LDA con otro corpus,
“corpus2”
>>> print(lda[doc_bow]) # Se vuelve a obtener la lista de tópicos para
“doc_bow” que probablemente habrán cambiado al añadir otro corpus de
entrenamiento
```

Figura 4. Ejemplo de funcionamiento de gensim

3.1.5 MongoDB

MongoDB [22] es un sistema de bases de datos de tipo NoSQL orientado a documentos centrado en la agilidad de sus operaciones y la escabilidad. Está desarrollado bajo el concepto de código abierto y se puede hacer uso del mismo de forma gratuita.

Entre las principales características de MongoDB se encuentran la escalabilidad, el rendimiento y una gran disponibilidad, pasando de una implantación de servidor único a grandes arquitecturas complejas de centros multidatos, lo que permite la adaptación para su uso en organizaciones.

Gracias al uso de la computación en memoria, ofrece buenos resultados de rendimiento tanto en lectura como escritura. Además, es capaz de ofrecer fiabilidad a nivel empresarial debido a que dispone de replicación de datos de forma nativa y tolerancia a fallos automática.

3.1.5.1 API de MongoDB para Python: pymongo

Pymongo [23] es una librería en Python que contiene las herramientas necesarias para trabajar con una base de datos MongoDB. Es un proyecto oficial de MongoDB y la recomendación de la organización para trabajar con sus bases de datos a través de Python.

Capítulo 4. Implementación y resultados.

Para la implementación de la solución se ha decidido utilizar el lenguaje Python, debido a su sencillez y rapidez de ejecución. En este capítulo se mostrarán qué herramientas de las anteriormente mencionadas se han utilizado junto con Python en su desarrollo.

4.1 Pipeline del procesamiento de correos electrónicos

La herramienta implementada se basa en varios módulos desarrollados en Python, descritos a continuación:

4.1.1 Recuperador de correos electrónicos

En este módulo se hace uso de la API de Google para realizar una conexión a la cuenta de GMail utilizada para las pruebas. Se leen los correos electrónicos de la bandeja de entrada, pudiendo especificar un criterio de búsqueda si se desea. A continuación, se almacena el contenido de cada correo electrónico en formato de texto plano en el sistema.

4.1.2 Clasificador por idiomas

Después de recuperar los correos electrónicos, el siguiente módulo se encarga de analizar cada fichero de texto para primero discernir la codificación de caracteres de sus palabras (necesario al tratar texto en diferentes idiomas) y luego analizar el texto utilizando la herramienta langdetect para detectar el idioma en el que está escrito. Una vez detectado su idioma, cada fichero de texto es trasladado a una carpeta correspondiente a su idioma.

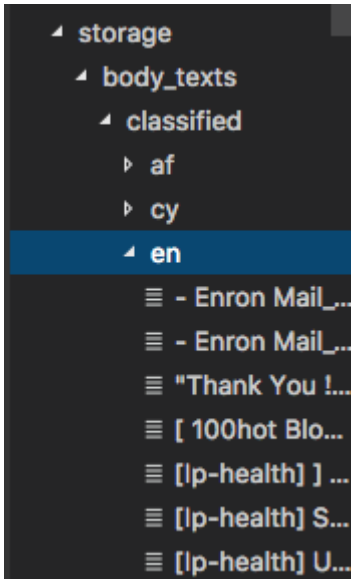

Figura 5. Ejemplo del almacenamiento local de correos recuperados y clasificados por idioma

4.1.3 Lematización y Modelado de tópicos

Una vez los correos electrónicos han sido clasificados por idioma se procede a realizar la identificación de tópicos latentes en los textos. Para ello se realizan dos tareas:

4.1.3.1 Lematización y limpieza del texto

Primero se recuperan los textos almacenados en el sistema por idioma y, si el idioma en el que están escritos es soportado por la librería de procesamiento del lenguaje natural (Freeling en este caso), se pasa el texto a la misma para obtener los lemas de cada palabra. Además, en este paso también se descartan los nombres propios, ya que no aportan semántica.

Tras lematizar el texto, los lemas obtenidos son procesados nuevamente para eliminar palabras vacías o "stopwords", haciendo uso de la librería NLTK, que posee corpus de palabras vacías para varios idiomas. También se eliminan los posibles símbolos, fechas y números que hayan podido quedar tras el proceso de lematización.

```
[u'really', u'appreciate', u'take', u'time', u'professional', u'today',
u'easily', u'pass', u'discussion', u'hope', u'understand', u'important',
u'issue', u'go', u'entire', u'evening', u'everyone', u'make', u'sure',
u'leave', u'stone', u'unturned', u'thanks', u'know', u'uphill', u'battle',
u'hope', u'challednge', u'thanks', u'concern', u'receive']
```

Figura 6. Ejemplo de un texto de un correo electrónico lematizado y sin palabras vacías.

4.1.3.2 Modelado de tópicos

Tras obtener un conjunto de lemas por cada idioma se procede a crear un modelo LDA utilizando la librería *gensim* para Python. Primero se crea un diccionario de *gensim* (un modelo bolsa de palabras) utilizando todos los conjuntos de lemas de cada texto del idioma. El objetivo de este diccionario es la posterior visualización de los tópicos.

En el siguiente proceso se divide el conjunto de lemas del idioma en dos subconjuntos. Uno será utilizado para entrenar el modelo LDA y el otro como conjunto de pruebas para estudiar la precisión de la herramienta.

```
[(0, 0.037345686083570577), (2, 0.033432514356450227), (3,
0.081186377370241605), (4, 0.026102459859072488), (8, 0.027652936668830376),
(10, 0.069030000124742738), (16, 0.64183949713864374), (17,
0.031456375253292916), (19, 0.047849675425353337)]
```

Figura 7. Ejemplo de la salida de *gensim* tras extraer los tópicos de un correo electrónico. Es una lista de t-uplas de dos elementos, siendo el primero el identificador de un tópico y el segundo la probabilidad de que el documento pertenezca al mismo.

4.1.4 Almacenamiento de resultados

El último módulo implementado se encarga de almacenar los resultados obtenidos en el paso anterior en una base de datos MongoDB. Los datos a almacenar son, por cada correo electrónico del conjunto utilizado para las pruebas, el asunto del correo (que contiene información sobre su contenido y será utilizado para contrastar los tópicos obtenidos por la herramienta), el

texto del correo sin procesar y el tópico que se le ha asignado al correo, junto con un índice por tópico e idioma.

Se puede ver como los módulos descritos interactúan entre sí en el siguiente diagrama:

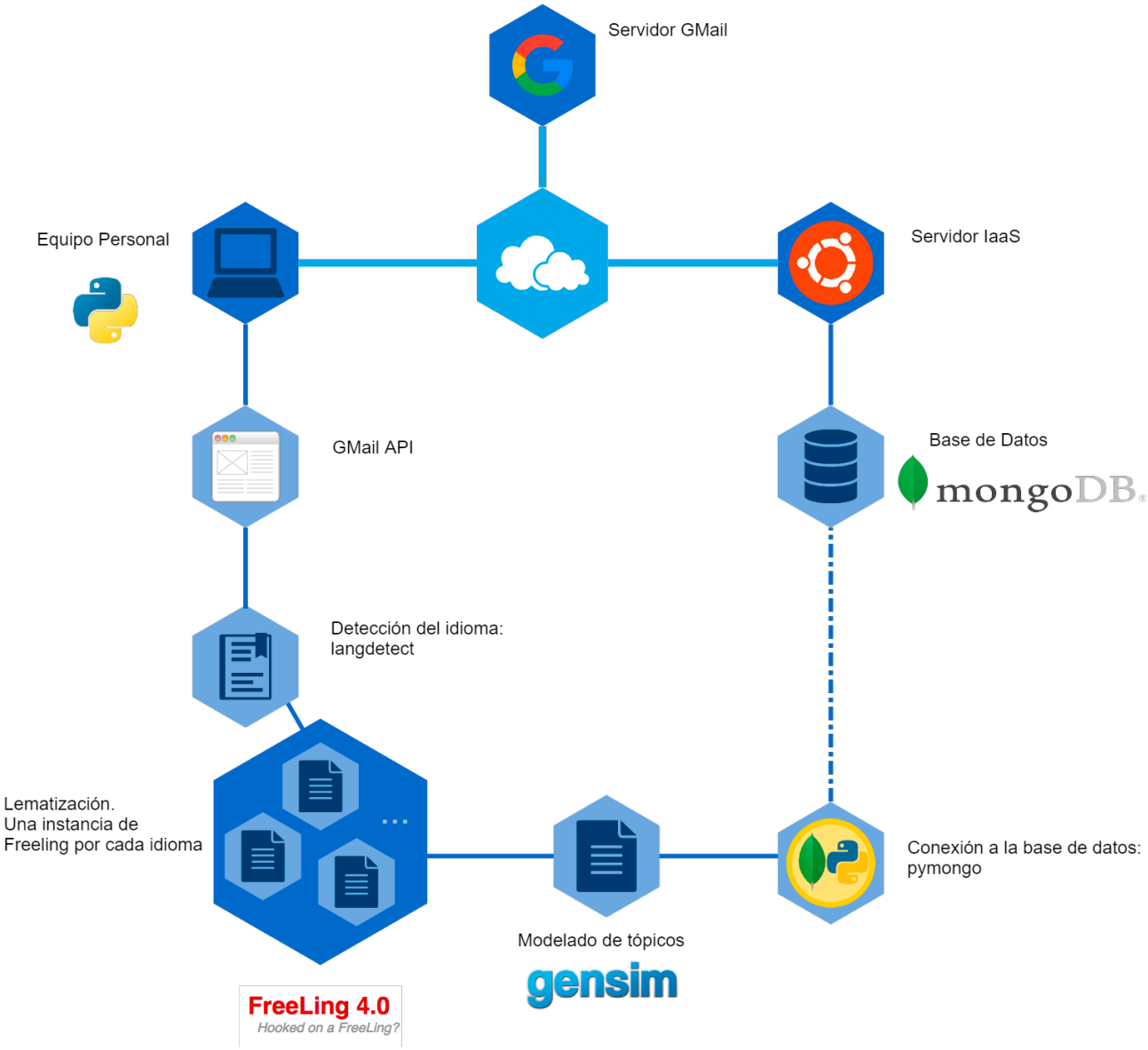

Figura 8. Diagrama de flujo de datos

4.2 Descripción de la base de datos

Para la base de datos de la herramienta se ha decidido hacer uso del sistema gestor NoSQL MongoDB, debido a su rapidez y escabilidad.

Gracias a la prácticamente inexistente restricción en sus estructuras no ha sido necesario especificar los tipos de los datos, siendo los mismos los siguientes:

- Nombre de la base de datos: **email**
  - Campos:
    - **mail_id (Índice único):** identificador único asignado por Google al correo electrónico.
    - **subject:** asunto original del correo electrónico.
    - **text:** texto sin procesar del correo electrónico.
    - **topic_id:** identificador de tópico por idioma.
    - **topic:** tópico asociado al correo electrónico.


```
{
    "_id": ObjectID("594c319da0ab6609d8dfc4bb"),
    "topic": [
        [
            "business",
            0.010198841605675645
        ],
        [
            "new",
            0.007777802662795284
        ],
        [
            "need",
            0.006947669061618188
        ],
        [
            "time",
            0.006827040475978696
        ],
        [
            "get",
            0.0065916354259021
        ]
    ],
    "text": "Again sorry Susan but I can not make it this week.  I wish I
couldbut I have \nbeen out too much latley  with work.  Thanks alot for the
invite.  Please \ngive Pam my best.  Have Fun !!",
    "mail_id": "15cd18bc4df0c87a",
    "topic_id": "31",
    "subject": "Re: - Enron Mail"
```


Figura 9. Ejemplo de documento que representa un resultado almacenado en MongoDB.

4.3 Análisis de resultados

Para analizar los resultados de la solución implementada, nos centramos en una muestra de 159 correos en inglés, español y alemán obtenidos de Linguee, donde 79 de los mismos se han utilizado para entrenar el modelo LDA y los 80 restantes para comprobar el funcionamiento del modelo. La precisión de los tópicos se mide utilizando el asunto del correo (que contiene el tema del que trata el mismo) y comparándolo con los tópicos obtenidos para dicho correo. Por ejemplo, un correo con asunto "curriculum" o "solicitud de empleo" tendrá un tópico preciso asignado si este contiene palabras como "solicitar" o "vitae". Con dicha muestra obtenemos los siguientes resultados:

- **Con 5 tópicos por idioma** la precisión total de los tópicos es de un **80%:**
    - **Un 74% de los tópicos en inglés han sido precisos.**
    - **Un 87% de los tópicos en español han sido precisos.**
    - **Un 80% de los tópicos en alemán han sido precisos.**
- **Con 10 tópicos por idioma** la precisión total de los tópicos es de un **88%:**
    - **Un 93% de los tópicos en inglés han sido precisos.**
    - **Un 90% de los tópicos en español han sido precisos.**
    - **Un 80% de los tópicos en alemán han sido precisos.**

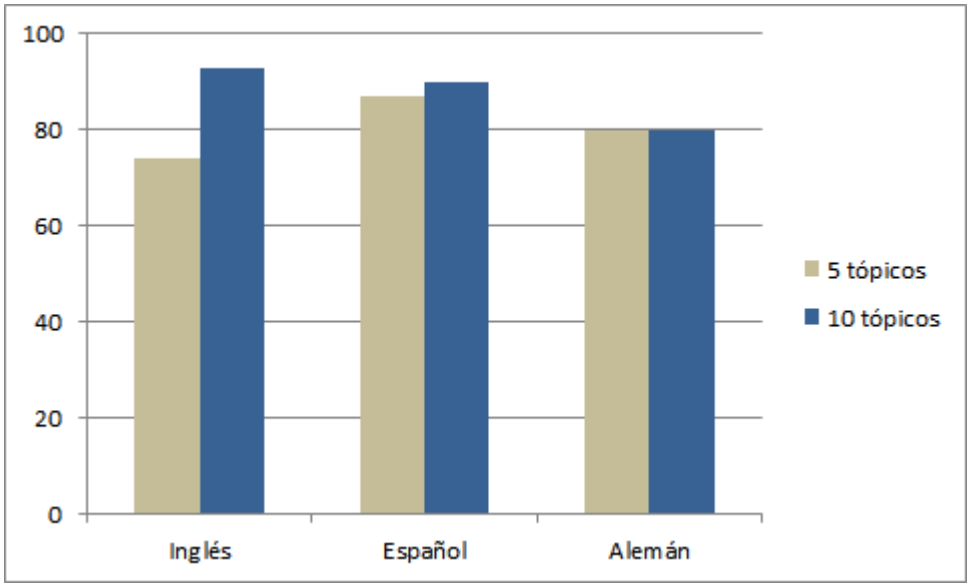

Figura 10. Gráfico que muestra cómo se distribuye el grado de precisión de los resultados obtenidos

Como se puede observar, al aumentar el número de tópicos generados por idioma, la precisión general de los resultados aumenta. Esto se debe a que, al limitar el número de tópicos, el modelo LDA tiende a crear tópicos con palabras ambiguas o que pertenecen a dos posibles temas a la vez. Un ejemplo claro es un tópico que contiene a la vez las palabras "reclamación" y "solicitar", por lo que se le asigna a la misma vez a correos que contienen una reclamación y a correos que solicitan información sobre una oferta de trabajo. Además, al limitar el número de tópicos también se crean tópicos con verbos que no aportan semántica, como "decir" o "comentar", que pueden aplicarse a un gran número de correos, pero que no aportan nada a la solución.

También puede aparecer en un tópico una palabra que pueda pertenecer a dos correos a la vez. Por ejemplo, la palabra "experiencia" puede asociarse a un correo sobre una oferta de trabajo donde el candidato menciona su experiencia laboral, pero también puede asociarse a un correo sobre una reclamación donde la persona afectada relata la mala experiencia que ha sufrido.

Por estas razones se decidió aumentar el número de tópicos por idioma a 10, lo que permite a *gensim* crear tópicos más ajustados a cada correo procesado. Aun así, no es recomendable aumentar demasiado el número de tópicos, ya que, si supera el número de correos, *gensim* asignará un tópico a cada correo basándose en su texto individualmente.

4.4 Aspectos a destacar

Al inicio del proyecto se decidió utilizar como librería principal de procesamiento del lenguaje natural Stanford CoreNLP, ya que posee soporte multi-idioma y debido a su facilidad de uso respecto a otras librerías de PLN. Además, también tiene una gran variedad de "wrappers" en muchos lenguajes de programación, incluido R. Sin embargo, cuando se decidió enfocar el proyecto hacia el descubrimiento de tópicos en los correos electrónicos, se descartó esta librería, ya que estos algoritmos requieren que las palabras sean lematizadas para aumentar la precisión y CoreNLP solamente es capaz de lematizar texto en inglés.

Por esta razón se ha elegido la librería Freeling para el procesamiento del lenguaje natural, debido a su capacidad de lematizar texto en otros idiomas además del inglés.

También en un inicio se consideró el uso de R como lenguaje de programación principal para la aplicación, debido a sus funcionalidades estadísticas y a la existencia de un paquete para el uso Stanford CoreNLP. Sin embargo, debido a la curva de aprendizaje y el hecho de abandonar CoreNLP como librería de PLN se sustituyó por Python como lenguaje para la implementación.

A pesar de que Freeling dispone de una funcionalidad de detección del idioma, tras realizar diversas pruebas se demostró que la librería de Python *langdetect* ofrecía una fiabilidad mayor en sus resultados, por lo que pasó a utilizarse ésta para clasificar los correos electrónicos en diferentes idiomas.

A la hora de detectar idiomas se observó un problema, algunos de los correos contenían caracteres con codificaciones diferentes: ASCII o UTF-8, lo que provocaba que la librería para identificar el idioma fallase. Por ello fue necesario implementar una función capaz de detectar la codificación y en caso necesario decodificarlos antes de procesarlos con *landgdetect*.

En cuanto al proceso de lematización, se encontraron algunos problemas con Freeling, especialmente a la hora de intentar lematizar texto en alemán. A pesar de que posee soporte para este idioma, es un soporte parcial que no permite la lematización. Por lo tanto, se decidió que para el alemán y otros idiomas soportados parcialmente por Freeling no se aplicaría esta tarea, sino

que solamente se eliminarían sus palabras vacías y símbolos que no añaden semántica.

Otro de los problemas encontrados con Freeling fue el hecho de que para cada idioma en el que se clasificaron los correos había que lanzar una instancia diferente de Freeling, lo que aumenta el tiempo de procesamiento de los correos electrónicos, teniendo en cuenta que ya de por sí Freeling es una herramienta que consume bastantes recursos.

4.4.1 Memoria

El principal uso de almacenamiento de la solución implementada proviene de la base de datos MongoDB. A pesar de que la colección en la que se almacenan los correos con sus tópicos solo ocupa 33 MB en disco (en el caso de 303 correos procesados), los ficheros creados por MongoDB para el correcto funcionamiento de la base de datos en su totalidad ascienden a 2,3 GB de espacio en disco.

4.4.2 Tiempo

El tiempo de ejecución de la solución implementada varía en función de los idiomas a lematizar y el número de correos electrónicos procesados. El elemento principal que influye en el tiempo de ejecución es la lematización realizada por Freeling.

- **303 correos en inglés, extrayendo 100 tópicos por correo**: 19 minutos de media.
- **303 correos en inglés, extrayendo 20 tópicos por correo**: 14 minutos de media.
- **159 correos en inglés, español y alemán, extrayendo 5 tópicos por correo**: 20 minutos de media.
- **159 correos en inglés, español y alemán, extrayendo 10 tópicos por correo**: 22 minutos de media.

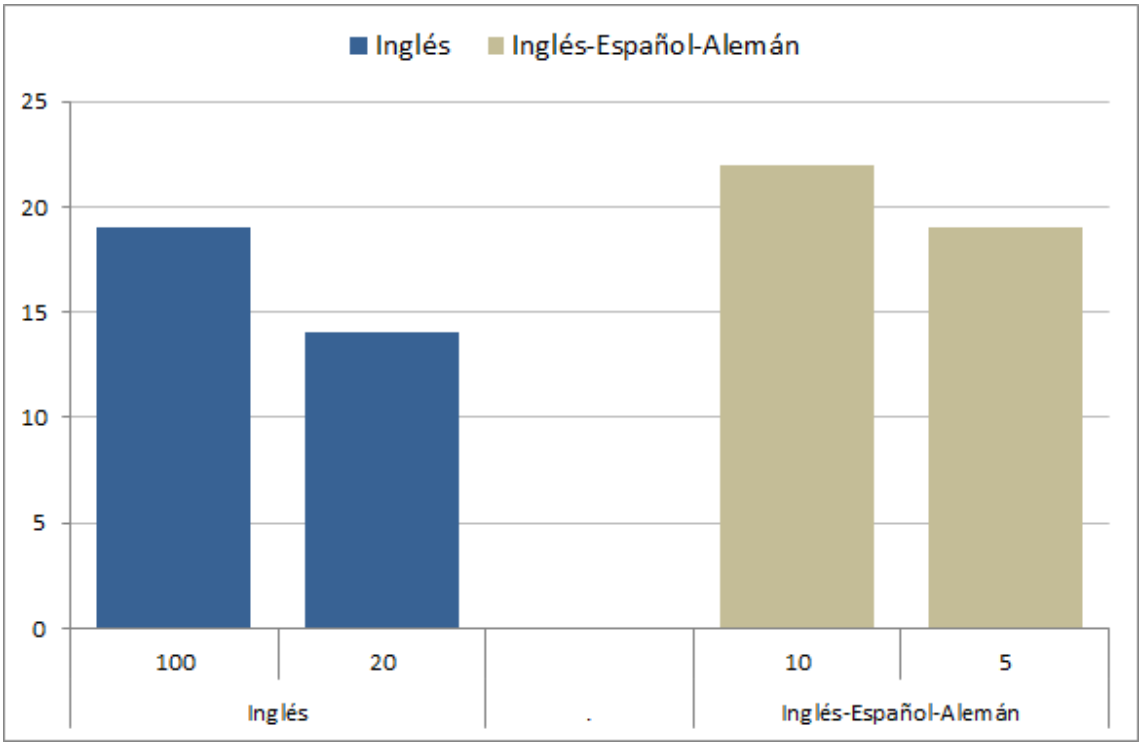

Figura 11. Gráfico que muestra la diferencia de tiempo consumido por los diferentes conjuntos de prueba.

Como se puede observar, cuando se procesan correos en diferentes idiomas, el tiempo de ejecución requerido es similar a cuando se procesa el doble de correos en un solo idioma. Esto se debe principalmente al hecho de que es necesario lanzar una instancia de Freeling por idioma para cada conjunto de correos a lematizar.

También se puede observar que la ejecución de *gensim* influye en el tiempo de ejecución, sobre todo cuando se trabaja con un número grande de tópicos como 100, pero no es una influencia tan notable como la de Freeling.

Capítulo 5. Conclusiones y líneas futuras

5.1 Conclusiones

Cada día que pasa la información recibida por las organizaciones es mayor, a veces hasta llegar a niveles abrumantes. Es por ello que muchas organizaciones recurren a sistemas informáticos para ayudarles a manejar dicha información.

Durante la realización de este Trabajo de Fin de Grado se ha podido realizar un estudio completo de las tecnologías y herramientas necesarias para solucionar este problema, en la forma de un catalogador de correos electrónicos recibidos por una organización. Además, también se han podido poner a prueba dichas tecnologías mediante la implementación de una solución práctica basada en las mismas.

Debido a las obvias restricciones de tiempo que existen a causa del marco educativo que envuelve al proyecto, la solución implementada ofrece una aproximación general al problema planteado y debería ser refinada para adaptarse a los datos concretos de la organización en la que se aplique.

5.2 Trabajo futuro

Algunas de las palabras vacías de los textos procesados por la herramienta desarrollada no desaparecen tras la limpieza del texto. Esto se debe a que el conjunto de palabras vacías ofrecido por NLTK no es muy rico, por lo que algunas no son eliminadas. Una línea de trabajo futuro sería encontrar un conjunto de palabras vacías multi-idioma con mayor cantidad de palabras.

Ya que Freeling no ofrece lematización para todos los idiomas que soporta, una manera de lematizar texto en estos idiomas y otros enteramente no soportados sería trabajar con diccionarios de lemas, con los que se pueden enfrentar las palabras de los textos para encontrar su forma lematizada.

Una de las principales desventajas de la herramienta implementada es el hecho de que aún no es capaz de decidir para qué departamento de una supuesta organización podría ir dirigido un correo electrónico. Una línea de trabajo futuro sería el desarrollo de un clasificador que, analizando las palabras de los tópicos asignados a los correos electrónicos, decida cuáles de los mismos van dirigidos a un departamento y cuáles a otro.

Capítulo 6.
Summary and Conclusions

Every day that passes the information received by organizations gets bigger, sometimes even to overwhelming amounts. Therefore, many organizations turn into computer systems in order to manage that information.

During the development of this project it has been possible to carry out a complete study of the technologies and tools needed to give a solution to this problem, focusing in the idea of cataloguing e-mails received by an organization. Moreover, it has also been possible to implement a tool that tests these technologies, giving an approximate solution to the problem.

Because of the obvious time restrictions that involve this project (having it being part of a degree's final subject) the developed tool only offers an approximate solution to the proposed objective and should be polished in order to adapt to the specific data of the organizations where it could be applied.

Capítulo 7. Presupuesto

Debido a que la totalidad de las herramientas empleadas en la implementación de la solución propuesta son herramientas libres de uso gratuito, el único factor a la hora de presupuestar el proyecto son las horas de trabajo dedicadas al mismo, aparte de la infraestructura hardware necesaria para el desarrollo.

Esta infraestructura hardware se compone de los siguientes elementos:

- **Equipo de trabajo:** Ordenador portátil (donde se ha desarrollado la solución implementada) con un procesador doble núcleo y 5 GB de memoria RAM.
- **Servidor:** Servidor virtual (donde se aloja la base de datos) con cuatro núcleos y 8 GB de RAM.

Tipos	Cantidad	Precio por unidad	Total
Horas de trabajo	150	60€ la hora	9000€
Equipo de trabajo	1	600€	600€
Servidor	1	75€ al mes	En función del uso

Tabla 2. Tabla resumen de los Tipos.

Bibliografía

[1] Haiyan Kang, Xiaojiao Yuan. (June 2014). Natural Language Processing Technologies for Multi-Level Intelligent Spam Mail-Filter. *International Journal of Machine Learning and Computing*, Vol. 4, No. 3, 271-274.

[2] Jorge A. Alvarado, Constanza Cuervo. (2013). Extracción de Funciones de un Cargo usando Minería de Texto en Correos Electrónicos. *Información Tecnológica*, Vol. 24(5), 61-68.

[3] Gordon V. Cormack, Thomas R. Lynam. (2007). *TREC 2007 Public Spam Corpus*. Junio 2017, de TREC. Sitio web: http://plg.uwaterloo.ca/~gvcormac/treccorpus07/

[4] William W. Cohen. (2015). *Enron Email Dataset*. Junio 2017, de CALO Project. Sitio web: https://www.cs.cmu.edu/~./enron/

[5] Gereon Frahling. *Linguee*. Junio 2017, de Linguee GmbH. Sitio web: http://www.linguee.es

[6] William B. Cavnar, John M. Trenkle. (1994). N-Gram-Based Text Categorization. *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, 161-175.

[7] Thomas Gottron, Nedim Lipka. (1994). A Comparison of Language Identification Approaches on Short, Query-Style Texts. *ECIR'2010 Proceedings of the 32nd European conference on Advances in Information Retrieval*, 611-614.

[8] David M. Blei, Andrew Y. Ng, Michael I. Jordan. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3, 993-1022.

[9] Python Software Foundation. (2001). *Python*. Junio 2017, de Python™. Sitio web: https://www.python.org/about/

[10] The R Foundation. (2002). *R*. Junio 2017, de The R Project for Statistical Computing. Sitio web: https://www.r-project.org/

[11] Google. (2017). *GMail API*. Junio 2017, de Google Developers. Sitio web: https://developers.google.com/gmail/api/

[12] Lluís Padró, Evgeny Stanilovsky (2012). FreeLing 3.0: Towards Wider Multilinguality. *Proceedings of the Language Resources and Evaluation Conference (LREC 2012).*

[13] Lluís Padró. (2016). FreeLing User Manual. Junio 2017, de TALP UPC. Sitio web: https://talp-upc.gitbooks.io/freeling-user-manual/content/basics.html

[14] Universitat Politècnica de Catalunya. *Introducción a las Etiquetas EAGLES.* Junio 2017, de Universitat Politècnica de Catalunya. Sitio web: http://www.lsi.upc.es/~nlp/tools/parole-sp.html

[15] Marcos Vanetta. (2016). *PyFreeling.* Junio 2017, de GitHub. Sitio web: https://github.com/malev/pyfreeling

[16] Bauer, J., Bethard, S., Finkel, J.R., Manning, C.D., McClosky, D., Surdeanu, M. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *ACL.*

[17] Stanford NLP Group. *Using Stanford CoreNLP on other human languages.* Junio 2017, de Stanford CoreNLP. Sitio web: https://stanfordnlp.github.io/CoreNLP/human-languages.html

[18] Bird, S., & Loper, E. (2002). NLTK: The Natural Language Toolkit. CoRR, cs.CL/0205028.

[19] Michal Danilák. (2014). *langdetect.* Junio 2017, de GitHub. Sitio web: https://github.com/Mimino666/langdetect

[20] Nakatani Shuyo. (2010). *language-detection.* Junio 2017, de GitHub. Sitio web: https://github.com/shuyo/language-detection

[21] Sojka, P., y Řehůřek, R. (2010). Software Framework for Topic Modelling with Large Corpora.

[22] MongoDB, Inc. (2017). *What is MongoDB?* Junio 2017, de MongoDB. Sitio web: https://www.mongodb.com/what-is-mongodb

[23] MongoDB, Inc. (2015). *PyMongo 3.4.0 Documentation.* Junio 2017, de MongoDB. Sitio web: https://api.mongodb.com/python/current/