Universidad Central “Marta Abreu” de las Villas.
Facultad Matemática Física y Computación
Ingeniería Informática

Trabajo de Diploma

Título: QtNLP-Diff: Comparador de casos de corpus lingüísticos y resultados de algoritmo de similitud.

Autor
Javier Sardiñas Morales

Tutores
Ing. Abel Meneses Abad
Lic. Yaisel Nuñez Arcia

Curso 2015- 2016

Declaración de autoría:

Título del documento:

Dado a los 25 días del mes de junio de 2016

Versión 1.0 publicada en junio del 2016

Copyright © 2016 UCLV, Abel Meneses Abad y Javier Sardiñas Morales

Licencia. Esta obra se publica bajo la licencia Creative Commons 3.0 BY-NC-SA que establece las siguientes condiciones:

Atribución: Debes reconocer y citar la obra de la forma especificada por el autor y el licenciante.

No Comercial: No se puede utilizar esta obra para fines comerciales.

Licenciar igual: Si se altera o transforma esta obra, o se genera una obra derivada, sólo se puede distribuir la obra generada bajo una licencia idéntica a esta.

Para ver una copia de esta licencia visitar la siguiente dirección en internet: http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode.

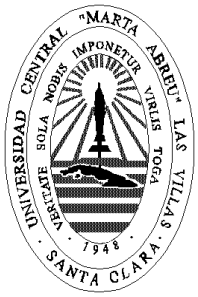
Marcas comerciales y marcas de servicios. Todas las marcas comerciales, marcas de servicios, logotipos y nombres de compañías mencionadas en esta obra son propiedad de sus respectivos dueños. Las mismas están protegidas bajo la ley de marcas comerciales y la ley de competencia desleal.

Derechos comerciales. Los derechos comerciales para exportación de esta obra son concedidos a la UCLV y los autores de forma exclusiva. En territorio cubano la utilización comercial de esta obra no debe interferir con el uso social de la misma. En caso de conflictos primará el derecho social sobre el comercial.

Autores: Ing. Abel Meneses Abad, Javier Sardiñas Morales

En caso de muerte los derechos morales de esta obra, su preservación, ejecución o modificación, quedarán bajo la responsabilidad moral y jurídica de:

Yanet Conde González, y Jane Meneses Conde



El que suscribe _Javier Sardiña Morales_____, hago constar que el trabajo titulado QtNLP-Diff: Comparador de casos de corpus lingüísticos y resultados de algoritmo de similitud fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de los estudios de la especialidad de __Ingeniería Informática_____, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Tutor Firma del Jefe del Laboratorio

Dedicatoria

A mi gran familia por su apoyo y comprensión en todo momento, especialmente a mis padres que han dado todo durante tantos años para que pudiera estudiar, graduarme y convertirme en un Ingeniero Informático.

Agradecimientos

A todas aquellas personas que han contribuido de una forma u otra al resultado final de este trabajo de diploma, muy especial a mi familia que son la base de todos mis esfuerzos y a los que les debo todo lo que soy hoy en día.

A mi novia quien me brinda su apoyo incondicional en los momento más difíciles y que nunca ha dejado de preocuparse por mí, y me demuestra su amor a diario, con cada gesto, palabra y acción.
A la familia de mi novia quienes me recibieron como un hijo, me brindaron su apoyo incondicional y soportaron durante todo este tiempo.

A mi tutores Abel Meneses Abad y Yaisel Nuñez Arcia por su apoyo permanente en todo momento.

A mi gran amigo Alexander Avello Silverio, quien nunca me dio la espalda y me dio la mano aun sabiendo que no recibiría nada a cambio.

A todas mis amistades por dedicarme parte de su tiempo, ayuda y apoyo incondicional.

Resumen

El presente trabajo propone el módulo QtNLP-Diff para la herramienta QtNLP, que permite estudiar los casos del XML de resultados de un algoritmo de detección de texto reusado, comparándolos con los del XML del corpus lingüístico original. El documento comprende la elaboración de un diseño de fácil uso por lingüistas con poco conocimiento en informática y por especialistas informáticos que trabajan en el área del Procesamiento del Lenguaje Natural (NLP) en las universidades cubanas. La comparación en pares de casos, provenientes de los XML del corpus y los resultados del algoritmo, es actualmente un proceso complejo e ineficiente. El módulo implementado facilita la caracterización de casos, para la validación de nuevos algoritmos de similitud. Para la gestión y análisis de la herramienta propuesta se utiliza la metodología ágil SXP, un híbrido de SCRUM y XP, poniendo énfasis en el diseño de la arquitectura del software, y la validación mediante pruebas de aceptación. Los resultados muestran que, con la utilización del corpus TNLP y resultados de algoritmos en idioma español, la herramienta permite a los investigadores construir fácilmente nuevos XMLs con casos de interés posteriormente a la revisión de casos.

Palabras claves: QtNLP, texto reusado, corpus lingüístico, similitud, SXP

Abstract

The work involves the development of QtNLP-Diff module for QtNLP tool, which allows to study cases XML files containing results of a detection algorithm applied to reused text with XML containing the original linguistic corpus. The document includes the development of a user-friendly design to facilitate the work of linguists with little knowledge in computer science and computer specialists working in the area of Natural Language Processing (NLP) in Cuban universities. The pairwise comparison of cases from the XML corpus and the results of the algorithm is currently a complex and inefficient process. The implemented module facilitates the characterization of cases for validation of new similarity algorithms. The SXP agile methodology was used for the management and analysis of the proposed tool. This methodology is a hybrid of SCRUM and XP, with an emphasis on the design of software architecture and validation through acceptance testing. The results show that use of the corpus TNLP and the results of Spanish language algorithms allow researchers to build new XMLs cases of interest easily, after reviewing cases.

Keywords: QtNLP, re-used text, linguistic corpus, similarity, SXP

Tabla de contenido

Lista de Figuras

Lista de Tablas

Introducción

La comparación es un proceso lógico que hace el ser humano a fin de identificar, mediante un análisis sensorial, los diferentes aspectos que se relacionan o no entre dos o varios objetos. Su principal fundamento consiste en detallar las semejanzas o diferencias que presenten elementos con cierto símil, ya que resulta ilógico realizar una comparación entre dos cosas que no tengan nada en común (Venemedia, 2015).

En el Centro de Estudios de Informática (CEI) de la Universidad Central "Marta Abreu" de Las Villas (UCLV), se realizan trabajos de investigación y desarrollo en esta línea. En este centro existe una investigación de conjunto con la Universidad de Camagüey sobre el procesamiento del español. En el marco de dicho convenio se creó la aplicación QtNLP, la cual se utiliza para el trabajo con corpus lingüísticos, y tiene como objetivo la creación, edición y análisis de corpus en español para tareas de Procesamiento de Lenguaje Natural (NLP). Es una aplicación fácil de usar por lingüistas con poco conocimiento de informática y también por especialistas informáticos que investigan en el área de NLP (Meneses and Videaux, 2015).

Esta aplicación carece de un módulo para comparar casos de corpus lingüísticos con casos de resultados de algoritmo de similitud textual, donde un caso se refiere a un par de fragmentos texto contenidos en dos documentos del corpus, definidos por su posición de inicio y longitud. En aplicaciones como la detección de plagios, los casos de corpus y de resultados de algoritmos de similitud a manejar para los lingüistas e investigadores que trabajan en esta área son de grandes dimensiones. Además, la caracterización de casos no detectados por nuevos algoritmos en un conjunto de datos voluminoso es compleja e ineficiente. Por consiguiente en el presente trabajo se plantean los objetivos siguientes.

Objetivo General

Desarrollar un módulo utilizando Python y Qt para la herramienta QtNLP que permita comparar casos de corpus lingüísticos con casos de resultados de algoritmos de detección de similitud textual.

Objetivos específicos:

1. Analizar el estado actual y tendencias de las herramientas de comparación de textos.
2. Diseñar el módulo QtNLP-Diff para la aplicación QtNLP.

3. Implementar el módulo QtNLP-Diff para la aplicación QtNLP.
4. Validar la solución implementada a través de pruebas de software.

Para guiar el desarrollo del trabajo se plantean la siguiente pregunta **de investigación:**
¿Cómo extender la aplicación QtNLP con la incorporación de un módulo que favorezca, la comparación de resultados de corpus lingüísticos con los resultados de algoritmos de similitud textual?

Justificación:
La detección de plagio es un área de investigación abierto en la actualidad. Uno de los sub-problemas en los que ha sido descompuesto es la alineación de textos. Esta no es más que la detección de similitud entre dos textos, ya sean copiados literalmente, parafraseados, entre otros. Los corpus lingüísticos dedicados a la detección de similitud textual son de grandes dimensiones. Algunos como el PAN-PC-12 contienen más de 15000 casos. Los resultados de los algoritmos que intentan solucionar este problema generan, a su vez, datos de la misma dimensión. Para los investigadores de este campo el estudio de varias centenas de casos no detectados entre varios miles existentes en el corpus, útiles para la validación de nuevos algoritmos, es un proceso necesario pero complejo e ineficiente. Por tanto, se necesita una solución que permita evaluar un caso en el menor tiempo posible, y facilite la obtención del estudio completo de casos para diseñar nuevos algoritmos de similitud que detecten estos casos y su posterior validación.

Hipótesis
El desarrollo del módulo para la herramienta QtNLP permitirá comparar los resultados de corpus lingüísticos con los resultados de algoritmos de detección de similitud textual de forma rápida y eficiente.

El presente trabajo está estructurado de acuerdo a la siguiente secuencia lógica: introducción, cuatro capítulos, conclusiones, recomendaciones, bibliografía.
El capítulo uno aborda descriptivamente los sustentos y fundamentos teóricos de la investigación. Se analizan aspectos del procesamiento del lenguaje natural, las herramientas gráficas y de consola para la comparación de textos, las bibliotecas de procesamientos XML y las características de la herramienta QtNLP.

El capítulo dos describe la metodología SXP para la gestión y panificación del proyecto. Se analiza el proceso del negocio y se proponen los diagramas en la forma AS-IS y TO-BE que lo representan. Contiene, además, los artefactos del diseño como son la pila del producto, el plan de iteraciones, las historias de usuarios, y los requisitos no funcionales.

El capítulo tres describe la arquitectura general del sistema. Se generan los diagramas de los documentos XML, de paquete, de componentes, y los prototipos de interfaz. Describe, además, la arquitectura de información de QtNLP-Diff.

El capítulo cuatro muestra las pruebas de aceptación realizadas al módulo QtNLP-Diff, utilizando las interfaces diseñadas.

Capítulo 1. Fundamentación Teórica.

Capítulo 1. FUNDAMENTACIÓN TEÓRICA.

En el presente capítulo se abordan aspectos generales sobre la similitud textual y el corpus lingüístico dentro del Procesamiento del Lenguaje Natural. Se realiza un análisis crítico sobre las diferentes herramientas para la comparación de textos gráficas y de consola. También se valoran las bibliotecas de procesamientos de XML, se describe la aplicación QtNLP mediante sus principales características.

1.1 Procesamiento del Lenguaje Natural en la lengua española.

La aplicación del Procesamiento de Lenguaje Natural más obvio, y quizá más importante en el momento actual, es la búsqueda de información. Por un lado, en Internet y en las bibliotecas digitales se almacena un gran cúmulo de conocimiento que puede dar respuestas a muchísimas preguntas que tenemos. Por otro lado, en la sociedad actual no se sabe discernir entre la información real o válida, ya que muchas veces no tenemos los medios o herramientas para encontrarla. Hoy en día la pregunta ya no es "¿se sabe cómo...?" sino "ciertamente se sabe, pero ¿dónde está esta información? (Carbonell, 1992).

Técnicamente, rara vez se trata de decidir cuáles son relevantes para la petición del usuario y cuáles no. Usualmente, una enorme cantidad de documentos se puede considerar como relevantes en cierto grado, siendo unos más relevantes y otros menos. Entonces, la tarea se entiende cómo medir el grado de esta relevancia para proporcionar al usuario primero el documento más relevante; si no le sirvió, el segundo más relevante, etc.(Carbonell, 1992).

1.1.1 Similitud textual.

En el presente trabajo se tuvo en cuenta el criterio de similitud textual siguiente:

"La similitud textual es una condición o propiedad que mide el grado de semejanza entre dos o más textos. Si bien la mayoría de los estudios de PLN están enfocados en el plagio y paráfrasis, existen otros ámbitos no muy explorados, como la adaptación, parodia o traducción. A la vez, es importante considerar que la similitud textual puede darse entre cualquier tipo de documentos, o bien por coincidencias entre los textos debido a cuestiones temáticas o de estilo y no como producto de un plagio. De esta manera, la detección de similitud textual debe ir más allá de la detección automática de

plagio y de paráfrasis. La plática presenta una visión amplia del concepto de similitud textual, las aplicaciones en tecnologías de lenguaje que resultan de medir la semejanza de documentos, la necesidad de crear un corpus que cubra distintos aspectos para la detección de similitud textual y algunos métodos que existen para detectarla" (Carmona, 2014).

1.1.2 Algoritmos de similitud textual.

Los algoritmos se similitud textual son usados principalmente para comparar secuencias biológicas como el ADN. Varios autores han propuesto adecuar estos algoritmos para que también, de forma interna, puedan hacer comparaciones semánticas entre los elementos de cada par de secuencias y así sean capaces de comparar textos (Carmona, 2014, Rosso, 2012, Stein, 2005).

Existen varios tipos de algoritmos como:

Algoritmo **Needleman-Wunsch** que sirve para realizar alineamientos globales de dos secuencias. Se suele utilizar en el ámbito de la bioinformática para alinear secuencias de proteínas o de ácidos nucleicos. Fue propuesto por primera vez en 1970, por Saul Needleman y Christian Wunsch. Se trata de un ejemplo típico de programación dinámica. El algoritmo funciona del mismo modo independientemente de la complejidad o longitud de las secuencias y garantiza la obtención del mejor alineamiento (Needleman and Wunsch, 1970).

Algoritmo de **Smith-Waterman** es una reconocida estrategia para realizar alineamiento local de secuencias biológicas (ADN, ARN o proteínas); es decir que determina regiones similares entre un par de secuencias. Este fue propuesto por Temple Smith y Michael Waterman en 1981. Está basado en el uso de algoritmos de programación dinámica, de tal forma que tiene la deseable propiedad de garantizar que el alineamiento local encontrado es óptimo con respecto a un determinado sistema de puntajes que se use (tales como matrices de substitución) (Smith and Waterman, 1981).

1.1.3 Corpus Lingüísticos.

Un corpus lingüístico es un conjunto amplio y estructurado de ejemplos reales del uso de la lengua. Estos ejemplos pueden ser textos (lo más común) o muestras orales (generalmente transcritas). Un corpus lingüístico es un conjunto de textos relativamente grande, creado independientemente de sus posibles fines de uso. Es decir, en cuanto a su

estructura, variedad y complejidad, un corpus debe reflejar una lengua o su modalidad de la forma más exacta posible y en cuanto a su uso, preocuparse de que su representación sea real. Los corpus tienen similitudes con los textos porque están compuestos por ellos; por otro lado, no son textos en sí, porque, a diferencia de los mismos, no tiene sentido analizarlos en su totalidad. Un texto tiene un principio y un fin y es, en mayor o menor grado, cohesivo y coherente, mientras que un corpus carece de tales características por no poseer una estructura, sino sólo una composición. Por esta razón conviene analizar un corpus recurriendo a herramientas y metodología propias (Mikelionienė, 2002).

Existen varios tipos de corpus como:

- **THE PAN PC-09:**

  Es una colección de 41 223 documentos en que 94 202 casos de plagio artificiales han sido insertados, es el primer cuerpo que tiene en cuenta las evaluaciones a gran escala de ambos métodos de detección de plagio internos y externos. Durante su construcción varios parámetros han sido variados de modo que las características de corte trasversal de los casos de plagio difieren (Martin Potthast, 2010).

- **Webis-CPC-11:**

  Webis-CPC-11. Este cuerpo comprende las paráfrasis a nivel de paso con 4 067 muestras positivas y 3 792 muestras negativas, usando el Amazon's *Mechanical* Turk para la multitud origen. Las contribuciones empíricas incluyen máquinas aprendiendo experimentos para explorar si las paráfrasis a nivel de paso se pueden identificar en un problema de clasificación de dos clases usando características de similitud de paráfrasis (Martin Potthast, 2012).

1.2 Herramientas de consola para comparaciones de textos.

Existen varias herramientas de consola para comparaciones de textos las cuales son utilizadas por varios desarrolladores, dentro de las que sobresalen SciPy, NumPy y Matplotlib.

1.2.1 SciPy.

Es una biblioteca de código abierto con herramientas y algoritmos matemáticos para Python, es distribuida bajo la licencia de software libre permisiva BSD (*Berkeley Software Distribution*) y su desarrollo se patrocina y sustentada por una comunidad de

desarrolladores. Contiene módulos para optimización, álgebra lineal, integración, interpolación, procesamiento de señales y de imagen, funciones especiales y otras tareas para la ciencia e ingeniería. Está dirigida al mismo tipo de usuarios que los de aplicaciones como MATLAB, GNU Octave, y Scilab. La estructura de datos básica usada por SciPy es en orden multidimensional proveído por el módulo de NumPy (community, 2014b).

Esta biblioteca contiene el módulo ndimage que se utiliza en los algoritmos de comparación del tipo DotPlot para visualizar las coincidencias de oraciones entre dos documentos. La dificultad de esta funcionalidad es que se genera una imagen que no se puede conectar a los documentos para visualizar las regiones de interés. Se necesita una estructura adicional que enlace los fragmentos con las regiones en la imagen, posibilidad no incluida en scipy.

1.2.2 NumPy.

Es una extensión de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices. El ancestro de NumPy, Numeric, fue creado originalmente por Jim Hugunin con algunas contribuciones de otros desarrolladores (community, 2014a).

Usar NumPy en Python da la funcionalidad comparable a MATLAB, permiten al usuario escribir rápidamente programas mientras la mayor parte de las operaciones trabajan con arreglos o matrices en lugar de la magnitud escalar (community, 2014a).

Se utiliza en algunos de los modelos de representación como TF-IDF (*Term Frequency – Inverse Document Frequency*), VSM, LSM que reducen el texto a representaciones de frecuencias de términos. Y estos modelos a su vez contienen funciones que determinan la similitud entre dos textos, generando un arreglo con los fragmentos encontrados. Sin embargo, no existe forma sencilla de visualizarlos, y se deben construir estructuras adicionales que enlacen el resultado al documento original.

1.2.3 Matplotlib.

Matplotlib es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python y su extensión matemática NumPy. Proporciona una interfaz de programación de aplicaciones orientada a objeto para incrustar lotes en aplicaciones usando juegos de herramientas de GUI de propósito general como wxPython, Qt, o GTK+. Varios juegos de herramientas son disponibles

que extienden la funcionalidad de Matplotlib. Escrita originalmente por John D. Hunter, distribuida bajo una licencia de estilo de BSD. Michael Droettboom se convirtió en el desarrollador principal de Matplotlib poco antes de la muerte de John (The matplotlib, 2014).

Al igual que SciPy esta herramienta permite visualizar a través de imágenes. Los resultados la comparación, representando las oraciones o los documentos dentro de un corpus como puntos de una imagen. Sin embargo, presenta la misma dificultad que scipy pues estos resultados no se enlazan directamente con los documentos comparados, y por lo tanto el desarrollador necesita construir una estructura nueva y visualizarla de una manera distinta que puede ser web, y muestra las regiones de interés vinculadas a los documentos reales.

1.3 Herramientas gráficas de comparaciones de textos.

Para comparar diferencias entre textos existen varias herramientas gráficas las cuales pistan los cambios que se le realizan a los mismos y son utilizadas por muchos desarrolladores, dentro de las que sobresalen KDiff, Bazaar y Meld.

1.3.1 KDiff.

KDiff es una herramienta gráfica que tiene como objetivos permitir a usuarios comparar dos archivos de texto o dos directorios diferentes. Muestra los dos archivos comparados mutuamente, de cierta manera esas líneas correspondientes están situadas en una posición conveniente, independiente de la posición de la barra de desplazamiento. Las líneas diferentes en los archivos comparados se resaltan en ambas vistas de archivo por tres colores diferentes (Joachim, 2014).

KDiff es parte de la compilación de software de KDE y por lo tanto primariamente usado en Linux y otros sistemas operativos parecidos a UNIX, así como Microsoft Windows. KDiff no computa en realidad las diferencias entre los archivos comparados, pero es solamente una postura gráfica a la utilidad de interfaz de línea de comandos (Joachim, 2014).

Ventajas que proporciona KDiff:

- combina dos o tres archivos de entrada de texto o directorios
- soporta Unicode, UTF-8 y otros codificadores, auto detección vía byte-order-mark.
- Impresión de diferencias.

- Manual de alineación de líneas.
- Automática fusión de la historia de control de versión e intuitiva interfaz de usuario.
- Cuenta con un menú de Plugin.
- Integración simplificada con IBM-Rational-Clearcase para Windows (Joachim, 2014).

1.3.2 Bazaar.

Bazaar es una herramienta de software libre con una comunidad grande de contribuyentes para ayudar a personas a trabajar en los equipos. Muestra los cambios que otras personas hacen a un grupo de archivos tal como código fuente de software para darle detalles de cada paso de su evolución. Bazaar es una controladora de versión de sistemas y ha sido durante mucho tiempo popular para los desarrolladores de software. La comodidad de uso, flexibilidad y la simple instalación lo hacen ideal no solo para los desarrolladores de software sino también para otros grupos que trabajen en conjunto de documentos, tales como escritores técnicos, diseñadores y traductores web (Canonical, 2011).

Ventajas que proporciona Bazaar:

- Brinda copia de seguridad de versiones antiguas.
- No hay que levantar ningún servidor.
- Tiene combinación inteligente lo que combinar cambios muchas veces no es un tormento.
- Creación de ramas y combinación de las mismas mucho mejor.
- Permite renombrar mejor.
- Organización del trabajo más fácil - se pueden desarrollar cambios separados en sus propias ramas.
- Los desarrolladores pueden combinar entre sí sus cambios en sus ramas personales cuando están trabajando en algo común.

1.3.3 Meld.

Meld es un comparador visual y al mismo tiempo una herramienta para los desarrolladores que ayuda a comparar archivos, directorios, y versiones controladas de proyectos. Proporciona dos y tres modos de comparar ambos archivos y directorios de

manera visual con coloración de código de diferentes líneas, y es soportado para muchos sistemas de control de versiones populares ya que tiene la capacidad de trabajar con diferencias entre versiones. Ayuda a repasar cambios de código. Es un software libre y de código abierto sujeto a los términos de le GPL GNU (General Public License), aparecen generalmente como un paquete en la mayor parte de las distribuciones de GNU/Linux (Willadsen, 2012).

Ventajas que proporciona Meld:

- Aplicación con una interfaz sencilla.
- Se pueden comparar dos o tres archivos.
- En el menú de 'Cambios' puede mezclarse rápida y fácilmente los archivos o carpetas para que queden iguales.
- Tiene configuraciones para filtrar contenido que no nos interese comparar usando expresiones regulares.
- Comúnmente se utiliza de conjunto con otras aplicaciones como editores de código, IDEs de programación y Administradores de archivos (Willadsen, 2012).

1.4 Bibliotecas de procesamiento XML.

Para el manejo de archivos en XML existen varias bibliotecas de procesamiento XML para Python las cuales son buenas herramientas para desarrolladores.

1.4.1 Xmltramp

Desarrollada por Aaron Swartz, es una herramienta para analizar documentos XML en una estructura de datos muy amigable para Python. La cual no cumple con la definición de una herramienta de enlace de datos; es decir, que no es un sistema que representa elementos y atributos del documento XML como objetos personalizados. Xmltramp es más como elementtree, conjunto de objetos ligeros que hacen que la información en el documento XML sea accesible a través de expresiones propias de Python. El objetivo declarado de Xmltramp es la simplicidad en lugar de una cobertura exhaustiva de funciones de XML (Uche, 2003).

1.4.2 Pxdom

Pxdom es una implementación de nivel 3 de W3C DOM para XML 1.0/1.1 con o sin espacios de nombre, usando Python y OMG-estilos. Todas las características descritas en el Core y recomendaciones LS son sostenidas, con las siguientes excepciones:

- validación;
- asincrónicos LS Parsers;
- verifica las características del nombre rigurosamente para XML 1.1.

Pxdom puede incluirse e importar también como un sub-módulo de otro paquete. Esta es una estrategia buena si desea distribuir una aplicación con base en DOM sin tener que preocuparse sobre las versiones del Python o PyXML instaladas en las máquinas de usuarios; las únicas dependencias son la manipulación de cadena de biblioteca estándar y los módulos relacionados con la URL (Clover, 2008).

1.4.3 PyRXP

PyRXP es una capa alrededor del lenguaje Python, excelente analizador de RXP. RXP es un espacio de nombres de validación-consciente escrito en C. En conjunto, proporcionan el marco XML de análisis más rápido disponible para Python. RXP fue escrito por Richard Tobin (ReportLab, 2012).

Los componentes generales son:

- Una representación estándar en memoria de un documento XML.
- Una herramienta de transformación ligera mejor que XSLT.

En general, posibilita obtener toda la estructura de un documento XML en la memoria en el menor tiempo posible (ReportLab, 2012).

1.5 QtNLP.

QtNLP es una aplicación de escritorio del tipo Front-End para el trabajo con corpus lingüísticos. Está desarrollada en Qt y Python. Tiene como objetivo la creación, edición y análisis de corpus en español para tareas de Procesamiento del Lenguaje Natural (NLP), fáciles de usar por lingüistas con poco conocimiento de informática; y también por especialistas informáticos que investigan en el área de (NLP). Basa su arquitectura en un modelo de Plugins (carpeta **modules**) que facilita su desarrollo desde funciones

básicas del procesamiento de las lenguas naturales, así como las interfaces simples para los lingüistas (Meneses, 2014).

QtNLP posee un expediente de proyecto (basado en la metodología SXP), donde se encuentra documentado todo el proceso de desarrollo. Así mismo cada función de código es documentada con docstrings y se incluyen las ayudas a estas funciones autogeneradas dentro de la documentación con Sphinx (Meneses, 2014).

Características:

- Gracias a la documentación de los códigos un nuevo desarrollador puede leer cada función programada en QtNLP.
- Cada clase o método programado podrá ser leído en esta documentación, en el momento necesario.
- El diseño de una estructura personalizada a la documentación de la herramienta hecha con Sphinx permite al desarrollador leer funciones de una en una para comprender sus relaciones a medida que las necesita.
- En su concepción planifica la inserción de un módulo QtNLP-Diff capaz de comparar casos de corpus con casos de resultados de algoritmos de similitud textual (Meneses, 2014)

1.6 Valoración personal de las tecnologías estudiadas.

Al realizar un análisis de las disimiles herramientas gráficas y de consola de comparación de textos, se puede afirmar que permiten hacer gran parte del proceso de comparación, pero no son de fácil uso para el usuario, estas no se adaptan al proceso de QtNLP al no cumplir con los requisitos para lograr comparaciones de casos de corpus con resultados de casos de algoritmos, pues estas herramientas comparan todo el documento, cuando realmente lo que se desea comparar son fragmentos de textos escogido por atributos, y además, no son capaces de comparan cuatro fragmentos de textos simultáneamente. Los archivos que almacenan los casos de corpus y los casos de resultados de algoritmos se encuentran en formato XML, Python posee tecnologías maduras para el manejo del lenguaje XML, una de las herramientas más utilizadas es Pxdom, ya que el usuario que trabaja con esta biblioteca no tiene que preocuparse sobre las versiones de Python y existe bibliografía complementaria sobre los manuales de uso de la herramienta.

1.7 Conclusiones parciales.

El capítulo uno aborda las bases conceptuales de la investigación y la fundamentación de la propuesta. Se enuncian los aspectos teóricos relativos al Procesamiento del Lenguaje Natural Español con el objetivo de argumentar las bases del funcionamiento de este. Se referencian herramientas gráficas y de consola para comparaciones de texto afirmando que permiten hacer gran parte del proceso de comparación, pero no son de fácil uso para el usuario, estas no se adaptan al proceso de QtNLP al no cumplir con los requisitos para lograr comparaciones de casos de corpus con casos de resultados de algoritmos. Se analizan las librerías que trabajan con XML ya que los casos de corpus y algoritmos se encuentran en este formato. El estudio de las características de la aplicación QtNLP permitió afirmar que este posee una interfaz de fácil uso por parte del usuario lo que la convierte en una herramienta de usabilidad.

Capítulo 2. Análisis del Sistema con SXP.

Capítulo 2. ANÁLISIS DEL SISTEMA CON SXP.

En el presente capítulo hace referencia a la de metodología SXP la cual se utiliza para la gestión y panificación del proyecto. Se describe el modelo de negocio junto a los diagramas de procesos de negocios en la forma AS-IS y TO-BE. Se generan los requisitos del sistema plasmándolos en la pila del producto, se conforma el plan de iteraciones, se realizan las historias de usuarios y los requisitos no funcionales.

2.1 Metodología de desarrollo SXP.

Todo desarrollo de software es riesgoso y difícil de controlar pero, si no se utiliza una metodología para guiar el proceso, lo que se obtiene son clientes insatisfechos con el resultado y desarrolladores aún más inconformes. Muchas veces no se toma en cuenta el utilizar una metodología adecuada, sobre todo cuando se trata de proyectos pequeños. Lo que se hace con este tipo de proyectos es separar rápidamente el aplicativo en procesos, cada proceso en funciones, y por cada función determinar un tiempo aproximado de desarrollo (Peñalver, 2008).

Cuando los proyectos que se van a desarrollar son de mayor envergadura, ahí sí toma sentido el basarse en una metodología de desarrollo, y empezar a buscar cuál sería la más apropiada para el desarrollo del software (Peñalver, 2008).

Las metodologías ágiles forman parte del movimiento de desarrollo ágil de software conocidos anteriormente como metodologías livianas, que se basan en la adaptabilidad de cualquier cambio como medio para aumentar las posibilidades de éxito de un proyecto. Se le denomina ágil por la habilidad de responder de forma versátil al cambio para maximizar los beneficios. Intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados (Peñalver, 2008).

2.1.1 Principios de las metodologías ágiles:

Realizar entregas cortas en el tiempo y continuas.

Adaptación a nuevos cambios durante el proceso de desarrollo.

Entregas periódicas y frecuentes que funcionen.

Los clientes forman parte del equipo de desarrollo.

Equipo con individuos motivados en un ambiente de confianza y seguridad.

La comunicación directa es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo. Se deben evitar el teléfono, los correos electrónicos, fax, etc.

La medida principal de progreso es el software que funciona.

Desarrollo sostenible. Es indispensable que exista paz y armonía en el equipo para que el proyecto tenga éxito.

Buen diseño y calidad técnica.

La simplicidad es algo básico.

Equipos auto-organizados.

El equipo debe realizar reflexiones periódicamente para plantearse cómo llegar a ser más efectivo (Peñalver, 2008).

Entre las más conocidas están:

- XP (Extreme Programming).
- SCRUM.
- Crystal.
- OpenUP
- ASD

2.1.2 SXP (híbrido cubano).

"SXP es una metodología compuesta por las metodologías SCRUM y XP que ofrece una estrategia tecnológica a partir de la introducción de procedimientos ágiles, que permitan actualizar los procesos de software para el mejoramiento de la actividad productiva, fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo y ayudando al líder del proyecto a tener un mejor control del mismo. SCRUM es una forma de gestionar un equipo de manera que trabaje de forma eficiente y de tener siempre medidos los progresos, tal que sepamos por dónde andamos. XP más bien es una metodología encaminada para el desarrollo; consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto" (Peñalver, 2008).

Consta de 5 fases principales:

- Investigación: flujo de trabajo que establece la caracterización de tecnologías afines con el proyecto.
- Planificación-Definición: es donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.
- Desarrollo: es donde se realiza la implementación del sistema hasta que esté listo para ser entregado.
- Entrega, puesta en marcha.
- Mantenimiento, donde se realiza el soporte para el cliente (Peñalver, 2008).

De cada una de estas fases se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la Lista del Producto, definición de las Historias de Usuario, diseño, implementación, pruebas, entre otras; de donde se generan artefactos para documentar todo el proceso. Las entregas son frecuentes, y existe una refactorización continua, lo que permite mejorar el diseño cada vez que se le añada una nueva funcionalidad (Peñalver, 2008).

SXP está especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad (Peñalver, 2008).

2.2 Principales roles (integrantes del equipo).

- Product Owner (Jefe de Proyecto): Javier Sardinas Morales

  Descripción: asegura de que el equipo Scrum trabaje de forma adecuada desde la perspectiva del negocio y escribe historias de usuario, las prioriza, y las coloca en la pila del producto.

- Interesados (clientes): Abel Meneses Abad y Manuel Llanes

  Descripción: Se refiere a la persona o entidad que hace posible el proyecto y para quienes el proyecto producirá el beneficio acordado que justifica su producción.

- Scrum Master (Scrum): Abel Meneses Abad

  Descripción: elimina los obstáculos que impiden que el equipo alcance el objetivo del sprint. Él no es el líder del equipo (porque ellos se auto-organizan), sino que actúa como una protección entre el equipo y cualquier influencia.

- Equipo de Trabajo (Javier, Alex, Abel, Llanes)

Descripción: tiene la responsabilidad de entregar el producto, es un equipo (3 a 9 personas) con las habilidades transversales necesarias para realizar el trabajo.

2.3 Modelado del Negocio Actual.

El proceso de comparar los casos de corpus lingüísticos con los resultados de algoritmos de detección de similitud textual, se realiza seleccionado dos archivos con formatos XML. Donde uno de estos archivos contiene resultados de casos de algoritmos y el otro, resultados de casos de corpus, donde el lingüista tiene que abrir dos editores de textos con el caso a comparar, tomando los atributos de las posiciones de los textos del algoritmo y compararlos con los atributos de las posiciones del corpus. Esto se realiza con un script de consola que compara los textos para detectar los nuevos tipos de casos. Debido a las grandes dimensiones de dichos conjuntos de datos este proceso resulta complejo e ineficiente pues toman demasiado tiempo compararlas y se requiere de habilidades computacionales.

2.3.1 IDEF0 (Definición de integración para modelado de función).

IDEF0 es una metodología, basada en SADT (*Structured Analysis and Design Technique*) que pretende representar de manera estructurada y jerárquica las actividades que conforman un sistema y los objetos o datos que soportan la interacción de esas actividades. Facilita el trabajo en situaciones de mayor complejidad y mayores exigencias en cuanto al tratamiento y nos va a guiar en la descripción de un proceso (función o actividad) que es considerado como la combinación de cinco unidades básicas que interactúan (Varun Grover, 2000).

2.3.2 Diagrama del Proceso de negocio As-Is.

En la Figura 2.1 se muestra el diagrama de Proceso de negocio en su forma As-Is y como el cliente realiza este proceso antes de la incorporación del módulo QtNLP-Diff.

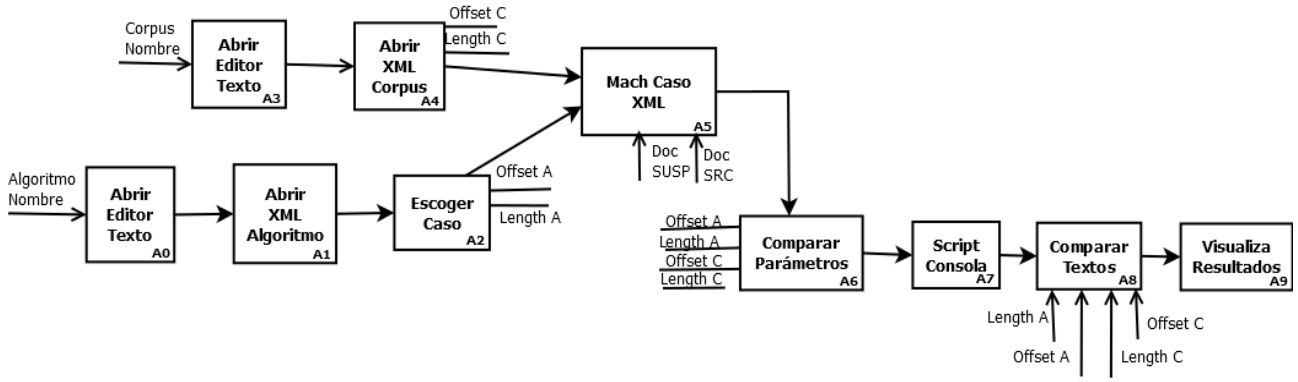


Figura 2.1 Proceso de Negocio forma As-Is

Descripción del diagrama de Proceso de negocio en la forma As-Is:

El lingüista abre un editor de texto, abre el XML del algoritmo y escoge un caso. Luego abre otro editor de texto, abre el XML del corpus y se selecciona el caso similar igualando los casos. Después se comparan los parámetros de los casos seleccionados mediante un Script de consola, se comparan los fragmentos de textos, y se visualizan los resultados.

2.3.3 Diagrama del Proceso de negocio To-Be.

En la Figura 2.2 se muestra el diagrama de Proceso de negocio en su forma To-Be y como el cliente realizará este proceso con la incorporación del módulo QtNLP-Diff.

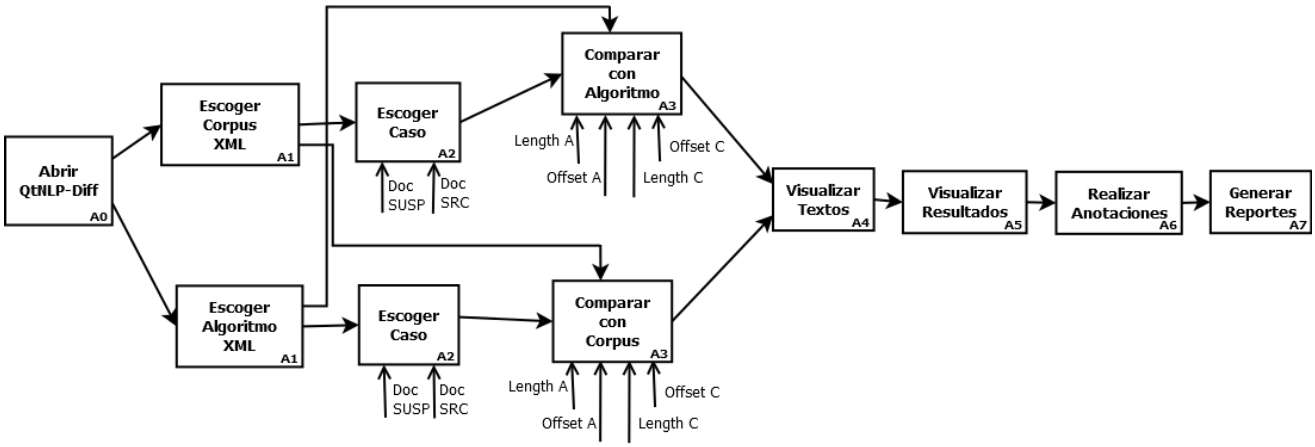


Figura 2.2 Proceso de Negocio forma To-Be

Descripción del diagrama de Proceso de negocio en la forma To-Be:

El lingüista abre el módulo QtNLP-Diff, abre el XML del corpus y escoge el caso a comparar. Previamente se abre el XML del algoritmo y se escoge el caso similar a comparar. Luego se visualizan los fragmentos de los textos con sus atributos y se muestra los resultados de la comparación. Posteriormente el lingüista realiza anotaciones a este caso comparado y se generan reportes en XML de los casos revisados.

2.4 Pila del producto

Es una lista priorizada que define el trabajo que se va a realizar en el proyecto. Cuando un proyecto comienza es muy difícil tener claro todos los requerimientos sobre el producto. Sin embargo, suelen surgir aquellos que son más importantes y que casi siempre son más que suficientes para un Sprint (Peñalver, 2008).

En la Tabla 2.1 se muestran los requerimientos del módulo QtNLP-Diff

Tabla 2.1 Pila del producto

Asignado a	Ítem	Descripción	Estimación	Estimado	HU	Estado
Prioridad		**Muy Alta**				
Javier	1	Gestionar casos del corpus	1	Alex	HU_1	Realizado
Javier	2	Gestionar casos del algoritmo	1	Alex	HU_2	Realizado
Prioridad		**Alta**				
Javier	3	Comparar resultados de corpus y algoritmos.	3	Alex	HU_3	Realizado
Javier	5	Realizar anotaciones de casos revisados	3	Alex	HU_5	Realizado
Javier	6	Generar reporte en XML.	3	Alex	HU_6	Realizado
Prioridad		**Media**				
Javier	4	Recuperar casos de corpus y algoritmos	3	Alex	HU_4	Realizado

Leyenda:

Ítem: Número de la funcionalidad

Estimación: Duración de la funcionalidad por semanas

HU: Historia de Usuario

2.5 Plan de iteraciones.

El plan de iteraciones es el artefacto documental donde se recogen las iteraciones a realizar con sus características, además del orden de las historias de usuario con su

planificación estimada para ser implementadas. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se pierde calidad en el software o no se cumplen los plazos (Peñalver, 2008)

En la Tabla 2.2 se recogen las iteraciones a realizar con sus características, además del orden de las historias de usuario a implementar y la duración total de la iteración.

Tabla 2.2 Plan de iteraciones

Iteración	Descripción de la iteración	Orden de la HU a implementar	Duración total
1	Investigación del tema de comparaciones de textos y herramientas para visualizar y editar la QtNLP-Diff.	Ninguna. Artefacto solo el capítulo 1 de la tesis.	17/01/2016-05/02/2016
2	Iteración de capacitación. Re-factorizar la investigación. Los documentos del diseño del sistema en SXP.	Ninguna Artefacto solo el capítulo 2 de la tesis.	08/02/2016-05/03/2016
3	Gestionar casos de corpus y algoritmos. Además comenzar los diseños de los GUI en Qt.	HU_1, HU_2	06/03/2016-31/03/2016
4	Realizar filtro de casos por el tipo de plagio y obtener caso por su nombre. También realizar comparaciones de casos de corpus y algoritmos. Además de diseñar sus respectivos GUI en Qt.	HU_3, HU_4	01/04/2016-27/04/2016
5	Realizar anotaciones de los casos y su respectiva GUI en Qt.	HU_5	27/04/2016-15/05/2016
6	Realizar generar reportes XML con su respectiva GUI en Qt.	HU_6	16/05/2016-15/06/2016

2.6 Historias de usuarios

La historia de usuario es la técnica utilizada en XP para especificar los requisitos del software, lo que equivale a los casos de uso en el proceso unificado. Las mismas son escritas por los clientes como las tareas que el sistema debe hacer y su construcción depende principalmente de la habilidad que tenga el cliente para definirlas. Son utilizadas como el único documento de requisitos que se genera en XP. Son escritas en

lenguaje natural, sin un formato predeterminado, no excediendo su tamaño de unas pocas líneas de texto(Jeffries, 2001).

En la Tabla 2.3, Tabla 2.4, Tabla 2.5, Tabla 2.6, Tabla 2.7 y Tabla 2.8 se aprecian las historias de los usuarios Gestionar casos del corpus, Gestionar caso del algoritmo, Comparar resultados de corpus y algoritmos, Recuperar casos de corpus y algoritmos, Realizar anotaciones de casos revisados y Generar reporte en XML del módulo QtNLP-Diff respectivamente.

Tabla 2.3 HU_1 Gestionar casos del corpus

HISTORIA DE USUARIO	
Número: 1	**Nombre Historia de Usuario:** Gestionar casos del corpus
Modificación de Historia de Usuario Número: No se ha modificado	
Usuario: Lingüistas	**Iteración Asignada:** 3
Programador responsable: Javier Sardiñas Morales	
Prioridad en Negocio: Muy-Alto	**Puntos Estimados:** 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Seleccionar corpus lingüístico en formato XML de donde se obtienen los casos, los cuales se mostraran en la interfaz del comparador.	
Observaciones: • Sin Observaciones	
Tareas de Ingeniería: 1. Diseñar prototipo de interfaz para seleccionar corpus y mostrar textos susp y src 2. Programar botón para seleccionar corpus 3. Trabajar con Pxdom para cargar corpus XML en donde se encuentran los casos 4. Cargar listwidget donde se observan todos los casos del corpus los cuales serán seleccionados por los lingüistas 5. Agregar el id correspondiente del caso en la lista de casos 6. Cargar en los textedit los archivos en formato txt susp y src para futuras comparaciones 7. Mostrar total de casos en la lista del corpus.	

Tabla 2.4 HU_2 Gestionar casos del algoritmo

HISTORIA DE USUARIO	
Número: 2	**Nombre Historia de Usuario:** Gestionar caso del algoritmo
Modificación de Historia de Usuario Número: No se ha modificado	
Usuario: Lingüistas	**Iteración Asignada:** 3
Programador responsable: Javier Sardiñas Morales	
Prioridad en Negocio: Muy-Alto	**Puntos Estimados:** 1
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 2
Descripción: Seleccionar algoritmo de similitud en formato XML donde se obtienen los casos, los cuales se mostraran en la interfaz del comparador.	
Observaciones: • Se reutilizó el conocimiento y el código de la HU_1 dado que es similar.	
Tareas de Ingeniería: 1. Diseñar prototipo de interfaz para seleccionar algoritmo y mostrar textos susp y src. 2. Programar botón para seleccionar algoritmo. 3. Trabajar con Pxdom para cargar algoritmo XML en donde se encuentran los casos. 4. Cargar listwidget donde se observan todos los casos del algoritmo los cuales serán seleccionados por los lingüistas. 5. Agregar el id correspondiente del caso en la lista de casos. 6. Cargar en los textedit los archivos txt susp y src para futuras comparaciones. 7. Mostrar cantidad de casos en la lista del algoritmo.	

Tabla 2.5 HU_3 Comparar resultados de corpus y algoritmos

HISTORIA DE USUARIO	
Número: 3	**Nombre Historia de Usuario:** Comparar resultados de corpus y algoritmos
Modificación de Historia de Usuario Número: No se ha modificado	
Usuario: Lingüista	**Iteración Asignada:** 4
Programador responsable: Javier Sardiñas Morales	
Prioridad en Negocio: Alto	**Puntos Estimados:** 3
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 4
Descripción: previamente seleccionados el corpus y algoritmo, se realizan comparaciones de los	

resultados de corpus y algoritmos en la interfaz del comparador de resultados.

Observaciones:

- La codificación de los textos que se utilizó es UTF-8.
- Los textos se cargan en textedit diferentes.
- Se muestra los valores del offset y length de cada texto susp y src.
- Se pintó las posiciones para un mejor entendimiento del cliente a la hora de analizar los casos

Tareas de Ingeniería:

1. Diseñar prototipo de interfaz para comparar casos de corpus y algoritmo.
2. Mostrar el nombre del XML corpus cargado.
3. Mostrar el nombre del XML algoritmo cargado.
4. Programar selección de texto por offset y length para el susp del corpus.
5. Programar selección de texto por offset y length para el src del corpus.
6. Programar selección de texto por offset y length para el susp del algoritmo.
7. Programar selección de texto por offset y length para el src del algoritmo.
8. Programar función para dibujar de color la selección de los textos.
9. Programar selección de los textos una oración antes y una oración después del offset y el length.
10. Cambiar comportamiento cuando se selecciona un caso de la lista corpus, entonces actualizar en la lista del algoritmo con los mismos o el mismo caso.
11. Cambiar comportamiento cuando se selecciona un caso de la lista algoritmo, entonces actualizar en la lista del corpus con los casos coincidentes.
12. Mostrar valores de offset y length en cada textedit.
13. Actualizar valores al cargar un nuevo caso.

Tabla 2.6 HU_4 Recuperar casos de corpus y algoritmos

HISTORIA DE USUARIO	
Número: 4	**Nombre Historia de Usuario:** Recuperar casos de corpus y algoritmos
Modificación de Historia de Usuario Número: No se ha modificado Sin modificaciones	
Usuario: Lingüistas	**Iteración Asignada:** 4
Programador responsable: Javier Sardiñas Morales	
Prioridad en Negocio: Media	**Puntos Estimados:** 3
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 3

Descripción: Buscador para facilitar las selección de los casos de las comparaciones de los casos del corpus y de los algoritmos.

Observaciones:

- Se utilizó la biblioteca de Pxdom para poder trabajar con los datos en los archivos XML.
- Se utilizó el mismo prototipo de interfaz para el buscador de los resultados de algoritmos.
- Se utilizó el mismo comportamiento para el buscador de los resultados de algoritmos.
- Los colores usados para los tipos de plagio son los predefinidos por QtNLP.
- Descripción de los colores.
    - Multicolor significado todos.
    - Blanco significado no plagio.
    - Rojo significado literal.
    - Azul significado parafrástico.
    - Amarrillo significado interlingual.
    - Naranja significado bad quotation.
    - Verde significado copyleft.

Tareas de Ingeniería:

1. Diseñar prototipo de interfaz para buscar casos.
2. Insertar iconos en colores para filtrar el tipo de plagio.
3. Programar filtros de búsqueda de tipos de plagio para agilizar la búsqueda.
4. Programar buscador en lineEdit para encontrar nombre del caso.
5. Programar comportamiento de los checkbox del filtro.
6. Mostrar información de tipo de plagio.

Tabla 2.7 HU_5 Realizar anotaciones de casos revisados

HISTORIA DE USUARIO	
Número: 5	**Nombre Historia de Usuario:** Realizar anotaciones de casos revisados
Modificación de Historia de Usuario Número: No se ha modificado Sin modificaciones	
Usuario: Lingüistas	**Iteración Asignada:** 5
Programador responsable: Javier Sardiñas Morales	
Prioridad en Negocio: Alta	**Puntos Estimados:** 3
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 4
Descripción: A partir de cálculos automáticos basados en los valores existentes de algunos campos permiten anotar más informaciones al caso de las que el algoritmo de detección genera.	

Observaciones:

- Atributos a insertar o modificar:
    - Plag_type: Si el caso del algoritmo coincide con un caso del corpus y este atributo no está generado por el algoritmo entonces transferir del corpus al XML_caso_revisado y poner no editable. Si no coincide con ningún caso, activar el combobox.
        - no plagi
        - literal
        - paraphrastic
        - interlingual
        - bad quotation
        - copyleft
    - Coincidencia: forma en la que machea un caso encontrado por el algoritmo, con alguno de los casos en el corpus (no editable, cálculo automático).
        - Match
        - Not match
        - Left match (coincide por la izquierda del caso del corpus)
        - Right match
        - Contained
        - Over-match
    - Topic match: coincidencia del susp y el src en cuanto a tópico (Si el caso del algoritmo coincide con alguno del corpus, transferir etiqueta del corpus al XML_caso_revisado) Si no el revisor debe leer ambos textos y decidir
        - Intra topic (del mismo tópico)
        - Inter topic (diferente tópico)
    - Paraphrase composition: clasificación por la cantidad de tipos de paráfrasis distintas que contiene el caso. Si el caso del algoritmo coincide con alguno del corpus, transferir etiqueta del corpus al XML_caso_revisado, de lo contrario activar el combobox.
        - pure (one type of paraphrase)
        - mixed (two types of paraphrase)
        - multiple (three types or more paraphrases)

- Length: tamaño relativo medido en palabras.(no editable, calculo automático)
  - Short <= 60; 60 < Medium <= 360; Long > 360
- Revisor: nombre de la persona que escribe estas anotaciones (se llena del fichero del profile). Nota cuando se carga el corpus generar un tack para obligar a poner el nombre y mostrar este campo no modificarlos. (no editable, leer de fichero)
- Text_reuse_susp_snippet: % del tamaño del caso susp con respecto a la longitud del documento susp. (no editable, calculo automático)
  - None < 5%
  - Low < 20%
  - Medium< 40%
  - High < 80%)
  - Extreme > 80%
- Case date: fecha del caso. Llenar con fecha de la revisión si el algoritmo no lo anotó. (editable)
- Generated by: generado automáticamente o manual, leer de la etiqueta del caso del corpus. Si no coincide rellenar con "descubierto". (no editable)
- Domain: dominio del texto. Ejemplo: informática, biología, física, arquitectura,… Si el caso del algoritmo coincide con alguno del corpus, transferir etiqueta del corpus al XML_caso_revisado. Y si no, poner editable.
- Document type: tipo de documento o lo que se conoce como "género discursivo". Llenar con estos 3 valores definidos por el equipo de QtNLP para esta versión. Si el caso del algoritmo coincide con alguno del corpus, transferir etiqueta del corpus al XML_caso_revisado, de lo contrario activar el combobox.
  - scientific paper
  - book
  - web page
- Resto de las etiquetas que aparecen en el corpus y que no estén en el algoritmo.
  - Si el caso del algoritmo coincide con alguno del corpus, transferir etiqueta del corpus al XML_caso_revisado, de lo contrario transferir = "" (vacío).

Tareas de Ingeniería:

1. Diseñar la interfaz.
2. Diseñar etiqueta de coincidencia. Diseñar los íconos de coincidencia. Definir los valores de la etiqueta.

3. Definir las etiquetas nuevas a agregar.
4. Definir los valores de las etiquetas, de acuerdo a estudio de corpus del profesor.
5. Definir resto de las etiquetas a transferir.
6. Rediseñar interfaz.
7. Programar comportamiento de las etiquetas.
8. Programar botón para guardar anotaciones.
9. Agregar iconos para mejor entendimiento.
10. Programar comportamiento de las anotaciones cuando el caso es nuevo.

Tabla 2.8 HU_6 Generar reporte en XML

HISTORIA DE USUARIO	
Número: 6	**Nombre Historia de Usuario:** Generar reporte en XML
Modificación de Historia de Usuario Número: No se ha modificado Sin modificaciones	
Usuario: Lingüistas	**Iteración Asignada:** 6
Programador responsable: Javier Sardiñas Morales	
Prioridad en Negocio: Alta	**Puntos Estimados:** 3
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 4
Descripción: Permite generar un nuevo corpus en XML con los datos de las anotaciones que se hayan guardados.	
Observaciones: • Se utilizó la biblioteca de Pxdom para poder trabajar con los datos en los archivos XML. • Los colores usados para los tipos de plagio predefinidos por QtNLP. • Los iconos usados para las coincidencias susp se definieron con el cliente.	
Tareas de Ingeniería: 1. Diseñar la interfaz. 2. Programar abrir la ubicación de los casos salvados. 3. Programar listar casos revisados. 4. Programar filtros por tipo de plagio. 5. Programar filtros por coincidencias susp. 6. Programar botón return. 7. Agregar iconos a los filtros. 8. Programar botón generate para generar nuevo corpus XML con casos de interés.	

2.7 Requisitos no funcionales

Un requisito no funcional es un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales. Por tanto, se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento (Giraldo, 2007).

En la Tabla 2.9 se muestra los requisitos no funcionales de módulo QtNLP-Diff

Tabla 2.9 Requisitos no funcionales.

RNF1	Usabilidad
Descripción	El sistema podrá ser usado por cualquier persona con conocimientos básicos de informática y navegación en internet.
RNF2	Software
Descripción	Las estaciones de trabajo deberán contar con soporte para Python y Qt.
RNF3	Hardware
Descripción	Para la aplicación servidora es necesaria una PC con microprocesador Pentium 4 3.0GHz, 512MB de RAM y una capacidad de 10GB en disco duro.
RNF4	Soporte
Descripción	Debe poder ser mantenido por el autor principal a través de un repositorio público y comunicación vía internet.
RNF5	Rendimiento
Descripción	El tiempo de respuesta no debe exceder los cinco segundos ante las solicitudes del usuario.
RNF6	Requisitos de Entrega
Descripción	Se debe incluir manual de usuario, ejecutable en .pcy, documentación del proyecto.

2.8 Conclusiones parciales.

En el capítulo dos (análisis del sistema con SXP) se describió la metodología SXP, híbrido de SCRUM y XP realizando el análisis del módulo. El modelo de negocio actual, permitió describir como el usuario realiza las comparaciones antes de la incorporación de QtNLP-Diff, evidenciándose en el diagrama AS-IS. El diagrama TO-

BE explica como el usuario realiza las comparaciones después de la implementación de la herramienta optimizando el proceso de negocio. La pila del producto reflejó los requisitos del sistema, permitiéndonos conocer los requerimientos del usuario. El plan de iteraciones permitió en cada iteración efectuar una revisión del trabajo realizado, acción que contribuyó a perfeccionar el módulo. En las historias de usuarios se plasmaron las tareas de ingeniería correspondientes para realizar cada requisito existente en el módulo.

Capítulo 3. ARQUITECTURA DE SOFTWARE.

Capítulo 3. ARQUITECTURA DE SOFTWARE

En el presente capítulo se describe la arquitectura general del sistema detallando cada uno de los componentes que conforman la misma y permitiendo sentar las bases para el desarrollo del módulo QtNLP-Diff. Se genera los diagramas de los documentos XML en estructura de árbol, el de paquete, de componente y los prototipos de interfaz. Se describe información de la arquitectura de QtNLP-Diff.

3.1 Gestión de la configuración.

QtNLP-Diff es una herramienta desarrollada en Qt y Python. Basa su arquitectura en un modelo de carpetas que facilita su desarrollo para realizar comparaciones de resultados de casos de algoritmo de similitud y corpus lingüísticos, así como las interfaces simples para los lingüistas. La raíz del proyecto está conformada por algunas carpetas que consideramos parte de la arquitectura, y algo importante a leer para iniciarse en el desarrollo de este proyecto:

Figura 3.1 Raíz del módulo

- data: Ubicación de los datos, casos de corpus, casos de resultados de algoritmos y reportes en formato XML.
- doc: Documentación de QtNLP-Diff: contiene el expediente de proyecto usando la metodología SXP y también materiales de apoyo para el entendimiento del módulo.
- lib: Bibliotecas externas al core de Python necesarias para ejecutar el proyecto con éxito. Como Pxdom.
- report: contiene las anotaciones de casos revisados generados por QtNLP-Diff los cuales son en formato XML.
- ui: contiene las interfaces gráficas de QtNLP-Diff.

3.2 Herramientas asociadas al desarrollo del sistema

Para el desarrollo de QtNLP-Diff se utilizan las siguientes herramientas ya que QtNLP se encuentra implementada con las tecnologías de Qt y Python, facilitando así la incorporación del módulo.

3.2.1 PyCharm.

Licencia del producto: Dual (Proprietary, Apache License)

Versión: 4.5.3

PyCharm es un IDE (Entorno de desarrollo integrado) desarrollado por la compañía Jetbrains, está basado en IntelliJ IDEA, el IDE de la misma compañía pero enfocado hacia Java y la base de Android Studio. Pycharm tiene cientos de funciones que lo pueden ver como una herramienta muy pesada, pero que valen la pena ya que ayuda con el desarrollo del día a día (Yograterol, 2014).

3.2.2 QtDesigner.

Licencia del producto: LGPL

Versión: 4.7.0

QtDesigner es una herramienta para diseñar y construir las interfaces gráficas del usuario. Puede componer y personalizar sus ventanas o diálogos en una manera y las pruebas usando estilos y resoluciones diferentes. Reproductores y formas creados con QtDesigner se integran a la perfección con código programado, usando mecanismos de signals y slots de Qt, que le permite asignar fácilmente el comportamiento de los elementos gráficos. Todas las propiedades establecidas en QtDesigner se pueden cambiar dinámicamente dentro del código. Está disponible para Windows, GNU/Linux y Mac OS X bajo diferentes licencias (The Qt Company, 2015).

3.2.3 PyQt.

Licencia del producto: GPL 3.0

Versión: 4.11.4

PyQt es la unión de la biblioteca gráfica Qt para el lenguaje de programación Python. Consta de varios módulos todos los cuales son parte del paquete de PyQt4 Python como por ejemplo QtGui componentes de la interfaz gráfica de usuario, QtCore clases no GUI base usado por otros módulos, QtXml clases para el manejo de XML, etc. (Boddie, 2015).

3.3 Tecnologías para el desarrollo de la aplicación.

Existen muchas tecnologías en el mundo las cuales ayudan a la hora de desarrollar una aplicación siendo muy útiles a desarrolladores. QtNLP-Diff utiliza las siguientes tecnologías pues QtNLP está implementada con las mismas.

3.3.1 Qt biblioteca multiplataforma.

Licencia del producto: LGPL-2.1 (Qt versión de código abierto) Qt Commercial License (Qt versión comercial)

Versión: 4.7.0

Qt es un marco de trabajo usado para desarrollar aplicaciones con interfaz gráfica de usuario, así como también para el desarrollo de programas sin interfaz gráfica, como herramientas para la línea de comandos y consolas para servidores. Qt es desarrollada como un software libre y de código abierto a través de Qt Project, donde participa tanto la comunidad, como desarrolladores de empresas (Lars, 2011).

3.3.2 Python.

Licencia del producto: Python Software, Foundation License

Versión: 2.7

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional (Jim, 2009).

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos). Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable (Jim, 2009).

3.3.3 XML.

Es un lenguaje de marcas utilizado para almacenar datos en forma legible. Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar documentos grandes. Brinda soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información (Rouse, 2016).

XML es un estándar para el intercambio de información estructurada entre diferentes plataformas. Juega un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil (Rouse, 2016).

3.4 Ingeniería del software asistida por computación (CASE).

- **Visual Paradigm:**

Licencia del producto: Proprietary with Free Community Edition
Versión: 6.0
Es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Pressman, 2002).
Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades (Pressman, 2002).

- **Dia:**

Licencia del producto: GPL2
Versión: 0.97.2
Es una aplicación informática de propósito general para la creación de diagramas, desarrollada como parte del proyecto GNOME. Está concebida de forma modular, con diferentes paquetes de formas para diferentes necesidades.
Dia está diseñada como un sustituto de la aplicación comercial Visio de Microsoft. Se puede utilizar para dibujar diferentes tipos de diagramas. Actualmente se incluyen diagramas entidad-relación, diagramas UML, diagramas de flujo, diagramas de redes, diagramas de circuitos eléctricos, etc. Nuevas formas pueden ser fácilmente agregadas, dibujándolas con un subconjunto de SVG e incluyéndolas en un archivo XML (McCann, 2013).

3.5 Sistemas de Control de Versiones.

El sistema de control de versiones utilizado en la implementación del módulo es Bazaar.
Licencia del producto: GPLv2 o mayor

Versión: 2.5.1
Descripción de su uso: Servidor de control de versiones utilizado para este proyecto.

3.6 Patrones de arquitectura (Modelo / Vista /Controlador).

Para lograr un mejor diseño e implementación del módulo QtNLP-Diff se tiene en cuenta el patrón arquitectónico Modelo-Vista-Controlador (MVC) como se muestra en la Figura 3.2 el cual es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario. Para ello el MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.

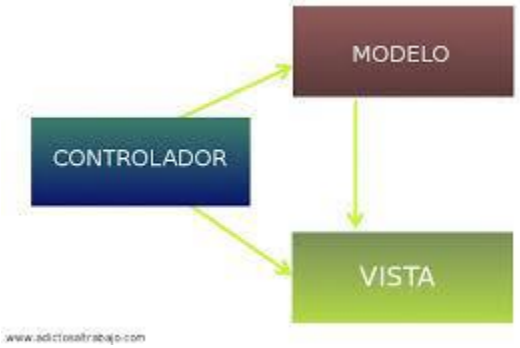


Figura 3.2 Modelo Vista Controlador

Según (Reenskaug and Coplien, 2009) el MVC tiene las siguientes características:

- Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado y responde a instrucciones de cambiar el estado. Este encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada. El modelo debe de preservar la integridad de los datos.
- Vista: Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador. Interactúa con la interfaz de usuario.
- Controlador: Controla el flujo entre la vista y el modelo (los datos). Reciben las entradas, usualmente como eventos, e interpreta las operaciones del usuario; codificando los movimientos, pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio ("service requests") para el modelo o la vista. Es el que debe de controlar los eventos.

En nuestro módulo la vista sería las interfaces gráficas de usuario creadas en el Qt Designer donde se precia la clase interfaz Ui_MainWindow con todos los componentes que se muestran al usuario, el modelo sería el código del módulo realizado en Python donde se aprecian en la clase Comparador con sus funciones relacionadas con las comparaciones y el controlador la librería PyQt4 la cual se encarga de unificar Python con Qt.

3.7 Estructura de árbol de documentos XML.

Un documento XML es siempre descriptivo. La estructura de árbol es a menudo referido como árbol XML y desempeña un papel importante para describir cualquier documento XML fácilmente. La estructura de árbol contiene raíz (padre) elementos, elementos secundarios y así sucesivamente. Con estructura de árbol, puede llegar a conocer todas las ramas y sub-ramas a partir de la raíz. El análisis comienza en la raíz, y luego se desplaza hacia la parte inferior de la primera rama de un elemento, toma la primera rama de allí, y así hasta el nodo hoja (Tutorialpoint, 2016) .

Documento XML del Corpus.

En el documento XML del corpus se encuentra el nodo padre <corpus> con los siguientes atributos name, versión, lang, total_cases, total_true_cases, total_annotations, total_true_annotations, license, copyright, owners, authors, country, creation_date, last_modification_date, xmlns, description. Después aparece el nodo <cases> hijo de <corpus>. A continuación tenemos el nodo <group> hijo del <cases> con los atributos NLP_problem_type, text_extension. Después aparece el nodo <case> hijo de <group> con los atributos id, description, plag_type, annotator_summary, automatic_summary, original_corpus, original_corpus_id, generated_by, generator_name, domain, document_type, topic_match, paraphrase_composition, length. A continuación aparecen los hijos < susp_snippet > con los atributos doc, offset, length, sentences_count, words_counttopic, topic y <src_snippet> doc, offset, length, sentences_count, words_count, topic.

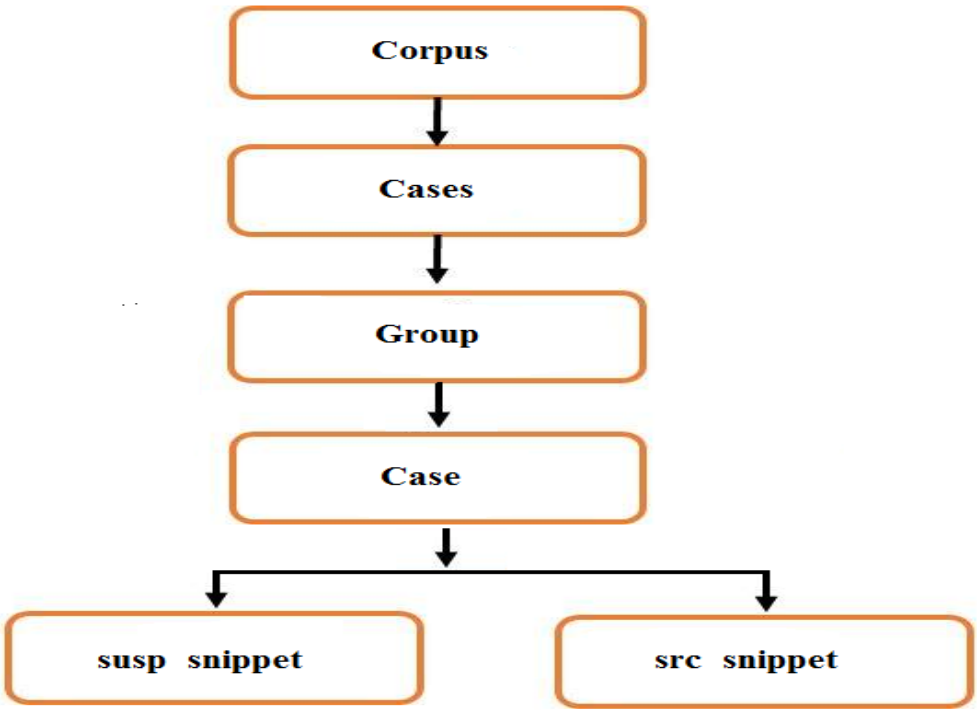


Figura 3.3 Árbol del XML del Corpus

Documento XML de Algoritmo.

En el documento XML del algoritmo se encuentra el nodo padre <algoritmo> con el siguientes atributos name. A continuación tenemos el nodo <group> hijo del <cases> Después aparece el nodo <case> hijo de <group>. Después aparece el nodo <case> hijo de <group> con los atributos id, description, plag_type, annotator_summary, automatic_summary, original_corpus, original_corpus_id, generated_by, generator_name, domain, document_type, topic_match, paraphrase_composition, length. A continuación aparecen los hijos < susp_snippet > con los atributos doc, offset, length, y <src_snippet> doc, offset, length.

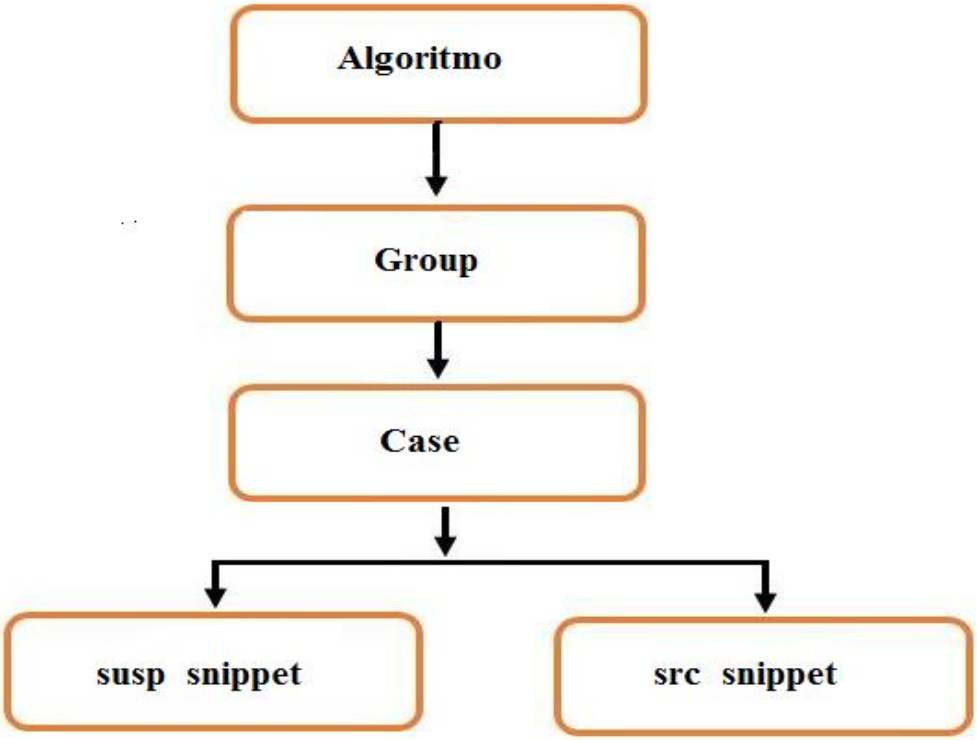


Figura 3.4 Árbol del XML del Algoritmo

Diagrama de case XML del Reporte.

En el documento XML del algoritmo se encuentra el nodo padre <corpus> con el siguientes atributos name. A continuación tenemos el nodo <group> hijo del <cases> Después aparece el nodo <case> hijo de <group>. Después aparece el nodo <case> hijo de <group> con los atributos id, description, plag_type, annotator_summary, automatic_summary, original_corpus, original_corpus_id, generated_by, generator_name, domain, document_type, topic_match, paraphrase_composition, length. A continuación aparecen los hijos < susp_snippet > con los atributos doc, offset, length, y <src_snippet> doc, offset, length.

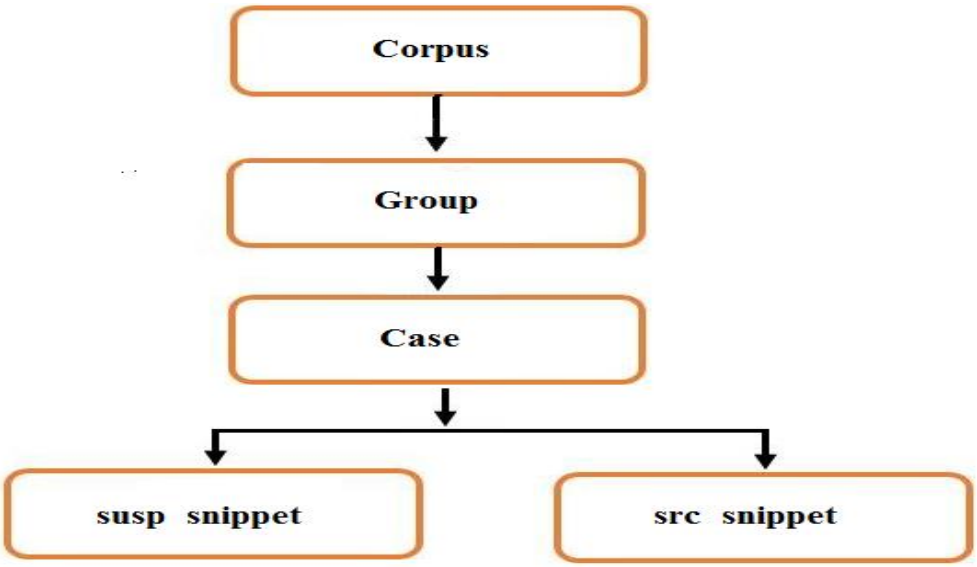


Figura 3.5 Árbol del XML del Reporte

3.8 Diagrama paquetes.

"Un diagrama de paquetes representa las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema" (OMG, 2007).

Este módulo se divide en cuatro grandes paquetes:

- Gestionar casos corpus: Este paquete es el encargado de cargar todos los casos del XML corpus.
- Gestionar casos de resultados de algoritmo: Este paquete es el encargado de cargar todos los casos del XML algoritmo.
- Comparar resultados de corpus y algoritmos: Este paquete es el encargado de visualizar las comparaciones entre los casos del XML corpus y los casos del XML algoritmo.
- Realizar anotaciones: Este paquete es el encargado de realizar las anotaciones ya sean realizadas automáticamente o por el cliente y las guarda en un formato XML.
- Generar reporte en XML: Este paquete es el encargado de generar un reporte en XML con las anotaciones de casos revisados y guardados previamente.

En la Figura 3.6 muestra el diagrama de paquete del módulo QtNLP-Diff.

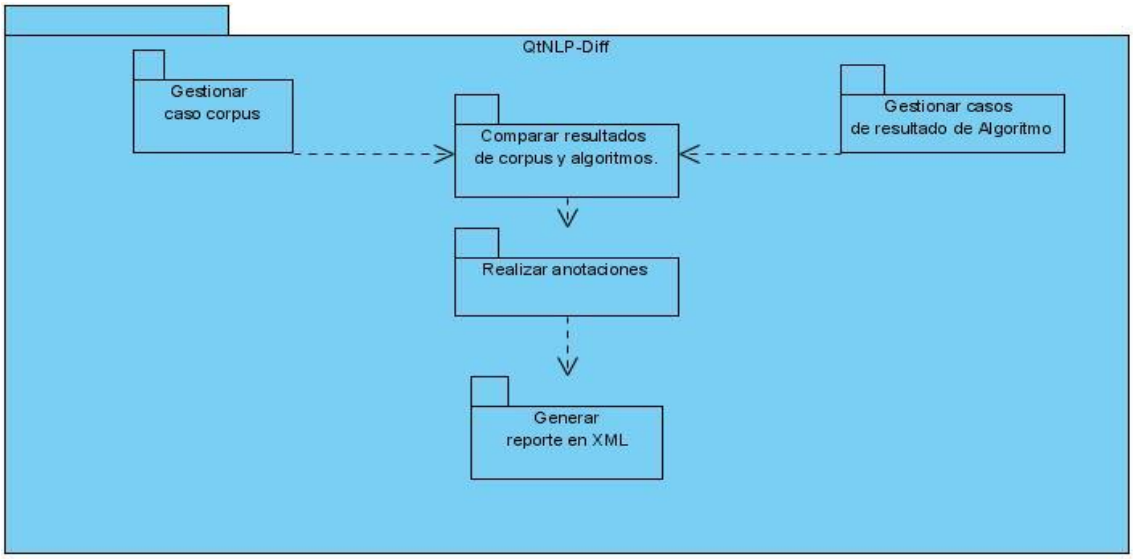


Figura 3.6 Diagrama de Paquete de QtNLP-Diff

3.9 Diagrama componente.

"Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de Componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema" (Grady Booch, 2005).

En la Figura 3.7 se representa el diagrama de componentes para el módulo que se propone. La aplicación concluida queda como un archivo ejecutable "QtNLP-Diff.pyc" que importa las bibliotecas "PyQt4" y "Pxdom" y contiene como ficheros independientes "Report XML", Corpus XML y Algoritmo XML.

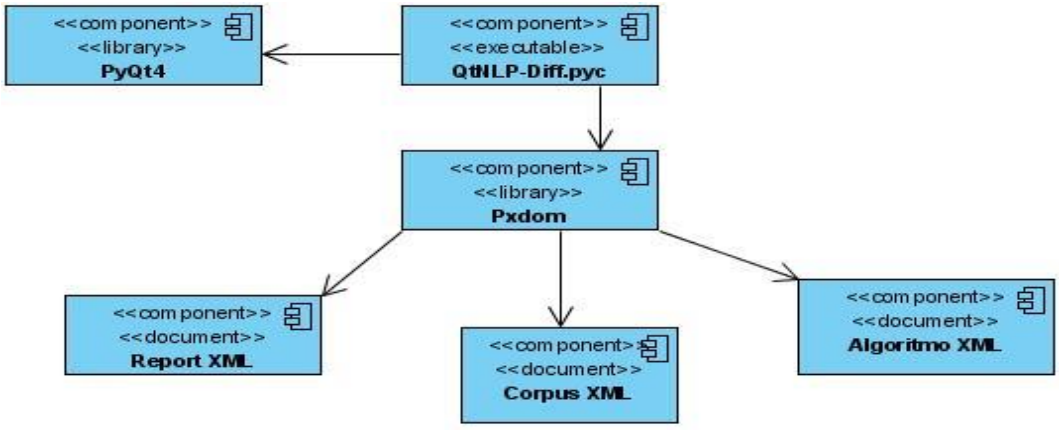


Figura 3.7 Diagrama de componentes de QtNLP-Diff

3.10 Prototipos de interfaz.

Los prototipos son un modelo limitado de un producto, que es probado en situaciones reales, creando así un proceso de diseño de iteración que genera calidad. Un prototipo puede ser desde un trozo de papel con sencillos dibujos a un complejo software. Son útiles para comunicar, discutir y definir ideas entre los diseñadores y las partes responsables. Los prototipos apoyan la evaluación de productos, clarifican requisitos de usuario y definen alternativas (Lacalle, 2006).

Prototipo interfaz “Gestionar casos del corpus”.

HU_1: Gestionar casos del corpus

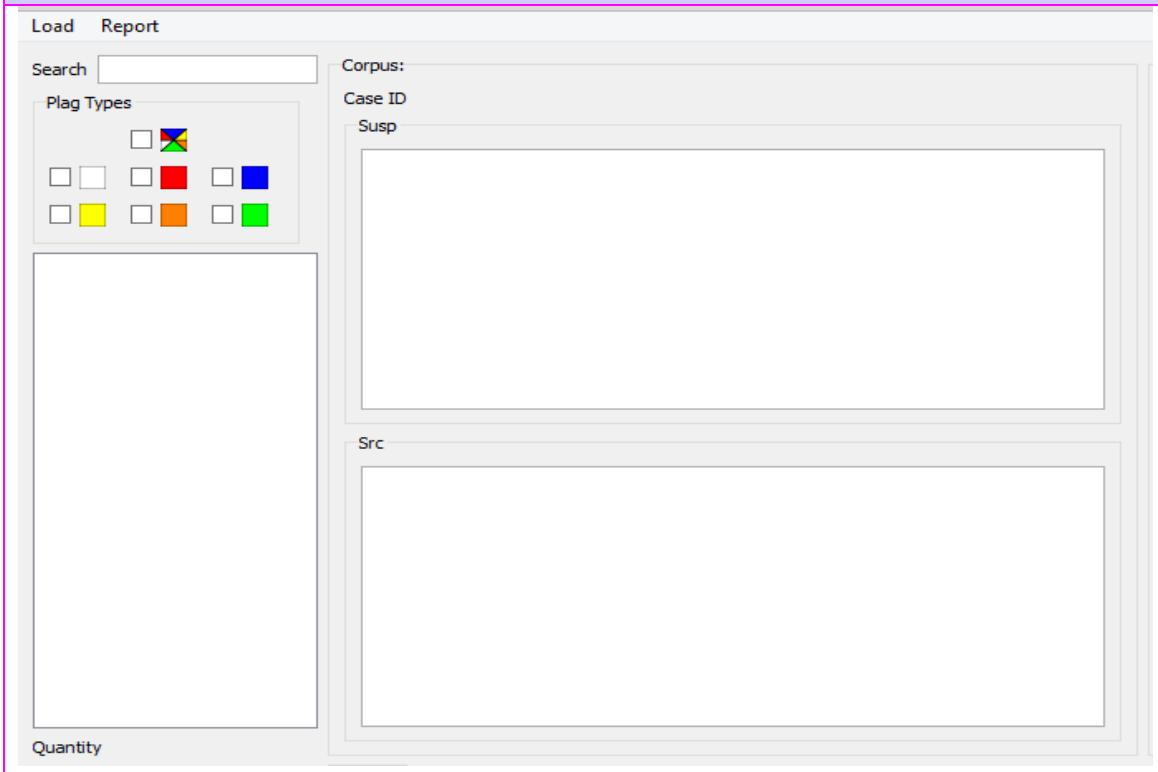


Figura 3.8 Prototipo de interfaz “Gestionar casos del corpus”

Prototipo interfaz “Gestionar casos del algoritmo”.

HU_2: Gestionar casos del algoritmo

Figura 3.9 Prototipo de interfaz “Gestionar casos del algoritmo”.

Prototipo interfaz “Comparar resultados de corpus y algoritmos”.

HU_3: Comparar resultados de corpus y algoritmos

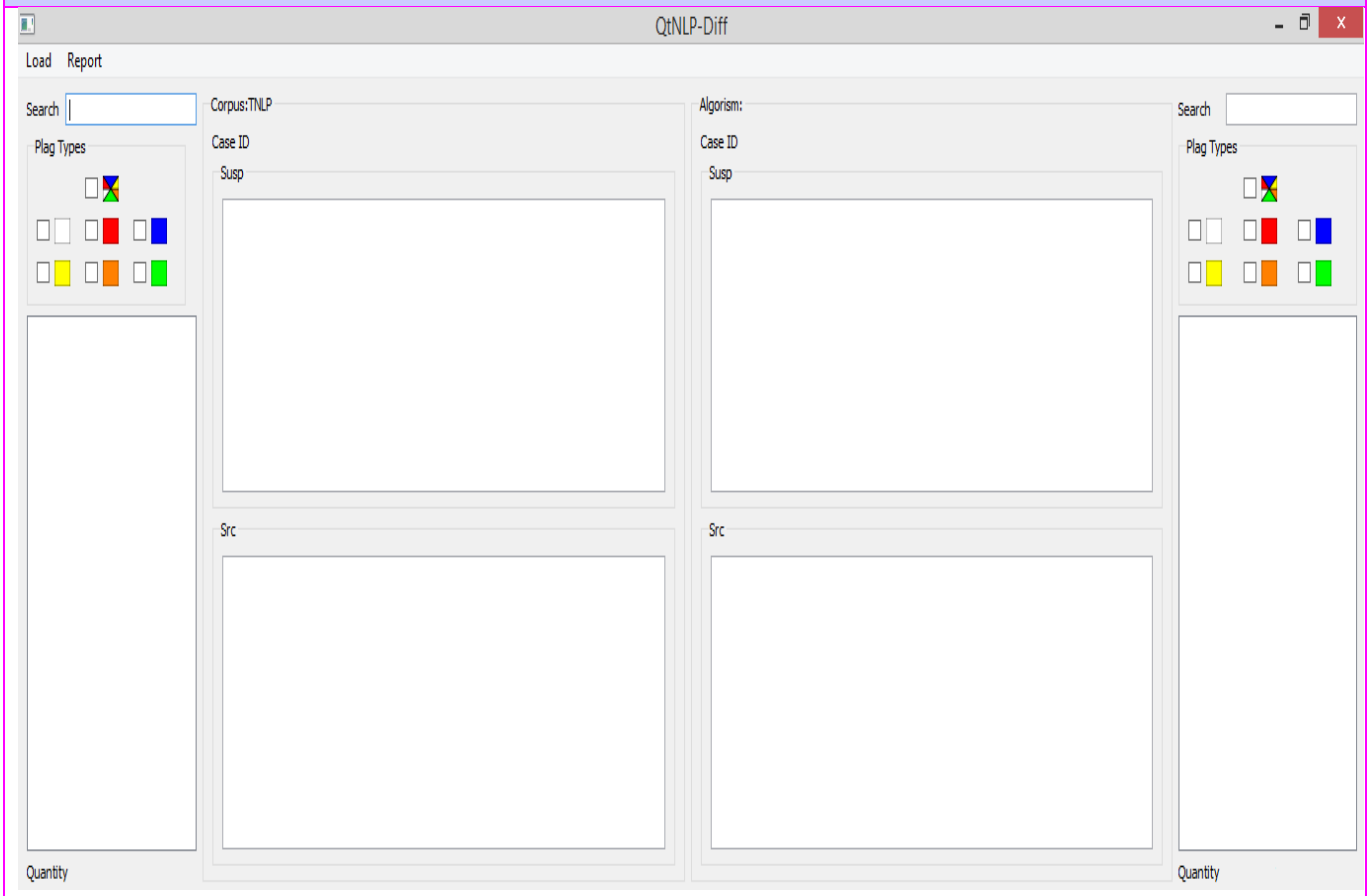


Figura 3.10 Prototipo de interfaz “Comparar resultados de corpus y algoritmos”

Prototipo interfaz “Recuperar casos de corpus y algoritmos”

HU_4: Recuperar casos de corpus y algoritmos

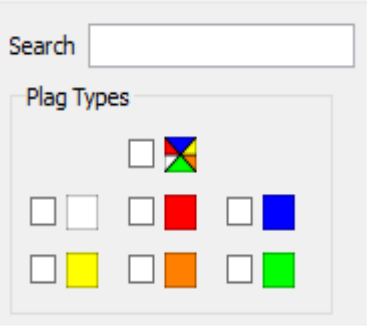


Figura 3.11 Prototipo de interfaz “Recuperar casos de corpus y algoritmos”

Prototipo interfaz “Realizar anotaciones de casos revisados”

HU_5: Realizar anotaciones de casos revisados

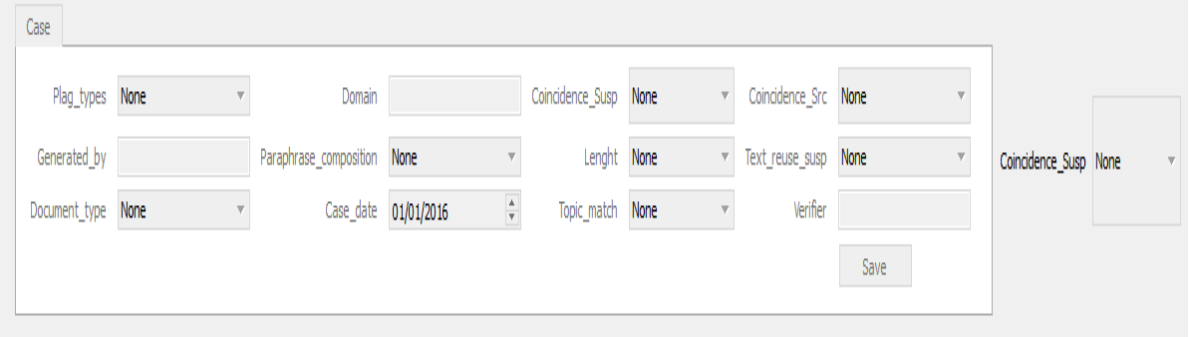


Figura 3.12 Prototipo de interfaz “Realizar anotaciones de casos revisados”

Prototipo interfaz “Generar reporte en XML”

HU_6: Generar reporte en XML

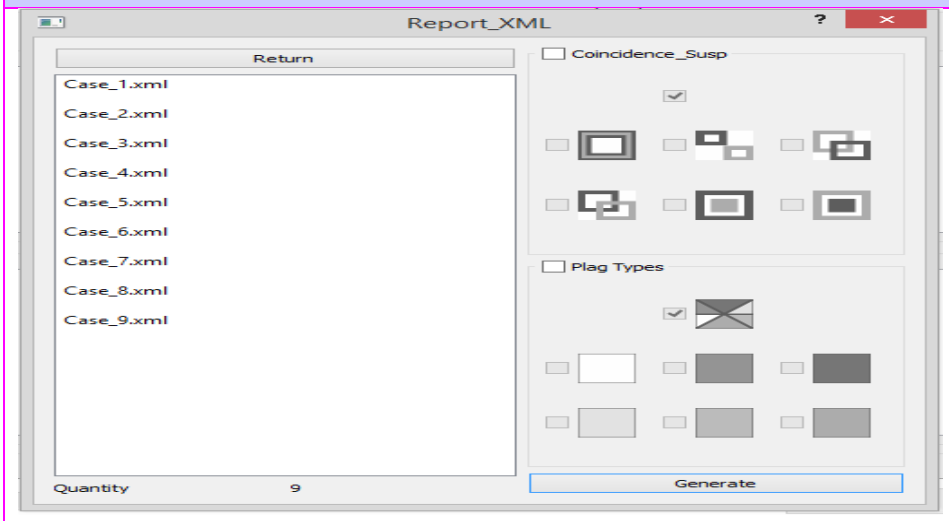


Figura 3.13 Prototipo de interfaz “Generar reporte en XML”

3.11 Definición de la Arquitectura de información.

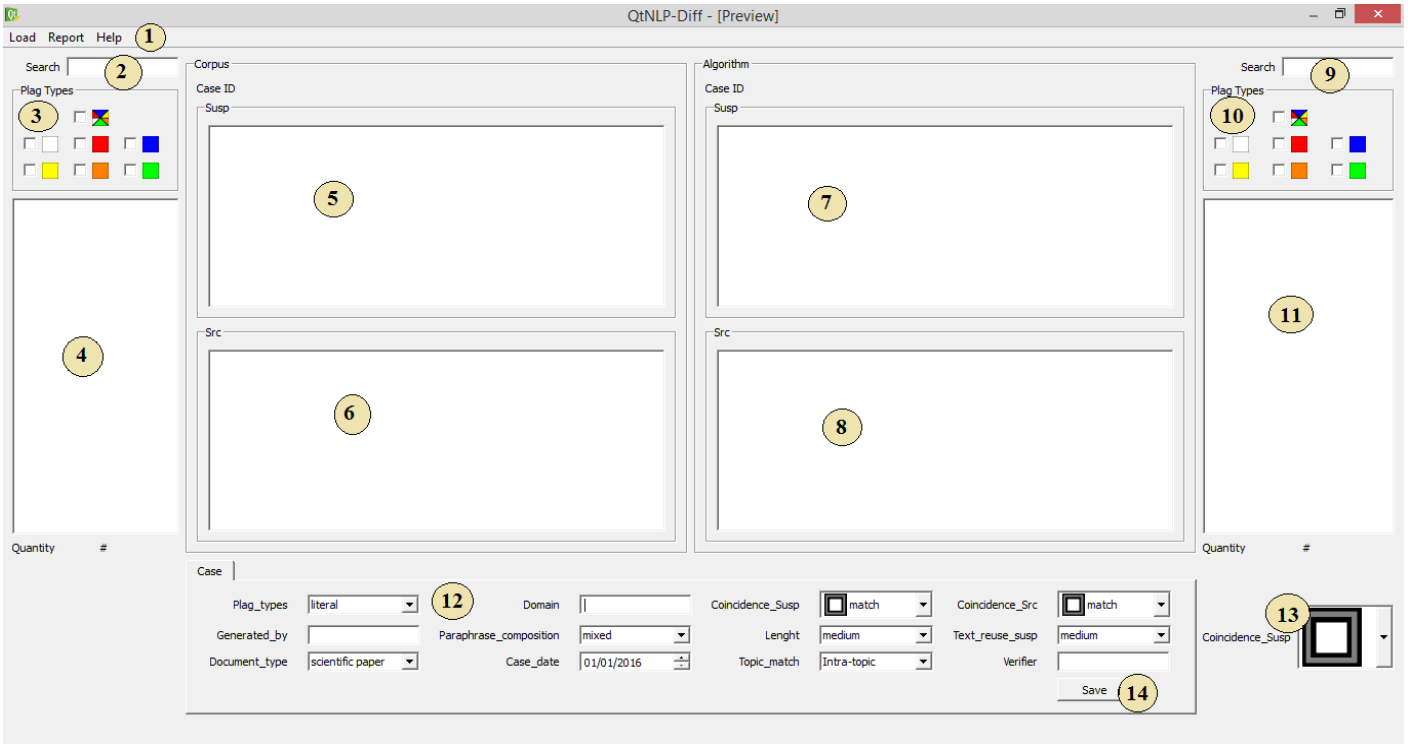


Figura 3.14 Vista principal de QtNLP-Diff

Descripción de los elementos que componen la Pantalla Principal:

- Elemento [1] “Barras de Menú”: Muestra los botones despegables denominados “Load”, “Report” y “Help”. Load tiene las dos opciones “Corpus” y “Algoritm”,

los cuales son los encargados de cargar los casos de los corpus y algoritmos según sean seleccionado por el cliente. "Report" tiene la opción "Open Report" la cual abre una ventana que contiene las carpetas donde se almacena todos los casos anotados que el usuario a realizado. "Help" tiene la opción del "QtNLP-Diff Help" la cual muestra el manual de usuario del módulo.

- Elemento [2] "Búsqueda casos corpus". Encargado de buscar un caso de corpus por su nombre escrito por el cliente.
- Elemento [3] "Filtro de búsqueda corpus": realiza un filtro de los casos del corpus que se está buscando según el tipo de plagio.
- Elemento [4] "Lista de casos corpus": Muestra una lista con todos los casos que existen en el corpus seleccionado.
- Elemento [5] "Caso Text Susp-Corpus". Muestra el fragmento de texto del caso susp del corpus seleccionado por los parámetros offset y lenght.
- Elemento [6] "Caso Text Src-Corpus". Muestra el fragmento de texto del caso src del corpus seleccionado por los parámetros offset y lengh.
- Elemento [7] "Caso Text Susp-Algoritmo". Muestra el fragmento de texto del caso susp del algoritmo seleccionado por los parámetros offset y lengh.
- Elemento [8] "Caso Text Src- Algoritmo". Muestra el fragmento de texto del caso src del algoritmo seleccionado por los parámetros offset y lengh.
- Elemento [9] "Búsqueda casos algoritmo". Encargado de buscar un caso de algoritmo por su nombre escrito por el cliente.
- Elemento [10] "Filtro de búsqueda algoritmo": realiza un filtro de los casos del algoritmo que se está buscando según el tipo de plagio.
- Elemento [11] "Lista de casos algoritmo": Muestra una lista con todos los casos que existen en el algoritmo seleccionado.
- Elemento [12] ""Anotaciones de datos": Muestra todos los datos que serán anotados por el cliente o por la aplicación para generar XML-Reporte.
- Elemento [13] ""Combobox": Muestra iconos de la coincidencia susp dependiendo de los tipos de casos que se seleccionan.
- Elemento [14] "Botón Save": Al presionar el botón "Save" se guarda un XML-Reporte después de ser anotados los casos por el cliente y la aplicación en una carpeta predeterminada.

3.12 Funcionamiento del módulo QtNLP-Diff

Los XMLs del corpus y de los resultados del algoritmo de similitud tienen varios nodos y dentro de estos varios atributos. En el nodo case, se encuentra los nodos *susp_snippet*, *src_snippet*, y dentro de estos un atributo *doc* que contiene la ruta a los documentos donde están los fragmentos similares. Al seleccionar un caso en la lista de casos reportados por el algoritmo se toman los atributos *doc* de este, y se analizan los atributos *doc* de cada caso en el corpus, mostrando en la lista solo aquellos que coinciden. Luego utilizando los atributos *offset* y *length* del caso seleccionado y uno seleccionado de los mostrados en la lista del corpus, se visualizan los fragmentos de texto de los documentos referidos en los atributos *doc*. Utilizando los valores de ambos atributos, *offset* y *length*, la aplicación calcula la coincidencia del caso reportado por el algoritmo con su similar del corpus, y coloca un valor en los atributos *coincidence_susp* y *coincidence_src*, estos valores se descrien en la HU_5. Adicionalmente el módulo calcula otras dos características del caso que son *lenght* y *text_reuse_susp* utilizando el tamaño en caracteres del fragmento y del documento. Visualiza entonces los fragmentos del texto y estas características, más las que aparecen en el resto de los XMLs, en los componentes gráficos. Al revisar caso por caso, del XML de resultados del algoritmo con sus respectivos similares en el corpus, se guardan todos los valores en un nuevo XML del caso que contiene los atributos iniciales y los atributos calculados automática o manualmente (los verificados por el lingüista), que quedan como nuevas características del caso para su análisis posterior. Utilizando luego la función de reporte el lingüista puede generar un único XML que contenga todos los casos revisados, teniendo en cuenta los atributos *plag_type* y *coincidence_susp*.

3.13 Conclusiones parciales.

En el capítulo tres se realiza la arquitectura de software al módulo a través del uso de las herramientas QtDesigner, PyCharm y PyQt y las tecnologías Python, Qt y XML. Estas herramientas se utilizaron para lograr la integración del módulo al sistema QtNLP. Se utilizó el patrón de arquitectura Modelo Vista Controlador para lograr un mejor diseño e implementación del módulo. Se realizaron los diagramas de XML en estructura árbol permitiendo la comprensión de los mismos. En el diagrama de paquete se muestran los paquetes que contiene QtNLP-Diff y en el diagrama de componentes se aprecia la integración de los componentes en el módulo. Se describió la arquitectura de la información para la comprensión de los componentes de la pantalla principal de QtNLP-

Diff. Adicionalmente se describe de forma resumida el funcionamiento de la herramienta en su totalidad, nombrando los atributos específicos del XML utilizados para su implementación.

Capítulo 4. Validación del Sistema.

Capítulo 4. VALIDACIÓN DEL SISTEMA

En el presente capítulo se plasman los casos de pruebas de caja negra a las que fue sometido el sistema, generándose pruebas de aceptación asegurando que sistema desarrollado cumple sus requisitos.

4.1 Casos de pruebas (Caja negra)

Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos (Dhamaryz, 2010).

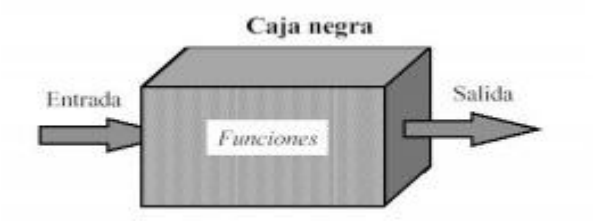


Figura 4.1 Secuencia de eventos en Caja negra.

La prueba de Caja Negra no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca. Muchos autores consideran que estas pruebas permiten encontrar:

1. Funciones incorrectas o ausentes.

2. Errores de interfaz.

3. Errores en estructuras de datos o en accesos a las Bases de Datos externas.

4. Errores de rendimiento.

5. Errores de inicialización y terminación.

Para preparar los casos de pruebas hace falta un número de datos que ayuden a la ejecución de estos casos, y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa, según, si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a

las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien (Dhamaryz, 2010).

Para lograr un producto con calidad es necesario trazarse un plan de pruebas desde el principio. Darle seguimiento a los cambios y desarrollar iterativamente (Dhamaryz, 2010).

4.2 Casos de pruebas de aceptación.

La prueba de aceptación es realizada por un grupo de usuarios finales o los clientes del sistema, para asegurarse que el sistema desarrollado cumple sus requisitos. La prueba de aceptación de usuario se distingue generalmente por la incorporación de un trayecto feliz o casos de prueba positivos (Pruebasdesoftware, 2015).

Tabla 4.1 Prueba de aceptación "Gestionar casos del corpus".

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_1	Nombre Historia de Usuario: Gestionar casos del corpus
Descripción de la Prueba: Prueba realizada al gestionar casos del corpus para ver comportamiento del módulo cuando carga los casos.	
Condiciones de Ejecución: Se debe tener instalado en el sistema el intérprete de Python y tener al menos un XML de casos del corpus.	
Entrada / Pasos de ejecución: 1. El usuario presiona en el menú el botón Load. El sistema despliega dos opciones de selección (Corpus y Algorithm). 2. El usuario selecciona el botón Corpus. El sistema muestra una interfaz de búsqueda para seleccionar XML corpus. 3. El usuario escoge el XML corpus. El sistema muestra un mensaje corpus cargado. 4. El usuario presiona ok. El sistema muestra el nombre del corpus seleccionado, muestra el total de casos y carga todos los casos del XML corpus.	
Resultado Esperado: cargar lista de casos del XML corpus.	
Evaluación de la Prueba: prueba satisfactoria.	

Tabla 4.2 Prueba de aceptación "Gestionar casos del algoritmo".

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_2	Nombre Historia de Usuario: Gestionar casos del algoritmo
Descripción de la Prueba: Prueba realizada al gestionar casos del algoritmo para ver comportamiento del módulo cuando carga los casos.	
Condiciones de Ejecución: Se debe tener instalado en el sistema el intérprete de Python y tener XML de casos del algoritmo.	
Entrada / Pasos de ejecución: 1. El usuario presiona en el menú el botón Load. El sistema despliega dos opciones de selección (Corpus y Algorithm). 2. El usuario selecciona el botón Algorithm. El sistema muestra una interfaz de búsqueda para seleccionar XML algoritmo. 3. El usuario escoge el XML algoritmo. El sistema muestra un mensaje algoritmo cargado. 4. El usuario presiona ok. El sistema muestra el nombre del algoritmo seleccionado, muestra el total de casos y carga todos los casos del XML algoritmo.	
Resultado Esperado: cargar todos los casos del XML algoritmo	
Evaluación de la Prueba: prueba satisfactoria	

Tabla 4.3 Prueba de aceptación "Comparar resultados de corpus y algoritmos".

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_3	Nombre Historia de Usuario: Comparar resultados de corpus y algoritmos.
Descripción de la Prueba: Prueba realizada al comparar resultados de corpus y algoritmos para ver comportamiento del módulo cuando se selecciona el caso del XML corpus.	
Condiciones de Ejecución: Cargados XML corpus y XML algoritmos y tener archivos src y susp.	
Entrada / Pasos de ejecución: 1. El usuario selecciona un caso en la lista de casos del XML corpus. El sistema carga los textos susp y src del XML corpus, brinda información de los parámetros del caso seleccionado y actualiza la lista de los casos del algoritmo mostrando el caso semejante. 2. El usuario selecciona el caso semejante de la lista del XML algoritmo. El sistema carga los	

textos susp y src del XML algoritmos y brinda información de los parámetros del caso seleccionado.
Resultado Esperado: mostrar los textos src y susp con sus parámetros de los casos seleccionados.
Evaluación de la Prueba: prueba satisfactoria

Tabla 4.4 Prueba de aceptación "Comparar resultados de corpus y algoritmos".

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_3	Nombre Historia de Usuario: Comparar resultados de corpus y algoritmos.
Descripción de la Prueba: Prueba realizada al comparar resultados de corpus y algoritmos para ver comportamiento del módulo cuando se selecciona el caso del XML algoritmo.	
Condiciones de Ejecución: Cargados XML corpus y XML algoritmos y tener archivos src y susp.	
Entrada / Pasos de ejecución: 1. El usuario selecciona un caso en la lista de casos del XML algoritmo. El sistema carga los textos susp y src del XML algoritmo, brinda información de los parámetros del caso seleccionado y actualiza la lista de los casos de la lista de los casos del corpus mostrando el caso semejante. 2. El usuario selecciona el caso semejante de la lista del XML corpus. El sistema carga los textos susp y src del XML corpus y brinda información de los parámetros del caso seleccionado.	
Resultado Esperado: mostrar los textos src y susp con sus parámetros de los casos seleccionados.	
Evaluación de la Prueba: prueba satisfactoria	

Tabla 4.5 Prueba de aceptación "Comparar resultados de corpus y algoritmos".

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_3	Nombre Historia de Usuario: Comparar resultados de corpus y algoritmos.
Descripción de la Prueba: Prueba realizada al comparar resultados de corpus y algoritmos cuando se el sistema activa valores de las anotaciones.	
Condiciones de Ejecución: Casos del XML corpus y XML algoritmos seleccionados, archivos src y susp del XML corpus abiertos y archivos src y susp XML algoritmo abiertos.	

Entrada / Pasos de ejecución:

1. El sistema muestra el tipo de plagio tomado del XML algoritmos, si este no se encuentra, entonces lo toma del XML corpus, y si tampoco se encuentra, entonces se habilita el combobox y el usuario lo escoge.

   Nota: Así ocurre con el resto de los valores de las anotaciones las cuales sean combobox y se puedan editar.
2. El sistema muestra la coincidencia susp dependiendo de los atributos offset y length del XML corpus y el XML algoritmo, si los valores de los atributos son los mismos entonces se muestra como mach ese caso.

   Nota: El comportamiento de los valores que muestra la coincidencia susp varía dependiendo de los atributos offset y length XML del corpus y del XML del algoritmo, estos pueden ser mach., no mach, left mach, right mach, over mach y contained.
3. El sistema muestra la coincidencia src dependiendo de los atributos offset y length del XML corpus y el XML algoritmo, si los valores de los atributos son los mismos entonces se muestra como mach ese caso.

   Nota: El comportamiento de los valores que muestra la coincidencia src varía dependiendo de los atributos offset y length de los XML del corpus y del XML del algoritmo, estos pueden ser mach., no mach, left mach, right mach, over mach y contained.

Resultado Esperado: mostrar los textos src y susp con sus parámetros de los casos seleccionados.

Evaluación de la Prueba: prueba satisfactoria

Tabla 4.6 Prueba de aceptación "Recuperar casos de corpus y algoritmos".

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_4	Nombre Historia de Usuario: Recuperar casos de corpus y algoritmos.
Descripción de la Prueba: Prueba realizada al buscador de casos de comparaciones de corpus y algoritmos cuando se busca un caso de la lista del XML corpus.	
Condiciones de Ejecución: cargada la lista de casos del XML corpus.	
Entrada / Pasos de ejecución: 1. El usuario escribe en el buscador de casos del corpus el nombre del caso que desea buscar. El sistema busca en la lista de casos del XML corpus, realiza una limpieza y muestra todos los resultados de los casos semejantes de la lista de casos del XML corpus.	

2. El usuario borra todo lo que escribió en el buscador. El sistema realiza una limpieza y carga todos los casos de la lista de casos del XML corpus.
Resultado Esperado: encontrar un caso de la lista de casos del XML corpus.
Evaluación de la Prueba: prueba satisfactoria

Tabla 4.7 Prueba de aceptación "Recuperar casos de corpus y algoritmos".

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_4	Nombre Historia de Usuario: Recuperar casos de corpus y algoritmos.
Descripción de la Prueba: Prueba realizada al buscador de casos de comparaciones de corpus y algoritmos cuando se busca un caso de la lista del XML algoritmo.	
Condiciones de Ejecución: cargada la lista de casos del XML algoritmo.	
Entrada / Pasos de ejecución: 1. El usuario escribe en el buscador de casos del algoritmo el nombre del caso que desea buscar. El sistema busca en la lista de casos del XML algoritmo, realiza una limpieza y muestra todos los resultados de los casos semejantes de la lista de casos del XML algoritmo. 2. El usuario borra todo lo que escribió en el buscador de casos del algoritmo. El sistema realiza una limpieza y carga todos los casos de la lista de casos del XML algoritmo.	
Resultado Esperado: encontrar un caso de la lista de casos del XML algoritmo.	
Evaluación de la Prueba: prueba satisfactoria	

Tabla 4.8 Prueba de aceptación "Recuperar casos de corpus y algoritmos".

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_4	Nombre Historia de Usuario: Recuperar casos de corpus y algoritmos.
Descripción de la Prueba: Prueba realizada al recuperar casos de corpus y algoritmos para ver comportamiento del filtro de tipo de plagio de la lista del XML del corpus.	
Condiciones de Ejecución: cargada la lista de casos del XML corpus.	
Entrada / Pasos de ejecución: 1. El usuario selecciona un tipo de plagio del filtro de la lista de casos del XML corpus. El sistema realiza una limpieza y muestra todos casos dependiendo del tipo de plagio seleccionado. 2. El usuario selecciona la opción de mostrar todos en el filtro de tipo de plagios de la lista de	

casos del XML corpus. El sistema realiza una limpieza y carga todos los casos de la lista de casos del XML corpus.
Resultado Esperado: mostrar casos según el tipo de plagio seleccionado de la lista de casos del XML corpus.
Evaluación de la Prueba: prueba satisfactoria

Tabla 4.9 Prueba de aceptación "Recuperar casos de corpus y algoritmos".

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_4	Nombre Historia de Usuario: Recuperar casos de corpus y algoritmos.
Descripción de la Prueba: Prueba realizada al recuperar casos de corpus y algoritmos para ver comportamiento del filtro de tipo de plagio de la lista del XML del algoritmo.	
Condiciones de Ejecución: cargado la lista de casos del XML algoritmo.	
Entrada / Pasos de ejecución: 1. El usuario selecciona un tipo de plagio del filtro de la lista de casos del XML algoritmo. El sistema realiza una limpieza y muestra todos casos dependiendo del tipo de plagio seleccionado. 2. El usuario selecciona la opción de mostrar todos en el filtro tipo de plagios de la lista de casos del XML algoritmo. El sistema realiza una limpieza y carga todos los casos de la lista de casos del XML algoritmo.	
Resultado Esperado: mostrar casos según el tipo de plagio seleccionado de la lista de casos del XML algoritmo.	
Evaluación de la Prueba: prueba satisfactoria	

Tabla 4.10 Prueba de aceptación "Realizar anotaciones de casos revisados".

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_5	Nombre Historia de Usuario: Realizar anotaciones de casos revisados
Descripción de la Prueba: Prueba realizada cuando el usuario realizar anotaciones para ver comportamiento del módulo.	
Condiciones de Ejecución: Casos del XML corpus y XML algoritmos seleccionados, archivos src y susp del XML corpus abiertos, archivos src y susp XML algoritmo abiertos	
Entrada / Pasos de ejecución:	

1. El usuario selecciona un caso del XML del corpus y el caso similar del XML algoritmo. El sistema muestra los atributos del XML algoritmo en los componentes de las anotaciones. Si estos atributos no están en el XML del Algoritmos entonces muestra los atributos del XML del corpus. Si los atributos del XML corpus tampoco se encuentran se activan los componentes para que el usuario realice las anotaciones.
Resultado Esperado: Comportamiento que el sistema le realiza a los componentes
Evaluación de la Prueba: prueba satisfactoria

Tabla 4.11 Prueba de aceptación "Realizar anotaciones de casos revisados".

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_5	Nombre Historia de Usuario: Realizar anotaciones de casos revisados
Descripción de la Prueba: Prueba realizada cuando el usuario realizar anotaciones para ver mensajes de error.	
Condiciones de Ejecución: Casos del XML corpus y XML algoritmos seleccionados, archivos src y susp del XML corpus abiertos y archivos src y susp XML algoritmo abiertos y atributos del XML corpus y del XML algoritmo ausentes.	
Entrada / Pasos de ejecución: 1. El sistema verifica los atributos del XML algoritmo, sino se encuentran entonces verifica los atributos del XML algoritmo y si estos tampoco se encuentran entonces se muestra un mensaje del componente length que no se puede calcular, también muestra un mensaje del componente text_reuse_snipet no se puede calcular	
Resultado Esperado: Mostrar mensajes de error	
Evaluación de la Prueba: prueba satisfactoria	

Tabla 4.12 Prueba de aceptación "Realizar anotaciones de casos revisados".

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_5	Nombre Historia de Usuario: Realizar anotaciones de casos revisados
Descripción de la Prueba: Prueba realizada cuando el usuario termino de realizar una anotación y lo quiere guardar	
Condiciones de Ejecución: Casos del XML corpus y XML algoritmos seleccionados, archivos src y susp del XML corpus abiertos y archivos src y susp XML algoritmo; todas los componentes de las anotaciones ya activados	
Entrada / Pasos de ejecución: 1. El usuario presiona el botón Save. El sistema guarda en el directorio Report en formato XML el caso revisado con el nombre del caso.	
Resultado Esperado: Guardar en XML el caso revisado	
Evaluación de la Prueba: prueba satisfactoria	

Tabla 4.13 Prueba de aceptación "Generar reporte en XML

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_6	Nombre Historia de Usuario: Generar reporte en XML.
Descripción de la Prueba: Prueba realizada al generar reporte cuando el usuario accede a la opción del Report en el menú.	
Condiciones de Ejecución: tener guardados casos revisados en la raíz report.	
Entrada / Pasos de ejecución: 1. El usuario presiona en el menú el botón Report. El sistema muestra la interfaz del Report XML con las carpetas con los casos anotados.	
Resultado Esperado: Abrir la interfaz del Report XML con las carpetas de la raíz del report.	
Evaluación de la Prueba: prueba satisfactoria	

Tabla 4.14 Prueba de aceptación "Generar reporte en XML

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_6	Nombre Historia de Usuario: Generar reporte en XML
Descripción de la Prueba: Prueba realizada al generar reporte cuando el usuario accede a una carpeta en la raíz del report.	
Condiciones de Ejecución: tener guardados casos revisados en la raíz report.	
Entrada / Pasos de ejecución: 1. El usuario selecciona la una carpeta. El sistema lista los casos que se encuentran en la carpeta seleccionada.	
Resultado Esperado: mostrar listar anotaciones de casos revisados	
Evaluación de la Prueba: prueba satisfactoria	

Tabla 4.15 Prueba de aceptación "Generar reporte en XML

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_6	Nombre Historia de Usuario: Generar reporte en XML
Descripción de la Prueba: Prueba realizada al generar reporte cuando el usuario accede a una carpeta en la raíz del report. y seleciona una opción del filtro	
Condiciones de Ejecución: tener listados casos revisados en la raíz report	
Entrada / Pasos de ejecución: 1. El usuario selecciona una opción del filtro. El sistema limpia la lista de casos anotados y muestra los casos anotados según el filtro seleccionado.	
Resultado Esperado: mostrar anotaciones de casos revisados según opción del filtro seleccionado	
Evaluación de la Prueba: prueba satisfactoria	

Tabla 4.16 Prueba de aceptación "Generar reporte en XML

CASO DE PRUEBA DE ACEPTACIÓN	
Código Caso de Prueba: HU_6	Nombre Historia de Usuario: Generar reporte en XML
Descripción de la Prueba: Prueba realizada al generar reporte cuando el usuario genera un reporte XML	
Condiciones de Ejecución: tener listados casos revisados en la raíz report	
Entrada / Pasos de ejecución: 1. El usuario presiona el botón generate. El sistema muestra una interfaz para guardar el reporte en XML. El usuario escribe un nombre al reporte XML y presiona guardar. El sistema guarda el reporte XML	
Resultado Esperado: guardar XML reporte	
Evaluación de la Prueba: prueba satisfactoria	

4.3 Conclusiones parciales.

El capítulo cuatro (validación del sistema) se centró en la realización de pruebas de Caja negra de aceptación al módulo QtNLP-Diff dando cumplimiento a las funcionalidades exigidas por el cliente y logrando un mejor producto final.

Conclusiones

Conclusiones

- Se analizaron las herramientas gráficas y de consola de comparación de textos, demostrándose que estas no se adaptan al proceso de QtNLP al no cumplir con los requisitos para lograr comparaciones de casos de corpus con casos de resultados de algoritmos pues están comparan todo el documento y lo que realmente se compara es un fragmento de textos escogido por atributos.
- Se describió la metodología SXP para realizar el análisis del módulo. Se levantaron los requisitos del sistema reflejados en la pila del producto. Se plasmaron las tareas de ingeniería correspondientes a cada requisito en las historias de usuarios.
- Se realizó la arquitectura de software al módulo a través del uso de las herramientas QtDesigner, PyCharm y PyQt y las tecnologías Python, Qt y XML. Lográndose la integración del módulo al sistema QtNLP. Se utilizó el patrón de arquitectura Modelo Vista Controlador facilitando el diseño e implantación del módulo. Con la implementación del módulo QtNLP-Diff se automatizó el proceso de comparación de casos de corpus con casos de resultados de algoritmos.
- Se realizaron pruebas de caja negra de aceptación al módulo QtNLP-Diff optimizándose la calidad del producto cumpliendo satisfactoriamente con los requisitos del cliente.

Recomendaciones

Recomendaciones

- Utilizar el módulo QtNLP-Diff para agilizar el proceso de comparaciones de casos de corpus con casos de resultados de algoritmos, pues este proceso es complejo e ineficiente para usuarios con poca capacidad en informática.
- Dar seguimiento a la propuesta para atender posibles recomendaciones y necesidades de los usuarios del sistema, una vez implementado.

Bibliografía

Bibliografía

BODDIE, D. 2015. *About PyQt* [Online].

CANONICAL. 2011. *Projects using Bazaar* [Online]. Available: http://doc.bazaar.canonical.com 2016].

CARBONELL, J. 1992. El procesamiento del lenguaje natural, tecnología en transición.

CARMONA, M. A. A. 2014. Detección de similitud semántica en textos cortos.

CLOVER, A. 2008. *pxdom 1.6* [Online].

COMMUNITY, T. S. 2014a. *NumPy v1.11 Manual* [Online].

COMMUNITY, T. S. 2014b. *SciPy* [Online]. [Accessed 2016.

DHAMARYZ 2010. Pruebas de Caja Negra.

GIRALDO, O. P. 2007. Ingeniería de Requisitos. *Requisito no funcional.*

GRADY BOOCH, J. R. 2005. EL lenguaje unificado del Modelado. *Diagrama de componentes.*

JEFFRIES, R., ANDERSON, A., HENDRICKSON, C. 2001. Extreme Programming Installed.

JIM, K. 2009. Python.

JOACHIM, E. 2014. *KDiff3 - Home* [Online].

LACALLE, A. 2006. *Que es un prototipo* [Online]. 2016].

LARS, K. 2011. The Qt Project is live!

MARTIN POTTHAST, P. R., BENNO STEIN, 2010. Corpus and Evaluation Measures for Automatic Plagiarism Detection.

MARTIN POTTHAST, S. B., BENNO STEIN 2012. Paraphrase Acquisition via Crowdsourcing and Machine Learning.

MCCANN, W. J. 2013. *Dia_Faq* [Online].

MENESES, A. & VIDEAUX, S. 2015. QtNLP: Editor de Corpus Lingüísticos.

MENESES, A. A. 2014. *ToNgueLP alpha-0.1 DOC* [Online].

MIKELIONIENĖ, J. 2002. KOMPIUTERINĖ LINGVISTIKA.

NEEDLEMAN, S. B. & WUNSCH 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins.

OMG, U. M. L. 2007. UML Infrastructure Specification. *Diagrama de paquetes.*

PEÑALVER, G. R. 2008. MA-GMPR-UR2 Metodología ágil para proyectos de software libre.

PRESSMAN, R. S. 2002. Ingeniería de Software, un enfoque práctico.

PRUEBASDESOFTWARE. 2015. *La prueba de aceptación* [Online].

REENSKAUG, T. & COPLIEN, J. O. 2009. The DCI architecture: A new vision of object-oriented programming.

REPORTLAB 2012. PyRXP.

ROSSO, P. 2012. Algoritmos bio-inspirados aplicados a problemas de Procesamiento del Lenguaje Natural.

ROUSE, M. 2016. *What is XML (Extensible Markup Language)* [Online]. 2016].

SMITH, T. & WATERMAN, M. 1981. Identification of common molecular subsequences.

STEIN, B. 2005. Fuzzy-fingerprints for text-based information retrieval.

THE MATPLOTLIB, D. T. 2014. *Matplotlib 1.5.1 documentation* [Online].

THE QT COMPANY. 2015. *Qt Designer Manual* [Online].

TUTORIALPOINT. 2016. *XML - Estructura de Árbol* [Online].

UCHE, O. 2003. *xmltramp* [Online].

VARUN GROVER, W. J. K. 2000. Process Think: Winning Perspectives for Business Change in the Information.

VENEMEDIA. 2015. *Definición de Comparación* [Online].

WILLADSEN, K. 2012. *Meld* [Online]. Available: http://meldmerge.org/ 2016].

YOGRATEROL. 2014. *PyCharm: El mejor IDE para tus proyectos en Python* [Online].