

Facultad de Ingeniería Informática

**Predicción de categorías de requisitos a partir del análisis de opiniones de usuarios**

**Entregable 4 Analista-Arquitecto-Diseñador**

Autor

Cristian Sori Aguilera IF - 43

Tutor

Dra.C Anaisa Hernández González

La Habana, Cuba

2022

# **Resumen**

A medida que las interacciones entre los humanos y las máquinas se hacen más frecuentes, cualquier pequeño paso hacia adelante en la tarea de hacerlas más fluidas será también un paso adelante en la construcción de ese nuevo mundo de la sociedad de la información. Evidentemente, la mejor manera de ganar fluidez y eficacia en estas interacciones es conseguir que éstas sean lo más naturales posibles, y ¿qué sistema de comunicación resulta más natural para los seres humanos que el lenguaje?.

• \_Específicamente, debemos desarrollar sistemas que lean y entiendan el texto de la forma en que lo hace una persona, formando una representación de el mundo del texto, con los agentes, objetos, escenarios y la relaciones, metas, deseos y creencias de los agentes, y todo lo demás que los humanos crean para comprender un fragmento de texto. Hasta que podamos hacer eso, todo nuestro progreso consiste en mejorar la capacidad de nuestros sistemas para hacer coincidir patrones.

En este trabajo nos estaremos enfocando en diseñar un sistema capaz de identificar entidades nombradas en un texto que contenga requerimientos de software.

# Diagrama de Casos de Uso del Sistema:



# Requisitos Funcionales del Sistema

1. 1- El analista pueda analizar cualquier texto en los formatos permitidos

2. 2- El analista pueda obtener un historial de textos desde la base de datos

3. 3- El analista pueda entrenar el modelo de acuerdo al tipo de texto a analizar

# Casos de Prueba:

|  |  |
| --- | --- |
| CASO DE PRUEBA | |
| Caso de Uso | Analizar Texto |
| Desarrollador | Cristian Sori |
| Probador | Cristian Sori |
| Fecha | 24/06/22 |
| Versión | 1.0.0 |
| Objetivo de la Prueba | Analizar un texto en el sistema |
| Descripción de la prueba | Prueba de caja negra diseñada para verificar que la funcionalidad está implementada de acorde a las especificaciones del cliente |
| Condiciones |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Combinaciones de valores de entrada | | | | Resultados Esperados | Resultados Reales |
| CP | escenario | variable de entrada | Valor de variable |
| CP1 | Texto vacio | textContent | null | No se puede analizar el texto | No se puede analizar el texto |
| CP2 | Texto no corresponde a requerimientos funcionales | textContent | Cualquier teto que no contenga requerimientos funcionales | El texto no contiene requerimientos de software | Error |
| CP3 | Texto Correcto | textContent | Cualquier texto que contenga RF | Obtener Entidades | Obtener Entidades |
| Observaciones: | | | | | |
|  | | | | | |

# Arquitectura Candidata:

## En este capítulo se realiza un estudio y selección de la tecnología para la implementación del sistema. Se plantea la arquitectura candidata, los patrones arquitectónicos y los estilos de arquitectura a los que pertenecen dichos patrones . Además, se concebirá el diseño y ejecución de pruebas de software que validen las funcionalidades del sistema que han sido implementadas .

## Tecnologías utilizadas :

**Python**

Python es un lenguaje de programación potente y fácil de aprender. Tiene estructuras de datos de alto nivel eficientes y un simple pero efectivo sistema de programación orientado a objetos. La elegante sintaxis de Python y su tipado dinámico, junto a su naturaleza interpretada lo convierten en un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en muchas áreas, para la mayoría de plataformas.

El intérprete de Python y la extensa librería estándar se encuentran disponibles libremente en código fuente y de forma binaria para la mayoría de las plataformas desde la Web de Python, https://www.python.org/, y se pueden distribuir libremente. El mismo sitio también contiene distribuciones y referencias a muchos módulos libres de Python de terceros, programas, herramientas y documentación adicional.

El intérprete de Python es fácilmente extensible con funciones y tipos de datos implementados en C o C++ (u otros lenguajes que permitan ser llamados desde C). Python también es apropiado como un lenguaje para extender aplicaciones modificables.12

Ventajas de Python para su uso en Inteligencia Artificial.

**Gran variedad de Bibliotecas.**Se puede entender una biblioteca como un módulo o un grupo de módulos publicados por diferentes fuentes como PyPi que incluyen un fragmento de código preescrito que permite a los usuarios alcanzar alguna funcionalidad o realizar diferentes acciones

Las bibliotecas de Python proporcionan elementos de nivel básico para que los desarrolladores no tengan que codificarlos desde el principio cada vez. De esta manera, reducen su tiempo de aprendizaje de las complejidades del *stack* y así pueden comenzar con el desarrollo de Inteligencia Artificial (IA) y pasar a la creación de Algoritmos y programas de IA.

1. Lenguaje flexible

**Flexibilidad.**Para el *Machine Learning* permite elegir OOPS o programación basada en secuencias de comandos. Además, tampoco es necesario volver a compilar el código fuente, puesto que los desarrolladores pueden implementar cambios y ver de forma inmediata los resultados.

2. **Legibilidad**

Python es un lenguaje muy fácil de leer, por lo que cada desarrollador que emplee esta herramienta puede entender el código de sus compañeros y cambiarlo, copiarlo o compartirlo. Igualmente hay herramientas como IPython, que es un shell interactivo que facilita funciones adicionales como pruebas, depuración, etc. optimizando el proceso de trabajo.

4. **Visualización**

Anteriormente se ha mencionado que Python ofrece una variedad de bibliotecas, y, entre ellas, existen excelentes herramientas de visualización**.**No obstante, para los desarrolladores de Inteligencia Artificial y Machine Learning es de vital importancia representar datos de forma legible para los humanos, para determinar patrones y dar sentido a todas las variables y factores.

También existen APIs que facilitan el proceso de visualización al permitir delinear informes claros de datos.

**SPACY.**

SPACY es una biblioteca gratuita de Python de código abierto que proporciona capacidades avanzadas para llevar a cabo el procesamiento del lenguaje natural (PNL) en grandes volúmenes de texto a alta velocidad. Le ayuda a crear modelos y aplicaciones de producción que pueden respaldar el análisis de documentos, las capacidades de chatbot y todas las demás formas de análisis de texto.

Los dos autores principales de spaCy, Matthew Honnibal e Ines Montani, lanzaron el proyecto en 2015. El marco spaCy, junto con un conjunto cada vez mayor de plug-ins y otras integraciones, proporciona características para una amplia gama de tareas de lenguaje natural. Se ha convertido en una de las bibliotecas de lenguaje natural más utilizadas en Python para casos de uso de la industria, y tiene una comunidad bastante grande, y con eso, mucho apoyo para la comercialización de la investigación avanza a medida que esta área continúa evolucionando rápidamente.

Su última versión(SPACY 3.0) aporta muchas mejoras para ayudar a construir, configurar y mantener modelos de PNL, incluyendo:

* • \_Tuberías basadas en transformadores recién entrenadas y recicladas que elevan significativamente las puntuaciones de precisión
* • \_Capacidades de configuración adicionales para crear su flujo de trabajo de entrenamiento y ajustar sus carreras de entrenamiento
* • \_Integración más fácil con otras herramientas como Streamlit, FastAPI o Ray para crear flujos de trabajo
* • \_Capacidades paralelas/distribuidas con Ray para ciclos de entrenamiento

Estas características se combinan para hacer que spaCy sea mejor que nunca en el procesamiento de grandes volúmenes de texto y configuraciones de ajuste para que coincidan con casos de uso específicos de una manera que proporcione una mejor precisión.

**Spacy vs NLTK.**

En este trabajo, se ha decidido usar Spacy en lugar de NLTK, a continuación se detallan las razones por las cuáles se llegó a esta conclusión .

**Velocidad**

NLTK tiene todas y cada una de las herramientas y técnicas que uno necesitaría para realizar la PNL. Es una enorme biblioteca con una amplia gama de envoltorios, tallos, analizadores y demás y seguramente todos los profesionales de PNL seguramente se asombrarán al ver tantas opciones. Pero es aconsejable elegir lo mejor de lo que está disponible. Además, NLTK es muy lento y, para ser justos, no se construyó teniendo en cuenta la velocidad, ya que se hizo por razones pedagógicas.

Spacy Tiene una precisión y una velocidad alucinantes para cada tarea de PLN específica, es hasta 400 veces más rápido que NLTK y proporciona algoritmos SOTA empaquetados en el

modelo en\_core\_web\_trf con la libertad de personalizar la canalización. La creciente popularidad de spaCy demuestra su destreza.

**Documentación Increíble**

La única luz en la oscuridad cuando uno comienza con una nueva biblioteca e incluso más tarde es su documentación. A pesar de que tenemos tutoriales en línea, para crear algo desde 0, debemos que profundizar en la documentación.Por lo tanto, es de suma importancia comparar las documentaciones de spaCy y NLTK. La documentación de spaCy es clara, nítida, sucinta, bien organizada y encubierta. Consiste en explicaciones detalladas, valiosos consejos y trucos, útiles módulos de código e ilustraciones y tutoriales en vídeo para empezar. Los fabricantes han aprobado varias preguntas frecuentes cuando y donde sea necesario, sin dejar ninguna duda en la mente de los lectores. La interfaz de usuario colorida e interactiva se suma a una gran experiencia de usuario.

Por el contrario, NLTK tiene una documentación inconsistente, difícil de seguir y bastante incompleta para ciertos paquetes que tarda un principiante en entender. El libro escrito por los creadores sobre el uso de NLTK para la PNL es muy detallado y ayuda a un novato a entender los conceptos en profundidad. Aun así, carece de la orientación que ayude a alguien a implementar la PNL. Esto hace que NLTK sea difícil de aprender y usar, ya que no hay una canalización clara de PNL mencionada en ninguna parte de la documentación. Sin duda, está cargado de fragmentos de código y ejemplos, pero ese no es el caso de todos los paquetes. Además, no hay ninguna documentación sobre la estructura y organización de cada cuerpo en la gran base de datos en datos de NLTK que pueda restringir al profesional de la PNL a aprovechar todo el potencial del conjunto de datos. Además, se da una introducción de cada API de canalización al final de la página después de todas las diferentes API algorítmicas para realizar esa tarea de PNL en lugar del principio. Digamos, por ejemplo, que el módulo de etiquetas se introduce después de explicar las API para Brill's Tagger, CRF Tagger, HMM Tagger, HunPos Tagger, Perceptron Tagger, entre otros. Hay un margen de mejora en la documentación para NLTK y la razón más plausible de la falta de uniformidad son las continuas adiciones a la biblioteca por parte de varios colaboradores.

**Internacionalización**

Aunque en etapas nacientes spaCy solo admitía el inglés(v1), pero con la última versión(v3) admite 17 idiomas para el modelado estadístico y más de 64 idiomas para la "tokenización alfa". Por otro lado, aunque NLTK admite más lenguajes en comparación con spaCy, no está ampliando el soporte a todos los módulos y algoritmos. Solo algunos paquetes proporcionan soporte multilingüe y todos ellos no son compatibles con todos y cada uno de los idiomas que los demás admiten. Por ejemplo, el módulo nltk.parse.stanford para el análisis, la tokenización y el etiquetado, el módulo nltk.stem.snowball para el stemming proporciona stemmers para alrededor de 15 idiomas, incluidos árabe, inglés, rumano, sueco, etc., nltk.test.unit ofrece métodos para evaluar el etiquetado pero el resto de los algoritmos basados en chunking,

classifying, clustering, parsing, sentiment analyzing, stemming, tagging, transformation based learning, unit testing, tralating y chatbot solo admiten el inglés.

Por lo tanto, la internacionalización no puede atribuirse a todo el conjunto de bibliotecas de procesamiento de texto proporcionadas por NLTK. Sin embargo, spaCy ofrece canalizaciones entrenadas y la libertad de personalizarlo para todos los idiomas que admite, lo que garantiza una internacionalización fiable y consistente.

Los datos de NLTK consisten en varios corpus, libros, paquetes populares, paquetes para ejecutar pruebas, paquetes de datos de terceros y todos los módulos disponibles en NLTK. Aunque es una interfaz de usuario simple, uno puede encontrarla abrumadora cuando la mira y termina ejecutando nltk.download('all'), que significa descargar todos los datos NLTK. Esto lleva a tener alrededor de 6 GB de datos NLTK en la memoria cuando en realidad uno estaría trabajando con solo un corpus a la vez y solo algunos de los paquetes.. La peor parte es que no se puede realizar ninguna tarea de PNL sin descargar los datos NLTK.

spaCy viene al rescate. Proporciona a todos los componentes en el módulo principal. Se podría optar por descargar un modelo de lenguaje de un modelo pequeño(sm), mediano (md), grande (lg) o basado en transformadores (trf) añadido recientemente. Incluso se puede comenzar con un modelo de lenguaje en blanco y comenzar con el entrenamiento de toda la canalización por su cuenta.[14]

**Django.**

¿Qué es Django?

Django es un marco de aplicaciones web gratuito y de código abierto escrito en Python. Un marco no es más que una colección de módulos que facilitan el desarrollo. Se agrupan y le permiten crear aplicaciones o sitios web a partir de una fuente existente, en lugar de desde cero. Así es como los sitios web, incluso los simples diseñados por una sola persona, todavía pueden incluir funcionalidades avanzadas como soporte de autenticación, paneles de gestión y administración, formularios de contacto, cuadros de comentarios, soporte de carga de archivos y más.. El sitio oficial del proyecto describe Django como "un marco web Python de alto nivel que fomenta un desarrollo rápido y un diseño limpio y pragmático. Construido por desarrolladores experimentados, se encarga de gran parte de la molestia del desarrollo web, por lo que puede centrarse en escribir su aplicación sin necesidad de reinventar la rueda. Es gratuito y de código abierto".

# Django se creó pensando en los desarrolladores front-end. "El lenguaje de plantillas de Django está diseñado para sentirse cómodo y fácil de aprender para aquellos acostumbrados a trabajar con HTML, como diseñadores y desarrolladores front-end. Pero también es flexible y altamente extensible, lo que permite a los desarrolladores aumentar el lenguaje de las plantillas según sea necesario".

# **Estilos y Patrones de Arquitectura**

Un patrón arquitectónico describe la relación entre un conjunto de componentes para

resolver un problema arquitectónico común en un contexto particular, de tal manera que

se puedan satisfacer la mayor cantidad de funcionalidades del sistema.

**Estilo Llamada-Retorno**

Este estilo no es más que la descomposición jerárquica en subrutinas que solucionan

una tarea o función definida. Los datos son pasados como parámetros y el manejador

principal proporciona un ciclo de control sobre las subrutinas. Permite al diseñador

del software construir una estructura de programa relativamente fácil de modificar y

ajustar a escala.

**Patrón n-Capas**

El patrón de software más común es el patrón arquitectónico en capas. Este tipo d

arquitectura son patrones de niveles donde los componentes están organizados en capas horizontales. Este es el método tradicional para diseñar la mayoría de los programas informáticos y está destinado a ser auto-independiente. Esto significa que todos los componentes están interconectados, pero no dependen unos de otros. En este patrón se puede ver la alta reutilización de los paquetes de la plataforma y se considera este enfoque como la mejor forma para representarlo.

**Estilo de Flujo de Datos**

Esta arquitectura se aplica cuando los datos de entrada se habrán de transformar en datos de salida mediante una serie de componentes para el cálculo o la manipulación.

**Patrón Tuberías y Filtros**

Este consta de un conjunto de componentes denominados «filtros», conectados entre sí por «tuberías», que transmiten los datos desde un componente al siguiente. Cada filtro trabaja de manera independiente de los componentes que se encuentren situados antes o después de él.

**Estilo Centrado en Datos**

El Software cliente tiene acceso a un almacén central. En algunos casos este es pasivo, es decir, el Software cliente accede a los datos independiente de cualquier cambio hecho a los datos o a las acciones de otros SW cliente.

**Patrón Repositorio**

El patrón repositorio consiste en separar la lógica que recupera los datos y los asigna a un modelo de entidad de la lógica de negocios que actúa sobre el modelo, esto permite que la lógica de negocios sea independiente del tipo de dato que comprende la capa de origen de

datos, en pocas palabras un repositorio medio entre el dominio y las capas de mapeo de datos, actuando como una colección de objetos de dominio en memoria.31En este trabajo , se cuenta con una base de datos que actúa como almacén de los textos analizados , y permite a cualquier software cliente intercambiar información con esta.

**Mecanismo de acceso a Datos**

Un mecanismo de acceso a datos es el que se encarga de garantizar la persistencia y el acceso a base de datos. En este proyecto se usa como mecanismo a el Framework Django, haciendo énfasis en su mecanismo ORM el cual permite interactuar con la base de datos sin necesidad de usar comandos SQL. El proyecto incluye un módulo llamado mocels.py, el cual define la estructura de las tablas de la base de datos que deseas crear. Para traducir los Objetos de Python en tablas de base de datos, entra en juego el ORM. El ORM es responsable de la comunicación con su base de datos, lo que incluye la traducción de sus modelos definidos a la estructura de base de datos correcta y la ejecución de cualquier operación de base de datos.33

**Conflictos de Implementación**

**Gestión de la configuración**

La gestión de la configuración evita que se realicen cambios importantes o modificaciones menores sin documentar.

Estos cambios pueden afectar el desempeño, provocar inconsistencias o derivar en el incumplimiento normativo, lo cual perjudica las operaciones y la seguridad de la empresa. Cuando se implementan cambios sin documentar en muchos sistemas y aplicaciones, se genera inestabilidad y tiempo de inactividad.

El proceso manual para identificar los sistemas que requieren atención, determinar las medidas de corrección, establecer la prioridad de las acciones y validar su finalización es demasiado complejo como para llevarlo a cabo en entornos grandes. Pero si los equipos de administración

de sistemas y desarrollo de software no cuentan con la documentación, las medidas de mantenimiento y el proceso de control de cambios necesarios, es posible que no estén al tanto de qué elementos hay en un servidor ni de qué software se ha actualizado.34

En este proyecto, la configuración se gestiona utilizando GitHub , el cual garantiza la resolución de conflictos de implementación a la hora del desarrollo del proyecto , permitiendo que cada desarrollador trabaje de forma independiente y además ,GitHub se encarga del versionado del código dejando trazas sobre los cambios realizados.

**Reutilización**

Para permitir la reutilización de código y evitar redundancias , en este proyecto se ha utlizado el gestor de paquetes de Python <pip> que se puede interpretar como Pip Instalador de Paquetes o Pip Instalador Python. Este es un sistema de gestión de paquetes sencillo utilizado para la instalación y administración de paquetes que pueden ser encontrados en el Python Package Index (PyPI). Python 2.7.9 y posteriores (en la serie Python2), Python 3.4 y posteriores incluyen este gestor (pip3 para Python3) por defecto.

# **Estándar de Codificación**

Los estilos de código o estándares de codificación son un conjunto de reglas o normas usadas para escribir código y que incluye una gran cantidad de aspectos dentro del proceso de codificación. Es un estilo de programación homogéneo que permite que los desarrolladores del sistema puedan interpretar de manera eficiente la escritura del código; asegurando que todos trabajen de forma coordinada y en un vocabulario común.

## **Nomenclatura**

### Paquetes

Por defecto todos los paquetes se escribirán en minúsculas y sin utilizar caracteres especiales.

Descripción de los paquetes:

* templates: En este paquete se encuentran las plantillas que serán utilizadas para visualizar el sitio web.
* config: En este paquete se encuentran las configuraciones de la aplicación
* migrations: En este paquete se encuentran todas las migraciones de la aplicación
* controllers: En este paquete se encuentran los controladores principales de la aplicación.
* tests: En este paquete se encuentran las pruebas unitarias del sistema.

### Clases

Los nombres de clases deben ser mezclas de mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúsculas (CamelCase):



### Métodos

* Los métodos deberán ser verbos (en infinitivo), en mayúsculas y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas (lowerCamelCase).
* No se permiten caracteres especiales.
* El nombre ha de ser lo suficientemente descriptivo, no importando a priori la longitud del mismo.

### Variables

Los nombres de las variables tanto de instancia como estáticas reciben el mismo tratamiento que para los métodos.

Se evitará en la medida de lo posible la utilización de caracteres especiales, así como nombre sin ningún tipo de significado funcional. Las excepciones son las variables utilizadas en bucles for, para esos casos se permite utilizar i, j, k, l y siempre en ese orden de anidamiento.

El primer bucle siempre será el que tenga la variable i como iterador. (Esta variable se definirá para el bucle en cuestión).

## **Estilo de codificación**

### Comentarios

Los comentarios serán utilizados para dar información adicional al desarrollador sobre la implementación del diseño de la clase. Se tiene, por tanto, que evitar referencias al diseño funcional de la misma.

El uso abusivo de los comentarios es desaconsejable, principalmente por el trabajo extra necesario para su correcto mantenimiento. Es preferible rediseñar el código para una mejor compresión del mismo.

Se tienen que evitar el uso de caracteres especiales dentro de los comentarios así como el uso de cajas u otro tipo de gráfico creado mediante códigos ASCII.



### Declaraciones

Para la declaración de las variables se utiliza una declaración de cada vez:

String letra = new String(“A”);

La declaración de las variables locales a una clase, método o bloque de código se realizan al principio del mismo y no justo antes de necesitarse la utilización de la variable. La única excepción a esta regla son las variables que gestionan los bucles *for*.

Las variables de avance  de bucles *for* no podrán ser modificadas de ninguna manera fuera de la propia sentencia del bucle.

La duplicidad de los nombres de variables en diferentes niveles dentro de la misma clase se tiene que evitar.

### Sentencias

* Una sentencia por línea de código.
* Todo bloque de sentencias entre llaves, aunque sea una sola sentencia después de un *if*.
* La declaración de los bucles for será usualmente de la forma:

for (int i = 0; i < condicion ; i++)

* Son obligatorias las tres condiciones del bucle *for*: inicialización, condición de finalización y actualización del valor de la variable de avance.
* La variable de avance del bucle nunca podrá ser modificada dentro del propio bucle, utilizar variables auxiliares en caso de ser necesario.