UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERIA

CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES

TEMA:

DISEÑO E IMPLEMENTACION DE INTERFAZ DE PROCESAMIENTO DE LENGUAJE NATURAL PARA CONSULTAR CONTENIDOS ACADEMICOS DEL AREA DE REDES DE DATOS ENFOCADOS AL MODELO TCP/IP

AUTORES:

Álvarez Chancay, Rafael Abraham; Palacios Wither, Arturo Emmanuel

Trabajo de titulación previo a la obtención del título de

INGENIERO EN SISTEMAS COMPUTACIONALES

TUTOR:

Ing. Murillo Bajaña, Eduardo Wenceslao

Guayaquil, Ecuador

21 de Marzo del 2017

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERIA

CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES

CERTIFICACIÓN

Certificamos que el presente trabajo de titulación, fue realizado en su totalidad por Álvarez Chancay, Rafael Abraham y Palacios Wither, Arturo Emmanuel, como requerimiento para la obtención del Título de Ingeniero en Sistemas Computacionales.

TUTOR

Ing. Murillo Bajaña, Eduardo Wenceslao

DIRECTORA DE LA CARRERA

Ing. Guerrero Yépez, Beatriz Del Pilar Mgs.

Guayaquil, a los 21 del mes de Marzo del año 2017

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERIA

CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES

DECLARACIÓN DE RESPONSABILIDAD

Nosotros,

Álvarez Chancay, Rafael Abraham y Palacios Wither, Arturo Emmanuel

DECLARAMOS QUE:

El Trabajo de Titulación, **"Diseño e Implementación de Interfaz de Procesamiento de Lenguaje Natural para Consultar Contenidos Académicos del Área de Redes de Datos Enfocados al Modelo TCP/IP"** previo a la obtención del Título de **Ingeniero en Sistemas Computacionales**, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.
En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Guayaquil, a los 21 del mes de Marzo del año 2017

LOS AUTORES:

Álvarez Chancay, Rafael Abraham

Palacios Wither, Arturo Emmanuel

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERIA

CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES

AUTORIZACIÓN

Nosotros,

Álvarez Chancay, Rafael Abraham y Palacios Wither, Arturo Emmanuel

Autorizamos a la Universidad Católica de Santiago de Guayaquil a la **publicación** en la biblioteca de la institución del Trabajo de Titulación: **"Diseño e Implementación de Interfaz de Procesamiento de Lenguaje Natural para Consultar Contenidos Académicos del Área de Redes de Datos Enfocados al Modelo TCP/IP"**, cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y total autoría.

Guayaquil, a los 21 del mes de Marzo del año 2017

LOS AUTORES:

Álvarez Chancay, Rafael Abraham

Palacios Wither, Arturo Emmanuel

REPORTE DE URKUND

AGRADECIMIENTO

Agradezco a mi familia por todo el apoyo que me ha dado, el cual me dio la motivación de seguir adelante con mis estudios, a mis compañeros que han hecho que estos cinco años de estudios sea una experiencia agradable estando en un ambiente de compañerismo y amistad.

Palacios Wither, Arturo Emmanuel

AGRADECIMIENTO

Agradezco a mis padres por el apoyo que siempre me brindan cual se la situación y por guiarme por el camino correcto inculcándome buenos modales, además de enseñarme que para cualquier situación por más difícil que sea hay una solución. A mi hermano por siempre estar cuando lo necesito a pesar que a veces se moleste.

Agradezco a la universidad por haberme dado la oportunidad de obtener la beca que con mi esfuerzo y dedicación pude mantenerla durante toda mi carrera.

Álvarez Chancay, Rafael Abraham

DEDICATORIA

Este trabajo de titulación se lo dedico a mis padres, sin el apoyo y sin sus consejos no hubiera sido capaz de llegar a este punto como la persona que soy.

Palacios Wither, Arturo Emmanuel

DEDICATORIA

Este proyecto de titulación se lo dedico a mis padres, de los cuales me siento sumamente orgulloso por su manera de ser, de cómo llevan su vida, de su dedicación a mi hermano y a mí; siendo cada día mí ejemplo a seguir. A mi abuelito Roque que ya no se encuentra con nosotros, ya que él siempre me preguntaba cómo me iba en mis estudios.

A los demás miembros de mi familia que siempre han estado pendiente de mí y mis logros, siempre deseándome lo mejor.

Álvarez Chancay, Rafael Abraham

UNIVERSIDAD CATÓLICA

DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERIA

CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES

TRIBUNAL DE SUSTENTACIÓN

Ing. Murillo Bajaña, Eduardo Wenceslao

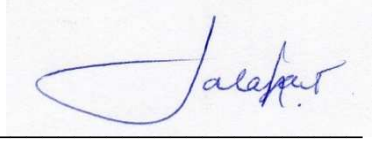
TUTOR

Ing. Guerrero Yépez, Beatriz Del Pilar Mgs.

DIRECTORA DE CARRERA

Ing. Cornejo Gómez, Galo Enrique

DOCENTE DELAGADO



Ing. Salazar Tovar, Cesar Adriano

OPONENTE

UNIVERSIDAD CATÓLICA

DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERIA

CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES

CALIFICACIÓN

Ing. Murillo Bajaña, Eduardo Wenceslao

TUTOR

Ing. Guerrero Yépez, Beatriz Del Pilar Mgs.

DIRECTORA DE CARRERA

Ing. Cornejo Gómez, Galo Enrique

DOCENTE DELEGADO

Ing. Salazar Tovar, Cesar Adriano

OPONENTE

ÍNDICE

RESUMEN

El proyecto propuesto consiste en el desarrollo e implementación de una interfaz capaz de analizar las consultas realizadas por estudiantes sobre temas relacionados a Redes siendo más preciso al Modelo TCP/IP basándose en el tema de Procesamiento de Lenguaje Natural o NLP siglas que corresponden a “Natural Language Processing” para que de esta manera los estudiantes no tengan la necesidad de hacer uso de lenguajes técnicos.

La interfaz está diseñada para que en ella puedan ingresar texto el cual después de un análisis y procesamiento por medio de la librería NLTK se llegue a elaborar una matriz atributo-valor solo con las palabras más relevantes para que los programas que implementen el algoritmo puedan realizar una extracción de información más exacta relacionado al tema consultado.

Para poder llevar a cabo el proyecto se utilizó una metodología cualitativo, además que se procedió a pedirle a un grupo de estudiante que realizaran preguntas libremente las mismas que permitieron conocer las palabras más comunes a la hora de hacer una consulta.

Palabras Clave: NLTK, TCP/IP, LENGUAJE NATURAL, INTERFAZ, NLP, MATRIZ, ATRIBUTO-VALOR, CONSULTAS.

ABSTRACT

The proposed project consists about the development and implementation about the development and design of an interface able to analyze queries made by students about subjects related to Networking, more specifically, the TCP/IP model using Natural Language Processing or NLP for short, allowing the students to make queries without the use of technical language.

The interface is able to receive text which is then analyzed and processed by using the NLTK library that at the end produces an atribute/value matrix only with the more relevant words so that the programs that implement this interface can extract information more precisely for their queries.

For this project to be developed a qualitative methology was used, in this project students were made to write queries freely, with no restrictions in order to analyze them in order to find a query pattern and find the most common words used by students.

Key Words: NLTK, TCP/IP, NATURAL LANGUAGE, INTERFACE, NLP, MATRIX, ATTRIBUTE-VALUE, QUERIES.

INTRODUCCIÓN

El procesamiento de lenguaje natural es un campo que ha estado presente desde los años 70s-80s hasta la actualidad, que se encarga del manejo y análisis del lenguaje, empleado por las personas y procesarlo a un lenguaje que sea entendible por una computadora. Una de los usos principales que se le puede dar a este campo es la comprensión del lenguaje de las personas, y a través de esa comprensión, poder extraer y usar esa información para algún fin determinado.

Un ejemplo similar a como funciona el procesamiento de lenguaje natural, son los motores de búsqueda como Google o Bing, a través de los cuales se puede investigar cualquier tipo de información, ya sea para fines académicos, de entretenimiento, profesionales, etc. Sin embargo, los buscadores solo devuelven resultados de las consultas realizadas por uno y no hay manera de poder observar y comprender como se analiza la consulta para poder llegar a resultados que muchas veces no tiene nada que ver con el tema lo que ocasiona pérdida de tiempo o desinterés en seguir buscando.

Una solución que se propone es desarrollar un programa dirigido al ámbito académico, el cual tendrá como objetivo procesar la consulta para así poder obtener una matriz atributo–valor, la misma que tendrá las palabras claves con la que cualquier aplicación que haga uso del programa puede saber lo que tendrá que buscar.

CAPITULO 1: FUNDAMENTACIÓN CONCEPTUAL

1.1 DESCRIPCIÓN DEL PROYECTO

El proyecto consiste en desarrollar un programa con una interfaz en la cual el usuario pueda escribir lo que se desea buscar de acuerdo al modelo *TCP/IP* en un lenguaje formal, con oraciones bien estructuradas. Después de haber ingresado la consulta y seleccionado la opción de procesar, el programa pasará a realizar el procesamiento del texto ingresado por el usuario con el fin de obtener la matriz atributo-valor la cual será usada por aplicaciones destinadas a la extracción de información, para esto se desarrollará una pequeña interfaz con el lenguaje Python, ya que es software libre; y además Python dispone de librerías *open source* (código abierto), que facilitarán el procesamiento del texto como es el caso de la librería *NLTK* (*Natural Lenguage Toolki*).

1.2 PLANTEAMIENTO DEL PROBLEMA

El método de búsqueda de información académicos tiene como objetivo dar a conocer a los estudiantes libros o páginas web con los temas que se tratarán en clase, ya que las opciones tecnológicas de búsquedas no son mayormente utilizadas por los estudiantes y por ende no sabrán dónde o cómo consultar algún tema específico. Uno de los elementos que generalmente evidencian el poco uso de los motores de búsquedas se ve reflejado en el promedio académico dado que a muchos estudiantes se les hace difícil de entender ciertos temas y no saben cómo y dónde obtener la información adecuada.

Usando la interfaz propuesta en este proyecto se podrá analizar, procesar y obtener la matriz atributo-valor de las consultas elaboradas por estudiantes con respecto a algún tema específico que se esté tratando en clase sin el uso de un lenguaje especial, la cual podrá ser utilizadas por cualquier programa destinado a la obtención de información.

La implementación de la interfaz es esencial para obtener un mejor análisis de lo que los estudiantes desean buscar con respeto a las materias.

1.3 PREGUNTA DE INVESTIGACIÓN

¿Es posible desarrollar una interfaz de procesamiento de lenguaje natural enfocada a una temática definida, pero que a su vez la interfaz tenga la capacidad de adaptarse a cualquier tema?

1.4 JUSTIFICACIÓN

Este trabajo de investigación busca sentar las bases para el uso de interfaces de procesamiento de lenguaje natural en un ámbito académico, cuyo impacto positivo se traduce en que permitirá procesar lo que los estudiantes o los docentes desean consultar sin la necesidad de usar un lenguaje técnico a través de una interfaz que después de analizar la consulta elaborará una matriz atributo-valor, la misma que podrá ser usada por algún programa dedicado a la extracción de información.

Las líneas de investigación que enmarcan este trabajo de titulación son electrónica y automatización identificada por la *UCSG* y la utilización de software libre establecida por carrera de Ingeniería en Sistemas Computacionales, ya que se emplearán librerías open source (código abierto) que facilitarán el procesamiento del lenguaje natural en la interfaz.

1.5 OBJETIVO GENERAL

- Diseñar una interfaz de procesamiento de lenguaje natural para el área de Redes de Datos, con un enfoque en el Modelo *TCP/IP*, que permita un mejor procesamiento de la información ingresada por parte de los estudiantes.

1.6 OBJETIVOS ESPECÍFICOS

- Evaluar las herramientas existentes en internet con el fin de analizar su funcionamiento, la manera que recibe la consulta.
- Diseñar una arquitectura de interfaz de procesamiento de lenguaje natural para poder realizar consultas relacionadas al área de Redes de Datos.
- Implementar la interfaz en un ámbito educativo de nivel superior para poder de esta manera demostrar que el proyecto realice el procesamiento del lenguaje natural.

- Elaborar la matriz atributo-valor que servirá para programas destinados a la extracción de información.

1.7 ALCANCE

La interfaz recibirá una entrada de texto por parte del usuario, en un lenguaje formal, con las oraciones en una estructura definida la cual podrán preguntar por fechas, autor o documentos, el algoritmo se encargara del procesamiento del texto ingresado mediante el uso de la librería *NLTK* (*Natural Language Toolkit*) la misma que descompondrá la oración y así obtener las palabras necesarias para poder ir llenando la matriz atributo-valor la cual servirá para una futura aplicación que realizara la consulta respectiva, correspondiente al campo de Redes de Datos, orientándose a todo lo relacionado con el modelo *TCP/IP*.
La salida es lo que ingresó el usuario, pero analizado e interpretado por la herramienta que a su vez será mostrado en una matriz con las palabras necesarias para poder llegar a extraer la información correcta.

1.8 MARCO TEORICO

1.8.1 Procesamiento de Lenguaje Natural

El procesamiento del lenguaje natural o también conocido por las siglas NLP que corresponden a “Natural Language Processing” tuvo su primera aparición en el año de 1950 cuando Alan Turing publicó un artículo llamado “Computing Machinery and Intelligence” siendo su traducción al español “Máquinas informáticas e Inteligencia” en la cual hizo referencia al *NLP* pero con el nombre de Prueba Turing. Según Turing (1950) la prueba se basa en el estudio de la habilidad que puede poseer una máquina para llegar a imitar el comportamiento humano, y responder un cuestionario de tal forma que el entrevistador no pueda diferenciar entre una persona y una computadora, en otras palabras, si una máquina puede llegar a pensar. El *NLP* se lo empleaba en traducciones, un ejemplo de este uso se lo pudo evidenciar durante la Guerra Fría para traducir de forma automática del ruso al inglés. Según Periñán (2012):

> El procesamiento del lenguaje natural […] es un campo del conocimiento al que han contribuido a su desarrollo disciplinas como la lingüística, la informática, la ciencia cognitiva y la ingeniería electrónica, esta última más estrechamente relacionada con las tecnologías del habla. (p. 14)

Fue en 1960 que se desarrollaron sistemas para el procesamiento de lenguaje trabajando con los denominados bloques de palabras (vocabulario utilizado para la traducción). Entre los años 70s y 80s apareció el primer programa para el entendimiento del lenguaje natural conocido como *SHRDLU* desarrollado por Terry Winograd y se empezó la explicación de la semántica. De acuerdo a Volpi (2012), el programa SHRDLU era un sistema en el cual una persona daba órdenes en lenguaje natural a un robot que se encontraba en este mundo virtual para que las ejecutaras y así de esta manera movieran ciertas figuras ya que este ambiente estaba formado de bloques de colores, que a su vez estaban compuesto por figuras como: (a) conos, (b) cubos y (c) paralelepípedos.

En los 90s apareció el NLP con métodos probabilístico, el mismo que se basa en una técnica de muestreo por medio de un proceso en el cual todos los individuos ya sean personas u objetos que conforman la muestra poseen el mismo número de oportunidades de ser o no seleccionado, y empezó a enfocarse en el aprendizaje automático lo que daría paso a lo llamado "Inteligencia Artificial". Finalmente, desde el año 2000 hasta la actualidad el *NLP* se enfoca principalmente en la semántica, discursos, técnicas simbólicas. Actualmente el *NLP* es conocido por formar parte, del área de "Inteligencia Artificial" y la Lingüística, la misma que se refiere a transformar el lenguaje empleado por las personas a un lenguaje que pueda ser interpretado y así poder procesar el lenguaje natural. De acuerdo a Benítez, Escudero, Kanaan y Masip (2013):

> La Inteligencia Artificial (IA) es una disciplina académica relacionada con la teoría de la computación cuyo objetivo es emular algunas de las facultades intelectuales humanas en sistemas artificiales. […]. El diseño de un sistema de inteligencia artificial normalmente requiere la utilización de herramientas de disciplinas muy diferentes como el cálculo numérico, la estadística, la informática, el procesado de señales, el control automático, la robótica o la neurociencia. (p. 3).

1.8.2 Aplicaciones de Procesamiento de Lenguaje Natural

Desde el nacimiento del procesamiento del lenguaje natural se lo ha aplicado en diferentes campos los cuales van desde una simple traducción hasta la búsqueda de información por ejemplo los motores de búsquedas (navegadores) que actualmente son usados (Google Chrome, Firefox, etc…).

Los principales campos donde se aplica el Procesamiento de Leguaje Natural son:

- Búsqueda de información, como los usados en documentos Word para encontrar alguna palabra especifica.
- Traducción a diferentes idiomas, como es el caso de Google Traductor.
- Interfaces PC-Usuario, un buen ejemplo de esta aplicación son los asistentes usados en celulares o en las computadoras como Cortana la misma que es el asistente de Windows 10.
- Análisis de lenguaje, los cuales son aplicados en el desarrollo de programas como por ejemplo compiladores.
- Reconocimiento de las expresiones del lenguaje empleadas por las personas.
- Síntesis de voz.

1.8.3 Análisis de Procesamiento de Lenguaje Natural

Para poder realizar correctamente el procesamiento de lenguaje natural se debe tener en cuenta los aspectos que la conforman ya que dependiendo de esto se podrá tener un mejor entendimiento; los aspectos del lenguaje que se deben considerar: (a) sintaxis, (b) semántica, (c) pragmática, (d) morfología, (e) fonética y (f) fonología.

Sintaxis

Según Casas (2015), la sintaxis forma parte de la gramática la cual se encarga del estudio acerca del orden y la relación que debe poseer cada palabra que forma una oración, es decir, la combinación que estas deben poseer para que una oración tenga sentido. Por ejemplo:

- Reprobó por no estudiar el estudiante materia una. (Esta es una oración con sintaxis incorrecta)

- El estudiante reprobó una materia por no estudiar. (Esta es una oración con sintaxis correcta)

Semántica

Según Escandell (2011), "la semántica es la disciplina que estudia el significado de las expresiones lingüísticas. Como estrategia metodológica, es común separar el estudio del significado de las palabras y el de las expresiones más complejas, como sintagmas y oraciones" (p. 13).

Por ejemplo:

- El juguete que compramos necesita pilas. (Denotación)
- Carlos se debe poner las pilas sino reprobará. (Connotación)

Pragmática

De acuerdo a Durán (2014), la pragmática se encarga del estudio de la forma de como una persona produce y puede llegar a interpretar la información, es decir, como se desarrolla la información. Por ejemplo:

- Ya es tarde. (La persona hace referencia al tiempo)
- Ya es tarde. (La persona esta incomoda, se quiere ir o quiere votar a alguien)

Morfología

Según García y Penny (2013), la morfología forma parte de la gramática encargada del estudio de la forma y definir la estructura de las palabras, y como estas puede dar paso a que se pueda generar otras palabras. Por ejemplo:

Marcos iba en carro.

Análisis morfológicos:

- Marcos (Nombre Propio)
- Iba (verbo)
- En (Indefinido)
- Carro (Nombre común)

Fonética y Fonología

Según Nguendjo (2013), la fonética estudia todos los elementos fonéticos que posee un idioma desde varios puntos como son: (a) acústica, (b) percepción y (c) producción. Y la Fonología se encarga del estudio de los elementos fonéticos dependiendo el sistema lingüístico.

1.8.4 Problemas del Procesamiento de Lenguaje Natural

Al momento de procesar lenguaje natural a parte de tener en cuenta los aspectos del lenguaje que se tratará se presentan dificultades de las cuales dependerá la efectividad para poder analizar el lenguaje.

Las dificultades a tener en cuenta son tres, las mismas que son:

- **Ambigüedad:** el término se lo emplea para expresar cuando una palabra o expresión puede tener diferentes significados, esto dependerá de cómo lo entiende una persona.
- **Separación:** espacio entre palabras.
- **Recepción incorrecta de datos:** esto sucede debido a los acentos y dificultad para expresarse verbalmente.

1.8.5 NLTK

NLTK son las siglas correspondientes al Natural Language Toolkit, que es una librería desarrollada para el lenguaje de programación de Python, la cual está orientada al procesamiento de lenguaje natural como también en áreas relacionadas como es la inteligencia artificial. Es un proyecto gratuito “Open Source”, liderado por una comunidad activa de programadores.

El NLTK se usa especialmente para poder ayudar en todo lo referente al procesamiento del Lenguaje Natural, en otras palabras, en áreas como “Inteligencia Artificial”, “Lingüística Empírica”, entre otras. El NLTK posee componentes y funciones los cuales ayudarán al

procesamiento del lenguaje, los mismos que serán detallados a continuación, dichos componentes son: (a) tokenizador, (b) postagger y (c) chunking.

Tokenizador

De acuerdo a Hardeniya (2015), tokenizar se refiere a un proceso mediante el cual un texto puede ser dividido en elementos llamados tokens, los mismos que son unidades indivisibles. Un tokenizador o también llamado segmentador es el primer componente utilizado al momento de procesar un texto. Cabe especificar que si existen muchos tokenizadores, es importante y necesario adaptarlos por objetivos.

Postagger

> De acuerdo a Correa (2015):
> Es una parte del software que permite leer el texto de un idioma y asignar las partes de una oración, en cada una de las palabras del texto como: verbo, adverbio, sustantivo u otra, es decir, permite etiquetar las palabras de una cadena y asigna el resultado dependiendo de que se trate esa palabra (p. 27).

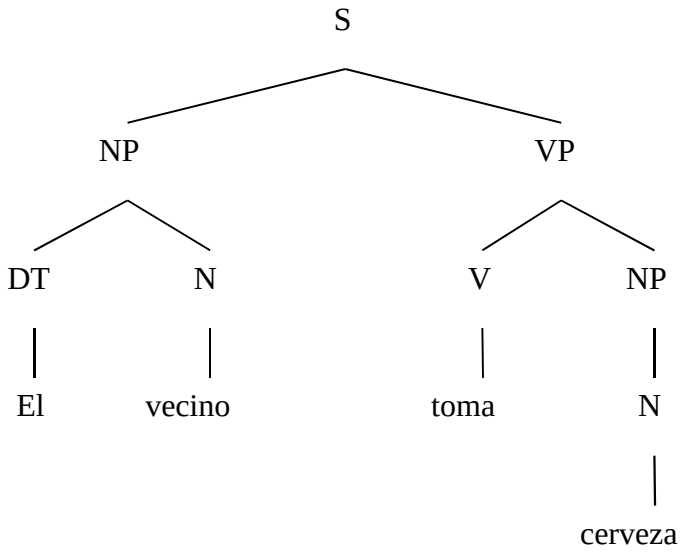
Chunking

De acuerdo a Perkins (2014), chunking se refiere al proceso en el cual se toma piezas de información y se las agrupa en unidades más grandes. Un punto importante es que al agrupar las piezas se puede mejorar el número o cantidad de información que se llegue a recordar.

1.8.6 Árbol de Dependencias

El árbol es una forma de analizar un texto, utilizando “Parsing de dependencia” el cual consiste en descomponer la frase para poder ordenar de una mejor manera cada uno de sus elementos, el árbol inicia con una raíz el cual será el verbo y desde este desplegar las demás dependencias; por ejemplo, a cada elemento de la frase se identificará sujeto, objeto directo y complementos.

Por ejemplo:



Otro método es el de usar “pos-tagger” que consiste en armar un diccionario con cada Token y etiquetarlos si son adjetivos, pronombres, verbo, entre otras, los mismos que se obtendrán con la ayuda del corpus.

1.8.7 Corpus

El corpus es el cuerpo del mensaje, es decir, es un conjunto de texto que trata de un lenguaje (idioma) especifico; al cual se le puede enseñar nuevos lenguajes los mismos que posteriormente podrán ser analizados.

Los corpus son utilizados para el procesamiento de lenguaje natural y trabaja de la mano con el módulo NLTK; la función del corpus en relación a NLP es la de facilitar la sintaxis usadas en el lenguaje que aprendió.

1.8.8 Tkinter

Tkinter es un binding o también expresado como una unión de la biblioteca gráfica Tcl/Tk que sirve para poder diseñar ventanas (utilización de interfaz gráfica) empleado en el lenguaje de programación de Python. Se trata de un módulo que proporciona las funciones necesarias para el desarrollo de interfaces para usuarios. De acuerdo a Chaudhary (2015) Tkinter se lo implementa como un módulo (-tkinter.py) para versiones de python2.x y

Tkinter/_inity_.py para versiones de Python 3.x. Al fijarnos en el código fuente, Tkinter es un contenedor de una extensión C que es utilizado en las bibliotecas Tcl/Tk.

Tkinter es muy adecuado para el desarrollo de aplicaciones dentro de un amplia variedad de áreas, que van desde pequeñas aplicaciones de escritorios hasta aplicaciones en modelos científicos e investigaciones que se lleven en diferentes disciplinas. Al utilizar y aprender a utilizar Python la persona debe tener conocimientos en GUI (Graphical User Interface), Tkinter es la forma más fácil y rápida para realizar un trabajo.

En base a Chaudhary (2015), Tkinter es una gran opción para la programación GUI, debido a las siguientes características que posee:

- Fácil de aprender (más sencillo que otro paquete GUI para Python).
- Código relativamente pequeño para producir aplicaciones GUI potentes.
- Es portátil en todos los sistemas operativos.
- De fácil acceso, debido a que viene preinstalado con la distribución estándar de Python.

Ninguno de los otros kits de herramientas de GUI de Python tiene todas estas características al mismo tiempo.

CAPITULO 2: METODOLOGIA DE LA INVESTIGACION

2.1 TIPO DE INVESTIGACION

El presente trabajo de titulación se enmarca en los parámetros de una investigación descriptiva, de acuerdo a Rojas (2015):

> Exhibe el conocimiento de la realidad tal como se presenta en una situación de espacio y de tiempo dado. Aquí se observa y se registra, o se pregunta y se registra. Describe el fenómeno sin introducir modificaciones: tal cual. Las preguntas de rigor son: ¿qué es?, ¿cómo es?, ¿dónde está?, ¿cómo ocurre?, ¿cuántos individuos o casos se observan?, ¿cuáles se observan? (p. 7).

La investigación descriptiva o también denominada investigación estadística, utiliza un enfoque cuantitativo, lo que hace la investigación descriptiva es, una vez que se obtengan los datos, proceden a describirlos o analizarlos. Responde a las preguntas: qué, quién, dónde, cuándo, cómo y por qué.

2.2 ENFOQUE METODOLOGICO

El enfoque que se utilizará en el presente trabajo de titulación se basa en la metodología cualitativa, de acuerdo a Graham (2012), la metodología cualitativa se refiere al sentido por la cual una investigación llega a obtener datos descriptivos, los mismos que pueden venir de palabras dichas por personas sean estas habladas o plasmadas en un papel, o también se las puede obtener mediante una observación planificada y bien elaborada.

La investigación cuantitativa es aquella en la que se maneja el uso de datos que puedan ser cuantificados, es decir convertidos a números, estos serán recogidos mediantes varias técnicas cuantitativas de levantamiento de información para que puedan ser estudiadas y analizadas.

2.3 UNIVERSO-POBLACIÓN

La Población elegida para el estudio del tema fueron los alumnos de la Carrera de Ingeniería en Sistemas de la Universidad Católica de Santiago de Guayaquil, conforme a Icart, Fuentelsaz y Pulpón (2006):

> Es el conjunto de individuos que tienen ciertas características o propiedades que son las que se desea estudiar. Cuando se conoce el número de individuos que la componen, se habla de población finita y cuando no se conoce su número, se habla de población infinita. (p. 55).

2.4 MUESTRA

La muestra con la que se elaborara la recolección de datos es una muestra intencionada la misma que la conforman los alumnos de la materia de Redes de Datos I. De acuerdo a Icart et al. (2006):

> Es el grupo de individuos que realmente se estudiarán, es un subconjunto de la población. Para que sea representativa, se han de definir muy bien los criterios de inclusión y exclusión y sobretodo, se han de utilizar las técnicas de muestreo apropiadas. (2006, p. 55).

El número de personas a la cual se le pidió que realizaran preguntas fue 30 las que se dividen en 15 estudiantes de redes 1 y 15 estudiantes fuera de esa materia, tomando como población los estudiantes de la carrera de sistema empleando la regla de 3 se determinó que el número de personas con la que trabajamos representa el 11% aproximadamente de la población.

Población: 288 estudiantes aprox.

Personas encuestadas: 30 estudiantes

288 ⟶ 100%

30 ⟶ X

X = 10.42%

El porcentaje con el que trabajamos es aceptable ya que el objetivo era tener una idea de que palabras son empleadas para realizar consultas y compararlas con el corpus *stopwords* y de esta manera modificarlos en caso que hallan palabras nuevas.

2.5 TECNICAS DE RECOPILACION DE DATOS

Para el presente trabajo, se decidió buscar ayuda por parte de los estudiantes de la materia de Redes de Datos I, debido a que el tema al cual se encuentra orientando el programa de procesamiento de lenguaje, es uno de los varios temas que se ven a lo largo de un semestre de estudios. Inicialmente la forma con la cual se procedió a recopilar información fue mediante una consulta realizada a los estudiantes, cabe mencionar que se pidió el debido permiso del docente, durante uno de sus días de clases, en el cual solicitamos que cada alumno en un bloc de notas elabore un listado que contenga 10 preguntas. Un punto muy importante que se pudo evidenciar es que al principio los estudiantes no entendían el tema de procesamiento de lenguaje natural, además de presentar cierta dificultad al momento de elaborar las preguntas.

Debido al tiempo se optó por recibir las preguntas que lograron elaborar sin considerar que fueran 10 preguntas que se solicitó por alumno. Después de receptar las preguntas se procedió con el debido análisis y tabulación de las mismas, para la primera recopilación de palabras no se establecieron reglas o limitantes a los estudiantes al momento de hacer el listado, pero para las tabulaciones futuras ya era necesario tenerlas establecidas, para poder obtener fácilmente una estructura general de oraciones empleada por los alumnos, además de un listado con las palabras más comunes usadas por los mismos.

2.6 APLICACIÓN Y PROCESAMIENTO: HERRAMIENTAS ESTADÍSTICAS

Con el fin de poder analizar y procesar la información obtenidas gracias a los estudiantes de la materia de Redes de Datos 1 se hizo uso de herramienta Excel la cual es parte de Microsoft Office.

Por medio de esta herramienta se procedió a la elaboración de gráficos y tablas a partir de la información obtenida.

2.7 ANÁLISIS DE RESULTADOS

Primera Visita: En la primera consulta que se realizó a los estudiantes de la clase de redes 1 se les pidió que cada uno realizara preguntas relacionadas a la materia sin la utilización de parámetros.

Se les dio libertad al momento de realizar las preguntas ya que el objetivo de esta primera visita era conocer la manera de como ellos escribían como por ejemplo:

- Con que palabras empezaban para hacer una consulta.
- Si preguntaban por fechas, autor, o tipo de papers.
- Si empleaban algún tipo de signos.

Tabla 1

Preguntas obtenidas de los alumnos

Total de Preguntas	**70**
Pronombre	20
Verbo	12
Sustantivo	11
Adverbio	27

Nota: Elaboración Propia

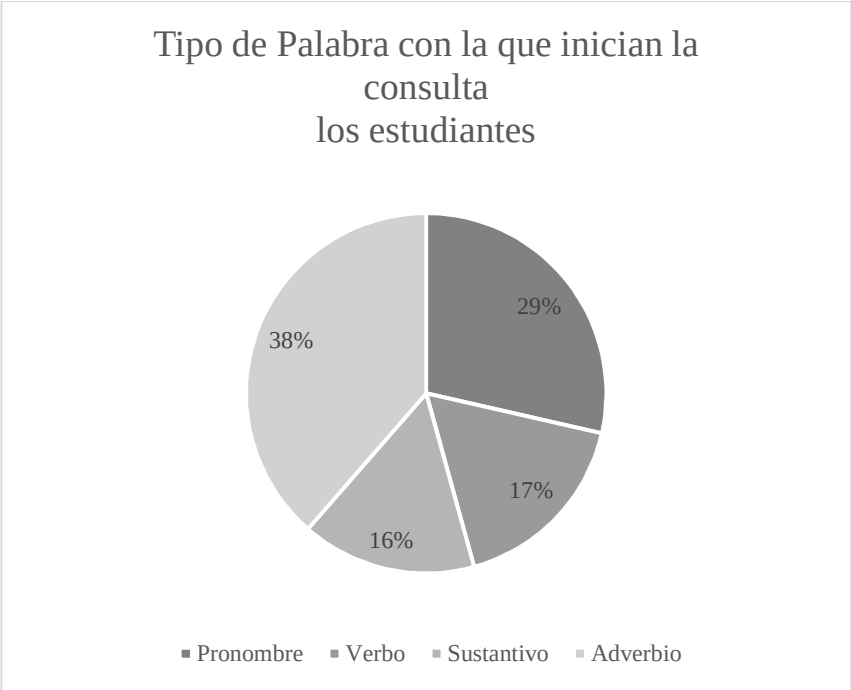


Figura 1: Diagrama pastel de porcentajes de la utilización de pronombres, verbos, sustantivos y adverbios al iniciar las preguntas

Tabla 2

Palabras más utilizadas para la realización de consultas.

Palabras	**# Veces**	**Palabras**	**# Veces**	**Palabras**	**# Veces**
Lista	5	Mostrar	3	La	1
Qué	19	Listar	4	Cuántas	1
Quién	1	Existe	2	Proveedores	1
Cómo	9	Listado	2		
Cuál	4	Se	1		
Cuáles	6	Importancia	1		
Clasificar	1	Top	2		
Definir	1	Dar	1		
Mejores	1	Quiero	1		
De	2	Para	1		

Total: 70

Nota: Elaboración Propia

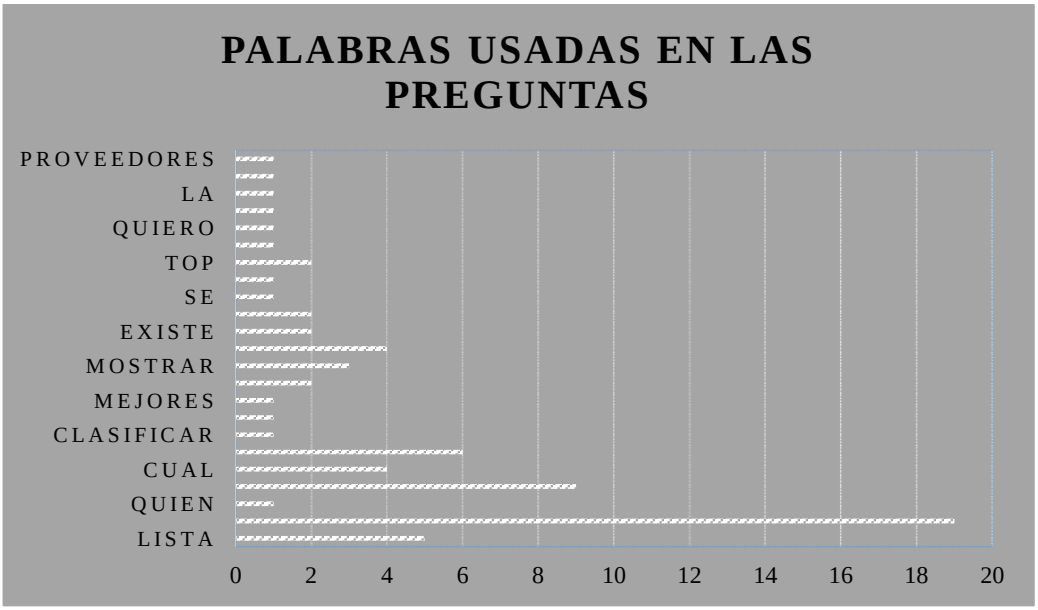


Figura 2: Cuadro comparativo de palabras usadas en consulta por los estudiantes.

En esta primera visita se observó que la palabra más empleada para iniciar una consulta fue “Que” siendo esta parte del grupo predominante de adverbios.

Segunda Consulta: En la segunda consulta que se realizó a los estudiantes de la misma clase de redes 1 se les pidió que cada uno realizara preguntas relacionadas a cualquier tema de su interés preferentemente de materias pero esta vez empleando ciertos parámetros.

A los estudiantes se les presento dificultades al momento de realizar las preguntas, ya que las preguntas tenían que emplear 3 reglas las cuales eran: (a) Fechas, (b) Autores y (c) Papers,

Tabla 3

Frases más usadas en la consulta implementado fechas, autores y papers

Fechas	**Autores**	**Papers**	**Fechas y Autores**
En los últimos 10 años	Obras más populares de autor	En pdf	El año en que se publicó y el autor del libro …
Más de 5 años	Mejores obras de autor	Computadoras cuánticas papers	El año y autor del libro
Desde marzo del 0000	Libros de autores sobre	Papers de xxxxxxx	
En qué fecha	Que autores poseen libros sobre		
Cuáles son las fechas que hay	Quién creó		
Más innovadores del 0000	Quién hizo		
En qué año	Quién es considerado		
En qué fecha se publico	Quién fue		
Cuándo se lanzó	Quién fabricó		
Cuál es el año	Cuál es el autor		
Entre los años 0000 – 0000	Del libro de autor		
Qué sucedió en 0000	Según el libro de NNNN		
Ocurrieron en el 00000	Según NNNN		
Avances tecnológicos en el 0000			

Nota: Elaboración Propia

El objetivo de estas dos visitas eran conocer las palabras más usadas por estudiantes para realizar consultas con el fin de poder compararlas con el corpus Stopwords el cual es un archivo que contiene una lista de palabras que no brindan un valor agregado y de esta manera ver que palabras que los estudiantes usaron no están en la lista e ir personalizando el corpus de acorde a la temática propuesta.

CAPITULO 3: PROPUESTA

3.1 HERRAMIENTAS ACTUALES

Además de desarrollar la interfaz de procesamiento de la temática, se revisaron algunas herramientas disponibles en la actualidad para ver cuál es el comportamiento de una interfaz de procesamiento de lenguaje natural, entre las herramientas que se revisaron están:

- TextRazor
- Demo de NLTK
- Herramienta NLP de la Universidad de Stanford

TextRazor es una aplicación web que permite el análisis y procesamiento de lenguaje natural, en un cuadro de texto el usuario ingresa lo que se desea analizar y la página lo que genera de salida es señalar las entidades/palabras clave que reconoce e indicar en dónde puede encontrar más información relacionada sobre esa entidad. Es capaz de reconocer entidades de cualquier temática, pero de cierto forma es limitada, ya que no filtra las palabras de relleno y no llega a reconocer todas las palabras que recibe por parte del usuario.

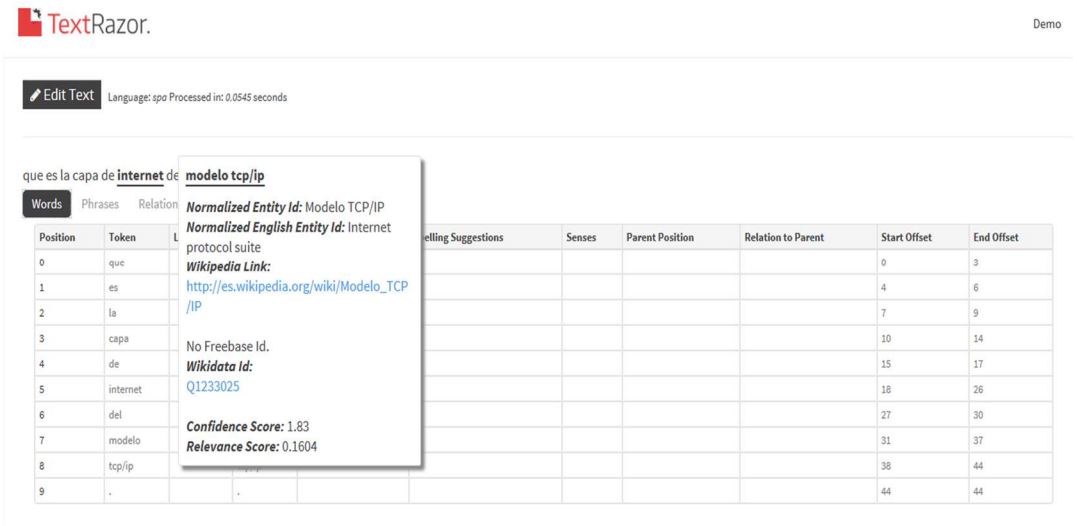


Figura 3: Consulta analizada por la herramienta TextRazor.

El demo del NLTK permitía ver las capacidades de la librería, pero en el caso del demo, utilizaba corpus que ya venían incluidos en el NLTK, haciendo de esta forma que esté limitado en reconocer solo el texto que esté presente en dichos corpus, además de otra observación que dichos corpus etiquetaban las palabras con códigos especiales, haciendo que sea más difícil entender a qué correspondía cada palabra etiquetada/tageada.

La herramienta de la Universidad de Stanford es similar al TextRazor, aplica un modelo llamado NER el cual significa name-entity-recognition, o en español, modelo de reconocimiento de nombre de entidades, al analizar el texto que haya sido ingresado por el usuario, el modelo le permite al programa indicar a qué clasificación es asociada. Las clasificaciones que el modelo de Stanford tiene las denomina entidades, y en el caso de la universidad de Stanford tiene un modelo estandarizado, que es usado en las librerías que emplean NER, el modelo como tal presenta siete clasificaciones:

- PERSONA
- ORGANIZACIÓN
- UBICACIÓN
- DINERO
- PORCENTAJE
- FECHA
- HORA

Igual que con las herramientas anteriores, el programa puede llegar a asociar las palabras con las entidades a las que pertenezca, a diferencia del TextRazor, el de Stanford no indicaba en dónde encontrar información adicional, pero igual que con las otras dos herramientas, puede llegar a reconocer y asociar palabras que pueden llegar a tratar de diversas temáticas al mismo tiempo.

No hay herramientas que procesen y analicen el lenguaje natural en base a una sola temática, pero que a su vez sean capaces de adaptarse a cualquier temática.

3.2 DESARROLLO

Para el desarrollo de la propuesta, lo primero que se realizó fue ingresar a la página del NLTK para aprender sobre la librería, entender su funcionamiento y cómo responde cuando se le da texto. Además de aprender sobre lo que ofrece el NLTK, también se tuvo que aprender sobre el lenguaje de programación Python, debido a que la librería fue hecha para trabajar con Python y no se tenía los fundamentos claros sobre el lenguaje de programación Python. El libro que habla sobre el uso y funcionamiento del módulo NLTK, Bird, Steven, Loper y Klein (2009), no sólo enseña los conocimientos acerca de su librería, también sirve como un libro guía para

aprender Python, haciendo que el aprendizaje de sus funciones y componentes accesibles tanto para los usuarios expertos como para los que recién están comenzando.

Se usó el libro como una guía, además de tener como referencias tutoriales sobre el manejo de la librería, los cuales están disponibles en el internet para los usuarios que estén interesados en el tema. Se empezó practicando con algunas funciones que se podían realizar empleando el NLTK, la primera fue Tokenización. Hay 2 maneras de tokenizar el texto, por oración y por palabra:

Oración: Hola mundo.
Oración tokenizada por palabra:
['Hola', 'mundo', '.']
Texto: Buen día. Me voy a la playa.
Texto tokenizado por oraciones:
['Buen día.', 'Me voy a la playa.']

Para propósitos de esta investigación se decidió trabajar haciendo uso del tokenizador por palabras, para así poder analizar y asignar el POS (*Part-of-Speech*) respectivo de cada palabra de la consulta que vaya a ingresar el usuario. Otro componente que se revisó y practicó es el *stemmer*, el cual su objetivo es el reducir la palabra a su raíz:

Listado = ["buscar", "buscando", "buscaré", "busqué", "busca"]
Raíz de las palabras:
Salida = busc, busc, busc, busqu, busc

Para el proyecto no se consideró emplear este componente, ya que obtener la raíz de las palabras no es una función necesaria para el análisis de la consulta que ingrese el usuario, porque para este proyecto se deben analizar las palabras completas, como las que ingresó el usuario. Una función que podría facilitar el análisis del proyecto es el *lemmatizer*, el cual en sí es similar al *stemmer*, ya que a las palabras lo que hace es buscar palabras similares o si están en plural, obtiene la palabra en singular. Utilizar el componente facilitaría el procesamiento de la consulta debido a que se podrían establecer palabras y relacionarlas con su versión en singular, haciendo que sea más sencillo el etiquetado de cada palabra, sin tener que estar

tomando en cuenta para cada palabra su versión en plural y tener que para cada variante estar etiquetando las palabras. Un ejemplo del lematizador sería el siguiente:

Listado = ["cats", "rocks", "papers", "cacti"]
Usando el lematizador:
Salida = cat, rock, paper, cactus

El problema que se presentó con el lematizador es que el NLTK no tiene la opción de utilizar el lematizador en otro idioma que no sea inglés. En el internet sí hay recursos que indican u ofrecen lematizadores en otros idiomas, español incluido, pero se decidió no utilizarlo en el proyecto, ya que los recursos que se encontraban, ofrecían los lematizadores en otros lenguajes de programación. Esto hubiese significado pasar el código al lenguaje que estamos empleando, lo cual tomaría tiempo solo verificando, corrigiendo y comprobando que el lematizador funcione, en caso que hubiera sido factible el implementarlo, este elemento facilitaría el procesamiento de las palabras, haciendo que pueda analizar rápidamente si las palabras están en plural, pasándolas a singular, lo que reduciría el volumen de texto que se emplearía para el entrenamiento.

Una observación que se presentó al momento de hacer las pruebas es como hacer el filtrado de palabras que no dan un valor agregado al texto, por ejemplo:

Busca libros acerca del protocolo tcp.

En esa oración no interesan las palabras “acerca” y “del”, ya que para el análisis no dan un valor que sea de utilidad, debido a esto se pidió a los estudiantes que realicen ellos a su manera preguntas acerca del tema, para poder determinar así qué palabras son las más empleadas y elaborar una estructura en la cual los usuarios puedan realizar la consulta. El problema con esto es que en cierta manera se limita la forma en que se realizan las consultas, porque de esa forma se están estableciendo reglas indicando a los usuarios qué escribir y qué no, haciendo que sea menos procesamiento de lenguaje natural. Una solución que se presentó venía incluida con el NLTK, y esa es el corpus *stopwords*, el cual tiene un conjunto de documentos, uno por idioma, los cuales tienen un listado de palabras que son consideradas de relleno para el análisis de un texto. Haciendo de esta manera que sea simple filtrar las consultas, para así analizar de esta manera sólo el texto relevante.

Por lo tanto, sólo se tuvo que actualizar el listado del documento para el idioma español, con las palabras adicionales que salieron de las tabulaciones realizadas de las preguntas hechas por los estudiantes; retomando el ejemplo anterior:

Buscar libros acerca del protocolo tcp.

Usando el corpus *stopwords*:

Oración filtrada = ['Investigar', 'libros', 'protocolo', 'tcp']

Para decidir cómo y con qué entrenar el programa se revisaron algunos inconvenientes, ya que si se quiere analizar un texto cualquiera, primero se debe entrenar los componentes, los cuales dependen de qué se quiere hacer con el texto analizado. Para este proyecto se consideró emplear el POS Tagger y usar el Modelo de Reconocimiento de Nombre de Entidades (NER), para etiquetar cada palabra con su respectivo POS y usar el reconocimiento de nombre de entidades para clasificar las palabras. Los problemas con este planteamiento inicial eran que no había recursos para poder entrenar el NER que viene incluido con el NLTK. Después de buscar en el internet no se encontraron recursos que indique cómo entrenar un modelo NER propio; lo más cercano a ello es el modelo desarrollado por la Universidad de Stanford, que estaba disponible en inglés, alemán y español. Sin embargo este modelo no se podía utilizar, ya que los nombres de entidades que usa ese modelo para clasificar son estandarizados, es decir, que no hay una forma de agregar entidades propias al NER. Cabe notar que mientras se realizó la búsqueda de información se encontró que los usuarios mismos dentro de foros de programación y otras páginas, preguntaban sobre la misma duda, cómo entrenar un modelo propio, recibiendo la misma respuesta, no hay recursos específicos sobre el tema, más aún si se desea entrenar el modelo en un idioma que no sea inglés.

El segundo planteamiento fue el de usar el POS Tagger para etiquetar y clasificar a la vez las palabras y luego realizar la fragmentación de la consulta, para así tener el bloque o matriz valor/atributo, el cual pasaría después a otro programa en el que se procesará la información recibida acorde a los requerimientos del mismo. En la mayoría de los casos, los corpus tienen su texto etiquetado tomando como base etiquetas POS estándar, los que indican si la palabra es pronombre, sustantivo, verbo, etc., pero también se puede etiquetar los corpus con etiquetas POS personalizadas, por ejemplo:

Buscar libros del modelo tcp/ip

Usando etiquetas POS estándar:

Buscar -> Verbo

Libros -> Sustantivo

Del -> forma parte del listado de palabras en el corpus *stopwords*

Modelo -> Sustantivo

Tcp/ip –> -

Ahora, usando etiquetas POS personalizadas para el proyecto cambian los atributos de cada palabra, de esta forma se puede catalogar el texto acorde a las necesidades del programa en el que se lo está usando:

Libros -> Tipo_Repositorio

Modelo -> Tema

Tcp/ip -> Tema_Principal

Una vez definido los componentes a usar se empezaron a realizar pruebas entrenando un POS Tagger, con un archivo de texto, el cual contiene texto de prueba. Una vez entrenado el tagger se procedió a probar ingresando texto que formaba parte del texto de prueba y texto que no era parte del mismo, efectivamente reconocía el programa las palabras con su respectiva etiqueta POS en caso de que fueran usadas para entrenar el tagger, y mostraba un mensaje de error en caso que se ingresara texto no relacionado con el archivo. El problema con esa situación es que en ese momento el tagger solo podía ser entrenado usando un solo documento, y se lo tenía que entrenar cada vez que se quería probar el funcionamiento del programa, la solución a esto fue el emplear una clase de la librería del NLTK llamada CategorizedTaggedCorpusReader, la cual permite establecer un conjunto de documentos como un corpus, y como identificar las categorías en las que se clasificarían, para el proyecto los documentos se denominan t_nombre_documento.txt, en este caso siendo la categoría nombre_documento, por ejemplo:

- t_verbos.txt
- t_adverbios.txt
- t_valores.txt
- t_pronombres.txt
- t_doc1.txt

En ese caso si el corpus solo constara de esos archivos, el corpus tendría un total de cinco categorías y el tagger detectaría solo las palabras que se encuentren dentro de esos archivos, de esta manera el tagger solo en una línea de código se entrenará con todos los documentos que estén almacenados en el directorio del corpus, y una vez terminado el entrenamiento se almacena el tagger en un archivo .pickle, el cual puede ser llamado en un futuro, ahorrando así la necesidad de estar entrenando a cada momento el tagger.

Otros problemas se presentaron cuando se realizaron pruebas con el programa, uno fue al ingresar texto de prueba, probando en minúsculas y mayúsculas para determinar si el tagger reconocería las palabras sin importar cómo estaban escritas, lo cual no fue el caso, el tagger discrimina las mayúsculas y minúsculas incluidas en los documentos empleados para el entrenamiento. Si bien se consideró ingresar en los textos las palabras tanto en minúsculas y mayúsculas, no es algo que se pueda realizar en un periodo, debido a la gran cantidad de palabras que se tendrían que agregar, además de que se tendrían que considerar escenarios en los que la consulta ingresada por el usuario tenga mayúsculas y minúsculas, por ejemplo:

BusCaR LiBroS de TCP/ip.

Si bien la oración no está escrita correctamente, sí se puede analizar, y procesar, el problema es que para que el tagger reconozca las palabras y las etiquete correctamente, el tagger tendría que haber sido entrenado previamente con texto que tenga las palabras escritas de esa misma forma, y las probabilidades de que eso ocurra son nulas. Para evitar esos inconvenientes se decidió para el tagger, que todo el corpus tenga el texto en minúsculas, y al momento de pasar la consulta escrita por el usuario, con una función en Python, hacer que la caracteres se pasen a minúscula todos.

Otro problema que al momento de verificar que la consulta ingresada por el usuario sea válida, no porque la consulta no se relacione con el tema, sino debido a que si los usuarios ingresan caracteres como por ejemplo () o [], el programa no los reconocerá, o si la consulta se la hace en forma de pregunta usando ¿? el tokenizador no separa el primer signo de interrogación (¿), así:

Oración: ¿Qué es TCP/IP?

Salida Tokenizada = ["¿Qué", "es", "TCP/IP", "?"]

Al inicio se consideró que en los textos que se fueran a incluir para el entrenamiento del tagger contengan texto que permita reconocer las palabras si empiezan con el signo de interrogación, pero eso tendría que tener adicionalmente todas las variantes de cómo se iniciaría una pregunta y no necesariamente ingresarán de esa forma la consulta los usuarios, si se llega a poner:

¿Protocolo tcp/ip?

En vez de:

¿Qué es el protocolo tcp/ip?

El programa no etiquetaría correctamente la primera oración, lo que lleva a la conclusión de que no se puede analizar la palabra a la que está junta. Para esto la solución inicial que se planteó fue entrenar un tokenizador del NLTK con texto para verificar si así logra separar la oración si empieza con el signo de interrogación. Para entrenar el tokenizador se empleó uno de los corpus en español que venía con el NLTK, ya que en el caso del tokenizador no era necesario mandarle texto que solo esté relacionado con el tema.

Aún después de haber entrenado el tokenizador y haberlo probado no se lograba separar, de modo que para esto la única solución viable era aceptar que puedan ingresar la pregunta empleando los signos de interrogación, para asegurarse que no puedan ingresar cualquier cosa. Se decidió hacer uso de las expresiones regulares en Python, lo cual permite elaborar un patrón de validación de una cadena de texto para determinar un patrón que verifique que si bien pueda iniciar la consulta con el signo de interrogación, al momento de ser tokenizada se le quita ese carácter a la consulta.

La fragmentación de la consulta es una función que permite establecer un bloque atributo/valor del texto que ya fue tokenizado y taggeado, para el proyecto es una función importante ya que el tagger por sí solo etiqueta toda la consulta pero para el bloque atributo/valor no se necesitan todas las palabras, solo las palabras que tengan etiquetas relevantes como Tema, Fecha, Tipo_Repositorio, etc., por ejemplo:

Oración: Buscar libros del modelo tcp/ip

Una vez filtradas, tokenizadas y etiquetadas las palabras se tiene como resultado el bloque valor/atributo:

```
(S
    (Bloque  Valor|Atributo
    libros/TIPO_REPOSITORIO
    modelo/TEMA
    tcp/ip/TEMA_PRINCIPAL_MODELO))
```

Si bien con todo lo anterior mencionado ya la interfaz valida, filtra, tokeniza, taggea y fragmenta, surge una observación durante el proceso de tageo, la cual está relacionada en sí con el orden de las palabras. Puede haber palabras que si se ven por sí solas no tienen un sentido o gran valor que aporten al tema, por ejemplo:

“red de área local”

Siguiendo el proceso que realiza la interfaz, se debería tagear de forma individual “red”,”de”,”área”,”local”, de esa forma máximo la palabra red se puede considerar que tenga un tag correspondiente dentro de los archivos de texto de entrenamiento, para esto se planteó usar un tokenizador diferente al que se empleaba en un principio, ya que ese es el que emplea el nltk por defecto. El tokenizador se denominda MWETokenizar, donde MWE significa multi-word expression, o en español, expresión de palabras-múltiples, dentro del cual se pueden armar conjuntos o tuplas de palabras que el tokenizador reconocerá como 1 sola expresión. Para que el tokenizador pueda reconocer las expresiones, se las deben agregar mediante una línea de programación, indicando entre paréntesis las palabras que formarán una expresión, por ejemplo:

“red de área local”

Tokenizador.add_mwe((‘red’,’de’,’área’,’local’))

Además que se las debe incluir en los archivos de textos usados para el entrenamiento del tagger, no sería de utilidad que se las añada al tokenizador si el tagger no puede identificarlos también, cuando se tienen que añadir expresiones con múltiples palabras al texto, se las debe agregar teniendo como separador el guión bajo (_), ese carácter es el que está por defecto, pero se lo puede cambiar. Para añadir “red de área local” se tendría que en el archivo agregarlo de la siguiente forma:

red_de_área_local/tema_sigla_red

Lo cual si es algo que puede ser analizado y procesado por la interfaz.

3.3 **PRUEBAS**

Para las pruebas de la interfaz, una vez que ya se realizó el proceso de entrenamiento con archivos de texto, lo que hizo es utilizar las preguntas que fueron generadas por los estudiantes al momento de hacer las encuestas para probar la interfaz, se tomaron solo las primeras 70 preguntas, que fueron obtenidas de los estudiantes actuales de la materia de Redes de Datos I, ya que estas preguntas en general las hicieron tomando en cuenta la materia, realizando consultas sobre la temática de Redes de Datos, en lo posible enfocándose al modelo TCP/IP.

Lo que se busca con las pruebas es ver el nivel de éxito de procesar/analizar las consultas, en base al desarrollo de la interfaz, una vez procesada la consulta, se pueden presentar dos escenarios:

- Éxito: la consulta fue procesada y se generó la matriz atributo/valor
- Fallo: la consulta no pudo ser procesada, lo cual puede ocurrir por cuatro razones:
    - Uso de caracteres especiales ({,},[,],etc.)
    - Errores ortográficos
    - Tema no relacionado a la temática
    - Texto está relacionado a la temática pero no forma parte del texto de entrenamiento.

Luego de haberse procesado las 70 preguntas de los estudiantes se presentaron los siguientes resultados:

- Éxito: 31 preguntas fueron procesadas correctamente
- Fallo: 39 preguntas no fueron procesadas por la interfaz, de las cuales:
    - Fallos por validación: 0
    - Fallos por errores ortográficos: 4
        - De esas cuatros preguntas, dos de ellas están relacionadas a la temática
    - Fallos por preguntas no relacionadas a la temática: 35
    - Fallos por texto aún no incorporado: 5

Sin considerar las siete preguntas que si están relacionadas pero no fueron procesadas por la interfaz, el porcentaje de éxito fue del 44,29%, que representa las 31 preguntas que fueron procesadas correctamente, teniendo con un 55,71% el porcentaje de fallo.

Si se toman las preguntas que se pueden corregir, el porcentaje de éxito aumentaría a un 54,28%, representando las 38 preguntas procesadas correctamente. La interfaz no se pudo probar utilizando el resto de preguntas debido a que esas fueron hechas por los estudiantes, tratando sobre cualquier temática, haciendo que no sea posible probar la interfaz, ya que habría un porcentaje de éxito de un 0%, debido a que todas las preguntas tratan de temáticas no relacionada a la enfocada en el proyecto.

Para asegurar un porcentaje de éxito alto, o reducir el porcentaje de posibles fallos, se debe entrenar en lo posible con el mayor número de textos relacionados con la temática. En la siguiente figura se puede ver el bloque valor/atributo generado por una de las consultas elaborada por los estudiantes, ya validada, filtrada, tokenizada, etiquetada y fragmentada, para tener como salida el bloque valor/atributo.

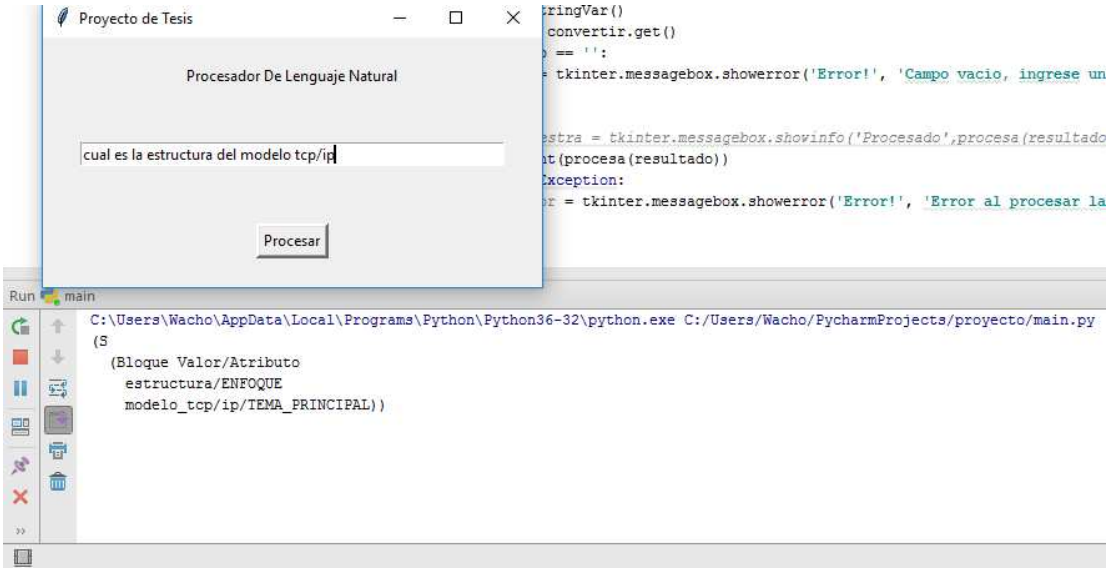
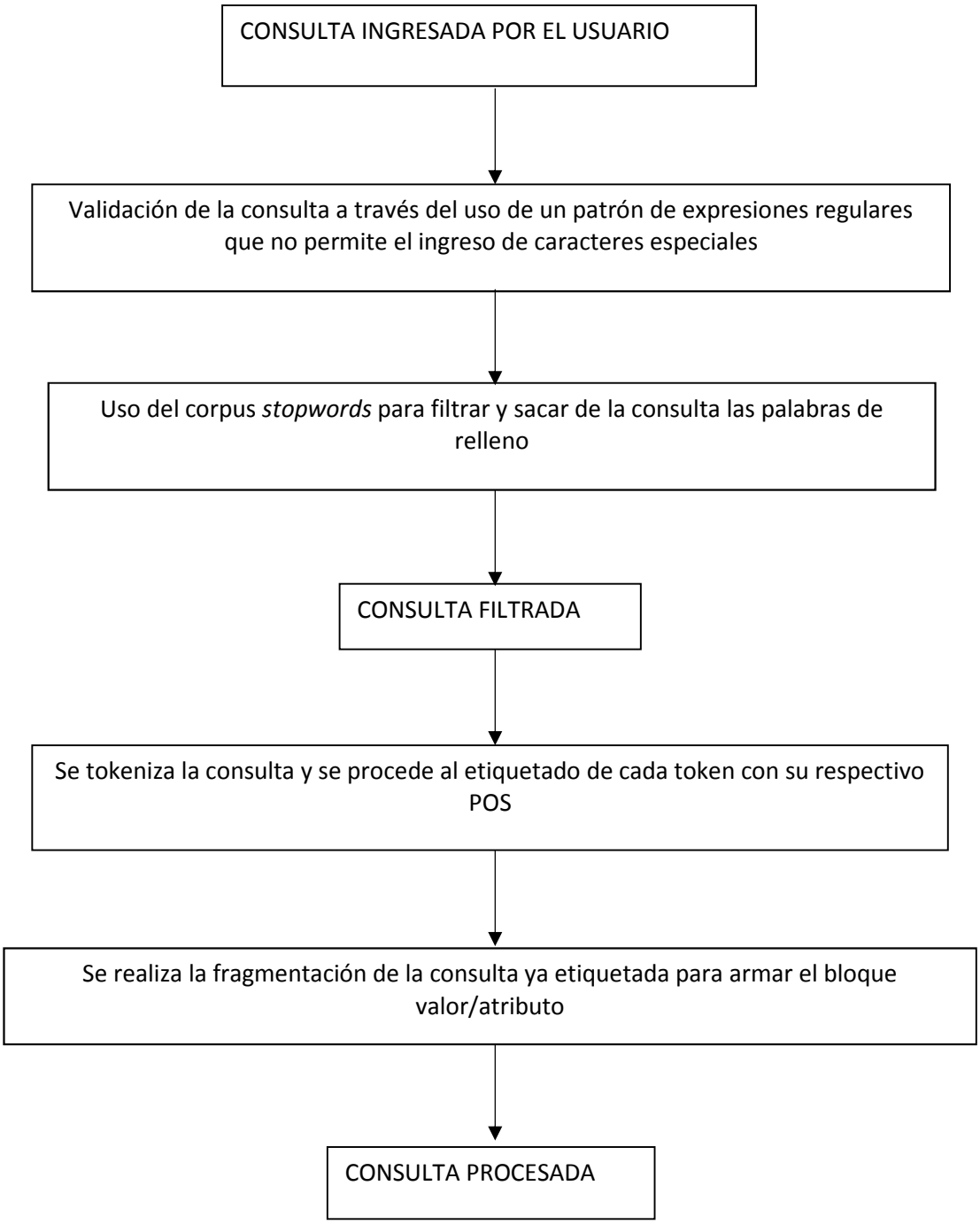
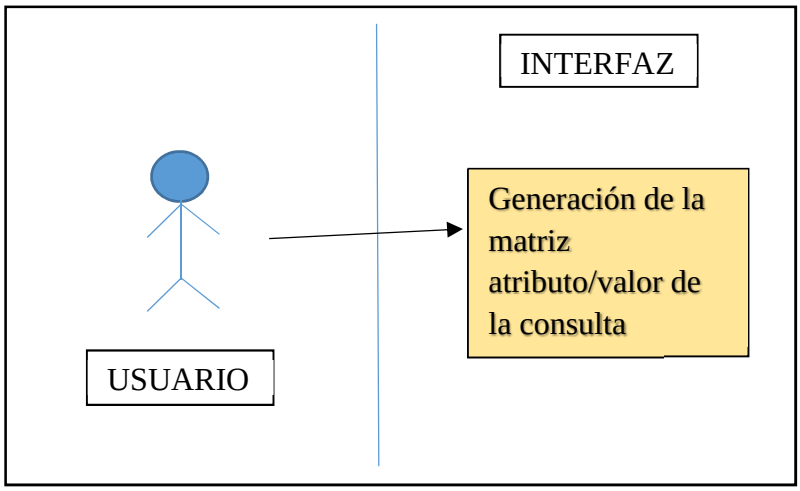


Figura 4: Consulta procesada utilizando la interfaz del proyecto

3.4 FLUJO DEL PROGRAMA



3.5 CASO DE USO



CASO DE USO DE LA INTERFAZ DE PROCESAMIENTO DE LENGUAJE NATURAL
Fuente: Elaboración Propia

Caso de Uso: CU01 – Proceso de Generación de la matriz atributo/valor de la consulta

Actor Principal: Usuario

Personal Involucrado e Intereses:

- El usuario desea realizar una consulta sobre un tema específico.

Suposiciones:

- El usuario realiza la consulta sobre la materia de redes, buscando solo lo relacionado con el modelo TCP/IP.

Precondiciones:

- El programa no aceptará consultas que traten sobre temas que no estén relacionados con el modelo TCP/IP.

Garantías de éxito:

- Se generó la matriz atributo/valor de la consulta hecha por el usuario.

Escenario principal de éxito (flujo básico):

1. El usuario ingresa su consulta en la interfaz.
2. El programa filtra las palabras que no agregan valor al análisis del texto.
3. El programa separa en *tokens* (unidades) cada elemento de la consulta.
4. El programa cataloga cada unidad con su respectivo tag, el cual indica a qué elemento pertenece dicha palabra o unidad.
5. El programa realiza la fragmentación de los *tokens*, para armar el bloque de atributo/valor solo con los atributos que son necesarios para la consulta.
6. El programa presenta al usuario la matriz atributo/valor de su consulta.

Extensiones (flujos alternativos):

1. El usuario ingresa una consulta sobre un tema no relacionado.
    a. El programa no procesará la consulta y mostrará un mensaje de error.
    b. El programa regresará a su estado inicial.

1. El usuario ingresa una consulta con caracteres especiales no admitidos por el programa.
    a. El programa no procesará la consulta y mostrará un mensaje de error.
    b. El programa regresará a su estado inicial.

CONCLUSIONES

En respuesta a los objetivos específicos establecidos para el desarrollo del proyecto, se han llegado a cuatro conclusiones:

En la actualidad existen herramientas que ya realizan el procesamiento de lenguaje natural, incluso hay las que van más allá e indican en dónde encontrar/extraer información, pero no hay una personalizada o enfocada a una sola temática, si bien este proyecto está enfocado a un tema, es adaptable a cualquier temática, con el debido entrenamiento del tagger y tokenizador usando el texto que uno necesite para el procesamiento y análisis.

La arquitectura de este proyecto pasa por un proceso el cual el usuario no percibe, solo ingresa la consulta y recibe como salida el bloque valor/atributo, pero detrás de eso la consulta pasa por un proceso de validación → filtro → tokenización → etiquetado → fragmentación.

Este proyecto por sí solo no realiza la búsqueda y extracción de información, solo el procesamiento de lenguaje natural, si bien se puede implementar en un ambiente educativo para que sea utilizado por los estudiantes/docentes, se debería de desarrollar un programa que sea capaz de recibir la salida que genera la interfaz de procesamiento para que con esa información realice la búsqueda y o extracción de documentos, etc. por medio de cualquier método que vaya a usar el programa, ya sea SQL, XML, etc.

La interfaz genera como salida una matriz valor/atributo que puede ser recibida por otro programa el cual a través de sentencias SQL, archivo JSON, XML, etc. Realice la búsqueda de información sobre la consulta realizada por el estudiante/docente.

RECOMENDACIONES

En base a lo realizado en el proyecto se han llegado a las siguientes conclusiones que pueden mejorar el mismo:

- Estandarizar los tags que son usados para las palabras, definir una estructura que le permita ser aplicada a cualquier temática.

- Aumentar la cantidad de textos/palabras usadas en el entrenamiento para tener de esta forma mayor contenido de la temática a analizar.

- Crear un lematizador en español que pueda utilizar el NLTK, ya que no hay recursos que faciliten uno para el idioma español para el lenguaje de programación Python.

- Investigar la forma de crear un propio modelo NER que permite crear clasificaciones propias para la temática o estandarizarlo para que sea aplicable a cualquier temática.

REFERENCIAS

Benítez, R., Escudero, G., Kanaan, S. & Masip, D. (2013). *Inteligencia Artificial Avanzada*. Barcelona, España: Editorial UOC

Casas, R. (2015). El dequeísmo: ¿un cambio en progreso de la sintaxis castellana? *Universidad Nacional Mayor de San Marcos*. 86(124), pp. 289-310

Chaudhary, B. (2015). *Tkinter GUI Application Development Blueprints*. Birmingham, Reino Unido: PACKT PUBLISHING.

Correa, M. (2015). Adaptación de una herramienta de procesamiento de lenguaje natural para el etiquetado de sentimiento y el análisis de lenguaje en español (tesis de pregrado). Universidad Técnica Particular de Loja, Loja, Ecuador.

Durán, M. (2014). Estructura sintáctico-pragmático del tema en el habla del español en Venezuela. *PARADIGMA*. 35(2), p. 107-127

Escandell, M. (2011). *Apuntes de Semántica léxica*. Madrid, España: UNED

Garcia, M. & Penny, M. (2013). *Gramática histórica de la lengua española: morfología*. Madrid, España: UNED

Graham, G. (2012). *El análisis de datos cualitativos en Investigación Cualitativa*. Madrid, España: Ediciones Morata.

Hardeniya, N. (2015). *NLTK Essentials*. Birmingham, Reino Unido: PACKT PUBLISHING.

Icart, M., Fuentelsaz, C. & Pulpón, A. (2006). *Elaboración y presentación de un proyecto de investigación y una tesina*. Barcelona, España: Ediciones de la Universitat de Barcelona.

Perkins, J. (2014). *Python 3 Text Processing with NLTK 3 Cookbook*. Birmingham, Reino Unido: PACKT PUBLISHING.

Nguendjo, I. (2013). Aspectos problemáticos de la enseñanza de la fonética española a estudiantes francófonos del oeste de Camerún. Íkala revista de lenguaje y cultura. 18, pp. 35-43

Periñán, C. (2012). En defensa del procesamiento del lenguaje natural fundamentado en la lingüística teórica. *ONOMÁZEIN*. (26), pp. 13-48

Rojas, M. (2015). Tipos de Investigación científica: Una simplificación de la complicada incoherente nomenclatura y clasificación. *REDVET*. 16(1), pp. 1-14

Turing, A. (1950). Computing Machinery and Intelligence. *Scientific American MIND*. 59 (236), 433-460.

Volpi, J. (2012). *No será la Tierra*. España: ALFAGUARA

Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python*. O'Reilly Media Inc.

GLOSARIO

- **Token**

  Un token es una palabra de texto, es considerada la unidad más simple de procesamiento por el NLTK.

- **Sentencia**

  Una sentencia es una secuencia ordenada de tokens, se puede decir que es una oración.

- **Part-of-Speech(POS)**

  El POS es la clasificación que le da el NLTK a cada palabra, ya sea que la palabra es un adjetivo, pronombre, sustantivo, etc.

- **POS Tagger**

  Es una rutina que está encargada de crear el diccionario con cada token y su respectivo POS.

- **Parser – Analizador**

  Una de las funciones del NLTK es el parser, el cual consiste elaborar el árbol de tokens con su respectivo POS a partir de una sentencia.

- **Etiquetado de POS**

  Consiste en identificar sentencias de texto y clasificar cada token con su respectivo POS.

- **Morfología**

  Se encarga de extraer los morfemos y raíces de cada token, y catalogarlos.

DECLARACIÓN Y AUTORIZACIÓN

Yo, **Álvarez Chancay, Rafael Abraham**, con C.C: # **0930564893** autor/a del trabajo de titulación: **"Diseño e Implementación de Interfaz de Procesamiento de Lenguaje Natural para Consultar Contenidos Académicos del Área de Redes de Datos Enfocados al Modelo TCP/IP"** previo a la obtención del título de **Ingeniero en Sistemas Computacionales** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, **21** de **Marzo** de **2017**

f. _____

Nombre: **Álvarez Chancay, Rafael Abraham**

C.C: **0930564893**

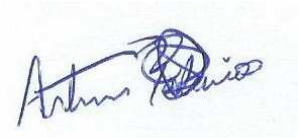
DECLARACIÓN Y AUTORIZACIÓN

Yo, **Palacios Wither, Arturo Emmanuel**, con C.C: # **0923522676** autor/a del trabajo de titulación: **"Diseño e Implementación de Interfaz de Procesamiento de Lenguaje Natural para Consultar Contenidos Académicos del Área de Redes de Datos Enfocados al Modelo TCP/IP"** previo a la obtención del título de **Ingeniero en Sistemas Computacionales** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, **21** de **Marzo** de **2017**



f. _____

Nombre: **Palacios Wither, Arturo Emmanuel**

C.C: **0923522676**

Presidencia de la República del Ecuador

Plan Nacional de Ciencia, Tecnología, Innovación y Saberes

SENESCYT
Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación

REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA			
FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN			
TÍTULO Y SUBTÍTULO:	Diseño e Implementación de Interfaz de Procesamiento de Lenguaje Natural para Consultar Contenidos Académicos del Área de Redes de Datos Enfocados al Modelo TCP/IP		
AUTOR(ES)	Álvarez Chancay, Rafael Abraham; Palacios Wither, Arturo Emmanuel		
REVISOR(ES)/TUTOR(ES)	Ing. Murillo Bajaña, Eduardo Wenceslao		
INSTITUCIÓN:	Universidad Católica de Santiago de Guayaquil		
FACULTAD:	Facultad de Ingeniería		
CARRERA:	Carrera de Ingeniería en Sistemas Computacionales		
TITULO OBTENIDO:	Ingeniero en Sistemas Computacionales		
FECHA DE PUBLICACIÓN:	21 de marzo del 2017	**No. DE PÁGINAS:**	55
ÁREAS TEMÁTICAS:	Utilización de Software Libre		
PALABRAS CLAVES/ KEYWORDS:	NLTK, TCP/IP, LENGUAJE NATURAL, INTERFAZ, NLP, MATRIZ, ATRIBUTO-VALOR, CONSULTAS		
RESUMEN/ABSTRACT El proyecto propuesto consiste en el desarrollo e implementación de una interfaz capaz de analizar las consultas realizadas por estudiantes sobre temas relacionados a Redes siendo más preciso al Modelo TCP/IP basándose en el tema de Procesamiento de Lenguaje Natural o NLP siglas que corresponden a "Natural Language Processing" para que de esta manera los estudiantes no tengan la necesidad de hacer uso de lenguajes técnicos. La interfaz está diseñada para que en ella puedan ingresar texto el cual después de un análisis y procesamiento por medio de la librería NLTK se llegue a elaborar una matriz atributo-valor solo con las palabras más relevantes para que los programas que implementen el algoritmo puedan realizar una extracción de información más exacta relacionado al tema consultado. Para poder llevar a cabo el proyecto se utilizó una metodología cualitativo, además que se procedió a pedirle a un grupo de estudiante que realizaran preguntas libremente las mismas que permitieron conocer las palabras más comunes a la hora de hacer una consulta.			
ADJUNTO PDF:	☒ SI	☐ NO	
CONTACTO CON AUTOR/ES:	**Teléfono:** 0982266030 / 0991059366	E-mail: rafa_alvarez_94@hotmail.com / reyarturo_palacios@hotmail.com	
CONTACTO CON LA INSTITUCIÓN (C00RDINADOR DEL PROCESO UTE)::	**Nombre: Valencia Macias, Lorgia del Pilar** **Teléfono:** +593-4-**2206950 ext 1020** **E-mail: lorgia.valencia@cu.ucsg.edu.ec**		
SECCIÓN PARA USO DE BIBLIOTECA			
Nº. DE REGISTRO (en base a datos):			
Nº. DE CLASIFICACIÓN:			
DIRECCIÓN URL (tesis en la web):			