

Laboratorio de Inteligencia Artificial

Título: Componentes de procesamiento de Lenguaje Natural (NLP) para la plataforma cubana de análisis de datos

Estudiantes:

Esther María Martín Hernández 3ro CC

Ariadna Arboláez Espinosa 3ro CC

CURSO 2022

Tutor: Lic. Alejandro A. Ramón Hernández

**RESUMEN**

Tanto en la vida cotidiana, como en el campo de la ciencia y el desarrollo socio-económico, es importante analizar todas las opiniones y extraer de ellas la mayor información posible que ayude en la toma de decisiones. Entre esta información relevante, la polaridad que expresa y los tópicos a los que se refieren constituyen aspectos importantes a trabajar. En este trabajo se propone diseñar e implementar un paquete de componentes de procesamiento de Lenguaje Natural (NLP),desarrollados en el lenguaje de programación Python a partir de algoritmos existentes desarrollados con anterioridad para las tareas de procesamiento de textos pero que no se integran fácilmente con la plataforma desarrollada actualmente en Cuba. Para esto se aplica un método de agrupamiento basado en la técnica del k-means y la polaridad se calcula con el uso del recurso SpanishSentiWordnett. El procedimiento permite conocer el grado de positividad y negatividad expresado en una información especificada y por otra parte distinguir los distintos tópicos tratados en esta.

**ABSTRACT**

Both in everyday life, and in the field of science and socio-economic development, it is important to analyze all opinions and extract from them as much information as possible that will help in decision-making. Among this relevant information, the polarity it expresses and the topics to which they refer are important aspects to work on. This project proposes to design and implement a package of Natural Language processing (NLP) components, developed in the Python programming language from existing algorithms previously developed for text processing tasks but that are not easily integrated with the platform currently developed in Cuba. For this, a grouping method based on the k-means technique is applied and the polarity is calculated using the SpanishSentiWordnett resource. The procedure allows to know the degree of positivity and negativity expressed in a specified information and on the other hand to distinguish the different topics treated in it.

**TABLA DE CONTENIDOS**

[INTRODUCCIÓN](#_Toc15363) 4

[CAPÍTULO 1: MINERÍA DE TEXTOS Y OPINIÓN 5](#_Toc21603)

[CAPÍTULO 2: PROCEDIMIENTOS Y ALGORITMOS PARA LA DETECCIÓN DE TÓPICOS Y EL CÁLCULO DE LA POLARIDAD 9](#_Toc11205)

- [Diagramas: 1](#_Toc89)2

[CAPÍTULO 3: IMPLEMENTACIÓN COMPUTACIONAL 1](#_Toc10734)6

[CONCLUSIONES 2](#_Toc14973)2

[RECOMENDACIONES 2](#_Toc24032)3

[REFERENCIAS BIBLIOGRÁFICAS 2](#_Toc16243)4

**INTRODUCCIÓN**

Como ya se ha mencionado, conocer numerosas y variadas opiniones es importante en muchos aspectos, ya que, a un usuario individual, le permite conocer la experiencia de otros usuarios con el producto o servicio que planea adquirir, y así tomar una decisión más certera, a una empresa que oferta algún producto o servicio, poder conocer las opiniones de los usuarios ayuda a que puedan los posibles errores y mejorar la calidad. Desde el punto de vista de una organización política es importante conocer el estado de opinión de la población ya sea de la organización de forma general o alguna acción que se haya desarrollado.

En la actualidad, el desarrollo de las tecnologías de la informática y las comunicaciones ha propiciado la aparición en internet de nuevos sitios ricos en opiniones. Entonces, la gran cantidad de opiniones generadas en un corto período de tiempo, dificulta su análisis manual, por lo que se hace cada vez más necesario el desarrollo de sistemas de minería de opinión que sean capaces de procesarlas de forma automática o semi-automática, es decir, técnicas y herramientas que permitan procesar de forma efectiva estas opiniones con vistas a facilitar la toma de decisiones.

En nuestro país se ha logrado poco a poco un desarrollo tecnológico notable y con ello aumentando la necesidad de procesar grandes cantidades de información. Es por eso que como **objetivo general** de la investigación se planteó diseñar e implementar en Python un paquete de componentes de procesamiento de NLP para la plataforma cubana de análisis de datos a partir de algoritmos existentes desarrollados con anterioridad pero que no se integran fácilmente con la plataforma.

Para poder cumplir con el objetivo planteado, será necesario ir cumpliendo **objetivos específicos** a medida se vaya avanzando, como son:

* Estudiar la literatura relacionada con tareas de procesamiento de Lenguaje Natural.
* Investigar dentro de todo lo relacionado a NLP los conceptos, términos y funcionamiento de estos, así como las tecnologías que se usarán para las implementaciones.
* Asimilar los algoritmos existentes para el Análisis de Sentimientos, la Detección de Tópicos, la sumarización y extracción de frases relevantes.
* Diseñar los módulos y clases que permitirán la integración de estos componentes con la plataforma.
* Implementar los diversos componentes y evaluar resultado de casos de ejemplos

**CAPÍTULO 1: MINERÍA DE TEXTOS Y OPINIÓN**

En este capítulo se aborda todo lo relacionado a la minería de textos, de opinión y se profundiza en tareas específicas como el cálculo de la polaridad y la detección de tópicos. Además se analizan los recursos y herramientas que contribuyen a la minería de textos y al procesamiento de Lenguaje Natural, así como los requisitos a tener en cuenta antes de implementar el paquete de componentes de NLP en python.

Para empezar, se debe aclarar que no es lo mismo minería de opiniones que de textos. La minería de texto implica extraer información útil de textos grandes que no sean necesariamente opiniones. Por ejemplo, extraer datos estructurados o patrones utilizando técnicas como lenguaje natural o clusterización para descubrir conexiones y relaciones entre un conjunto de documentos. Mientras que la minería de opinión también se puede utilizar para analizar grandes cantidades de texto, admite un enfoque diferente: evaluar el contenido subjetivo en el material, como las opiniones y los sentimientos hacia determinadas temáticas.

La Minería de Opiniones es una forma automatizada e inteligente para recopilar y estudiar los aspectos positivos y negativos involucrados en un tema determinado. Se utiliza para determinar las opiniones sobre una marca, producto o servicio mediante el análisis cuidadoso del contenido presente en línea o fuera de línea (provienen principalmente de foros, entrevistas, encuestas y mensajes). Por ejemplo, la minería de opiniones se puede emplear para analizar los comentarios de los clientes hacia un producto o una marca a través del análisis en profundidad del contenido disponible socialmente y las opiniones de usuarios no expertos pueden servir como complemento de puntos de vistas, así como toda la información recopilada durante este proceso, puede ayudar tanto a las organizaciones comerciales como a sus clientes a comprender mejor su comportamiento, permite identificar a la empresa problemas específicos en sus productos o servicios, además de tomar decisiones estratégicas teniendo en cuenta la opinión que tienen sus usuarios . Por lo tanto, podemos decir que la minería de opinión se ha convertido en un componente invaluable para tomar decisiones empresariales, políticas y sociales oportunas.ç

La Minería de Texto se refiere al procesamiento automatizado y continuado del texto extraído de documentos, con el objetivo de descubrir los temas y conceptos claves y las relaciones ocultas y las tendencias existentes sin necesidad de conocer las palabras o los términos exactos que los autores utilizaron para expresar dichos conceptos. Para organizar texto automáticamente durante el procesamiento de mineria textual se utiliza varias técnicas como son: tokenización, extracción semántica o lenguaje natural (NLP), identificación de frases/palabras claves (Keywords), detección del tono (Sentiment detection), clasificación (Clustering) etc.. Esta tecnología es usada por ejemplo, por agencias gubernamentales, corporaciones, bibliotecarios, investigadores académicos etc., debido a su capacidad para simplificar la ardua labor manual necesaria revisando documentación literaria para recopilar información relevante acerca de un caso u objetivo específico.

Estrechamente relacionado con las técnicas de la minería de textos, se encuentran las tareas del Procesamiento del Lenguaje Natural (Natural Language Processing: NLP) es un conjunto de procesos informáticos dentro del campo de la Inteligencia Artificial que se encarga principalmente del análisis lingüístico automatizado de textos. Esta técnica particular ayuda a las computadoras a interpretar las oraciones humanas al interpretar imágenes, audios, vídeos y textos ya sea directamente o por medio de interpretación multi-lingüista con diferentes fuentes documentadas así como a entender mejor la correspondencia entre palabras escritas e ideas similares y asociarlas con significados adecuados asignados. Dentro de las tareas del NLP, en esta resultan de interés aquellas que realizan análisis sintáctico (etiquetamiento de los términos en partes de la oración y la delimitación en oraciones), semántico (extracción de relaciones, análisis de sentimiento, reconocimiento y segmentación en tópicos) y de discurso (el resumen automático y la resolución de co-referencias).

Uno de los factores principales para cualquier trabajo a implementar son las herramientas y recursos que contribuyen a la minería de textos y de opinión, menconadas a continuación

Herramientas:

- Procesamiento de lenguaje natural (NLP) para procesar y analizar textos.

- Analizadores de sentimiento para clasificar las opiniones en polaridades (positivo, negativo, neutral).

- Algoritmos de agrupamiento para descubrir patrones en la estructura del texto y detectar temas recurrentes.

- Algoritmos basados en reglas para extraer información relevante y características específicas relacionadas con el tema.

Recursos:

- Bases de datos etiquetadas con etiquetas contextuales, como la lista de palabras frecuentemente usada en la minería de textos (WordNet).

- Grandes conjuntos de datos disponibles, como eventos deportivos, noticias o usuarios sociales que permiten aplicar técnicas avanzadas a una amplia gama de problemáticas.

- Repositorios o fuentes abiertas que facilitan la recopilación masiva de contenido.

En el presente proyecto se hizo uso de estos recursos unido a las tecnologías de python. Se menciona a continuación las herramientas y recursos usadas:

- SentiWordNet es un recurso léxico etiquetado computacionalmente del inglés disponible para descarga libre en http://sentiwordnet.isti.cnr.it/. Está constantemente siendo desarrollado por el Italia Consiglio Nazionale delle Ricerche (CNR) y se basa en la estructura de WordNet, un lexicón en inglés con definiciones de palabras semánticamente organizadas. SentiWordNet proporciona tanto sentimientos positivos como negativos asociados a cada palabra o frase a partir del análisis de corpus textuales usando algoritmos automatizados. Ofrece información sobre los conceptos relacionados con el significado de las palabras, incluidos los sentimentales y valoraciones morales asociadas con esas palabras. Se puede usar para evaluar los sentimientos que subyacen en documentos, para entender mejor el ánimo general de conversaciones en línea, y para ayudar a clasificar contenido según su tendencia emocional general.

- TreeTagger es una herramienta para la etiquetado automático de lenguaje natural. Está diseñada para realizar el análisis morfosintáctico, es decir, extraer información sobre los componentes gramaticales de frases y proporcionar información a bajo nivel acerca de su significado. Es capaz de reconocer palabras como sustantivos, verbos, adjetivos o adverbios. TreeTagger fue desarrollado por Helmut Schmid en Alemania en 1995 como resultado de un proyecto conjunto entre la Universidad de Stuttgart y École Française d'Extrême-Orient. Usandolo se puede construir corpora y bases de datos morfológicas que sean importantes en investigación científica sobre lenguaje natural.

- Flask es un microframework web para aplicaciones desarrolladas con el lenguaje de programación Python. Se caracteriza por ser muy ligero, pero además permite a los usuarios construir robustas aplicaciones web fácilmente. Está diseñado para comenzar rápidamente y escalar sin problemas en complejidad cuando sea necesario.

- Spacy es una biblioteca de procesamiento de lenguaje natural (NLP) para Python. Está construido con la última tecnología de NLP e incluye modelos precargados entrenados para reconocer características como entidades, tópicos, sintaxis y relevancia semántica. Spacy ofrece un conjunto único y completo de herramientas y funcionalidades para procesar grandes cantidades de contenido con fines comerciales y científicos. Algunas de sus principales características son: procesamiento a nivel token, detección de errores gramaticales, análisis sintáctico avanzado, modelización semántica profunda y combinado virtualmente con herramientas visuales.

- Sklearn (Scikit-learn) es una biblioteca que brinda un conjunto de herramientas de aprendizaje automático y análisis estadístico. Estas herramientas permiten a los crear modelos predictivos capaces de realizar tareas como clasificación, regresión, clustering, reducción de dimensionalidad, selección de características y otros. La interfaz única e intuitiva de Sklearn permite a los usuarios acceder a una variedad de algoritmos sin tener que conocer la lógica compleja subyacente.

Las técnicas de agrupamiento son herramientas útiles para el procesamiento automático y verificaión coherencia en NLP, lo que permite encontrar información valiosa desde muchas perspectivas y modelar comportamientos correlacionados analítica y semánticamente. Existen variadas, como Latent Semantic Indexing (LSI), Probabilistic Latent Semantic Analisys (pLSA), Latent Diritchlet Allocation(LDA), k-means, entre otros. Pero de todos solo usaremos este último en el proyecto.

Es importante aclarar que está la posibilidad de usar Docker, que es una herramienta de código abierto que tiene una API en python para realizar tareas como construir contenedores, desplegarlos en diferentes servidores y supervisarlos. Pero hasta el avance actual del proyecto, con Flask y demás recursos se ha resuelto todas las funcionalidades que hasta el momento han sido necesarias. Los resultados alcanzados, no son culminantes, por lo que a medida del avance futuro del proyecto se irán presentando más complejidades y es posible que intoducir entonces el uso de Docker traiga consigo ventajas más significativas que en este momento.

**CAPÍTULO 2: PROCEDIMIENTOS Y ALGORITMOS PARA LA DETECCIÓN DE TÓPICOS Y EL CÁLCULO DE LA POLARIDAD**

En este capítulo se presentan detalladamente los procedimientos y algoritmos utilizados en el desarrollo del proyecto y además se detallan los elementos de diseño.

El término polaridad tiene una serie de usos diferentes, pero en esta disertación se utiliza principalmente para referirse al sentimiento positivo o negativo expresado por una palabra. Sin embargo, existe una distinción importante entre la polaridad a priori de una palabra y su polaridad contextual.

La polaridad a priori de una palabra se refiere a si una palabra típicamente evoca algo positivo o algo negativo cuando se saca de contexto. Por ejemplo, la palabra hermoso tiene una polaridad a priori positiva, y la palabra horrible tiene una polaridad previa negativa. La polaridad contextual de una palabra es la polaridad de la expresión en la que aparece la palabra, considerando el contexto de la oración y el discurso.

Aunque las palabras a menudo tienen la misma polaridad contextual, muchas veces difieren. Una palabra con una polaridad a priori positiva puede tener una polaridad contextual negativa, o viceversa. Por ejemplo, en la oración 1 la palabra "denunciado" tiene negativo y “aprobado” tiene una polaridad positiva mientras que en la oración 2 No odiar tiene positivo y odio tiene negativo según algoritmo basado en SentiWordNet.

Los principales procedimientos para calcular la polaridad en el NLP son:

- Análisis de sentimientos basado en contenido: se basa en un conjunto de reglas para determinar la polaridad o el sentimiento asociado con una palabra dada. Estas reglas pueden tomar la forma de etiquetas asignadas a las palabras e incluir etiquetas como “positivo”, “negativo”, “neutro” e incluso más sutiles como “emotivo”, “skepticismo” y cosas por el estilo.

- Análisis de sentimientos basado en corpus: se basan en gran medida en la teoría lingüística para analizar cómo se utiliza el lenguaje para expresar emociones. Esta técnica involucra el uso del contexto alrededor de una palabra particular o series de frases para determinar si es positiva o negativa.

- Modelos estadísticos: esta técnica aprovecha los datos previamente etiquetados para generar patrones dentro de los textos que correspondan a la polaridad deseada. Estas tecnologías son capaces de aprender sobre relaciones entre elementos gramaticales y expresiones particulares que estén fuertemente correlacionadas con la polaridad/sentimiento deseado.

- Aprendizaje profundo: Este tipo de herramientas usan redes neuronales artificiales profundas para crear modelos predictivos a partir de grandes cantidades de datos históricos adecuadamente etiquetados (normalmente muchas veces más que otros métodos). Debido a su capacidad para codificar relaciones complejas y abstracciones implícitas, suelen ser los mejores al buscar información contextual acerca del significado interno e intencional del texto, lo que hace que ofrezcan resultados consistentemente precisos cuando se tratad e identificación del sentimiento.

Resumiendo, se debe seguir entonces para el cálculo de la polaridad el siguiente algoritmo:

1. Mostrar/localizar el texto a evaluar.

2. Tokenizar el texto para obtener palabras y oraciones.

3. Identificar la polaridad (positiva/negativa) de cada palabra/oración usando un diccionario de polaridad.

4. Si hay palabras no etiquetadas, utilizar técnicas de aprendizaje automático para etiquetarlas con su polaridad.

5. Calcular la polaridad general del texto en base a los resultados anteriores, tomando en cuenta el peso de cada palabra/oración respectivamente según el contexto del texto globalmente analizado.

En fin, esta forma de análisis causa la extracción automatizada del significado semántico del texto que incluye análisis etiquetado y anotación para identificar distintos matices, aspectos y terminologías relacionadas con los sentimientos expresados por los usuarios. Los resultados del análisis son generalmente mostrados en formato numérico para evidenciar la fuerza global y/u orientación política generalmente asociados con las menciones o artículos que se examinan. Esta tecnología permite a los profesionales obtener un ecosistema completo alrededor de los datos recopilados mediante el análisis previo sin necesidad de una labor manual intensiva.

La detección de tópicos consiste en un algoritmo de procesamiento de lenguaje natural (NLP) que identifica y extrae tópicos específicos relevantes del texto. Esta técnica se suele aplicar, sobre todo, para el procesamiento automático del lenguaje en análisis textual. Se utiliza para comprender los documentos y extraer información relevante de ellos además de que pueden detectar conceptos o palabras clave para priorizar el contenido. Se a través de esta técnica se descubre el vocabulario básico para realizar inferencias apropiadas así como para establecer relaciones entre conceptos adyacentes y también poder extraer una idea esencial.

El clustering/agrupamiento es una técnica de exploración de datos utilizada para descubrir grupos o patrones en un conjunto de datos. Existen dos estrategias estándar para la generación de los agrupamientos: el método de particiones y el método jerárquico. Particularmente, el algoritmo de k-means pertenece al método de particiones, en el que cada cluster está representado por un centro (centroide) que es la media de los puntos de datos del cluster. La idea es hacer una clasificación en la que los objetos dentro del mismo cluster sean lo más similares posibles (alta cohesión intraclase), a la vez que los objetos en clústeres diferentes sean lo más disímiles posibles (baja interrelación entre clases).

A modo general, se debe seguir para la detección de tópicos el siguiente algoritmo:

1. Preparar los datos:

- Recopilar toda la información y documentación existente sobre el tema.

- Preprocesar los datos para detectar elementos tales como palabras clave, frases, etc., utilizando herramientas informáticas avanzadas de NLP.

- Filtrar los datos para eliminar vocabulario no relevante y remover palabras menores a un cierto umbral de importancia comúnmente aceptado (stops words).

2. Identificar las palabras clave:

- Utilizar algoritmos basados en aprendizaje profundo para identificar patrones relevantes en los conjuntos de información previstos hasta ahora.

- Encontrar nuevas relaciones semánticas entre las palabras relevantes encontradas desde el paso 1 mediante técnicas.

3. Categorizacion del tema:

- Utilizar clusterización basada en reglas o mediante el uso de algoritmos (por ejemplo KMeans) para crear grupos de datos dentro del contenido que presenten relaciones significativas entre sí.

- Usar la minería de texto para encontrar y relacionar elementos que describan los tópicos hallados previamente.

Anteriormente se ha mencionado en varias ocasiones el algoritmo KMeans como ejemplo de técnica de agrupamiento a aplicar en la detección de tópicos dentro del NLP, su funcionamiento explicado de forma más específica es el siguiente:

1. Seleccionar un número k de grupos y definir k centróides aleatoriamente en el espacio de características.

2. Asignar cada observación a su centroide más cercano, basado en la distancia euclidiana.

3. Calcular la lógica media para cada nuevo cluster (media de todos los puntos de datos dentro del cluster).

4. Mover el centroide asociado al valor promedio calculado en el paso 3.

5. Repetir los pasos 2-4 hasta que no se produzcan cambios significativos o se alcance un límite predefinido.

En fin, esta técnica de NLP es el proceso para identificar y agrupar opiniones, sentimientos o contenido basado en palabras clave. Esta información proporciona importante contexto sobre cómo un producto o servicio se ve desde una perspectiva externa. La detección de temas puede ser utilizada para ayudar a las empresas a entender mejor la percepción del consumidor sobre su marca. El análisis de temas también puede ayudar a las organizaciones a realizar investigaciones sobre tendencias, monitorear los puntos calientes en potencial y medir cómo un acontecimiento o campaña se está desempeñando.

A continuación se presentan los diferentes diagramas correspondientes a cada microservicio.

Diagrama de clases (Cálculo de la polaridad):

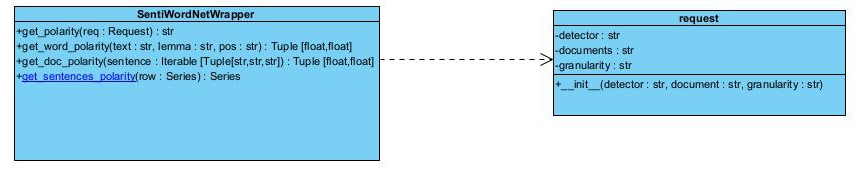


Diagrama de casos de uso(Cálculo de la polaridad):

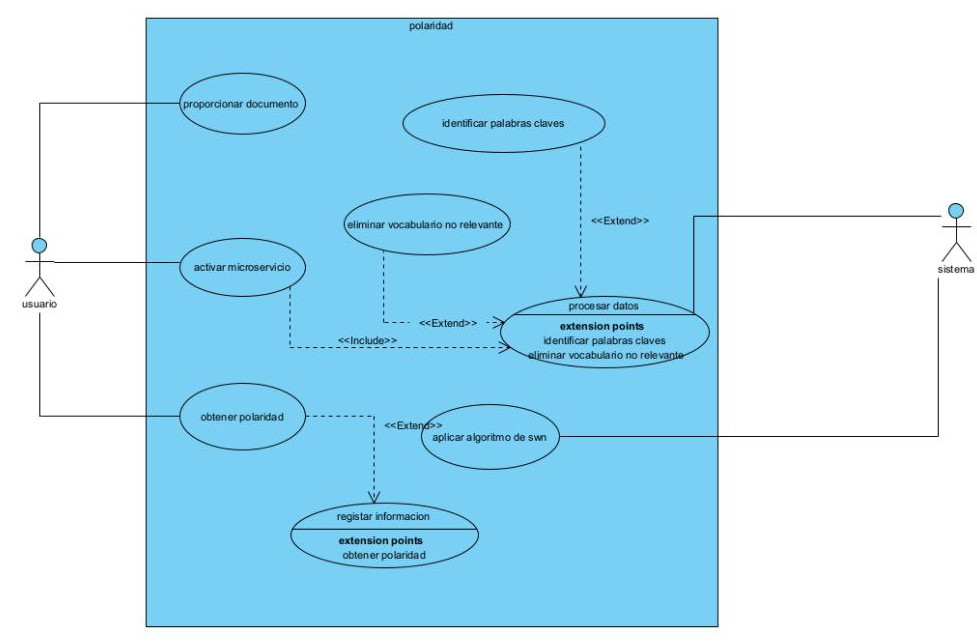


Diagrama de actividades(Cálculo de la polaridad):

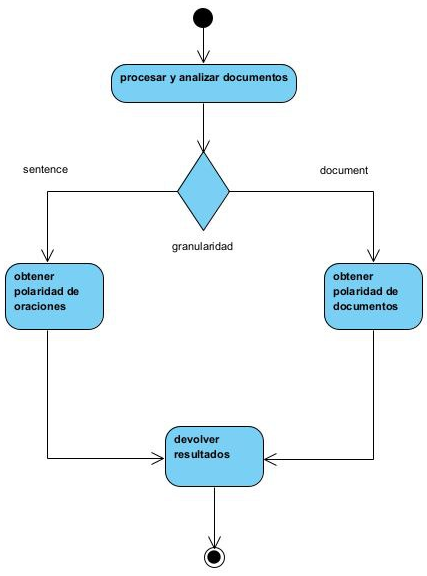


Diagrama de clases (Detección de tópicos):

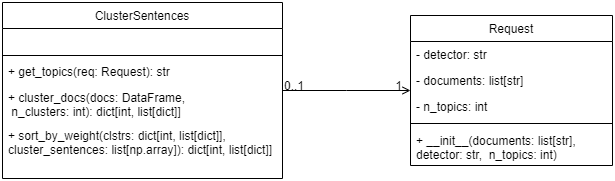


Diagrama de casos de uso (Detección de tópicos):

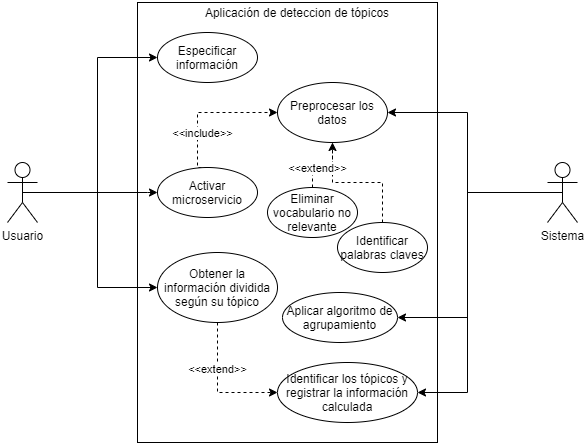
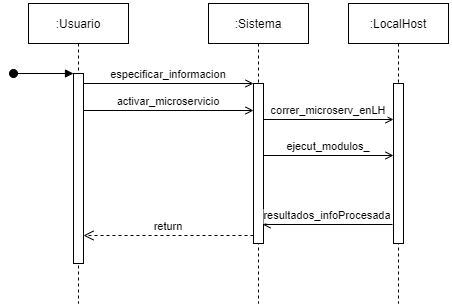


Diagrama de actividades (Detección de tópicos):



Diagrama de flujo(Ambos juntos):



**CAPÍTULO 3: IMPLEMENTACIÓN COMPUTACIONAL**

En este capítulo se presentan una explicación más detallada del código, enfatizando en la estructura de los módulos programados y el comportamiento de sus funciones.

Los módulos están organizados en diferentes carpetas acorde a sus funciónes en la aplicación, a continuación se menciona de forma ordenada según estas carpetas la descripción de su contenido.

CÁLCULO DE LA POLARIDAD:

* **utils**

El archivo load\_resourses contiene dos funciones: load\_swn y load\_stopwords.

La función load\_swn carga el archivo de SentiWordNet (SWN) y lo procesa para obtener un diccionario de palabras y sus puntuaciones de polaridad. El archivo de SWN es un archivo de texto que contiene una lista de palabras y sus puntuaciones de polaridad para diferentes partes del discurso (sustantivos, verbos, adjetivos y adverbios). Cada línea del archivo contiene una palabra y sus puntuaciones separadas por tabuladores. La función procesa cada línea del archivo, separa la palabra y las puntuaciones y luego las agrega al diccionario de SWN.

La función load\_stopwords carga un archivo de stopwords y lo procesa para obtener una lista de stopwords. Una stopwords es una palabra comúnmente utilizada que no aporta significado a un texto y puede ser ignorada en algunos casos. El archivo de stopwords es un archivo de texto que contiene una lista de palabras separadas por saltos de línea. La función procesa cada línea del archivo y agrega la palabra a la lista de stopwords.

El archivo preprocess contiene una función preprocess que se utiliza para procesar un conjunto de documentos de texto. La función realiza varias tareas de preprocesamiento en cada documento:

1. Carga el modelo de procesamiento de lenguaje de Spacy y agrega el componente "sentencizer" para dividir el documento en oraciones.
2. Carga una lista de stopwords utilizando la función load\_stopwords.
3. Itera sobre cada documento utilizando el componente "sentencizer" de Spacy para dividir el documento en oraciones.
4. Para cada oración, itera sobre cada token (palabra) y verifica si cumple con las siguientes condiciones:

- No es una stopword.

- No está en mayúsculas.

- Su parte del discurso (POS) está en un conjunto de POS válidos (VALID\_POS).

1. Si el token cumple con las condiciones anteriores, se guardan el texto del token, su lema y su POS en las listas texts, lemmas y pos, respectivamente.
2. Al finalizar la iteración sobre las oraciones del documento, se guardan las listas texts, lemmas y pos en un diccionario de documentos. Si se habilitó la opción split\_sentences, también se guarda una lista de oraciones en el diccionario.
3. Al finalizar la iteración sobre todos los documentos, se convierte el diccionario en un DataFrame de Pandas y se devuelve.

La función preprocess limpia y prepara un conjunto de documentos para su análisis. Al eliminar las stopwords y las palabras en mayúsculas, se reducen las palabras irrelevantes y se facilita el procesamiento de los documentos. Al lematizar las palabras y conservar solo ciertas partes del discurso, se simplifica la representación de los documentos y se hace más fácil comparar palabras similares.

* **services**

La clase SentiWordNetWrapper, se encarga de obtener la polaridad de un conjunto de documentos de texto utilizando el archivo de SentiWordNet (SWN).

La clase tiene un método llamado get\_polarity que recibe como entrada un objeto de la clase Request y devuelve la polaridad de cada documento en formato JSON. El objeto Request contiene los documentos a procesar y la granularidad deseada (oración o documento completo). La función get\_polarity utiliza la función preprocess para limpiar y preparar los documentos y luego utiliza una de las dos funciones siguientes para obtener la polaridad: get\_doc\_polarity o get\_sentences\_polarity.

La función get\_doc\_polarity recibe una oración y devuelve la polaridad del documento completo. La función itera sobre cada token de la oración, obtiene su polaridad utilizando la función get\_word\_polarity y acumula las puntuaciones positivas y negativas. Luego, devuelve la polaridad promedio dividiendo la puntuación positiva y negativa acumulada por la longitud de la oración.

La función get\_sentences\_polarity recibe una fila del DataFrame de Pandas que contiene los documentos preprocesados y devuelve la polaridad de cada oración del documento y la polaridad del documento completo. La función itera sobre cada oración del documento, obtiene su polaridad utilizando la función get\_doc\_polarity y acumula las puntuaciones positivas y negativas. Luego, devuelve la polaridad promedio de cada oración y la polaridad promedio del documento dividiendo las puntuaciones positivas y negativas acumuladas por la cantidad de oraciones en el documento.

La función get\_word\_polarity recibe un token y su lema y parte del discurso y devuelve la polaridad del token. La función busca la polaridad del token en el archivo de SWN y, si no se encuentra, busca la polaridad del lema. Si tampoco se encuentra la polaridad del lema, devuelve puntuaciones de polaridad cero.

La clase SentiWordNetWrapper obtiene la polaridad de un conjunto de documentos de texto utilizando el archivo de SWN. Al dividir los documentos en oraciones o analizarlos completos, se pueden obtener diferentes niveles de granularidad en los resultados.

DETECCIÓN DE TÓPICOS:

* **request**

La clase *request* del módulo *request.py* se utiliza para mapear los datos de entrada en una clase, donde los datos serían el o los documentos, el nombre de la técnica de agrupamiento a utilizar que en este caso está fijado KMEANS y el número de tópicos para este, que en este caso para realizar las pruebas hasta el momento se fijó 5.

* **resouces**

Contiene el archivo de texto *stop\_words\_es.txt* con todas aquellas palabras de las cuales se puede prescindir al analizar e interpretar una información

* **services**

La clase *ClusterSentences* dentro del módulo *cluster\_sentences.py* tiene dos métodos principales: *get\_topics* y *cluster\_docs*. El método *get\_topics* toma como entrada un objeto Request y devuelve una cadena en formato JSON que contiene una lista de listas de oraciones agrupadas por tópicos. Primero, se aplican algunos preprocesamientos a la lista de documentos utilizando la función *preprocess\_clusters*. Luego, se llama al método *cluster\_docs* para agrupar las oraciones en clusters y se devuelve una lista de oraciones de cada cluster en formato JSON. El método *cluster\_docs* toma como entrada un dataframe de oraciones preprocesadas y un número de clusters opcional (por defecto 10). Utiliza el algoritmo de agrupamiento KMeans de scikit-learn para agrupar las oraciones en clusters y etiquetar cada oración con una etiqueta de cluster. Luego, se crea un diccionario que asigna a cada etiqueta de cluster una lista de oraciones que pertenecen a ese cluster. Finalmente, se llama al método *sort\_by\_weight* para organizar las oraciones de cada cluster en orden ascendente según su distancia al centro del cluster y se devuelve el diccionario de clusters ordenado. El método *sort\_by\_weight* toma como entrada un diccionario de clusters y los centros de cada cluster. Calcula la distancia entre cada oración y el centro del cluster al que pertenece y la almacena en una nueva clave 'distance' en el diccionario de oración. Luego, se ordena cada cluster por la distancia de sus oraciones al centro del cluster y se devuelve el diccionario de clusters ordenado.

* **utils**

En el módulo *preprocess.py* hay definido una única función *preprocess\_clusters* que toma como entrada una lista de documentos y devuelve un dataframe con dos columnas: 'sent' y 'vector'. La función utiliza un modelo de procesamiento de lenguaje natural de spaCy para aplicar un pipe a la lista de documentos. Esto significa que el modelo procesará cada documento en la lista y realizará algunas operaciones sobre ellos. En este caso, se deshabilitan algunos componentes del modelo, como el analizador sintáctico y el analizador de entidades nombradas. Luego, se itera sobre cada oración del documento y se crea un diccionario llamado 'sent' que almacena la oración como texto. A continuación, se itera sobre cada token (palabra) de la oración y se comprueba si el token tiene un vector asociado, si su parte del discurso es un sustantivo, un pronombre o un verbo y si no está en la lista de stopwords. Si todas estas condiciones se cumplen, se añade el vector del token al diccionario 'sent'. Finalmente, si el diccionario 'sent' contiene una clave 'vector', se normaliza y se añade a la lista 'data\_vectors'. Al final de la función, se crea un dataframe a partir de la lista 'data\_vectors' y se devuelve.

En el módulo *resources.py* se carga un modelo de spaCy y agrega un componente personalizado llamado 'sentencizer' al pipeline del modelo. El modelo se carga utilizando la función load() de spaCy y se especifica la ruta del modelo como una variable de entorno llamada 'SPACY\_MODEL'. Luego, se define una función interna llamada *\_\_load\_stopwords()* que lee un archivo de stopwords y devuelve una lista de stopwords. La ruta del archivo se especifica como una variable de entorno llamada 'STOPWORDS\_PATH'. Finalmente, se llama a la función *\_\_load\_stopwords()* y se almacena el resultado en la variable 'stopwords'.

* **topic**

El módulo *app.py* es un script de Python que implementa un servicio web (también conocido como microservicio) utilizando el módulo Flask, este es un marco (framework) de Python para desarrollar aplicaciones web de manera sencilla. El script comienza importando algunos módulos necesarios: Flask, request, Response, load\_dotenv, Request, y ClusterSentences. *Flask* es el marco Flask mencionado anteriormente. *request* es un objeto global de Flask que contiene información sobre la solicitud HTTP actual. *Response* es una clase de Flask que se utiliza para construir una respuesta HTTP. *load\_dotenv* es una función de un módulo llamado dotenv que se utiliza para cargar variables de entorno desde un archivo .env. *Request* y *ClusterSentences* son clases personalizadas que se han definido en otros archivos, mencionados anteriormente. A continuación, se crea una instancia de Flask con app = Flask(name). Luego, se define un diccionario llamado TOPIC\_DETECTOR que asigna a la clave 'KMEANS' la clase *ClusterSentences*. Luego se define una función de enrutamiento (route) utilizando el decorador *@app.route*. Una función de enrutamiento es una función de Python que se ejecuta cuando se accede a una determinada dirección URL. En este caso, la función se ejecutará cuando se envíe una solicitud POST a la dirección /get\_topics. La función comienza creando una instancia de la clase *Request* con req = Request(request.json). Esta línea toma el cuerpo de la solicitud POST (que se almacena en request.json) y lo pasa a la función *Request* como argumentos de palabra clave (kwargs). Luego, se crea una instancia de la clase *ClusterSentences* con TOPIC\_DETECTOR[req.detector](). Esta línea utiliza la clave req.detector del diccionario TOPIC\_DETECTOR para obtener la clase correcta y luego crea una instancia de ella. A continuación, se llama al método get\_topics de la instancia de *ClusterSentences* con result = TOPIC\_DETECTOR [req.detector]().get\_topics(req). Este método procesa los documentos enviados en la solicitud POST y devuelve una lista de tópicos. Finalmente, se construye y devuelve una respuesta HTTP con return Response(result, mimetype='application/json').

**CONCLUSIONES**

Se diseñaron procedimientos para el cálculo de la polaridad y detección de tópicos en textos, desarrollando un paquete de componentes en Python que consta con etapas que constituyen cada una, un módulo independiente lo que permite que pueda modificarse cada etapa sin afectar las restantes. Para ello, se han estudiado diferentes técnicas y algoritmos existentes para el análisis de sentimientos, detección de tópicos, sumarización y extracción de frases relevantes, entre totros, y se han diseñado e implementado dichos módulosa partir de algoritmos ya existentes para lograr integrar estos componentes con la plataforma. El paquete incluye funcionalidades para calcular la polaridad de un conjunto de documentos de texto a partir de SentiWordNet y para detectar tópicos en un conjunto de documentos mediante el uso de la técnica de agrupamiento k-means. Se logró cumplir entonces el objetivo planteado y se evaluó los resultados de los componentes implementados mediante casos de ejemplo.

**RECOMENDACIONES**

Implementar más de una técnica posible de agrupamiento, para poder escoger en diferentes casos.

Implementar la categorización de tópicos, es decir, además de agruparlos, registrar una idea esencial de lo que abarca.

Implementar y diseñar usando html y css, la aplicación ilustrativa integrando los módulos ya programados, de forma que la introducción de los datos a analizar sea a través de un buscador de archivos de texto dentro de esta interfaz.

**REFERENCIAS BIBLIOGRÁFICAS**

Bo Pang, L. L. (2008). A Survey of Opinion Mining and Sentiment Analysis . *Journal of Fundations and Trends in Information Retrieval* .

Deepak Singh Tomar, P. S. (n.d.). A Text Polarity Analysis Using Sentiwordnet .

Dwyer, G. (n.d.). *Flask by Example.*

G.A. de Almeida, G. d. (2013). Sentiment Analysis: A Review. *Journal of Information Science and Engineering* .

Grinberg, M. (n.d.). *Flask Web Development with Python".*

Hernández, A. A. (n.d.). Detección de polaridad por tópicos para textos cortos en español.

J. Ben Schafer, J. L. (2007). Topic Modeling: An Introduction. *Journal of the American Society for Information Science and Technology*.

Lamere, P. (2014). Topic Modeling: A Basic Introduction. *revista Library Technology Reports* .

Lee, B. (n.d.). *Sentiment Analysis: Mining Opinions, Sentiment, and Emotions.*

mr.Thavasi, d. (n.d.). Enhanced Sentiment Analysis and Polarity Classification Using .