UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

PROGRAMA DE DOCTORADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

SOFTWARE PARA DETECCIÓN AUTOMÁTICA DE SIMILITUD EN DOCUMENTOS DE INVESTIGACIÓN EN LA UNIVERSIDAD NACIONAL DEL ALTIPLANO PUNO 2016

PRESENTADA POR:

ELVIS AUGUSTO ALIAGA PAYEHUANCA

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTORIS SCIENTIAE EN CIENCIAS DE LA COMPUTACIÓN

PUNO, PERÚ

2017

UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

PROGRAMA DE DOCTORADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

SOFTWARE PARA DETECCIÓN AUTOMÁTICA DE SIMILITUD EN DOCUMENTOS DE INVESTIGACIÓN EN LA UNIVERSIDAD NACIONAL DEL ALTIPLANO PUNO 2016

PRESENTADA POR:

ELVIS AUGUSTO ALIAGA PAYEHUANCA

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTORIS SCIENTIAE EN CIENCIAS DE LA COMPUTACIÓN

APROBADA POR EL SIGUIENTE JURADO:

PRESIDENTE

.....

Dr. JUAN REYNALDO PAREDES QUISPE

PRIMER MIEMBRO

.....

Dr. HENRY IVÁN CONDORI ALEJO

SEGUNDO MIEMBRO

.....

D.Sc. YALMAR TEMISTOCLES PONCE ATENCIO

ASESOR DE TESIS

.....

Dr. ANGEL MANUEL OLAZABAL GUERRA

ÁREA: Ciencias de la computación
TEMA: Desarrollo de software antiplagio en la investigación
LÍNEA: Verificación, validación, testing y análisis.

Puno, 23 de Enero del 2017.

DEDICATORIA

Con mucho cariño para mis queridos padres Alcides Augusto Aliaga Coaquira y Lucila Payehuanca Justo (Q.E.P.D), para mis queridas hermanas Karina e Ibeth y para mis sobrinos Joaquín y Romina.

AGRADECIMIENTOS

- A la Universidad Nacional del Altiplano por la formación tanto a nivel de pregrado y posgrado que he recibido.
- A los miembros del Jurado Dr. J. Reynaldo Paredes Quispe, Dr. Henry I. Condori Alejo, D.Sc. Yalmar T. Ponce Atencio y Dr. Angel M. Olazabal Guerra; por todas sus sugerencias y correcciones en el trabajo de investigación.
- Al director general de investigación de la Universidad Nacional del Altiplano – Puno, M.Sc. Ernesto N. Tumi Figueroa por brindarme su apoyo y facilidades para la experimentación del Software en las instalaciones del vicerrectorado de investigación.

ÍNDICE GENERAL

ÍNDICE DE CUADROS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

El trabajo de investigación es una propuesta de un Software para la verificación y detección de manera automática de la similitud en los trabajos de investigación en la Universidad Nacional del Altiplano de Puno. Los procesos para la detección de la similitud se divide en tres partes, en primer lugar se realiza las tareas de pre-análisis, que consiste en la conversión de los documentos en los formatos PDF, DOC a un archivo de texto plano, una vez echo eso inmediatamente se realiza la eliminación de los espacios mayores a uno, la conversión a minúsculas y la eliminación de caracteres especiales. La segunda parte consiste en un análisis a priori, en la que utilizamos algoritmos no tan costosos para determinar si el documento en cuestión es posiblemente plagiado. Finalmente se tiene el análisis riguroso que tienen como entrada todos los documentos que pasaron la primera fase y a través de estos procesos se va a determinar cuál es el nivel de similitud del documento solicitado para el análisis, esto se hará expresándolo en porcentaje, así mismo se muestra cuáles son los archivos fuentes de esas coincidencias. Para realizar las pruebas iniciales y posteriormente la puesta en marcha del Software, se tomó los proyectos de tesis de pregrado que actualmente se tienen presentados a partir del semestre 2016-I en la UNAP, el Software, para detectar la similitud, realiza comparaciones a través de algoritmos de detección de similitudes contra un repositorio local de trabajos de investigación y se tuvo como resultado que el 5% de los documentos verificados contienen por lo menos un 20% de similitud con los documentos del corpus de referencia.

Palabras clave: algoritmo, automático, detección de similitud, investigación, repositorio.

ABSTRACT

The research work is a proposal of a Software for the automatic verification and detection of similarity in the research works at the National University of the Puno Highlands. The processes for the detection of similarity is divided into three parts, we first perform the pre-analysis tasks, which consists of the conversion of documents in PDF, DOC formats to a flat text file, once echo That immediately removes the spaces greater than one, the conversion to lowercase and the elimination of special characters. The second part consists of an a priori analysis, in which we use algorithms not so expensive to determine if the document in question is possibly plagiarized. Finally we have the rigorous analysis that have as input all the documents that passed the first phase and through these processes will determine the level of similarity of the document requested for the analysis, this will be done expressing it in percentage, Shows the source files of those matches. In order to carry out the initial tests and later the start-up of the Software, the undergraduate thesis projects that have been presented since the semester 2016-I in the UNAP, the Software, were taken to detect similarity, make comparisons through Of similarity detection algorithms against a local repository of research papers and it was found that 5% of the documents verified contain at least 20% similarity with the documents of the reference corpus.

Keywords: algorithm, automatic, detection of similarity, investigation, repository.

INTRODUCCIÓN

El trabajo de investigación tiene como finalidad de mejorar el proceso de verificación y determinación de plagio en todos los documentos de investigación presentados en la Universidad Nacional del Altiplano de Puno, para lo cual lo hemos dividido en cuatro capítulos, los cuales están divididos de la siguiente manera:

En el Capítulo I, se refiere a la problemática de investigación, es decir al planteamiento y formulación del problema, los antecedentes de la investigación, papers referentes al tema de investigación, el objetivo general y objetivos específicos de la investigación.

En el Capítulo II, se desarrolla el marco teórico referidos los temas del estudio de los algoritmos para la detección de plagio, aplicaciones web, metodología de desarrollo ágil XP, herramientas y tecnología actual para el desarrollo de software tales como AngularJS, NodeJS, PHP, C++, también tenemos la definición de los términos básicos utilizados en la investigación y finalmente se formuló la hipótesis de la investigación.

En el Capítulo III, se presenta la metodología utilizada en el trabajo de investigación; también se especifican el tipo y diseño de investigación y finalmente se tiene la población y muestra utilizada en la investigación.

En el Capítulo IV, se exponen los resultados de la investigación, donde se consideran la evaluación de los procesos que actualmente utiliza la Universidad para la detección de plagio y también analizamos el proceso de detección de plagio considerando la utilización de un software que nos permita realizar éste trabajo de manera automática. Como parte del trabajo de investigación de la nueva propuesta de detección de plagio a través de

un software se hace el análisis de algoritmos existentes y posteriormente se presenta una propuesta de mejoras que se pueden aplicar para tener mejores resultados en las comparaciones de texto; posterior a ello se muestra el desarrollo (análisis, diseño e implementación, pruebas) de la plataforma de software para la verificación y detección de plagio en los trabajos de investigación en la Universidad Nacional del Altiplano y para garantizar la calidad del Software y verificar la satisfacción de los usuarios se hizo uso de la norma ISO/IEC 9126.

Y finalmente tenemos las conclusiones a las cuales se ha llegado con la investigación, las recomendaciones y anexos.

CAPÍTULO I

PROBLEMÁTICA DE INVESTIGACIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

En la actualidad, la violación de los límites de la propiedad intelectual y el plagio es pan de cada día, basta con aplicar un copiar y pegar, esto se refleja en todo tipo de trabajos de investigación, éste accionar es gravísimo si se realiza en los trabajos de investigación con la que una persona va a obtener un título profesional o un grado académico.

Es un problema cada vez más recurrente en los ámbitos académicos, profesionales y políticos, tal como se dio el caso en las últimas elecciones presidenciales con un candidato a la presidencia de la república de nuestro país, en la que pusieron en tela de juicio la originalidad de su trabajo de investigación para la obtención del grado de doctor, razón por la cual le sacaron de carrera a sus aspiraciones de conseguir la presidencia de la república.

En todas las Universidades de la Región Puno (públicas y privadas), no se tiene un control de sí un proyecto o una de tesis es plagiado, ni siquiera realizan comparaciones con trabajos de investigación presentados en la misma Universidad (repositorio de tesis) y menos en otras Universidades de la región. Lo anterior es porque casi en la totalidad de universidades no existen

repositorios digitales, pero muchas universidades ya están iniciando trabajo en la digitalización de las tesis creando repositorios locales de todos sus trabajos de investigación a nivel de pregrado y postgrado, éstos repositorios solo se están utilizando a manera de consultas como referencias.

Existen algunos software que realizan el trabajo de verificar la originalidad de un determinado documento tales como: *plagscan, docode, viper*, *etc*. que son herramientas de pago, y si es adquirido en éste caso por la Universidad Nacional del Altiplano, estaría muy dependiente de la empresa desarrolladora de dicho software y para obtener actualizaciones se tiene que desembolsar fuertes cantidades de pagos adicionales y de lo que se trata de tener en una universidad pública herramientas propias de tal manera que en cualquier momento podamos modificarlas y actualizarlas para obtener versiones mejoradas y que las verificaciones se haga contra los repositorios locales de la Universidad.

Actualmente la Universidad Nacional del Altiplano en el semestre 2016-I inició con la implementación de una plataforma para la automatización de todo el proceso para la presentación y gestión de las tesis en pregrado, se registra los datos principales del proyecto de tesis en una aplicación web y se anexa el proyecto en un formato PDF en un esquema pre establecido por el vicerrectorado de investigación de la UNA-PUNO, pero solo se verifican aspectos de forma, mas no se hacen verificaciones si dicho perfil de proyecto de tesis presenta copias de otros trabajos de investigación con la que se graduaron en la Universidad Nacional del Altiplano, sin esa verificación que es muy importante se dan como aceptados y se procede directamente a la asignación de los jurados respectivos y lo mismo ocurre con los borradores de tesis, no son verificados en lo que referente a plagio.

1.2 FORMULACIÓN DEL PROBLEMA

¿Es posible la detección de plagio en documentos de investigación en la Universidad Nacional del Altiplano haciendo uso de una plataforma de software con un corpus de referencia?

1.4 OBJETIVOS DE LA INVESTIGACIÓN

1.4.1 Objetivo general

Implementar una plataforma de software para la verificación y detección de plagio en documentos de investigación con un corpus de referencia en la Universidad Nacional del Altiplano – Puno.

1.4.2 Objetivos específicos

a. Analizar los procedimientos que se realizan en la actualidad y haciendo de un software con un corpus de referencia para determinar si en los trabajos de investigación presentados en la Universidad Nacional del Altiplano existen o no plagio.
b. Implementar y evaluar la plataforma de software para la detección de plagio con un corpus de referencia en trabajos de investigación en la Universidad Nacional del Altiplano.

1.5 ALCANCES Y LIMITACIONES

1.5.1 Alcances

El presente trabajo de investigación está dirigida principalmente para su uso en la oficina de investigación general de la Universidad que es una dependencia del vicerrectorado de investigación (VRI), que es la encargada de la gestión de todos los trabajos de investigación que se presentan en la Universidad a nivel de pregrado, referente a postgrado la encargada de las revisiones de las tesis es la oficina de investigación de

la escuela de postgrado. También puede ser utilizada por las coordinaciones de investigación de cada una de las facultades de la Universidad.

1.5.2 Limitaciones

El presente trabajo de investigación se limitó en el ámbito de todos los trabajos de investigación presentados a nivel de pregrado y postgrado en la Universidad Nacional del Altiplano - Puno, y el repositorio principal con la que se realiza las comparaciones de plagio es la que se tiene en vicerrectorado de investigación implementado en coordinación con CONCYTEC (consejo nacional de ciencia, tecnología e Innovación tecnológica), puesto de lo que se trata en ésta primera versión es de poner más énfasis en los algoritmos utilizados para detección de plagio y es por esa razón que se trabaja con un corpus de referencia.

CAPÍTULO II

MARCO TEÓRICO

1.3 ANTECEDENTES DE LA INVESTIGACIÓN

A continuación se mencionan los antecedentes de trabajos de investigación realizados a nivel nacional y mundial.

Para (Jacobs, 2010), la investigación plantea intentar responder a la pregunta ¿qué factores propician el plagio en la elaboración de trabajos académicos realizados fuera del aula por parte de alumnos de la UPC? Dada la naturaleza de la pregunta, el equipo de investigación no se planteó hipótesis, sino que más bien se propuso un estudio exploratorio para descubrir cuáles son los mencionados factores. Como conclusión general de esta investigación, se propone que los esfuerzos de prevención, control y análisis de los problemas de plagio dependen principalmente de los propios docentes. Y, en tanto ellos parecen estar un poco menos informados que los alumnos, cualquier estrategia debe involucrarlos directamente.

(Weber-Wulff, 2013), en su artículo Why My Students Don´t Plagiarise, describe la manera como diseña su curso de tal forma que evita el plagio. La idea central de MacDonald es que los alumnos deben tener una clara comprensión de que el aprendizaje en la universidad no significa reproducir datos sino desarrollar habilidades intelectuales. En el desarrollo de su caso, MacDonald explica cómo

su curso gira alrededor de seminarios y tutorías personalizadas en lugar de clases magistrales, los trabajos son presentados en varios borradores que pueden ser corregidos por la clase en su totalidad y además se hace uso de herramientas tecnologías como TurnItIn.com. Todo esto permite a los alumnos aprender en el proceso, crear y corregir hasta lograr un producto propio. El resultado fue el esperado, no se produjo el plagio.

(Patil, 2015), afirma que el uso indebido del Internet es un fenómeno que está incrementándose rápidamente y para el cual, al menos algunos docentes, no parecen estar preparados. De acuerdo con el artículo What`s the answer?, mientras que la mayoría de docentes está optando por eliminar todo recurso tecnológico de uso de los alumnos durante procesos de evaluación, en un esfuerzo por combatirel problema, los docentes de cerca de 900 universidades y 3200 escuelas secundarias en Estados Unidos están haciendo uso de servicios como TurnItIn, una herramienta tecnológica capaz de comparar los trabajos de los alumnos con archivos de hasta 6 billones de páginas. Sin embargo, ¿es esta la respuesta adecuada?, ¿en realidad ataca el problema?.

Las universidades también están realizando un importante trabajo que se puede llamar de prevención al tener informados a sus alumnos acerca de la definición de plagio y de las consecuencias de cometerlo. La Universidad de Maryland cuenta con un juramento de honor, documento que los alumnos deben firmar a su ingreso a la institución, además de contar con un código de integridad académica y de folletos informativos. La Universidad de Indiana incluye la definición de plagio académico dentro delos datos de los sílabos de los cursos. La Universidad de la Américas de Puebla cuenta con un manual de integridad académica. La mayoría de universidades en Perú, como la UPC, la Universidad

de Lima y la Pontificia Universidad Católica del Perú, cuentan con pautas para la sanción del plagio. La Universidad de Alberta cuenta con una página web dedicada a la promoción de la integridad académica que incluye sugerencias acerca de cómo incluir el concepto de plagio en, sugerencias que incluyen: clarificación de conceptos, sílabos de los cursos, hablar del tema en clase, recomendaciones a los docentes acerca de la manera en la que se debe diseñar las tareasw, etc.

Existe una diversidad de software que nos permite detectar el nivel de plagio en un determinado documento, pero casi en su totalidad para su uso se debe hacer el pago, el tipo de licencia que otorgan son por la cantidad de estudiantes, imaginemos el monto de dinero que se tiene que pagar si es que se quiere usar dicho Software en una determinada Universidad donde la cantidad de estudiantes en bastante y eso multiplicado con la cantidad de trabajos de investigación que elaboran cada uno de ellos.

2.1 SUSTENTO TEÓRICO

2.1.1 Plagio

Según la RAE (Real Academia Española de la Lengua), plagiar es: “copiar en lo sustancial obras ajenas dándola como propias”.

2.1.2 Detección de plagio

La investigación referente a la detección de plagio ha crecido enormemente en los últimos años, la mayoría de las investigaciones quedan desactualizadas en un periodo de tiempo corto. Debido a las distintas formas de plagio unos más elaborados que otros, podemos encontrar plagios con permutación, borrado o agregado de contenido (de cierto tamaño, párrafo o textos más extensos).

Cualquier sistema de detección de plagio debe cumplir ciertas propiedades que lo hagan estable frente a cualquier tipo de plagio, (Schleimer, Wilkerson & Aiken, 2003) proponen las siguientes propiedades:

- No ser sensible a los espacios, signos de puntuación o mayúsculas.
- Supresión del ruido: los pasajes considerados deben tener un cierto tamaño, de forma tal de estar seguros de que se trata de un plagio y descartar coincidencias producidas por expresiones idiomáticas comunes.
- Independencia de la posición: la permutación, borrado o agregado de contenido (de cierto tamaño, párrafos o pasajes más extensos) no debería afectar la detección de plagio.

Así la detección automática de plagio puede abarcar distintos problemas, como menciona, (Elizalde, 2011):

- Detección de plagio con un corpus de referencia, mediante comparación de documentos.
- Detección sin corpus de referencia o estilometria, analizando el texto para descubrir inconsistencias en el estilo y/o vocabulario. Este problema está ligado al de la atribución de autoría: decidir si un texto pertenece a un autor, analizando las características de su escritura en otros textos.
- Detección de plagio en distintos idiomas, generalmente entre inglés y otro idioma.
- Dado un plagio, determinación de la dirección, es decir quien plagio a quien.

2.1.3 Metodologías para la detección de plagio con corpus de referencia

Las metodologías para la detección de plagio con corpus de referencia en general se basan en comparaciones entre los documentos. Al existir un corpus de referencia se realizan ciertos análisis globales que permitan detectar plagios más elaborados, es decir, con permutación, eliminación o agregado de contenido al texto.

Normalmente se tienen corpus de referencia muy grandes (millones de documentos), por lo tanto se realiza una fase preliminar de análisis en el cual se determina la similitud de cada uno de los documentos del corpus de referencia con el documento analizado, aquellos que tengan una similitud que supere un umbral mínimo se consideran potenciales candidatos a ser la fuente de donde se hizo el plagio, de esa manera se reduce enormemente la cantidad de documentos a analizar detalladamente que se realiza en la fase detallada o final.

En la fase detallada es donde se realiza un análisis más riguroso sobre el conjunto de documentos que pasaron el umbral mínimo de similitud con el documento analizado obtenido en la fase preliminar. Así se realiza una comparación del documento analizado con cada uno de los que pertenecen al corpus de referencia y pasaron la fase preliminar. En esta fase se detectan los pasajes de texto plagiado en cada uno de los documentos y se determina el porcentaje de similitud entre los documentos.

Los sistemas de detección de plagio con corpus de referencia tiene en general la misma estructura. (Stein & Rosso, 2010) describen la

estructura:

- Se analizan los documentos del corpus de referencia y se calcula un hash o una representación de algún tipo (por ejemplo un vector de palabras) para cada uno de ellos.
- Se almacenan estas representaciones en un índice invertido.
- Cuando se desea verificar si un documento contiene plagio, se calcula su representación y se la compara contra las que están en el índice, obteniendo un coeficiente que denota la similitud.
- Se seleccionan los archivos más cercanos para hacer una comparación más detallada.
- La comparación detallada produce como resultados los pasajes que se sospecha fueron plagiados.
- Podría haber eventualmente una fase de post-procesamiento: volver a analizar los pasajes obtenidos con otro modelo para corroborar que se trata de plagio, filtrar los pasajes muy cortos, etc.

2.1.4 Métodos sintácticos

Estos métodos consideran a la unidad de comparación que se determina previamente (palabras, frases, párrafos, etc.) únicamente como símbolos, sin darles significado. Esto tiene la ventaja de detectar plagios más elaborados al aislar los símbolos, pero su desventaja es que el significado de las unidades de comparación puede variar según el contexto en el que se encuentren.

Estos métodos usan emparejamiento de cadenas para detectar y medir la similitud entre los documentos, también modelos de espacio vectorial junto con fingerprints. Así tenemos los principales métodos sintácticos:

- Modelo de espacio vectorial.
- Fingerprinting
- N-gramas

2.1.5 Métodos semánticos o lingüísticos

Estos métodos son más complejos que los métodos sintácticos ya que las unidades de comparación (palabras, frases, párrafos, etc.) no son tratados como símbolos, sino que son relacionados con otros términos según el significado (como sinónimos, antónimos, homónimos, etc.). Su implementación es más complicada que los sintácticos, por lo que la investigación en esta área es mucho menor que los métodos semánticos. La mayoría de los métodos analizan los textos a nivel de frases y utilizan estos métodos para un análisis detallado, ya que el costo de este enfoque es más alto que los sintácticos, esto se ve en las investigaciones (Alzahrani & Salim, 2010), (Tsatsaronis, Varlamis, Giannakopoulos & Kanellopoulos, 2010) y (Muftah & Jabr, 2009).

2.1.6 Modelo de espacio vectorial

El modelo de espacio vectorial surge del área de recuperación de información (Information Retrieval, en inglés), y es una manera de representar objetos en general (para el caso de detección de plagio, el texto que contienen los documentos). Los objetos son representados por un vector en donde las componentes del vector son las características esenciales del objeto cuantificadas, así la cantidad de componentes, es decir, la cantidad de dimensiones del vector depende de las características esenciales que son necesarias para representar el objeto. Mientras más complejo sean los objetos la cantidad de dimensiones

aumenta con lo cual la complejidad de representarlas y realizar operaciones sobre estas para poder determinar ciertas relaciones entre dos objetos.

Generalmente se representan las componentes del vector como términos del documento y sus frecuencias. Como peso se suele utilizar el tf-idf, esto es term frequency-inverse document frequency. Este enfoque considera a las palabras menos frecuentes como más relevantes del texto, por ejemplo, la palabra “de” en español es muy frecuente y por lo tanto su presencia no es muy significativa, en cambio la palabra “algoritmo” en cambio no es muy frecuente y su presencia es mucho más significativa.

Posteriormente se usa una función de distancia entre dos vectores para determinar el grado de similitud entre los documentos. Esta medida se puede obtener con el coseno del ángulo entre los vectores, además se pueden considerar muchas funciones de distancias desde estadísticos hasta enfoques matemáticos. (Rehurek, 2008) propone que este modelo conlleva tres elecciones distintas: que característica utilizar para representar el texto, los pesos que se le asigna a cada dimensión y la medida de distancia entre vectores.

Al representar de esta manera un documento hace que las componentes del vector sean vistas como independientes entre sí, esto significa que las palabras tienen el mismo significado sin importar el contexto, lo cual no necesariamente es cierto. Por eso es un método sintáctico, ya que trata a la unidad de comparación (en este caso las componentes del vector) como símbolos. Así no se tiene en cuenta el orden de la unidad

de comparación, esto tiene un lado positivo y otro negativo. Al no tomar en cuenta el orden de la unidad de comparación (palabras, frases, párrafos, etc.) es posible detectar plagios más elaborados, en el otro lado se pierde información al no poder considerar el orden de las palabras para la detección de plagio.

La complejidad de los cálculos sobre vectores depende de la cantidad de dimensiones, a medida que los documentos aumentan en longitud tienen más componentes, por lo que se usan métodos para reducir las cantidad de dimensiones y únicamente considerar las más relevantes (Rehurek, 2008) y (Zechner, Muhr, Kern & Granitzer, 2010), proponen y usan métodos para reducir la cantidad de componentes del vector.

(Raghavan & Wong, 1986) hacen un análisis crítico sobre el modelo de espacio vectorial explicando sus limitaciones en el área de recuperación de información (Information Retrieval, en inglés), refieren que dado la matriz de términos de los documentos y la interpretación de ellos como componentes de vectores de los documentos, el modelo no es completo. Una forma de hacer esto completo es tener una manera para determinar la similitud componente a componente.

El modelo de espacio vectorial asume que los plagios ocurren en documentos de un mismo tópico, lo cual dejaría sin efecto algunos casos en los que un autor hace mención superficialmente de otra disciplina (probablemente haciendo plagio) como menciona (Stein & Meyer, 2006).

El modelo de espacio vectorial es usado para la detección de plagio a partir de la determinación de similitud entre los documentos (los cuales son representados como vectores) como se describió, pero no señala

donde ocurre exactamente el plagio. Para resolver esto (Zechner, Kern & Granitzer, 2004) usan un vector por frase, así las que superan un cierto umbral mínimo de similitud son consideradas plagio. Este enfoque no es capaz de detectar la permutación de frases plagiadas, es decir, plagios más elaborados, además tener un texto dividido en frases tiene problemas, tener como delimitadores a puntos, índices, no es suficiente.

El modelo de espacio vectorial puede ser usado como una primera selección de documentos del corpus de referencia, luego a partir de esta información establecer ciertas líneas del documento como posibles plagios y finalmente hacer una función de comparación entre estas líneas sospechosas como refieren (Devi, Rao, Sundar & Akilandeswari, 2010).

Debido al alto costo de realizar operaciones sobre vectores (especialmente en textos; gran cantidad de componentes) se puede evaluar la similitud de dos documentos, si esto supera un cierto valor mínimo, entonces podemos realizar una comparación más detallada, es decir, dividir el documento en partes más cortas y volver a calcular la similitud, esto se hace recursivamente, este enfoque es usado por (Stein & Meyer, 2006).

2.1.7 Fingerprinting

Es una técnica muy utilizada, consiste en la creación de hashes para porciones de texto. Formalmente es una pequeña secuencia de bytes que caracterizan un archivo muy largo. Por ejemplo fingerprints puede ser obtenido aplicando cualquier función hash a un archivo. En sistemas de detección de plagio fingerprints son normalmente más avanzados que simples código hash, cada fingerprint contiene varios atributos numéricos

que reflejan la estructura del documento. Atributos comunes son la frecuencia de palabras, la cantidad media de palabras por línea, el número de palabras únicas. Si dos fingerprints son muy cercanos (de acuerdo a algún criterio, usualmente se usan funciones de distancia como el coseno) los documentos correspondientes son tratados como similares.

Las funciones de hash criptográficas tienen el denominado efecto avalancha, es decir, ante un cambio de algún bit en la entrada, la salida será drásticamente diferente. Es decir si una parte de texto difiere muy poco, esta tendrá un hash completamente diferente, esto tiene un lado positivo y negativo. No será posible tener longitudes de partes de texto grandes, ya que al diferir mínimamente esto se considerara como diferentes y no podríamos hacer uso de la definición de textos similares, en donde ocurre normalmente un plagio más elaborado, permutando, agregando o quitando texto. A la vez cuanto menor sea la longitud de las porciones de texto, más costoso será el análisis (mayor tiempo de pre procesamiento y mayor espacio de almacenamiento).

(Kasprzak, Brandejs & Kripac, 2009) usan como longitud de partes n-gramas de palabras o caracteres, es decir, n palabras o caracteres consecutivos. Se toman solo algunas partes y luego se comparan, ya que mantener todas las partes es muy costoso. Para resolver esto (Schleimer, Wilkerson & Aiken, 2003) proponen el algoritmo Winnowing para hashing de k-gramas el cual mejora el almacenamiento y la comparación (reduce la cantidad de hashing almacenados).

La metodología fingerprinting ha sido criticado a lo largo de los años, por ejemplo (Stein & Eissen, 2006) critican a los fingerprints por ser computacionalmente caras y señalan que la longitud de las porciones de texto tiene que ser pequeña, ellos proponen el uso de fuzzy fingerprints.

Fuzzy fingerprints es un método que combina fingerprints y el modelo de espacio vectorial, en este enfoque en lugar de tomar directamente los pasajes del documento, usan una representación en forma de vector de los mismos. (Stein, 2005), divide los términos que aparecen en un documento en clases de equivalencia de forma tal que un término corresponde a una clase si comienza con cierto grupo de prefijos, para formar el vector, se calcula la frecuencia de cada clase de equivalencia para cada longitud de porción, y se almacena la desviación de estas con respecto a los valores calculados en base al corpus. Finalmente se aplica un método de dispersión al vector y se obtiene un hash de cada longitud de porción. Una vez construido el hash del documento, la comparación se hace de igual manera que en el caso de las fingerprints. Una colisión en el hash indica que la porción de texto correspondiente ha sido plagiado.

2.1.8 N-gramas

Ha sido ampliamente usado en la detección de plagio durante los últimos años, usualmente se toman n-gramas (es decir n caracteres o palabras consecutivas), se calcula sus frecuencias, hay veces únicamente el universo de n-gramas y finalmente se evalúa con alguna función de distancia.

Este enfoque se puede ver como una combinación del modelo de espacio

vectorial y fingerprinting, aprovechando las bondades que ofrecen ambos, el conjunto de n-gramas obtenidos es posible interpretarlo como un hash y también se puede ver como un vector, donde las frecuencias de los n-gramas son los pesos y luego se usa una función de distancia.

Usualmente se usan bigramas, trigramas, cuatro-gramas y cinco-gramas de palabras que representan las características del espacio de modelo vectorial.

(Cedeño & Rosso, 2009) usan bigramas y trigramas, (Ferret, 2004) se usan trigramas, mientras que (Clough, 2010) y (Kasprzak, 2009) usan cinco-gramas.

(Shivaji, 2015) usa los algoritmos de Karp Rabin y string matching para la detección de plagio. Karp Rabin normalmente es usado para la obtención de los n-gramas de una manera eficiente, debido a la reutilización de información, pero esto podría generar colisiones.

Para lidiar con esto, además del valor hash se puede agregar información adicional como las letras más frecuentes y si hay empates las lexicográficamente menores para poder reducir considerablemente la existencia de colisiones y mejorar la complejidad del algoritmo que tiene como unidad de comparación las palabras del texto. Esta es la idea usada en el algoritmo que se propone en el presente proyecto de investigación.

2.1.9 Karp Rabin

Para el procesamiento de cadenas la mayoría de los algoritmos usan hashing para poder comparar textos, esto debido a su simplicidad y su complejidad esperada, a cada unidad de comparación (que pueden ser letras, palabras, frases, párrafos, entre otros) se les asigna una clave el

cual debería identificar únicamente a la unidad de comparación. De esa manera cuando se quieren comparar las unidades únicamente nos referimos a sus claves y no propiamente a la palabra, frase o párrafo, esto significa menor costo de comparación.

Sin embargo el problema surge con la función que confiere a cada unidad de comparación un hash, es decir podrían existir dos unidades de comparación U1 y U2 diferentes con el mismo hash, a esto se le conoce como colisión en la función hash. Existen varias maneras de poder resolver esto, unas usando mayores cantidades de almacenamiento y otros aumentando la complejidad.

El algoritmo de Karp Rabin usa polinomios para la función hash y un módulo sobre el cual se define el universo de las hashes que se les asigna a cada unidad de comparación. El algoritmo de Karp Rabin puede reducir la complejidad de calcular los hashing de las porciones de texto, debido a la reutilización de información procesada como la porción de texto anterior, ya que podrían diferir únicamente en un carácter. El cálculo es rápido, pero se pierde en alguna medida el efecto avalancha, es decir al alterar un byte en la entrada, la salida podría no variar drásticamente. Se puede lograr un mejor desenvolvimiento si se usan no solamente hashing, podemos usar información adicional para tener el efecto avalancha y una complejidad de cálculo rápida.

(Kiran, 2015) usa Karp Rabin y string matching para la detección de plagio, usan la categoría de izquierda a derecha para procesar los textos, a partir del cual hacen uso de información previamente calculada para obtener los hashes de porciones de texto junto a algoritmos de string

matching, se tiene una precisión del 85% y minimiza el porcentaje de detección fallida de plagio en 10%.

2.1.10 Programación dinámica

La programación dinámica es una metodología muy usada para la reducción de complejidades en algoritmos, así un problema con complejidad no polinomial, puede ser resuelta en un tiempo polinomial usando programación dinámica. En la detección de plagio con corpus de referencia, normalmente no se hace mención del uso de este enfoque, ya que es usado como una optimización de los algoritmos y no propiamente como el enfoque que tiene.

Por ejemplo para el algoritmo de Rabin Karp una manera eficiente de calcular la función hash de porciones de texto es haciendo un pre procesamiento de todo el texto, para luego reutilizar información calculada para obtener el hash de porciones de texto, esto hace que sea eficiente. Es decir se hace el uso de información previamente calculada y almacenada, evitando el recalculo innecesario, esto es, programación dinámica.

Similarmente en muchas partes de los algoritmos usados para la detección de plagio se hace uso de la programación dinámica.

2.1.11 Calidad

Conjunto de propiedades y características de un producto o servicio, que le confieren aptitud para satisfacer unas necesidades explícitas o implícitas (ISO 8402).

2.1.12 Calidad de software

La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario (IEEE, Std. 610-1990).

2.1.13 ISO/IEC 9126

Esta norma Internacional fue publicado en 1992, la cual es usada para la evaluación de la calidad de software, llamado “Information technology – Software product evaluation - Quality characteristics and guidelines for their use”; o también conocido como ISO 9126 (o ISO/IEC 9126).

Este estándar describe 6 características generales:

Funcionalidad, textualmente la define: “A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied set of users”.

- **Confiabilidad,** textualmente la define: “A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time”.
- **Usabilidad,** textualmente la define: “A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users”.

Eficiencia, *q*ue textualmente la define: “A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions”.

- **Mantenibilidad**, textualmente la define: “A set of attributes that bear on the mffort needed to make specified modifications”.

Portabilidad, textualmente la define: “A set of attributes that bear on the ability of software to be transferred from one environment to another”.

2.1.14 Modelo de calidad (ISO/IEC 9126)

La ISO/IEC 9126 permite especificar y evaluar la calidad del software desde diferentes criterios asociados con adquisición, requerimientos, desarrollo, uso, evaluación, soporte, mantenimientos, aseguramiento de la calidad y auditoria del software.

Los modelos de calidad para el software de describen de la siguiente manera:

2.1.14.1 Calidad interna y externa

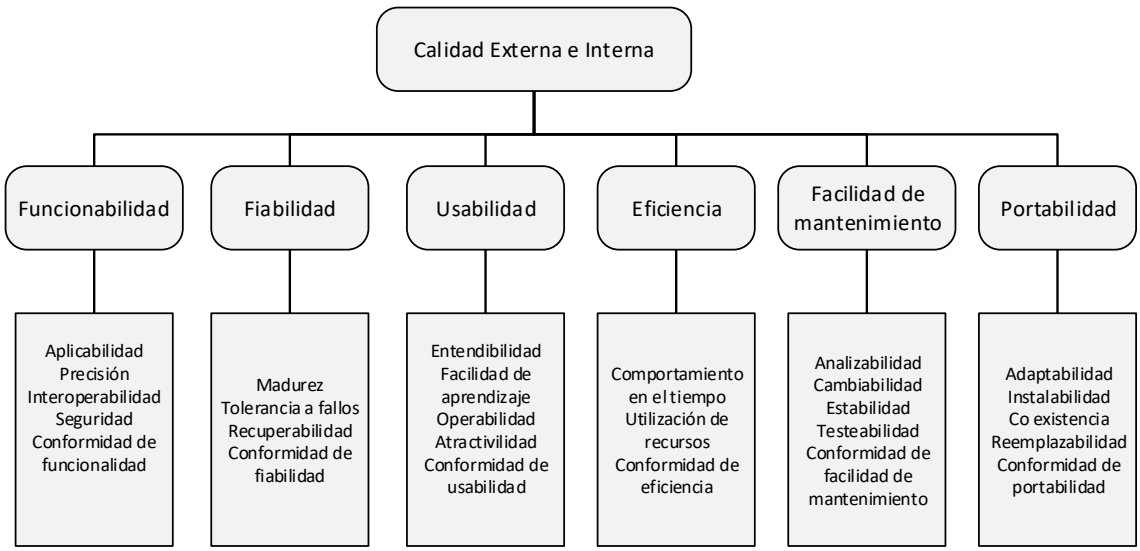
La norma ISO/IEC 9126 define la calidad interna como: “la totalidad de las características del producto software desde una perspectiva interna. La calidad interna es medida y evaluada en base a los requerimientos de calidad interna. Los detalles de la calidad del producto software pueden ser mejorados durante la implementación, revisión y prueba del código software, pero la naturaleza fundamental de la calidad del producto software representada por la calidad interna permanece sin cambios a menos que sea re diseñado”; y a la calidad externa como: “la totalidad de las características del producto software desde una perspectiva externa. Es la calidad cuando el software es ejecutado, la cual es típicamente medida y evaluada mientras se prueba en un ambiente simulado con datos simulados y usando métricas externas. Durante las pruebas, muchas fallas serán descubiertas y eliminadas. Sin embargo, algunas fallas todavía pueden

permanecer después de las pruebas. Como es difícil corregir la arquitectura de software u otros aspectos fundamentales del diseño del software, el diseño fundamental permanece sin cambios a través de las pruebas".

FIGURA 1

MODELO DE CALIDAD DEL PRODUCTO DE SOFTWARE PARA LA CALIDAD EXTERNA E INTERNA

.



Fuente: ISO/IEC 9126

CUADRO 1

CARACTERÍSTICAS, CALIDAD INTERNA Y EXTERNA

Características	Definición
Funcionalidad	La capacidad del producto software para proveer las funciones que satisfacen las necesidades explícitas e implícitas cuando el software se utiliza bajo condiciones específicas.
Fiabilidad	La capacidad del producto software para mantener un nivel especificado de funcionamiento cuando se está utilizando bajo condiciones especificadas.
Usabilidad	La capacidad del producto software de ser entendido, aprendido, usado y atractivo al usuario, cuando es usado bajo las condiciones especificadas.
Eficiencia	Capacidad del producto software para proveer un desempeño apropiado, de acuerdo a la cantidad de recursos utilizados y bajo las condiciones planteadas.
Facilidad de	Capacidad del producto software para ser modificado. Las modificaciones pueden incluir

mantenimiento	correcciones, mejoras o adaptación del software a cambios en el entorno, y en requerimientos y especificaciones funcionales.
Portabilidad	La capacidad del software para ser trasladado de un entorno a otro.

Fuente: ISO/IEC 9126

CUADRO 2

SUB CARACTERÍSTICAS, CALIDAD INTERNA Y EXTERNA

	Sub características	**Descripción**
Funcionabilidad	Aplicabilidad	La capacidad del producto software para proveer un conjunto apropiado de funciones para las tareas y objetivos especificados por el usuario.
	Precisión	La capacidad del producto software para proveer de resultados o efectos acordados con un grado necesario de precisión.
	Interoperabilidad	La capacidad del producto software a interactuar con uno o más sistemas específicos.
	Seguridad	La capacidad del producto software para proteger la información y los datos de modo que las personas o sistemas no autorizados no puedan leerlos o modificarlos y a las personas o sistemas autorizados no se les denegará el acceso.

	Conformidad de la funcionabildiad	La capacidad del producto software de adherirse a los estándares, convenciones o regulaciones legales o prescripciones similares referentes a la funcionalidad.
Fiabilidad	Madurez	La capacidad del producto software para evitar fallas como resultado de errores en el software.
	Tolerancia a fallos	La capacidad del producto software para mantener un nivel especificado de funcionamiento en caso de errores del software o de incumplimiento de su interfaz especificada.
	Recuperabilidad	La capacidad del producto software para restablecer un nivel especificado de funcionamiento y recuperar los datos afectados directamente en el caso de una falla.
	Conformidad de fiabilidad	La capacidad del producto software para adherirse a las normas, convenciones o regulaciones relativas a la fiabilidad.
Usabilidad	Entendibilidad	La capacidad del producto software para permitir al usuario entender si el software es aplicable, y cómo puede ser utilizado para las tareas y condiciones particulares de la aplicación.
	Facilidad de aprendizaje	La capacidad del producto software para permitir al usuario aprender su aplicación.
	Operabilidad	La capacidad del producto software para permitir al usuario operarlo y controlarlo.

	Atractibilidad	La capacidad del producto software de ser atractivo al usuario.
	Conformidad de usabilidad	La capacidad del producto software para adherirse a las normas, convenciones, guías de estilo o regulaciones relacionadas a la usabilidad.
Eficiencia	Comportamiento en el tiempo	La capacidad del producto software para proveer tiempos apropiados de respuesta y procesamiento, y ratios de rendimiento cuando realiza su función bajo las condiciones establecidas.
	Utilización de recursos	La capacidad del producto software para utilizar apropiadas cantidades de recursos cuando éste funciona bajo las condiciones establecidas.
	Conformidad de eficiencia	La capacidad del producto software para adherirse a normas o convenciones relacionadas a la eficiencia.
Facilidad de mantenimiento	Analizabilidad	La capacidad del producto software para ser diagnosticado por deficiencias o causas de fallas en el software o la identificación de las partes a ser modificadas.
	Cambiabilidad	La capacidad del producto software para permitir que una determinada modificación sea implementada.

	Estabilidad	La capacidad del producto software para evitar efectos inesperados debido a modificaciones del software.
	Testeabilidad	La capacidad del producto software para permitir que las modificaciones pueden ser validadas.
	Conformidad de facilidad de mantenimiento	La capacidad del producto software para adherirse a estándares o convenciones relativas a la facilidad de mantenimiento.
Portabilidad	Adaptabilidad	La capacidad del producto software para ser adaptado a diferentes entornos definidos, si aplicar acciones o medios diferentes de los previstos para el propósito del software considerado.
	Instalabilidad	La capacidad del producto software para ser instalado en un entorno definido.
	Co existencia	La capacidad del producto software para co existir con otro producto software independiente dentro de un mismo entorno compartiendo recursos comunes.
	Reemplazabilidad	La capacidad del producto software para ser utilizado en lugar de otro producto software, para el mismo propósito y en el mismo entorno.
	Conformidad de portabilidad	La capacidad del producto software para adherirse a estándares o convenciones relacionadas a la portabilidad.

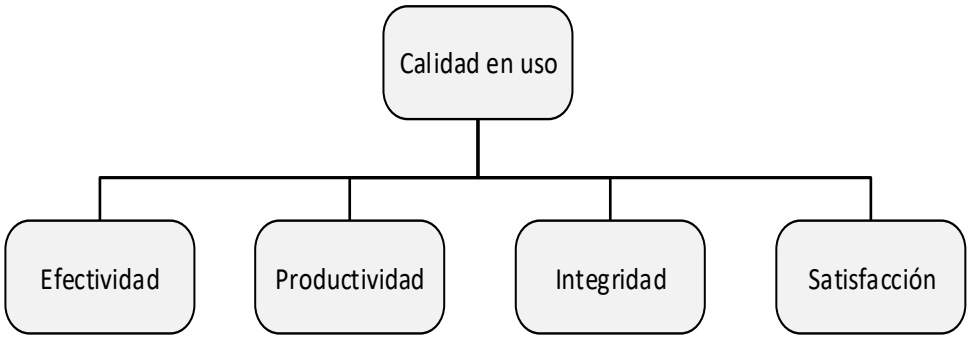
Fuente: ISO/IEC 9126

2.1.14.2 Calidad en uso

La norma ISO/IEC 9126-1 define la calidad en uso como: “la perspectiva del usuario de la calidad del producto software cuando éste es usado en un ambiente específico y un contexto de uso específico. Ésta mide la extensión para la cual los usuarios pueden conseguir sus metas en un ambiente particular, en vez de medir las propiedades del software en sí mismo”.

FIGURA 2

MODELO DE CALIDAD DEL PRODUCTO DE SOFTWARE PARA LA CALIDAD EN USO



Fuente: ISO/IEC 9126

CUADRO 3

CARACTERÍSTICAS DE LA CALIDAD EN USO, DEFINIDO EN ISO/IEC 9126-1

Características	Definición
Efectividad	La capacidad del producto software para permitir a los usuarios lograr las metas especificadas con precisión y completitud en un contexto específico de uso.
Productividad	La capacidad del producto software para permitir a los usuarios emplear cantidades apropiadas de recursos en relación a la efectividad lograda en un contexto específico de uso.
Integridad	La capacidad del producto software para lograr niveles aceptables de riesgo de daño a las personas, negocio, software, propiedad, o entorno en un contexto específico de uso.
Satisfacción	La capacidad del producto software para satisfacer a los usuarios en un contexto de uso específico.

Fuente: NTP ISO/IEC 9126-1 (Normalización, 2015).

2.1.15 Organización Internacional de Normalización (ISO)

Normalización (2015), ISO (Organización Internacional de Normalización) es una organización de membresía no gubernamental independiente y mayor desarrollador mundial de Normas Internacionales voluntarias. Están en 162 países, que son los organismos nacionales de normalización de todo el mundo, con una Secretaría Central, que tiene su sede en Ginebra, Suiza. Más información sobre nuestra estructura y cómo nos regimos.

2.1.16 Ingeniería del software

La ingeniería del Software es una disciplina de la ingeniería que se interesa por todos los aspectos de la producción de software, desde las primeras etapas de la especificación del sistema hasta el mantenimiento del sistema después de que se pone en operación (Sommerville, 2011).

2.1.17 Proceso de desarrollo de software

Un proceso de Software es una serie de actividades relacionadas que conduce a la elaboración de un producto de software. Estas actividades pueden contribuir al desarrollo de software desde cero en un lenguaje de programación estándar (Sommerville, 2011).

También, Sommerville (2011) nos dice que existen muchos diferentes procesos de software, pero todos deben incluir cuatro actividades que son fundamentales para la ingeniería del software:

a. *Especificación del software*, Tiene que definirse tanto la funcionalidad del software como las restricciones de su operación.
b. *Diseño e* implementación *del software*, Debe desarrollarse el

software para cumplir con las especificaciones.

c. *Validación del software*, Hay que validar el software para asegurarse de que cumple lo que el cliente quiere.

d. *Evolución del Software*, El software tiene que evolucionar para satisfacer las necesidades cambiantes del cliente.

2.1.18 Metodología XP

(Stwward, 2002), nos indica que el ciclo de vida de un proyecto XP es muy dinámico y se puede separar en las siguientes fases:

Fase de exploración.- Es la fase en la que se define el alcance general del proyecto. En esta fase, el cliente define lo que necesita mediante la redacción de sencillas historias de usuarios. Los programadores estiman los tiempos de desarrollo en base a esta información. Debe quedar claro que las estimaciones realizadas en esta fase son primarias (ya que estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más en detalle en cada iteración. Esta fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado.

Fase de Planificación.- La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación.

Fase de Iteraciones.- Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de

usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios. El cliente, por lo tanto, también debe participar activamente durante esta fase del ciclo. Las iteraciones son también utilizadas para medir el progreso del proyecto. Una iteración terminada sin errores es una medida clara de avance.

Fase de puesta en Producción.- Si bien al final de cada iteración se entregan módulos funcionales y sin errores, puede ser deseable por parte del cliente no poner el sistema en producción hasta tanto no se tenga la funcionalidad completa. En esta fase no se realizan más desarrollos funcionales, pero pueden ser necesarias tareas de ajuste.

2.1.18.1 Desarrollo de código en XP

Disponibilidad del cliente

Uno de los requerimientos de XP es tener al cliente disponible durante todo el proyecto. No solamente como apoyo a los desarrolladores, sino formando parte del grupo. El involucramiento del cliente es fundamental para que pueda desarrollarse un proyecto con la metodología XP. Al comienzo del proyecto, el cliente debe proporcionar las historias de usuarios. Pero, dado que estas historias son expresamente cortas y de "alto nivel", no contienen los detalles necesarios para realizar el desarrollo del código. Estos detalles deben ser proporcionados por el cliente, y discutidos con los desarrolladores, durante la etapa de desarrollo. No se requieren de largos documentos de especificaciones, sino que los detalles son proporcionados por el cliente, en el momento adecuado,

"cara a cara" a los desarrolladores.

Uso de estándares

Si bien esto no es una idea nueva, XP promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo, y que facilite la recodificación.

Programación dirigida por las pruebas ("Test-driven programming")

En las metodologías tradicionales, la fase de pruebas, incluyendo la definición de los tests, es usualmente realizada sobre el final del proyecto, o sobre el final del desarrollo de cada módulo. La metodología XP propone un modelo inverso, en el que, lo primero que se escribe son los test que el sistema debe pasar. Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas.

Programación en pares

XP propone que se desarrolle en pares de programadores, ambos trabajando juntos en un mismo ordenador. Si bien parece que ésta práctica duplica el tiempo asignado al proyecto (y por ende, los costos en recursos humanos), al trabajar en pares se minimizan los errores y se logran mejores diseños, compensando la inversión en horas. El producto obtenido es por lo general de mejor calidad que cuando el desarrollo se realiza por programadores individuales.

Integraciones permanentes

Todos los desarrolladores necesitan trabajar siempre con la "última versión". Realizar cambios o mejoras sobre versiones antiguas causan graves problemas, y retrasan al proyecto. Es por eso que XP promueve publicar lo antes posible las nuevas versiones, aunque no sean las

últimas, siempre que estén libres de errores. Idealmente, todos los días deben existir nuevas versiones publicadas. Para evitar errores, solo una pareja de desarrolladores puede integrar su código a la vez.

2.1.18.2 Pruebas en XP

Pruebas unitarias

Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, como se mencionó anteriormente, las pruebas deben ser definidas antes de realizar el código .Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código.

En este sentido, el sistema y el conjunto de pruebas debe ser guardado junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo.

Detección y corrección de errores

Cuando se encuentra un error ("bug"), éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto.

Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada.

Los clientes son responsables de verificar que los resultados de estas

pruebas sean correctos. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación.

2.2 DEFINICIÓN DE TÉRMINOS BÁSICOS

2.2.1 CORPUS DE REFERENCIA

En un repositorio de trabajos de investigación que tiene un cierto grado de representatividad, por lo tanto ha de ser suficientemente extenso como para representar todas las variedades relevantes de trabajos de investigación.

2.2.2 Plagio

Según la RAE (Real Academia Española de la Lengua), plagiar es: "copiar en lo sustancial obras ajenas dándola como propias".

2.2.3 PAN

PAN es una serie de eventos científicos y tareas compartidas sobre textos digitales forenses.

2.2.4 ISO/IEC 9126

Es un estándar internacional para la evaluación de la calidad del software.

2.2.5 Sistema

Una composición integrada que consiste en uno o más procesos, hardware, software, instalaciones y personas, que proveen una capacidad para satisfacer una necesidad establecida o un objetivo.

2.2.6 Software

Todo o parte de los programas, procedimientos, reglas y documentación asociada a un sistema de procesamiento de información.

2.2.7 Validación

Confirmación por inspección y provisión de evidencia objetiva de que los requerimientos particulares para un uso específico son alcanzados.

2.2.8 Verificación

Confirmación por examen y provisión de evidencia objetiva que los requerimientos específicos han sido alcanzados.

2.2.9 Indicador

Una medida que se puede utilizar para estimar o para redecir otra medida.

2.2.10 Escala

Un conjunto de valores con propiedades definidas.

2.2.11 Calificación

La acción de evaluar el valor medido al nivel de calificación adecuado.

2.2.12 Atributo

Una característica física o abstracta mesurable de una entidad. Los atributos pueden ser internos o externos.

2.2.13 Calidad

Son todas las características de una entidad que forman parte de su habilidad para satisfacer las necesidades propias e implícitas.

2.2.14 Calidad interna

Es la totalidad de atributos del producto que determinan su habilidad para satisfacer las necesidades propias e implícitas bajo condiciones específicas.

2.2.15 Calidad externa

La extensión para la cual un producto satisface necesidades explícitas e

implícitas cuando es usado bajo condiciones específicas.

2.2.16 Calidad en uso

La perspectiva del usuario de la calidad del producto cuando éste es usado en un ambiente específico y un contexto de uso específico.

2.2.17 Métrica

Es un método definido de valoración y su escala de valoración.

2.3 HIPÓTESIS DE LA INVESTIGACIÓN

2.3.1 Hipótesis general

El Software nos permite la verificación y determinar el grado de similitud en documentos de investigación con el corpus de referencia en la Universidad Nacional del Altiplano existe o no plagio.

CAPÍTULO III

METODOLOGÍA

3.1 TIPO DE INVESTIGACIÓN

La presente investigación corresponde a una investigación de tipo cuasi experimental, puesto que aplica conocimientos científicos y tecnológicos para desarrollar y evaluar una aplicación de Software para mejorar los procesos de detección de plagios en los trabajos de investigación de la Universidad Nacional del Altiplano de Puno.

3.2 DISEÑO DE LA INVESTIGACIÓN

El diseño de investigación corresponde al tipo cuasi-experimental de estudio de caso con una sola medición:

$$G - - - X - - - 0$$

Donde:

G: Grupo de sujetos

X: Tratamiento, estimulo o condición pre-experimental

O: Una medición de los sujetos del grupo.

En este tipo de diseño se administra un tratamiento a un grupo y después se aplica una medición en una o más variables para observar cual es el nivel del grupo en estas variables.

3.3 POBLACIÓN Y MUESTRA

3.3.1 Población

De todas las Universidades que se encuentran en nuestra región, la Universidad que más graduados tiene es la Universidad Nacional del Altiplano, incluso se encuentra en el quinto lugar en el ranking de Universidades públicas con mayor cantidad de graduados, (2 604 en el 2014, esto según los datos estadísticos de la ANR 2015).

A partir del semestre 2016-II, todos los proyectos de investigación para graduarse en la Universidad Nacional del Altiplano se realizan electrónicamente a través de un software, hasta diciembre del 2016 se tuvo una cantidad de 560 trabajos de investigación presentados y es justamente nuestra población para el presente trabajo de investigación.

3.3.2 Muestra

La muestra que se ha tomado es no probabilística, se considera el 40% de la población, es decir 224 trabajos de investigación indistintamente de las tres áreas (ingenierías, sociales y biomédicas).

3.4 METODOLOGÍA DE DESARROLLO DE LA PLATAFORMA DE SOFTWARE

Para el desarrollo e implementación de la Plataforma de Software se está utilizando una metodología ágil, en específico la metodología de Programación Extrema (XP). (Sánchez, 2007) define la Programación Extrema como una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el

equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

3.4.1 Planificación

El primer paso es definir las historias de usuario con el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles.

3.4.2 Diseño

La metodología XP sugiere que hay que conseguir diseños simples y sencillos para conseguir un diseño fácilmente entendible e implementable que a la larga costará menos tiempo y esfuerzo desarrollar. Se realizó el diseño de la arquitectura del sistema, la base de datos y el diseño de interfaces a través de prototipos evolutivos.

3.4.3 Implementación

3.4.3.1 En el lado cliente

Para el desarrollo de la Plataforma de Software en el lado cliente hemos hecho uso de **AngularJS** que se basa en javascript. AngularJS es una herramienta que nos ayudará mucho en el desarrollo de la plataforma de software, es un framework desarrollado por Google.

Elementos de una aplicación Angular:

Componentes

Los componentes controlan secciones de nuestro HTML, nuestras vistas

El componente es una clase, ésta clase va a tener propiedades (la

información de la vista) y métodos (el comportamiento) que nos van a permitir modificar la vista. Para crear una aplicación en Angular se requiere por lo menos un componente (raíz), pero generalmente se tiene más.

Enrutamientos

Enrutamientos (Routing) se utiliza para mostrar diferentes áreas de nuestra aplicación dependiendo de la dirección URL que el usuario utilice.

Directivas

Las directivas nos sirven para hacer cambios en el DOM des nuestras páginas. Existen dos tipos: Estructurales y de atributo.

3.4.3.2 En el lado servidor

En el lado servidor se hizo uso del lenguaje de programación PHP, pero para garantizar aspectos bastantes críticos tales como la seguridad se hizo uso de *Frameworks*, en éste caso se optó por el lenguaje de programación PHP y C++.

PHP

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

MariaDB

Para la implementación de la plataforma de software, necesitamos un gestor de base de datos que nos permita gestionar todos los datos que involucran el proceso de detección de plagio, desde el momento de la solicitud de verificación de un determinado

documento por parte del usuario hasta que el mismo usuario pueda ver los resultados de dicha solicitud. Para ello hemos elegido el gestor de base de datos MariaDB que es un sistema de gestión de bases de datos derivado de MySQL con licencia GPL.

C++

Es un lenguaje de programación robusto y muy potente, orientado a objetos, muy flexible, altamente portable. Por todas esas características es que lo utilizamos para la implementación de los algoritmos de comparaciones y van a ser invocados desde la plataforma web.

3.4.4 Evaluación

Para la evaluación del modelo propuesto se utilizará la norma ISO/IEC 9126 donde detalla el modelo a usar para la calidad del producto software, que a su vez se divide en dos partes: la Calidad Interna y Calidad Externa.

3.4.5 Prueba de hipótesis

$$Z_c = \frac{p - P}{\sqrt{\frac{p * q}{n}}}$$

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1 ANALISIS DE LOS PROCEDIMIENTOS QUE SE REALIZAN EN LA ACTUALIDAD Y HACIENDO USO DE UN SOFTWARE CON UN CORPUS DE REFERENCIA PARA DETERMINAR SI EN LOS TRABAJOS DE INVESTIGACIÓN DE LA UNIVERSIDAD NACIONAL DEL ALTIPLANO EXISTE O NO PLAGIO

4.1.1 Procedimiento actual que se realiza para la determinación si en los trabajos de investigación presentados en la Universidad Nacional del Altiplano existe o no plagio.

En la actualidad a nivel de pregrado se están haciendo las presentaciones de los proyectos de tesis en todas las escuelas profesionales de la Universidad a través de un software de registro de los datos generales del proyecto y se adjunta dicho proyecto en un formato preestablecido por vicerrectorado de investigación de la Universidad en un archivo del tipo PDF.

Posterior a ello solo se hace la verificación de forma mas no si dicho trabajo de investigación es una copia de algún otro trabajo que ya se ha presentado a la Universidad y se procede a la asignación de jurados. Lo mismo ocurre en cuanto a los borradores de tesis.

También en los trabajos de investigación de postgrado en la oficina de investigación de la escuela de postgrado lo único que se hace es la revisión de forma y de igual manera sin la verificación de posible plagio dicha oficina entrega las constancias de visto bueno para las sustentaciones.

4.1.1 Procedimiento actual que se realiza para la determinación si en los trabajos de investigación presentados en la Universidad Nacional del Altiplano existe o no plagio a través de un software con un corpus de referencia.

En vista de que en la actualidad en la Universidad Nacional del Altiplano no se realiza ningún tipo de verificación y tomando en cuenta si se hace la verificación de forma manual es prácticamente imposible tomando en cuenta el tamaño de los repositorios de tesis u otros trabajos de investigación que se tienen en la Universidad y ello va a seguir incrementándose con el transcurrir del tiempo, la propuesta de solución para la verificación y detección de plagio tiene que ser a través de un software.

En el presente trabajo de investigación se resuelve el problema de la detección de plagio con un corpus de referencia (repositorio de tesis de la UNA anteriormente publicadas) mediante el uso de nuevos algoritmos para su implementación en la plataforma de Software.

Debido a la gran cantidad de documentos a comparar, se presentan problemas como la complejidad en tiempo, complejidad en espacio (memoria para el almacenamiento) y el grado de certeza en la detección de plagio, por ello se tienen dos fases, la primera que es un análisis

preliminar entre los documentos, el cual determina el grado de similitud entre ellos, todos aquellos que estén encima de un valor mínimo se consideran candidatos potenciales de los cuales se hizo el probable plagio, así esto reduce el universo de posibles documentos del corpus de referencia, teniendo en la segunda fase menor cantidad de documentos a analizar, pudiendo hacer uso de algoritmos con complejidades más altas que detecten el posible plagio con un grafo de certeza más alta.

En la fase detallada o definitiva se analiza sobre un universo más pequeño, dando la posibilidad de evaluarlo más rigurosamente, algoritmos con mayor precisión pero con complejidades muy altas pueden resolver esto.

Se usa la metodología sintáctica para la detección de plagio, el cual trata a las palabras, frases, caracteres, párrafos, entre otros como símbolos, sin darles alguna interpretación o significado.

4.1.2 Propiedades generales del algoritmo de detección de plagio propuesto

- No es sensible a los espacios, signos de puntuación o mayúsculas: Para esto se hará un pre procesamiento de los documentos.
- Los pasajes considerados tienen un tamaño mínimo, de tal forma que se garantice que se trata de un plagio y no de coincidencias por expresiones idiomáticas comunes: Para ello se toma entre 2-gramas hasta 5-gramas.
- Independencia de la posición: Permitirá maximizar la detección de plagio sin importar la permutación, borrado o agregado de contenido en los documentos.

4.1.3 Procedimiento general para la detección de plagio

- Para la etapa de análisis preliminar se usa N-gramas que combina hashing (una idea similar al usado por Rabin Karp con algunas variaciones) y Vector Space model con varias funciones de distancia.
- Para la etapa de análisis detallado o definitivo se usa hashing (para tamaños de palabras que van entre tres y cuarenta palabras) y el uso de un algoritmo greedy propuesto que une intervalos de matching próximos maximizando la cobertura de palabras, detectando plagios aún más elaborados.

4.1.4 Desarrollo del algoritmo

4.1.4.1 FASE 01: Análisis preliminar

En esta fase inicial se hace un análisis preliminar del documento con un corpus de referencia. Para ello se involucra varias etapas:

a. Pre procesamiento

Se hace un previo pre procesamiento del texto para evitar que pequeñas modificaciones dificulten la detección de plagio, (Elizalde, 2013) convierte el texto a minúsculas y elimina signos diacríticos presentes y signos de puntación, manteniendo espacios entre palabras.

Debido a la elección de palabras como unidad de comparación se realizaron modificaciones como no considerar números ya que normalmente son alterados para evitar ser detectados, además se convierte todo a minúsculas y se elimina tildes y signos de puntuación.

b. N-gramas

Se usan trigramas, cuatro-gramas y cinco-gramas de palabras que representan las características de nuestro modelo de espacio vectorial, de los cuales se determina cual es el de mejor desempeño para la detección de plagio con corpus de referencia.

(Cedeño & Rosso, 2009) usan bigramas y trigramas, en (Ferret, 2004) se usan trigramas, mientras que (Clough, 2010) y (Kasprzak, 2009) usan cinco-gramas.

(Sonawane, 2015) usa los algoritmos de Rabin Karp y string matching para la detección de plagio. Para el presente proyecto de investigación se emplea Rabin Karp para el cálculo eficiente de n-gramas, debido a la reutilización de información, pero esto podría generar colisiones.

Para lidiar con esto, además del valor hash se agrega información adicional como las letras más frecuentes y si hay empates las lexicográficamente menores para poder reducir considerablemente la existencia de colisiones y mejorar la complejidad del algoritmo que tiene como unidad de comparación las palabras del texto. A partir de esto se construye el vector que representa al documento.

c. Calculo de pesos de las componentes del vector

Como peso de las componentes del vector se usa tf-idf, es decir, term frequency-inverse document frequency. La idea general es que aquellas palabras que aparecen con mayor frecuencia

tienen menos relevancia al momento de comparar documentos, por ejemplo, la palabra “de” en español es muy común, pero no es tan determinante en la detección de plagio, en cambio la palabra “algoritmo” es usado en menor frecuencia y por lo tanto su uso tiene mayor relevancia en la detección de plagio con un corpus de referencia.

d. Funciones de distancia

Una vez definidos los vectores de los documentos, evaluamos la similitud entre ellos, esto haciendo uso de funciones de distancia entre los vectores que representan los documentos.

Se usan las funciones de coseno. A partir de estas funciones se elige la que mejor desempeño tiene en la FASE 01, el cual tiene como objetivo reducir al mínimo el universo de documentos de los cuales probablemente se hizo el plagio.

A partir de estas funciones se obtienen el porcentaje de similitud entre dos documentos, teniendo un umbral mínimo sobre la cual consideramos que hay probable plagio, evaluando esto en la FASE 02, que es una etapa más detallada y donde los cálculos son más costosos.

4.1.4.1.1. Diseño del algoritmo FASE 01

Debido a la restricción de complejidad en tiempo y espacio para esta primera fase es necesario el uso de enfoques que nos permitan tener algoritmos eficientes con un grado de certeza muy alta.

Se usa el enfoque n-gramas que es una combinación del modelo de espacio vectorial y fingerprinting, aprovechando las bondades de ambos para poder tener un mejor desenvolvimiento del algoritmo. Para la parte de fingerprinting se hace uso de trigramas, cuatro gramas y cinco gramas, usando Karp Rabin para el cálculo del hash el cual es muy eficiente. A partir de ello se construye el vector en nuestro modelo de espacio vectorial, usando como peso de las componentes del vector tf-idf, es decir, term frequency-inverse document frequency, para finalmente usar funciones de distancia que determinan la similitud de los documentos.

Las funciones de distancia empleadas son coseno, Jaccard, Dice, Overlap y algunas funciones estadísticas.

Para un óptimo desenvolvimiento del algoritmo se prueban las diferentes combinaciones de los n-gramas y funciones de distancia eligiendo el mejor de ellos para poder ser usado en la FASE 01.

4.1.4.1.2. Análisis del algoritmo FASE 01

Para esta primera fase es necesario tener complejidades muy buenas, puesto que el corpus de referencia (repositorio de tesis de la UNA anteriormente publicadas) es muy grande, pudiendo llegar a miles de documentos.

Para la obtención de trigramas, cuatro gramas y cinco gramas se tienen una complejidad en tiempo lineal, es decir, O(n) (donde n es la cantidad de palabras en el documento), puesto que sehace el uso de Karp Rabin, similarmente la complejidad en espacio es lineal, almacenamos la frecuencia de cada palabra diferente

presente en el texto, aquí se manejan únicamente el valor del hash y su frecuencia en el texto.

Para el modelo de espacio vectorial se tiene en el peor caso que la cantidad de dimensiones del vector es igual a la cantidad de diferentes trigramas, cuatro gramas o cinco gramas presentes en el texto, así se tiene una complejidad en tiempo lineal y también una complejidad en espacio lineal.

Para el cálculo de las funciones de distancia, una vez calculado el vector de los documentos a comparar, únicamente usamos la definición de cada una de las funciones de distancia, siendo esta lineal en tiempo y espacio.

Por lo tanto se tiene una complejidad en tiempo y espacio lineal para todos los pasos que forman parte del algoritmo en la FASE 01.

4.1.4.2. FASE 02: Análisis detallado o definitivo

Después de haber calculado el grado de similitud en la FASE 01 contra todos los documentos del corpus de referencia (repositorio de tesis de la UNA anteriormente publicadas) solo se consideran para esta fase, aquellos documentos que tienen un grado de similitud con el documento analizado que superen un valor mínimo. Así la cantidad de documentos a evaluar en esta fase se reduce considerablemente, admitiendo complejidades mayores en los algoritmos usado en esta fase.

En esta fase se determina los fragmentos de texto que emparejan exactamente en los documentos, para luego extender estos

emparejamiento exactos a emparejamiento aproximados los cuales detectan plagios más elaborados, es decir, permutación, agregación o eliminaron de palabras en el texto.

Además de mostrar los fragmentos de probable plagio, también se obtiene el porcentaje de similitud entre el documento analizado contra todos los documentos del corpus de referencia consideradas en esta fase.

En la FASE 02 se tienen las siguientes etapas:

a. Pre procesamiento

Se realiza un pre procesamiento para poder calcular los n-gramas de los documentos eficientemente (para obtener el hash de cada n-grama usando Karp Rabin), las posiciones de los fragmentos de texto, además para poder verificar si un cierto fragmento de texto ya ha sido emparejado previamente con el uso de programación dinámica.

b. Emparejamientos exactos de fragmentos de texto

En esta etapa se determina los fragmentos que emparejan exactamente entre el documento analizado y los documentos del corpus de referencia del cual probablemente se hizo el plagio.

Para ello es necesario definir el tamaño de los fragmentos de texto a analizar. Si este valor es muy alto la complejidad incrementa proporcionalmente con este valor, en el otro lado si fuera muy chico probablemente sea difícil extender los

emparejamientos exactos a emparejamientos aproximados, los cuales detectan plagios más elaborados.

Para poder tener mayor probabilidad de obtener emparejamientos exactos de gran tamaño, se evalúa en forma decreciente, es decir, se comienza con un tamaño de los fragmentos igual a 40 (40-gramas, se toma este valor al considerarlo apropiado para nuestro corpus de referencia) y se calcula los emparejamientos exactos entre el documento analizado y cada uno de los documentos del corpus de referencia que son considerados en esta etapa. Luego se hace lo mismo pero para un tamaño de los fragmentos igual a 39 y así sucesivamente hasta un tamaño de los fragmentos igual a 3. Cada vez que se obtienen los emparejamientos exactos se verifica que no haya solapamiento entre ellos, para optimizar esto se hace el pre procesamiento mencionado en la anterior etapa.

c. Extensión de los emparejamientos exactos en emparejamientos aproximados

Una vez obtenido los emparejamientos exactos, se intenta extenderlos a emparejamientos aproximados, los cuales tienen mayores tamaños, es decir, intentar unir intervalos de emparejamientos exactos.

Un emparejamiento exacto es aquel en donde los fragmentos de texto son exactamente los mismos, en cambio un

emparejamiento aproximado es aquel en donde algunos intervalos son emparejamientos exactos y otros no.

Los emparejamientos aproximados permiten detectar plagios más elaborados, cuando se permutan, eliminan o agregan palabras al texto plagiado.

Para poder conseguir emparejamientos aproximados de mayor tamaño, se intenta extender aquellos emparejamientos exactos que tengan mayor tamaño y se procede en ese orden de mayor a menor, de esta manera tendremos mayor probabilidad de conseguir emparejamientos aproximados de gran tamaño, lo que nos permite detectar plagios más elaborados.

Al unir intervalos de emparejamiento exacto y formar emparejamientos aproximados es necesario considerar la cantidad de palabras entre emparejamiento exactos que no coinciden, para el algoritmo empleado usamos una cantidad de palabras igual a 4, ya que un valor más alto normalmente descarta la posibilidad de plagio en nuestro corpus de referencia.

4.1.4.2.1. Diseño del algoritmo FASE 02

Se espera que después de la FASE 01, la cantidad de documentos del corpus de referencia de donde probablemente se hizo el plagio sea mucho menor. Así en esta FASE 02 la complejidad en tiempo y espacio no es tan rigurosa como en la FASE 01.

Se usan n-gramas para poder obtener los emparejamientos exactos con la ayuda del algoritmo Karp Rabin y programación dinámica para poder verificar eficientemente que tos emparejamientos exactos no se solapen.

Luego para extender los emparejamientos exactos en emparejamientos aproximados se usa el enfoque de los algoritmos greedy, esto es, se empieza por extender los emparejamientos exactos de forma decreciente por tamaño, es decir primero los de mayor tamaño, luego los de menor tamaño, de esta manera se tendrá mayor probabilidad de obtener emparejamientos aproximados de gran tamaño los cuales probablemente hayan sido plagiados.

4.1.4.2.2. Análisis del algoritmo FASE 02

Para poder obtener los emparejamientos exactos se tiene una complejidad en tiempo en el peor caso de O(nm) (donde “n” es la cantidad de palabras del documento analizado y “m” la cantidad de palabras del documento del corpus de referencia del cual se hizo probable plagio), esto podría suceder cuando la mayoría de fragmentos de texto que se repiten muchas veces en el documento analizado también se repitan muchas veces en el documento del corpus de referencia, existiendo un total de O(nm) emparejamientos exactos.

Pero en la práctica este caso es muy poco probable, esperándose que la cantidad de emparejamientos sea lineal y las razones son las siguientes:

- En el algoritmo se consideran como mínimo emparejamientos exactos de tamaño cuatro (es decir se consideran emparejamientos exactos de al menos cuatro palabras consecutivas).
- Una tesis en la Universidad Nacional del Altiplano es muy poco probable que contenga algunos fragmentos de texto que aparezcan con una frecuencia mayor al 20%.

Por ejemplo imaginemos un documento con 40000 palabras de los cuales se tendría alrededor de 10000 fragmentos de texto de tamaño 4, la probabilidad que uno de los fragmentos se repita más del 20% (es decir más de 2000 veces) es en la práctica muy improbable, por lo tanto la complejidad en tiempo en la práctica es O(n + m) que sería la cantidad de emparejamientos exactos. Similarmente la complejidad en espacio del algoritmo es la misma que la complejidad en tiempo para esta etapa.

Para poder extender los emparejamientos exactos en emparejamientos aproximados se usa el enfoque de los algoritmos greedy, se procede de mayor tamaño a menor tamaño sobre los emparejamientos y se intenta extenderlos, para ello se usa una cola de prioridad, el cual es una estructura de datos que permite en un tiempo logarítmico extraer el de mayor tamaño, así como insertar y eliminar elementos de la cola de prioridad.

Cuando se unen emparejamientos, se crea uno nuevo el cual ahora representa a todos los anteriores, así la cantidad de uniones entre emparejamientos es lineal. Para el peor caso se tiene una

complejidad de 0(nmlogn) (el factor logn es debido al uso de la cola de prioridad), pero similarmente a la obtención de los emparejamientos exactos se tiene en la práctica una complejidad en tiempo de 0((n+m)logn) y una complejidad en espacio de 0 (n + m).

Para probar la efectividad del nuevo algoritmo propuesto se ha tomado como base para las pruebas el **repositorio local** de tesis de la Universidad que actualmente con un total de ***615 tesis digitalizadas*** y son tesis de todas las escuelas profesionales pertenecientes a los tres áreas (ingenierías, sociales y biomédicas). Este es el ***corpus de referencia*** con la que se va a trabajar para realizar cada uno de las pruebas de los algoritmos y están preparados (en formato de texto plano) listo para realizar las comparaciones.

Lamentablemente no existen muchos *corpus* para evaluar el algoritmo. Se han generado algunos de manera sintética, donde algoritmos automáticos deciden aleatoriamente la cantidad y longitud del texto que será insertado en un documento anfitrión, que se convertirá en el sospechoso. El problema con estos *corpus* es que utilizan un modelo que solo mezcla las partes de los documentos, haciendo una ingenua simplificación del proceso de plagio, principalmente del plagio con alta reescritura, que es justamente el problema más complejo de la detección de plagio. Los casos de plagios generados por estos medios, obviamente, no son muy apegados a los realizados por humanos. En primer lugar, en los plagios humanos existe una mayor cohesión temática entre los documentos sospechosos y las fuentes, pues las fuentes son elegidas

conscientemente; otro aspecto es que, obviamente, el tipo de modificaciones y ofuscamiento del plagio es mucho más fino que cuando se hace sintéticamente.

Como se indica en la descripción del algoritmo propuesto, consta de dos fases, la primera se refiere a la determinación del porcentaje de coincidencia del documento analizado con el corpus de referencia, si éste valor supera un cierto porcentaje (en el caso nuestro se estableció de más del 20%, entonces se pasa a la siguiente etapa del análisis).

La efectividad del algoritmo mucho depende de los n-gramas establecidos y el por esa razón que para encontrar la mayor efectividad del algoritmo realizamos las pruebas variando éste valor en un rango de tres a cinco.

Hay que tener en cuenta que estos *corpus* sintéticos fueron creados primordialmente para la búsqueda de plagio y no para detección de plagio, por lo que las complicaciones en esa tarea está más orientada la complejidad de trabajar con cantidades de datos y no a la de manejar los detalles más finos del lenguaje como se requiere en la DAP (detección automática de plagio).

Para la evaluación de los métodos de la detección de plagio automático se integraron los pares de documentos; cada respuesta proporcionada por los participantes se tomó como documento sospechoso y la respuesta de referencia que se les entregó como fuente. En el caso de la evaluación binaria de la DAP se tomaron todas las estrategias de plagio, copia cercana y las revisiones ligera y severa, como documentos plagiados.

Los resultados de los experimentos con cubrimiento de n-gramas con la variación de la n sobre el *corpus de referencia* se encuentran en la tabla.

CUADRO 4

RESULTADOS VARIANDO N-GRAMA

n-grama	Detección de Plagio	%
1	0.7823730	78
2	0.3203708	32
3	**0.2312730**	**23**
4	0.14091961	14
5	0.1021438	10

De lo anterior podemos notar que el mejor indicador que se obtiene es cuando se establece el n-grama en tres, es decir tri-gramas. Por lo tanto las pruebas siguientes se han hecho en base a tri-gramas.

El cuadro siguiente muestra el resultado de las comparaciones de un los 224 de documentos de la muestra de investigación contra el repositorio de 615 tesis, como se ha explicado antes el algoritmo propuesto consta de dos fases, en la primera fase se realiza las verificaciones y se obtiene un indicador de semejanza con un determinado archivo del repositorio, si éste indicador es mayor o igual al 20%, entonces se pasa a la evaluación detallada o definitiva.

CUADRO 5

DETERMINACIÓN DE PLAGIO CON 224 DOCUMENTOS DE LA MUESTRA

CONTRA EL REPOSITORIO DE REFERENCIA

Documento	Evaluación Fase 01, 02	AREA	OBSERVACIONES
1	0.017005	INGENIERÍA	NO EXISTE PLAGIO
2	0.108538	INGENIERÍA	NO EXISTE PLAGIO
3	0.010513	INGENIERÍA	NO EXISTE PLAGIO
4	**0.440505**	**INGENIERÍA**	**SI EXISTE PLAGIO**
5	0.017125	INGENIERÍA	NO EXISTE PLAGIO
6	0.005114	INGENIERÍA	NO EXISTE PLAGIO
7	0.026595	INGENIERÍA	NO EXISTE PLAGIO
8	0.057223	INGENIERÍA	NO EXISTE PLAGIO
9	0.016096	INGENIERÍA	NO EXISTE PLAGIO
10	**0.322403**	**INGENIERÍA**	**SI EXISTE PLAGIO**
11	0.011497	INGENIERÍA	NO EXISTE PLAGIO
12	0.008844	INGENIERÍA	NO EXISTE PLAGIO
13	0.018374	INGENIERÍA	NO EXISTE PLAGIO
14	0.010360	INGENIERÍA	NO EXISTE PLAGIO
15	0.016052	INGENIERÍA	NO EXISTE PLAGIO
16	0.008074	INGENIERÍA	NO EXISTE PLAGIO
17	0.004983	INGENIERÍA	NO EXISTE PLAGIO
18	0.016254	INGENIERÍA	NO EXISTE PLAGIO
19	0.006138	INGENIERÍA	NO EXISTE PLAGIO
20	0.012016	INGENIERÍA	NO EXISTE PLAGIO
21	**0.610280**	**INGENIERÍA**	**SI EXISTE PLAGIO**
22	0.013654	INGENIERÍA	NO EXISTE PLAGIO

23	0.013166	INGENIERÍA	NO EXISTE PLAGIO
24	0.019146	INGENIERÍA	NO EXISTE PLAGIO
25	0.013701	INGENIERÍA	NO EXISTE PLAGIO
26	0.034228	INGENIERÍA	NO EXISTE PLAGIO
27	0.021495	INGENIERÍA	NO EXISTE PLAGIO
28	0.008226	INGENIERÍA	NO EXISTE PLAGIO
29	0.004197	INGENIERÍA	NO EXISTE PLAGIO
30	0.015720	INGENIERÍA	NO EXISTE PLAGIO
31	0.008738	INGENIERÍA	NO EXISTE PLAGIO
32	0.008526	INGENIERÍA	NO EXISTE PLAGIO
33	0.011883	INGENIERÍA	NO EXISTE PLAGIO
34	0.005100	INGENIERÍA	NO EXISTE PLAGIO
35	0.013467	INGENIERÍA	NO EXISTE PLAGIO
36	0.016451	INGENIERÍA	NO EXISTE PLAGIO
37	0.005898	INGENIERÍA	NO EXISTE PLAGIO
38	**0.459244**	**INGENIERÍA**	**SI EXISTE PLAGIO**
39	0.006496	INGENIERÍA	NO EXISTE PLAGIO
40	0.016149	INGENIERÍA	NO EXISTE PLAGIO
41	0.010579	INGENIERÍA	NO EXISTE PLAGIO
42	0.005964	INGENIERÍA	NO EXISTE PLAGIO
43	0.008530	INGENIERÍA	NO EXISTE PLAGIO
44	0.009677	INGENIERÍA	NO EXISTE PLAGIO
45	0.011881	INGENIERÍA	NO EXISTE PLAGIO
46	**0.341963**	**INGENIERÍA**	**SI EXISTE PLAGIO**
47	0.012625	INGENIERÍA	NO EXISTE PLAGIO
48	0.002783	INGENIERÍA	NO EXISTE PLAGIO
49	0.015343	INGENIERÍA	NO EXISTE PLAGIO

50	0.005887	INGENIERÍA	NO EXISTE PLAGIO
51	0.013536	INGENIERÍA	NO EXISTE PLAGIO
52	0.113408	INGENIERÍA	NO EXISTE PLAGIO
53	0.063721	INGENIERÍA	NO EXISTE PLAGIO
54	0.020627	INGENIERÍA	NO EXISTE PLAGIO
55	0.006642	INGENIERÍA	NO EXISTE PLAGIO
56	0.013634	INGENIERÍA	NO EXISTE PLAGIO
57	0.004559	INGENIERÍA	NO EXISTE PLAGIO
58	**0.380850**	**INGENIERÍA**	**SI EXISTE PLAGIO**
59	0.021479	INGENIERÍA	NO EXISTE PLAGIO
60	0.017788	INGENIERÍA	NO EXISTE PLAGIO
61	0.009239	INGENIERÍA	NO EXISTE PLAGIO
62	0.031286	INGENIERÍA	NO EXISTE PLAGIO
63	0.010245	INGENIERÍA	NO EXISTE PLAGIO
64	0.002986	INGENIERÍA	NO EXISTE PLAGIO
65	0.010627	INGENIERÍA	NO EXISTE PLAGIO
66	0.013125	INGENIERÍA	NO EXISTE PLAGIO
67	0.007621	INGENIERÍA	NO EXISTE PLAGIO
68	0.020004	INGENIERÍA	NO EXISTE PLAGIO
69	0.006935	INGENIERÍA	NO EXISTE PLAGIO
70	0.005851	INGENIERÍA	NO EXISTE PLAGIO
71	0.026080	INGENIERÍA	NO EXISTE PLAGIO
72	0.005674	INGENIERÍA	NO EXISTE PLAGIO
73	0.011135	INGENIERÍA	NO EXISTE PLAGIO
74	0.009948	INGENIERÍA	NO EXISTE PLAGIO
75	0.011163	SOCIALES	NO EXISTE PLAGIO
76	0.004737	SOCIALES	NO EXISTE PLAGIO

77	0.006557	SOCIALES	NO EXISTE PLAGIO
78	0.006672	SOCIALES	NO EXISTE PLAGIO
79	0.015081	SOCIALES	NO EXISTE PLAGIO
80	0.010081	SOCIALES	NO EXISTE PLAGIO
81	0.012583	SOCIALES	NO EXISTE PLAGIO
82	0.004917	SOCIALES	NO EXISTE PLAGIO
83	0.014369	SOCIALES	NO EXISTE PLAGIO
84	0.006091	SOCIALES	NO EXISTE PLAGIO
85	0.005192	SOCIALES	NO EXISTE PLAGIO
86	0.023405	SOCIALES	NO EXISTE PLAGIO
87	0.011573	SOCIALES	NO EXISTE PLAGIO
88	0.016532	SOCIALES	NO EXISTE PLAGIO
89	0.006971	SOCIALES	NO EXISTE PLAGIO
90	0.005755	SOCIALES	NO EXISTE PLAGIO
91	0.006507	SOCIALES	NO EXISTE PLAGIO
92	**0.356796**	**SOCIALES**	**SI EXISTE PLAGIO**
93	0.012651	SOCIALES	NO EXISTE PLAGIO
94	0.009093	SOCIALES	NO EXISTE PLAGIO
95	0.006403	SOCIALES	NO EXISTE PLAGIO
96	0.018119	SOCIALES	NO EXISTE PLAGIO
97	0.010918	SOCIALES	NO EXISTE PLAGIO
98	0.010909	SOCIALES	NO EXISTE PLAGIO
99	0.010882	SOCIALES	NO EXISTE PLAGIO
100	0.010040	SOCIALES	NO EXISTE PLAGIO
101	0.007755	SOCIALES	NO EXISTE PLAGIO
102	0.009338	SOCIALES	NO EXISTE PLAGIO
103	0.017321	SOCIALES	NO EXISTE PLAGIO

104	0.005107	SOCIALES	NO EXISTE PLAGIO
105	0.014241	SOCIALES	NO EXISTE PLAGIO
106	0.009825	SOCIALES	NO EXISTE PLAGIO
107	**0.259642**	**SOCIALES**	**SI EXISTE PLAGIO**
108	0.019211	SOCIALES	NO EXISTE PLAGIO
109	0.012397	SOCIALES	NO EXISTE PLAGIO
110	0.015022	SOCIALES	NO EXISTE PLAGIO
111	0.009077	SOCIALES	NO EXISTE PLAGIO
112	0.005636	SOCIALES	NO EXISTE PLAGIO
113	0.019332	SOCIALES	NO EXISTE PLAGIO
114	0.004319	SOCIALES	NO EXISTE PLAGIO
115	0.007382	SOCIALES	NO EXISTE PLAGIO
116	0.011582	SOCIALES	NO EXISTE PLAGIO
117	0.025484	SOCIALES	NO EXISTE PLAGIO
118	0.008341	SOCIALES	NO EXISTE PLAGIO
119	0.031656	SOCIALES	NO EXISTE PLAGIO
120	0.007449	SOCIALES	NO EXISTE PLAGIO
121	0.029623	SOCIALES	NO EXISTE PLAGIO
122	0.014161	SOCIALES	NO EXISTE PLAGIO
123	0.009368	SOCIALES	NO EXISTE PLAGIO
124	0.005635	SOCIALES	NO EXISTE PLAGIO
125	0.012303	SOCIALES	NO EXISTE PLAGIO
126	0.011406	SOCIALES	NO EXISTE PLAGIO
127	0.020592	SOCIALES	NO EXISTE PLAGIO
128	0.011109	SOCIALES	NO EXISTE PLAGIO
129	0.014297	SOCIALES	NO EXISTE PLAGIO
130	0.037532	SOCIALES	NO EXISTE PLAGIO

131	0.008235	SOCIALES	NO EXISTE PLAGIO
132	0.014337	SOCIALES	NO EXISTE PLAGIO
133	0.011211	SOCIALES	NO EXISTE PLAGIO
134	0.013903	SOCIALES	NO EXISTE PLAGIO
135	0.004301	SOCIALES	NO EXISTE PLAGIO
136	0.018765	SOCIALES	NO EXISTE PLAGIO
137	0.013343	SOCIALES	NO EXISTE PLAGIO
138	0.013031	SOCIALES	NO EXISTE PLAGIO
139	0.009232	SOCIALES	NO EXISTE PLAGIO
140	0.007233	SOCIALES	NO EXISTE PLAGIO
141	0.005322	SOCIALES	NO EXISTE PLAGIO
142	0.013445	SOCIALES	NO EXISTE PLAGIO
143	0.012475	SOCIALES	NO EXISTE PLAGIO
144	0.010012	SOCIALES	NO EXISTE PLAGIO
145	0.027441	SOCIALES	NO EXISTE PLAGIO
146	0.006818	SOCIALES	NO EXISTE PLAGIO
147	0.007719	SOCIALES	NO EXISTE PLAGIO
148	0.013031	SOCIALES	NO EXISTE PLAGIO
149	0.006872	BIOMÉDICAS	NO EXISTE PLAGIO
150	0.059034	BIOMÉDICAS	NO EXISTE PLAGIO
151	0.040966	BIOMÉDICAS	NO EXISTE PLAGIO
152	**0.264652**	**BIOMÉDICAS**	**SI EXISTE PLAGIO**
153	0.022672	BIOMÉDICAS	NO EXISTE PLAGIO
154	0.074703	BIOMÉDICAS	NO EXISTE PLAGIO
155	0.009232	BIOMÉDICAS	NO EXISTE PLAGIO
156	0.014692	BIOMÉDICAS	NO EXISTE PLAGIO
157	0.013329	BIOMÉDICAS	NO EXISTE PLAGIO

158	0.011793	BIOMÉDICAS	NO EXISTE PLAGIO
159	0.009935	BIOMÉDICAS	NO EXISTE PLAGIO
160	0.011204	BIOMÉDICAS	NO EXISTE PLAGIO
161	0.010832	BIOMÉDICAS	NO EXISTE PLAGIO
162	0.013648	BIOMÉDICAS	NO EXISTE PLAGIO
163	0.014622	BIOMÉDICAS	NO EXISTE PLAGIO
164	0.035244	BIOMÉDICAS	NO EXISTE PLAGIO
165	0.014769	BIOMÉDICAS	NO EXISTE PLAGIO
166	0.011569	BIOMÉDICAS	NO EXISTE PLAGIO
167	0.010255	BIOMÉDICAS	NO EXISTE PLAGIO
168	0.013600	BIOMÉDICAS	NO EXISTE PLAGIO
169	0.025714	BIOMÉDICAS	NO EXISTE PLAGIO
170	0.007622	BIOMÉDICAS	NO EXISTE PLAGIO
171	0.015377	BIOMÉDICAS	NO EXISTE PLAGIO
172	0.007243	BIOMÉDICAS	NO EXISTE PLAGIO
173	0.005626	BIOMÉDICAS	NO EXISTE PLAGIO
174	0.006712	BIOMÉDICAS	NO EXISTE PLAGIO
175	0.013056	BIOMÉDICAS	NO EXISTE PLAGIO
176	0.007879	BIOMÉDICAS	NO EXISTE PLAGIO
177	0.016446	BIOMÉDICAS	NO EXISTE PLAGIO
178	0.009190	BIOMÉDICAS	NO EXISTE PLAGIO
179	0.019618	BIOMÉDICAS	NO EXISTE PLAGIO
180	0.009631	BIOMÉDICAS	NO EXISTE PLAGIO
181	0.004407	BIOMÉDICAS	NO EXISTE PLAGIO
182	0.005955	BIOMÉDICAS	NO EXISTE PLAGIO
183	0.007762	BIOMÉDICAS	NO EXISTE PLAGIO
184	0.035814	BIOMÉDICAS	NO EXISTE PLAGIO

185	0.035281	BIOMÉDICAS	NO EXISTE PLAGIO
186	0.007105	BIOMÉDICAS	NO EXISTE PLAGIO
187	**0.286780**	**BIOMÉDICAS**	**SI EXISTE PLAGIO**
188	0.013281	BIOMÉDICAS	NO EXISTE PLAGIO
189	0.011051	BIOMÉDICAS	NO EXISTE PLAGIO
190	0.045707	BIOMÉDICAS	NO EXISTE PLAGIO
191	0.007628	BIOMÉDICAS	NO EXISTE PLAGIO
192	0.006056	BIOMÉDICAS	NO EXISTE PLAGIO
193	0.008453	BIOMÉDICAS	NO EXISTE PLAGIO
194	0.009576	BIOMÉDICAS	NO EXISTE PLAGIO
195	0.010461	BIOMÉDICAS	NO EXISTE PLAGIO
196	0.006185	BIOMÉDICAS	NO EXISTE PLAGIO
197	0.017727	BIOMÉDICAS	NO EXISTE PLAGIO
198	0.002682	BIOMÉDICAS	NO EXISTE PLAGIO
199	0.012709	BIOMÉDICAS	NO EXISTE PLAGIO
200	0.012652	BIOMÉDICAS	NO EXISTE PLAGIO
201	0.016339	BIOMÉDICAS	NO EXISTE PLAGIO
202	**0.251067**	**BIOMÉDICAS**	**SI EXISTE PLAGIO**
203	0.003884	BIOMÉDICAS	NO EXISTE PLAGIO
204	0.005412	BIOMÉDICAS	NO EXISTE PLAGIO
205	0.014960	BIOMÉDICAS	NO EXISTE PLAGIO
206	0.007414	BIOMÉDICAS	NO EXISTE PLAGIO
207	0.004149	BIOMÉDICAS	NO EXISTE PLAGIO
208	0.012209	BIOMÉDICAS	NO EXISTE PLAGIO
209	0.009500	BIOMÉDICAS	NO EXISTE PLAGIO
210	0.024787	BIOMÉDICAS	NO EXISTE PLAGIO
211	0.011764	BIOMÉDICAS	NO EXISTE PLAGIO

212	0.011430	BIOMÉDICAS	NO EXISTE PLAGIO
213	0.008246	BIOMÉDICAS	NO EXISTE PLAGIO
214	**0.209788**	**BIOMÉDICAS**	**SI EXISTE PLAGIO**
215	0.017384	BIOMÉDICAS	NO EXISTE PLAGIO
216	0.018520	BIOMÉDICAS	NO EXISTE PLAGIO
217	0.009470	BIOMÉDICAS	NO EXISTE PLAGIO
218	0.011332	BIOMÉDICAS	NO EXISTE PLAGIO
219	0.016873	BIOMÉDICAS	NO EXISTE PLAGIO
220	0.022952	BIOMÉDICAS	NO EXISTE PLAGIO
221	0.015024	BIOMÉDICAS	NO EXISTE PLAGIO
222	0.012574	BIOMÉDICAS	NO EXISTE PLAGIO
223	0.027693	BIOMÉDICAS	NO EXISTE PLAGIO
224	0.007621	BIOMÉDICAS	NO EXISTE PLAGIO

Como se indicó antes, el algoritmo propuesto consta de dos fases, la primera se refiere a la determinación del porcentaje de coincidencia del documento analizado con el corpus de referencia, si éste valor supera un cierto porcentaje (en el caso nuestro se estableció de más del 20%, entonces se pasa a la siguiente etapa del análisis).

4.1.5 Discusión

Es muy importante que se considere la primera fase, puesto que los algoritmos de detección de plagio en la segunda fase son bastante costosos (complejidad), y en la primera fase lo que se hace es reducir considerablemente el número de tesis de nuestro repositorio con la que se va a realizar la verificación exhaustiva.

Encontrar que el mejor método de referencia en la clasificación de los casos difíciles es el de los conjuntos de atributos de n-gramas, apoya la idea de que tener más atributos dedicados a cada uno de los diversos tamaños de las coincidencias exactas, ayuda a caracterizar el plagio y por tanto clasificarlo mejor. La efectividad del algoritmo mucho depende de los n-gramas establecidos y el por esa razón que para encontrar la mayor efectividad del algoritmo realizamos las pruebas variando éste valor en un rango de tres a cinco.

Finalmente, como se ha expuesto en la revisión del estado del arte, el tamaño del n-grama para un buen desempeño no es el mismo en *corpus* de naturalezas distintas, y en el caso de nuestros corpus de evaluación obtuvimos que para otros corpus de referencia, los mejores resultados fueron con una n igual a 2 ó 3.

4.2 IMPLEMENTAR Y EVALUAR LA PLATAFORMA DE SOFTWARE PARA LA DETECCIÓN DE PLAGIO CON UN CORPUS DE REFERENCIA EN TRABAJOS DE INVESTIGACIÓN

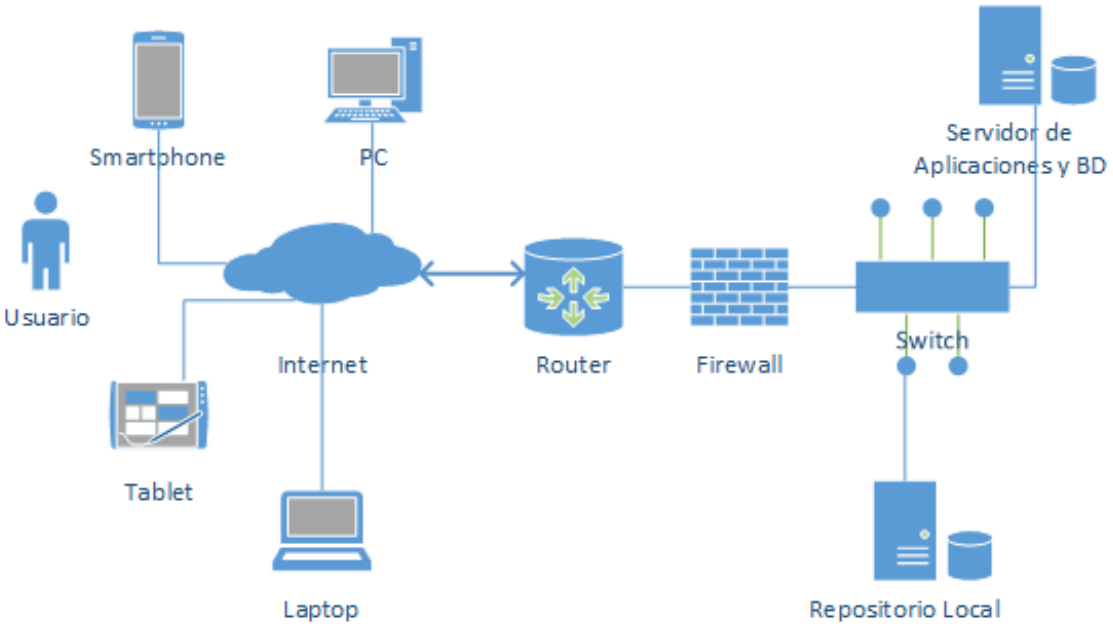
En esta parte del documento nos vamos a avocar a los resultados del objetivo específico número tres, que se refiere a la implantación y evaluación de la plataforma de software para determinar el nivel de plagio en los trabajos de investigación.

4.2.1.6 Diseño arquitectónico

Para la implementación de la plataforma se utilizó la siguiente arquitectura:

FIGURA 3

ARQUITECTURA DE LA PLATAFORMA DE SOFTWARE



En la anterior imagen tenemos un repositorio local, en éste servidor se encuentra almacenado todos los trabajos de investigación de la Universidad Nacional del Altiplano, éste repositorio la Universidad lo ha

implementado en coordinación con CONCYTEC (Consejo Nacional de Ciencia, Tecnología e innovación tecnológica), éste servidor se encuentra en la ciudad universitaria, específicamente en las instalaciones de vicerrectorado de investigación.

También tenemos un servidor de aplicaciones y base de datos, en éste servidor se aloja la plataforma de software con su respectiva base de datos, la aplicación web que gestiona la solicitud de verificación de plagio se encuentra alojada en éste equipo, en la que tenemos instalado aplicaciones tales como apache, PHP, MariaDB, Node.JS. A través de éste servidor también se ejecutarán los procesos críticos referentes a los algoritmos de comparaciones y lo harán contra el repositorio local, estos algoritmos están implementados en el lenguaje de programación C++.

Por cuestiones de seguridad siempre es necesario contar con sistemas de seguridad tanto a nivel físico y lógico y es por esa razón que en nuestro diseño arquitectónico se incluye un firewall, la seguridad lógica dependerá de las implementaciones de seguridad que se tenga en la Universidad Nacional del Altiplano, puesto que la plataforma se encuentra instalada dentro del campus universitario y la salida a Internet se realiza a través del acceso a internet que se tiene de manera integral para la Universidad.

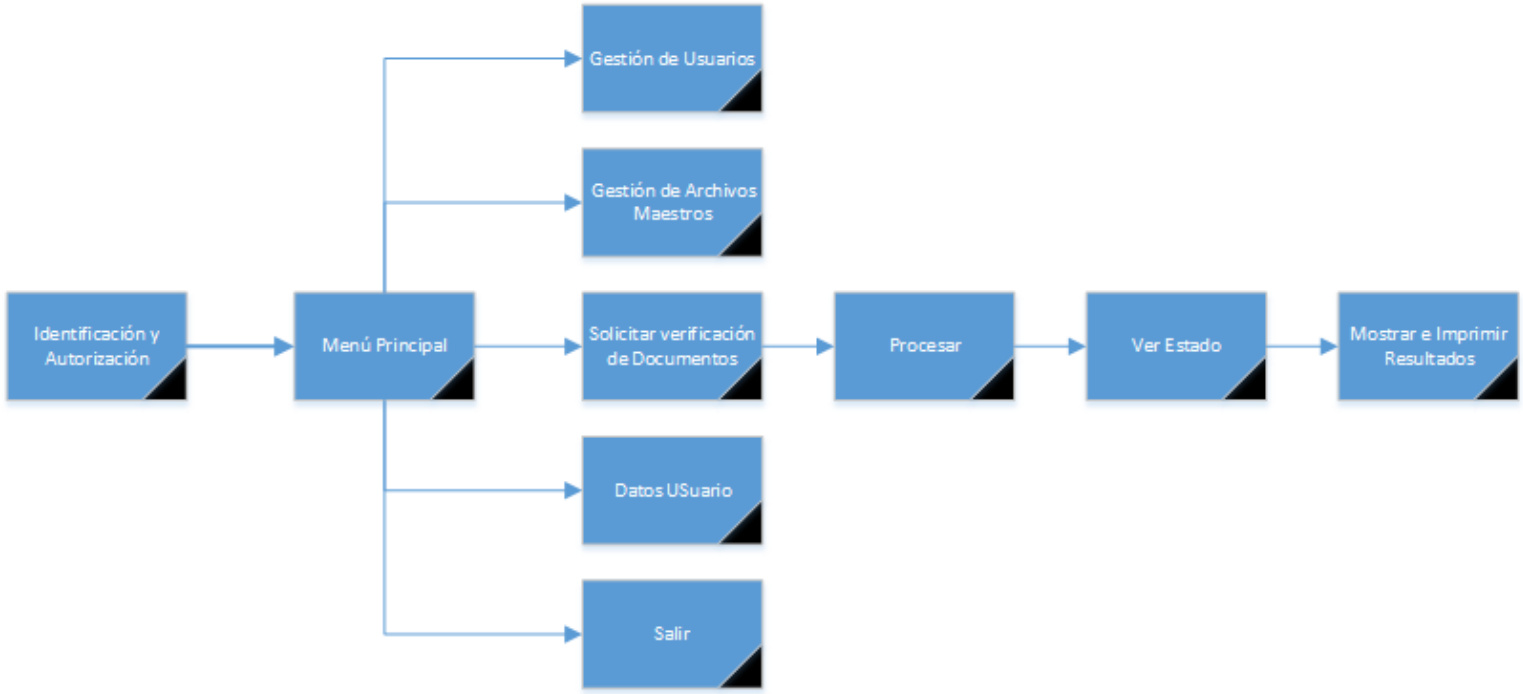
Finalmente el acceso a la plataforma se puede hacer desde cualquier equipo o dispositivo móvil que permita acceder a la plataforma Web y realizar las operaciones de solicitar la verificación de algún documento de investigación y nuestros algoritmos realizarán su trabajo internamente y al final mostrar resultados de dichas verificaciones contra los repositorios locales.

4.2.1.7 Diseño de navegación

En la siguiente figura se ilustra el diagrama de navegación en la plataforma de software.

FIGURA 4

DISEÑO DE NAVEGACIÓN EN LA PLATAFORMA DE SOFTWARE

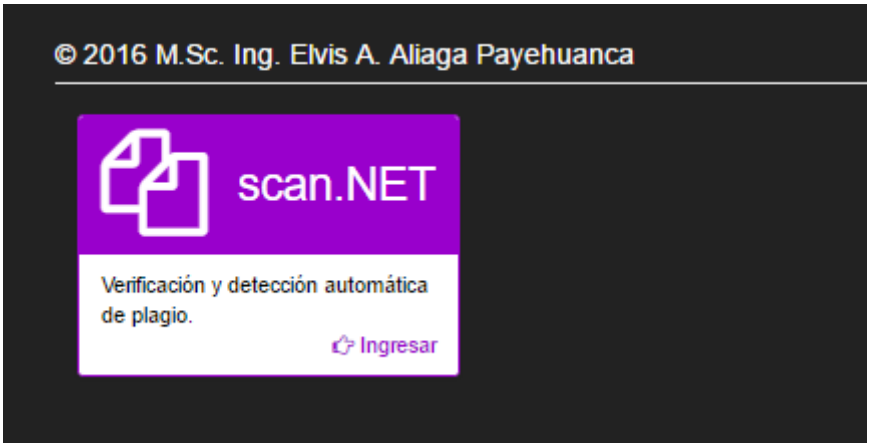


4.2.1.8 Diseño de prototipos

Para el diseño de interfaces se desarrolló prototipos evolutivos, desde las versiones iniciales funcionando para su entrega iteración tras iteración según el plan de entregas priorizada conjuntamente con el cliente se van a ir mejorando hasta finalmente tener una versión definitiva.

FIGURA 5

ICONO DE ACCESO DIRECTO A LA PLATAFORMA

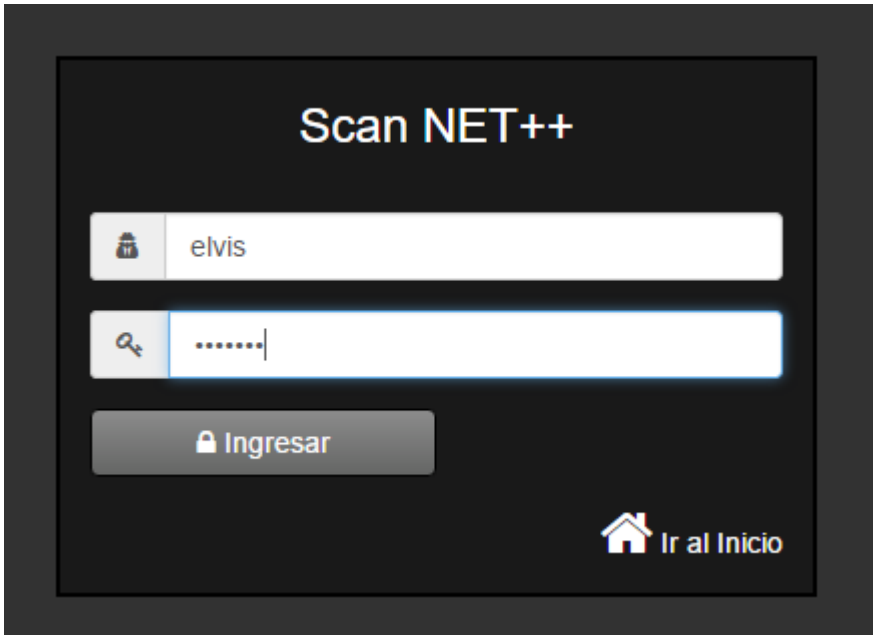


La plataforma de software para la detección de plagio en documentos de investigación, es una aplicación web, dicha aplicación se encuentra disponible vía internet y puede ser utilizado por cualquier usuario a través de la web.

Se recomienda la utilización del navegador web Chrome de Google, aunque funciona utilizado cualquier otro navegador.

FIGURA 6

VENTANA DE CONTROL DE ACCESO AL SOFTWARE



Todos los usuarios que hacen uso de la plataforma de software deben de estar previamente registrados. En primer lugar tienen de identificarse a través de la ventana anterior y una vez que en esta etapa se tenga éxito la plataforma procederá a autorizarle el acceso a los recursos del mismo que tiene permiso acceder.

La aplicación tiene sus opciones de recordar contraseña, esto con la finalidad de restablecer la contraseña a los usuarios en caso de olvido de las mismas, el procedimiento utilizado es a través del envío de un correo electrónico a sus cuentas previamente ingresadas al momento de su registro inicial.

Se tiene limitado los intentos de ingreso a la aplicación, es decir solo le es permitido cinco veces, si no se tiene éxito en esta cantidad de veces la cuenta automáticamente es bloqueada.

FIGURA 7

INTERFAZ PARA SELECCIONAR UN ARCHIVO PARA SU ANÁLISIS

A través de ésta interfaz se va a proceder seleccionar un archivo desde los dispositivos de almacenamiento. En ésta oportunidad se está limitando en cuanto al tipo de archivo aceptando solo archivos del tipo TXT, DOC y PDF. También estamos limitando (pero solo es cuestión de configurar los servidores web para eliminar tal restricción) el tamaño de los archivos a analizar a 2MB (mega bytes).

FIGURA 8

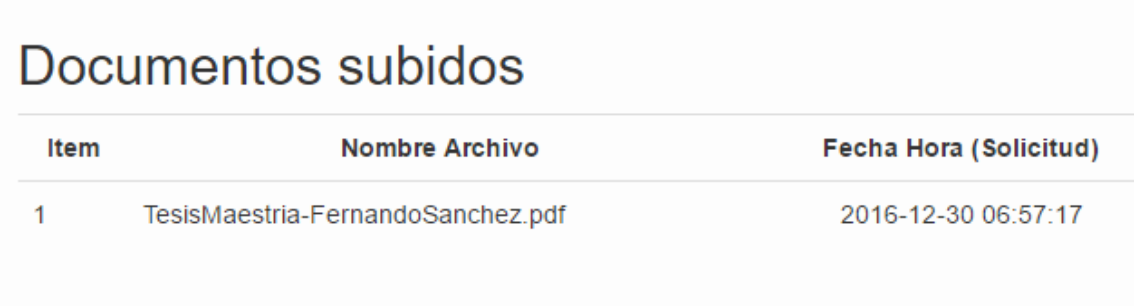
INTERFAZ PARA ENVIAR UN ARCHIVO PARA SU ANÁLISIS

La anterior figura nos muestra la interfaz una vez ya seleccionado el archivo, en la que se el nombre del archivo y el tamaño del mismo. Ya se encuentra listo para proceder a enviar el archivo que el usuario solicita para su verificación.

Una vez seleccionado el archivo, se procede a subir a los servidor dicho archivo, tarea de demorará según el tamaño del mismo y las características de la conexión a internet del usuario, se indica con una barra de progreso el estado de dicho proceso y una vez culminado se habilita de manera automática para que se pueda envirar otros archivos para ser analizados.

FIGURA 9

INTERFAZ PARA MOSTRAR LOS DOCUMENTOS ENVIADOS



Todos los archivos enviados para análisis se irán acumulando y lo mostramos indicando el nombre del documento con su respectiva extensión, la fecha que ha sido enviado para su análisis, el número de palabras de dicho documento y también tenemos como se muestra en la figura de abajo dos opciones adicionales que el mismo usuario podrá realizar: Iniciar proceso de verificación o la eliminación del documento.

FIGURA 10

INTERFAZ PARA ENVIAR UN ARCHIVO PARA SU ANÁLISIS

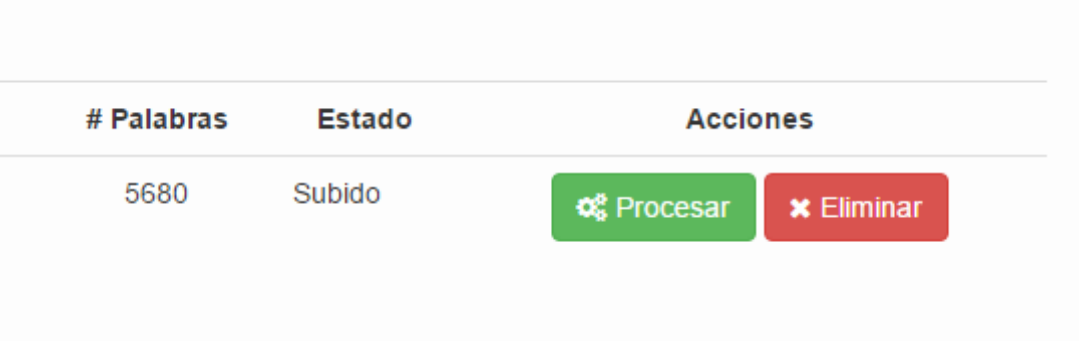


FIGURA 11

INTERFAZ PARA MOSTRAR LOS DOCUMENTOS EN PROCESO DE ANÁLISIS

En la figura anterior se muestran todos los archivos que ya iniciaron el procesamiento de verificación de coincidencias, éste proceso va a tardar dependiendo de la carga del procesador y del tamaño del archivo solicitado y de la cantidad de nuestros repositorios locales, en la vista se muestran la fecha de inicio de procesamiento, el número de palabras, el estado y tenemos una opción de actualización de la vista, aunque ésta actualización de realiza de manera automática a partir del servidor una vez haya terminado algún proceso, la tecnología usada para el manejo de los eventos desde el servidor es Node.JS.

La implementación del proceso de verificación, en sus fases de análisis preliminar y análisis definitivo se ha implementado con el lenguaje de programación de C++, estos procesos corren en segundo plano y son invocados desde PHP una vez que se ha hecho la preparación de los archivos.

FIGURA 12

INTERFAZ PARA MOSTRAR LOS DOCUMENTOS ANALIZADOS

En la figura anterior mostramos la interfaz que nos permite mostrar todos los documentos analizados, es decir documentos que ya se tiene resultados de las verificaciones de coincidencias contra los repositorios de vicerrectorado de investigación de la Universidad. Se muestran el nombre del documento, fecha y hora de solicitud de verificación, fecha y hora de finalización, el porcentaje de plagio de manera general, estado y finalmente tenemos una opción de ***revisar,*** que a través de ésta opción se puede ver los documentos fuentes de nuestros repositorios de donde han sido copiados.

Figura 13

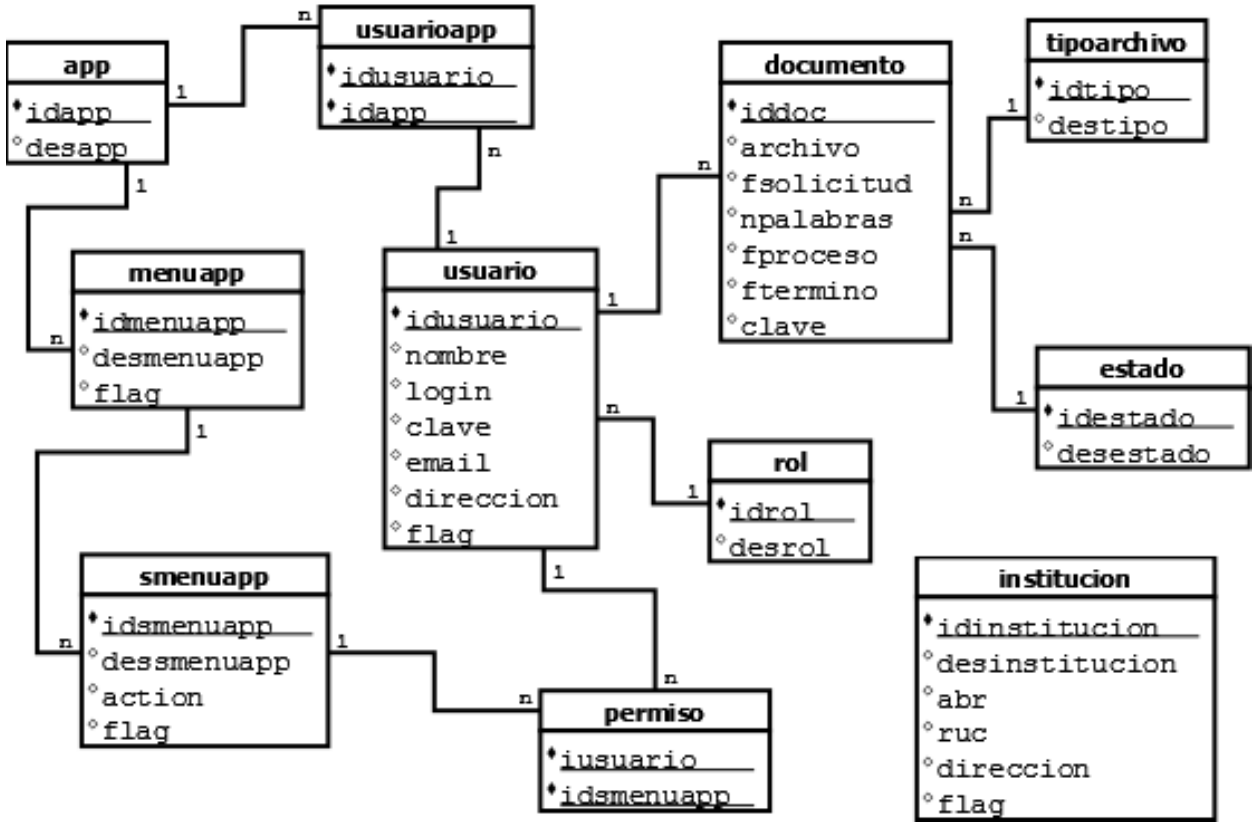
INTERFAZ PARA ENVIAR UN ARCHIVO PARA SU ANÁLISIS

Finalmente en la detección de plagio se determinan las coincidencias de que se han detectado, en ella se indica todos los documentos fuentes de las que han sido copiados y podemos explorar archivo por archivo y en ella indicar el número de coincidencias e indicar los párrafos o conjunto de palaras que han sido copiadas.

4.2.1.9 Diseño lógico de base de datos

FIGURA 14

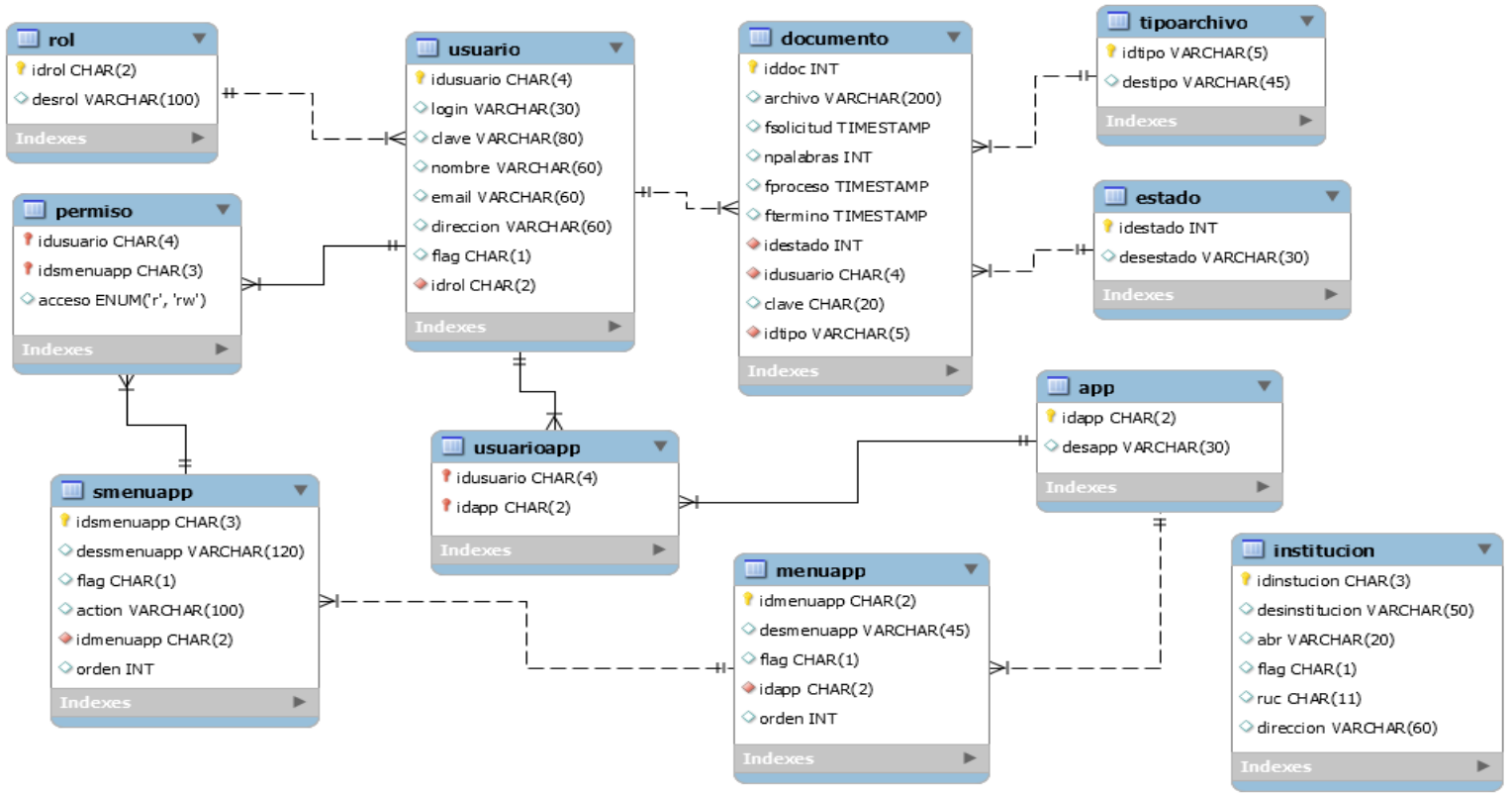
DISEÑO LÓGICO DE LA BASE DE DATOS



4.2.1.10 Diseño físico de la base de datos

FIGURA 15

DISEÑO FÍSICO DE LA BASE DE DATOS



4.2.1.11 Discusión

Según (Sánchez, 2007) la metodología XP no da mucho énfasis en el diseño, pero la metodología RUP lo considera como primordial y en la presente investigación también, puesto que permite representar el que se debe realizar y el cómo realizarlo, por lo que se está de acuerdo con lo que indica la metodología RUP, dado que sin una base no se base lo que se implementa, así mismo permite que se pueda realizar mantenimiento con mayor facilidad. También la metodología XP nos permite obtener la documentación necesaria del software.

4.2.2 EVALUACIÓN DE LA PLATAFORMA DE SOFTWARE

Luego de ver los resultados del objetivo específico número uno (Análisis, testeo de los diferentes algoritmos de detección de plagio existentes y proponer procesos y algoritmos más eficientes y optimizados con una complejidad O(n log n), memoria O(n) para la implementación en la Plataforma de Software) que es quizá el más importante, pero no deja de ser menos importante la ***calidad del producto de software,*** no debemos abocarnos solamente en los algoritmos de comparaciones que lógicamente se va a referir a los resultados del mimo y a la velocidad de respuesta, sino, hay que tener en cuenta otras cualidades, para buscar una integridad al afirmar que el software es de calidad.

Nosotros podemos decir que nuestro software es de calidad, pero quien realmente va a evaluar ello es el cliente. Y es por ello que en éste objetivo específico presentaremos los indicadores de calidad de la Plataforma de Software; basados en los estándares de calidad: la norma ISO/IEC 9126 (ISO/IEC 9126-2: métricas externas; ISO/IEC 9126-3: métricas internas; ISO/IEC 9126-4: métricas de calidad de uso); de la ISO (Organización

Internacional de Normalización) y la IEC (Comisión Electrotécnica Internacional).

Para la implementación de la plataforma de software previamente se realizó la evaluación respectiva, para dicha evaluación se tomó aspectos tales como la funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad.

4.2.2.1 Evaluación de la plataforma de software

Los requisitos necesarios previos para la evaluación de calidad del software se muestran en el siguiente cuadro:

CUADRO 9

REQUISITOS PARA APLICAR EL MODELO DE CALIDAD ISO/IEC 9126

Requisitos para aplicar el modelo de medición	**Tipo de calidad a medir**
Historias de usuario	Calidad interna
Tareas de ingeniería	
Diseños	
Código	
Pruebas	
Plataforma de software (producto final)	Calidad externa y calidad en uso

La calidad de la plataforma de software ha sido evaluada a través de las características y sub características de la calidad interna, externa y de uso según la norma ISO/IEC 9126. La fórmula utilizada para la sumatoria de características y sub características son las siguientes:

$$V_{sc} = \frac{\sum \boldsymbol{m}}{\mathbf{n}}$$

Donde:

V_{sc} : Valor de la sub característica

m : Valor de la métrica

n : Número de métricas

$$V_c = \frac{\sum mV_{sc}}{\mathbf{nsc}}$$

Donde:

V_c : Valor de la característica

V_{sc} : Valor de la sub característica

nsc : Número de sub características

4.2.2.1.1 Evaluación de la calidad interna y externa

En el cuadro 10 se muestra un 86% de calidad interna obtenida por la plataforma de software, lo que significa que el sistema tiene atributos de adecuación, seguridad, madurez, inteligibilidad y atractividad. La figura 18 ilustra que la característica de la funcionalidad resalta con un 99% sobre las características de fiabilidad y usabilidad.

CUADRO 10

EVALUACIÓN DE LA CALIDAD INTERNA (ISO/IEC 9126-3)

Característica	Sub	Métrica	m	Vsc	Vc	Valor total
Funcionalidad	Adecuación	Adecuación funcional	0.80	0.80	0.90	0.89
		Completitud de la implementació	0.79			
	Seguridad	Acceso controlable	0.99	0.99		
Fiabilidad	Madurez	Detección de fallas	0.80	0.90	0.90	
		Manejo de fallas	1.00			
Usabilidad	Inteligibilidad	Funciones evidentes	0.90	0.87	0.87	
		Funciones entendibles	0.84			
	Atractividad	Interacción atractiva	0.87	0.87		

En el cuadro 11 se muestra un valor total de 90% de calidad externa obtenida, éste valor consolida los valores que individualmente tienen los atributos de exactitud, interoperabilidad, madurez, inteligibilidad, operatividad, atractivilidad, comportamiento en el tiempo, instalación y coexistencia lo que a su vez nos indican que en la plataforma de software dichos atributos son aceptables.

CUADRO 11

EVALUACIÓN DE LA CALIDAD EXTERNA (ISO/IEC 9126-2)

Característica	Sub	Métrica	m	Vsc	Vc	Valor total
Funcionalidad	Exactitud	Precisión de las expectativas	0.70	0.70	0.84	0.90
	Interoperabilidad	Intercambio de datos	0.98	0.98		
Fiabilidad	Madurez	Densidad de fracaso frente a casos de prueba	0.88	0.94	0.94	
		Resolución de fallas	1.00			
Usabilidad	Inteligibilidad	Entendimiento de entradas y	0.93	0.93	0.86	
	Operatividad	Comprensión de mensajes en uso	0.80	0.80		
	Atractividad	Interacción atractiva	0.85	0.85		
Eficiencia	Comportamiento en el tiempo	Tiempo de respuesta	0.95	0.93	0.93	
		Tiempo de	0.90			
Portabilidad	Instalación	Facilidad de instalación	0.96	0.96	0.93	
	Coexistencia	Disponibilidad de coexistencia	0.90	0.90		

4.2.2.1.2 Evaluación de la calidad en uso

En el cuadro 12 mostramos que un 91% de calidad en uso obtenida, lo que significa que el modelo tiene atributos de eficacia, tiempo de proceso de la tarea, salud y seguridad del usuario, y satisfacción del usuario.

CUADRO 12

CUADRO DE LA EVALUACIÓN LA CALIDAD EN USO (ISO/IEC 9126-4)

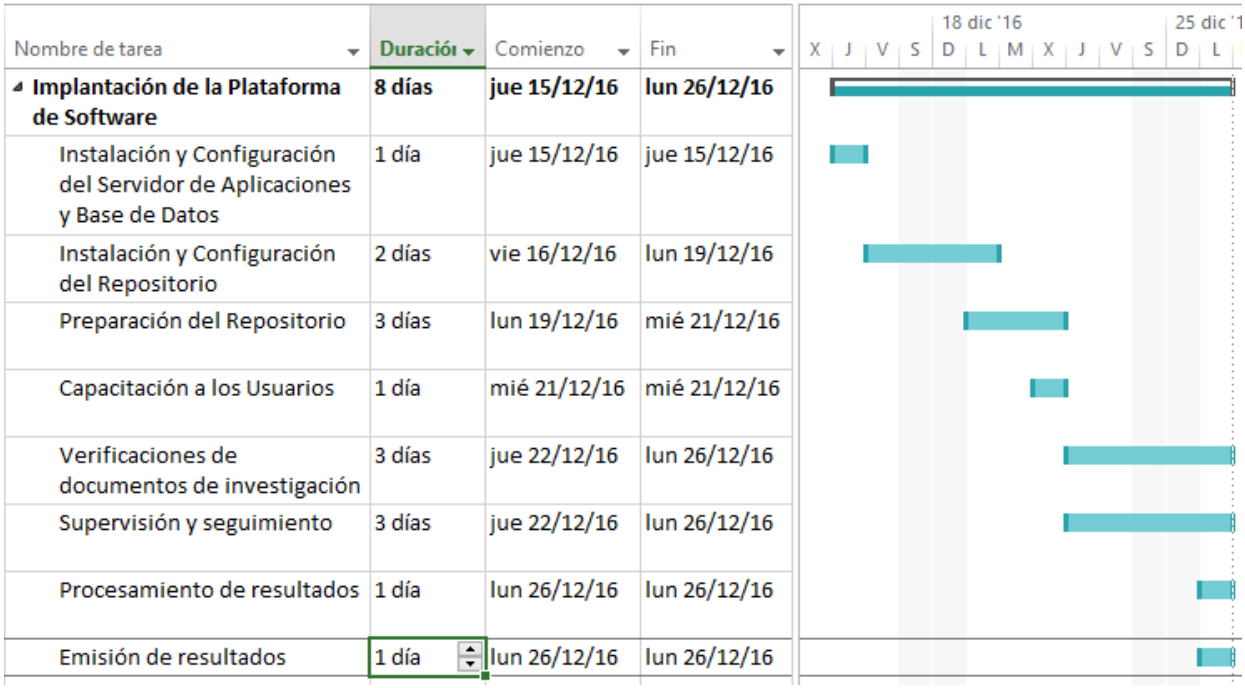
Característica	Métrica	m	Vc	Valor total medido
Efectividad	Eficacia en la tarea	0.95	0.95	
Productividad	Tiempo de tarea	0.90	0.90	
Satisfacción	Satisfacción del usuario	0.88	0.88	

4.2.2.2 Plan de implantación

En la figura 23 se muestra el plan de implantación de la plataforma de software, en éste plan se consideró las tareas tales como la instalación y configuración del servidor de aplicaciones y base de datos, instalación y configuración del repositorio de trabajos de investigación, preparación del repositorio, capacitación a los usuarios de la plataforma, la verificación de documentos de investigación contra el repositorio, procesamiento de resultados y finalmente la emisión de dichos resultados.

FIGURA 16

PLAN DE IMPLANTACIÓN DE LA PLATAFORMA DE SOFTWARE



Nombre de tarea	Duración	Comienzo	Fin
Implantación de la Plataforma de Software	**8 días**	**jue 15/12/16**	**lun 26/12/16**
Instalación y Configuración del Servidor de Aplicaciones y Base de Datos	1 día	jue 15/12/16	jue 15/12/16
Instalación y Configuración del Repositorio	2 días	vie 16/12/16	lun 19/12/16
Preparación del Repositorio	3 días	lun 19/12/16	mié 21/12/16
Capacitación a los Usuarios	1 día	mié 21/12/16	mié 21/12/16
Verificaciones de documentos de investigación	3 días	jue 22/12/16	lun 26/12/16
Supervisión y seguimiento	3 días	jue 22/12/16	lun 26/12/16
Procesamiento de resultados	1 día	lun 26/12/16	lun 26/12/16
Emisión de resultados	1 día	lun 26/12/16	lun 26/12/16

4.2.3 Discusión

De acuerdo a la ISO 9126 nos permite medir la calidad de un sistema sobre tres enfoques: calidad interna, calidad externa y calidad de uso, con lo cual se está de acuerdo dado que nos permitió verificar la calidad del modelo desarrollado con métricas solidas que nos da como referencia esta ISO, donde esta métrica nos permitió establecer que el sistema tendría la calidad necesaria para la salida a producción y no se produzca percances en la ejecución del plan de implementación.

4.2.4 Prueba de hipótesis para la proporción

a. **Hipótesis**

HIPOTESIS NULA (Ho): La proporción de los contenidos de un archivo no son considerados como plagio.

HIPOTESIS ALTERNATIVA (Ha): La proporción de los contenidos de un considerados como plagio.

b. **Probabilidad de significancia**

α = 0.05

Como n es mayor a 30, se utiliza la prueba estadística de la para lo cual se considera a Zt = Z(0.05) = 1.96

c. **Prueba estadística**

$$= \frac{p - P}{\sqrt{\frac{p * q}{n}}}$$

Zc = -0.146428571

0.015044804

Zc = -9.732833622

d. **Nivel de decisión**

Como Zc = -9,73 es mayor que Zt = 1,96, entonces se acepta la hipótesis alternativa.

CONCLUSIONES

Una vez terminada la investigación y dar por cumplimiento a los objetivos planteados, se concluye que:

En efecto, de acuerdo a los resultados obtenidos, la proporción de los contenidos de los archivos revisados son considerados como plagio al 5% de probabilidad de significancia, por lo tanto si existe plagio comparados con el repositorio de tesis en los documentos de investigación en los trabajos de investigación presentados en la Universidad Nacional del Altiplano de Puno.

Es muy importante que se considere la primera fase, puesto que los algoritmos de detección de plagio en la segunda fase son bastante costosos (complejidad), y en la primera fase lo que se hace es reducir considerablemente el número de tesis de nuestro repositorio con la que se va a realizar la verificación exhaustiva.

Referente al tamaño del n-grama para un buen desempeño no es el mismo en *corpus* de naturalezas distintas, y en el caso de nuestros corpus de evaluación obtuvimos que para otros corpus de referencia, los mejores resultados fueron con una n igual a 2 ó 3.

El uso de la norma ISO/IEC 9126 nos permitió obtener un nivel aceptable en cuanto a la calidad del software, para ésta evaluación se tomaron en cuenta la *calidad interna, externa y de uso*. Referente a la calidad interna se obtuvo un ***89% de calidad interna***, lo que significa que la Plataforma de Software tiene

características aceptables de funcionabilidad (adecuación, seguridad) con un 90%, fiabilidad (madurez) con un 90%, usabilidad (inteligibilidad, atractividad) con un 87% de calidad. En lo que respecta a la calidad externa, se obtuvo un ***90% de calidad externa,*** lo que significa que la Plataforma de Software tiene características aceptables de funcionabilidad (exactitud, interoperabilidad) con un 84%, fiabilidad (madurez) con un 94%, usabilidad (intelegibilidad, operatividad, atractividad) con un 86%, eficiencia (tiempo de respuesta, tiempo de espera) con un 93% y finalmente portabilidad (instalación, coexistencia) con un 93% de calidad. Finalmente, tenemos la calidad de uso, se obtuvo un ***91% de calidad de uso,*** lo que significa que la plataforma de software tiene características aceptables de efectividad (eficiencia en la tarea) con un 95%, productividad (tiempo de tarea) con un 90% y satisfacción (satisfacción del usuario con los resultados obtenidos) con un 88% de calidad.

RECOMENDACIONES

Después del trabajo realizado, se abren varias posibles trabajos a desarrollar o ampliaciones a realizar:

Probar el método propuesto y las ideas en las que se basan en otros *corpus* de plagio. Esto se podría realizar con el *corpus* de la competencia internacional de detección de plagio.

Teniendo ya desarrollada la detección del plagio, es conveniente desarrollar el resto de las etapas para poder realizar la identificación de plagio automático; en especial la búsqueda de plagio, que ha adquirido una mayor atención en el último año debido a la creación de la competencia internacional. Para ello es necesario realizar una etapa anterior a la detección que asegure obtener un subconjunto de unas pocas fuentes, entre los cuales se encuentre la verdadera fuente del plagio.

Hacer consideraciones especiales para los casos en los que el plagio está constituido primordialmente por cambios de palabras por sus sinónimos o antónimos. Es cierto que en el presente trabajo se ha considerado que este problema es manejable, si se hace la suposición de que el cambio es una simple eliminación seguida de una inserción de una nueva palabra. Pero, independientemente de esta simplificación, resulta interesante analizar cuáles podrían ser las opciones para poder sustentar la identificación de las

sustituciones que conserven el mismo sentido de aquellas que no lo hagan, sin requerir complejos recursos adicionales.

Finalmente se debería de trabajar en la detección de plagio en caso de texto traducido, puesto que es más complicado dichas detecciones de plagio puesto que no se encuentran las fuentes (en nuestro caso en el idioma castellano) sino que han sido traducidos de otros idiomas, especialmente el inglés.

BIBLIOGRAFÍA

- Alba L. (2008). SOA Arquitectura Orientada al Servicio. BIT, 167 - 168.
- Alberto, L., & Cedeño, B. (2008). Detección automática de plagio en texto.
- Buendía, L., & Berrocal De Luna, E. E. (2010). La Ética de la Investigación Educativa.
- Cemborain, M. S., & Valarino, E. (2011). PROGRAMAS PARA DETECTAR PLAGIO.
- Charya, N., Doshi, K., Bawkar, S., & Shankarmani, R. (2015). International Journal of Innovative and Emerging Research in Engineering Intrinsic Plagiarism Detection in Digital Data. *International Journal of Innovative and Emerging Research in Engineering*, *2*(3).
- Chunga Chinguel, G. (2014). Conoce herramientas gratuitas para detectar plagiosProfesorOnline. Retrieved May 5, 2016, from http://www.profesoronline.net/2014/11/05/conoce-herramientas-gratuitas-para-detectar-plagios-en-los-trabajos-de-investigacion/
- IEEE (2006). Plagiarism.
- Franco P., Gupta P. (2013). Detección de plagio translingue utilizando el diccionario estadístico de babelnet.

- Jacobs, M. (2010). *Factores que propician el plagio en la elaboración de trabajos de investigación en los alumnos de la Universidad de Ciencias Aplicadas*.
- Joskowicz, J. (2008). Reglas y Prácticas en eXtreme Programming. España:Universidad de Vigo.
- Kang, N., Gelbukh, A., & Han, S. (2006). PPChecker: Plagiarism Pattern Checker in Document Copy Detection (pp. 661–667). Springer Berlin Heidelberg. http://doi.org/10.1007/11846406_83
- Kulathuramaiyer N. (2013). Coping with the copy-paste-syndrome.
- Laudon, K., & Laudon, J. (2004). Sistemas de Información Gerencial. Mexico: Pearson Education.
- Luparenko, L. A. (2014). PLAGIARISM DETECTION TOOLS FOR RESEARCH WORKS: ANALYSIS OF SOFTWARE SOLUTIONS. *Information Technologies and Learning Tools*, *40*(2), 151–169.
- NTP-ISO/IEC 9126-4. (2005). Métricas de Calidad en Uso. Lima.
- Oberreuter, G., L 'huillier, G., Ríos, S. A., & Velásquez, J. D. (2011). Approaches for Intrinsic and External Plagiarism Detection Notebook for PAN at CLEF 2011.
- Patil, A. V. (2015). Plagiarism Software's Useful to Researchers : Analysis of few Softwares. *ASIAN JOURNAL OF MULTIDISCIPLINARY STUDIES*, *3*(12).
- Sampieri, R., Fernández-Collado, C., & Lucio, P. (2010). Metodología de la investigación Quinta Edición. México: McGraw-Hill/Interameriaca Editores
- Sanchéz, A. (2007). Metodologías agiles para el desarrollo de software. España

- Sánchez J. (2013). Detección automática de plagio basada en la distinción y fragmentación del texto reutilizado.
- Sommerville, I. (2011). *Ingeniería del Software* (Novena). 2011.
- Urbina S., Ozollo R., Gallardo J., Marti C., Torres A., Torrens M. (2012). Análisis de herramientas para la detección del ciberplagio.
- Weber-Wulff, D., Möller, C., Touras, J., & Zincke, E. (2013). Plagiarism Detection Software Test 2013.
- ISO/IEC, ISO/IEC 9126-4:2004 Software Engineering – Product quality. Part 4: Quality in Use Metrics, Secretaría General de ISO, Ginebra, 2004.

ANEXO 1: HISTORIAS DE USUARIO

Historia de usuario	
Número: 1	**Usuario:** Administrador / Usuario
Nombre historia: Identificación y autorización del usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Iteración asignada:** 1
Quiero: Ser en primer lugar identificado como usuario del sistema para comprobar si la identidad de cada usuario, verificando en primer lugar si son los que dicen ser. Posterior a la identificación sigue el proceso de autorización a los recursos que tiene permiso acceder.	
Para : Ingresar al Sistema para posteriormente ser autorizado a los diferentes recursos del Sistema según mi perfil de usuario.	

Historia de usuario	
Número: 2	**Usuario:** Administrador
Nombre historia: Administración de usuarios	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Medio
Puntos estimados: 4	**Iteración asignada:** 1
Quiero: Una interfaz que permita la gestión de usuarios que acceden a la plataforma de software: crear, modificar, eliminar, buscar.	
Para : Que nos debe de permitir dar altas a usuarios nuevos, modificar sus datos, dar de baja a los mismos, además de lo anterior nos debe de permitir administrar las autorizaciones de acceso a los recursos de la plataforma, es decir definir privilegios.	

Historia de usuario	
Número: 3	**Usuario:** Administrador
Nombre historia: Gestión de archivos maestros	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Medio
Puntos estimados: 4	**Iteración asignada:** 2
Quiero: Una interfaz que me permita gestionar todos los archivos maestros de la plataforma, tales como: estados, tipos de documentos, etc.	
Para : Tener opciones de crear, modificar, eliminar, buscar registros en las tablas maestras.	

Historia de usuario	
Número: 4	**Usuario:** Administrador / Usuario
Nombre historia: Subir archivo solicitado para la comparación	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 4	**Iteración asignada:** 1
Quiero: Una interfaz que me permita en primer lugar seleccionar un archivo en formato: txt, doc, pdf; una vez seleccionada debe de subirlo a los servidores.	
Para : Que una vez en los servidores se haga los análisis y comparaciones correspondientes.	

Historia de usuario	
Número: 5	**Usuario:** Administrador / Usuario
Nombre historia: Primera fase: Preparación del archivo	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 4	**Iteración asignada:** 1

Quiero: Un módulo que permita realizar la primera fase, que consiste en la conversión de los archivos Doc, Pdf según corresponda y convertirlos a textos plano y almacenarlos en el mismo servidor con el mismo nombre y la extensión *.txt.
Para : Que tener dichos archivos listos e iniciar las tareas de comparación.

Historia de usuario	
Número: 6	**Usuario:** Administrador / Usuario
Nombre historia: Segunda fase: Comparación preliminar contra el repositorio	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 5	**Iteración asignada:** 2
Quiero: Realizar una comparación del documento ya preparado en la primera fase, contra el repositorio local de todos los documentos de la Universidad, es decir se tiene que realizar la comparación ***preliminar*** de coincidencias del documento solicitado para su revisión con cada uno de los archivos que se encuentran en el repositorio.	

Para : Encontrar coincidencias y así poder determinar el plagio.

Historia de usuario	
Número: 7	**Usuario:** Administrador / Usuario
Nombre historia: Tercera fase: Comparación definitiva contra el repositorio	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 5	**Iteración asignada:** 2
Quiero: Realizar una comparación del documento ya preparado en la primera fase, contra el repositorio local de todos los documentos de la Universidad, es decir se tiene que realizar la comparación ***definitiva*** de coincidencias del documento solicitado para su revisión con cada uno de los archivos que se encuentran en el repositorio.	
Para : Encontrar coincidencias y así poder determinar el plagio.	
Historia de usuario	

Número: 8	**Usuario:** Administrador / Usuario
Nombre historia: Tercera fase: Determinación del plagio	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 5	**Iteración asignada:** 2
Quiero: Una vez encontrado las coincidencias del documento solicitado para su revisión, se debe de determinar el nivel de plagio que existe de dicho documento a partir del repositorio local de la Universidad, para la determinación de dicho plagio se debe de tener en cuenta aspectos como si una referencia está correctamente citada no constituye plagio, o palabras cortas que son comunes en cualquier redacción, etc.	
Para : Que tener el porcentaje de plagio y mostrar resultados.	

Historia de usuario	
Número: 9	**Usuario:** Administrador / Usuario
Nombre historia: Mostrar documentos analizados	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Medio
Puntos estimados: 4	**Iteración asignada:** 3
Quiero:	

Un módulo que permita en primer lugar mostrar los resultados, ésta opción es para los documentos ya finalizados, es decir, después de realizadas las comparaciones.
Para : Que nos permita ver que partes del documento son plagio, además al mismo tiempo nos debe de indicar cuales son las fuentes de donde se copiaron.

Historia de usuario	
Número: 10	**Usuario:** Administrador / Usuario
Nombre historia: Preparación del repositorio local	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 4	**Iteración asignada:** 3
Quiero: Un módulo que permita realizar la preparación de los archivos del repositorio local de la Universidad Nacional del Altiplano de Puno, puesto que los trabajos de investigación se encuentran digitalizados, pero el formato que utilizan es PDF, éstos archivos deben de ser preparados para que las comparaciones se realicen directamente.	

Para : Que tener dichos archivos listos e iniciar las tareas de comparación.

Historia de usuario	
Número: 11	**Usuario:** Administrador / Usuario
Nombre historia: Imprimir resultados	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada:** 3
Quiero: Un módulo que permita imprimir resultados de las comparaciones de los archivos enviados para su análisis respectivo contra los repositorios locales, en la misma impresión debe de mostrarse los resultados de dichas comparaciones, indicando que partes del documento son copias e indicando también los orígenes de donde son copiados.	
Para : Que los usuarios puedan imprimir sus resultados.	

ANEXO 2: TAREAS DE INGENIERÍA

Tarea de ingeniería	
Número: 1	**Número de historia:** 1 y 2
Nombre tarea: Identificación, autorización y gestión de usuarios	
Tipo de tarea: Desarrollo	**Puntos estimados:** 4
Fecha inicio: 01-10-2016	**Fecha fin:** 20-10-2016
Programador responsable: Elvis A. Aliaga Payehuanca	
Descripción: El usuario debe acceder a través de un usuario y una contraseña, dicha contraseña se almacena en la base de datos. Es indispensable que la contraseña se almacene encriptado utilizada alguna función del mismo gestor de base de datos.	

Tarea de ingeniería	
Número: 2	**Número de historia:** 3
Nombre tarea: Gestión de archivos maestros	
Tipo de tarea: Desarrollo	**Puntos estimados:** 4
Fecha inicio: 03-11-2016	**Fecha fin:** 05-11-2016
Programador responsable: Elvis A. Aliaga Payehuanca	
Descripción:	

Se debe de hacer la implementación de los CRUDs para cada uno de las tablas maestras de la plataforma de software, es decir debe de tener las opciones de Crear, Leer, Modificar, Eliminar datos en cada una de las tablas maestras.

Esta implementación también debe de ser restringida a través de los privilegios que cada usuario tienes, esto se debe de hacer en concordancia con la tarea de ingeniería N°1.

Tarea de ingeniería	
Número: 3	**Número de historia:** 4
Nombre tarea: Subir archivo solicitado para la comparación	
Tipo de tarea: Desarrollo	**Puntos estimados:** 4
Fecha inicio: 18-10-2016	**Fecha fin:** 20-10-2016
Programador responsable: Elvis A. Aliaga Payehuanca	

Descripción:

Se debe de implementar una interfaz Web que nos permita subir un archivo solo en uno de los siguientes formatos: PDF, DOC, TXT. La aplicación mientras se está subiendo el archivo debe de mostrar una barra de progreso, esto con la finalidad de indicarle al usuario el porcentaje de avance de la subida de su archivo.

El archivo en el formato indicado, se debe de guardar en una carpeta en el servidor de aplicaciones, el nombre que se le tiene que asignar es una secuencia aleatoria de veinte caracteres, esto pata evitar las duplicidades en los nombres.

Tarea de ingeniería

Número: 4	**Número de historia:** 5
Nombre tarea: Primera fase: Preparación del archivo	
Tipo de tarea: Desarrollo	**Puntos estimados:** 4
Fecha inicio: 15-10-2016	**Fecha fin:** 26-10-2016
Programador responsable: Elvis A. Aliaga Payehuanca	

Descripción:

Los archivos solicitados para su revisión, se encuentran en la carpeta files, el nombre asignado es una secuencia de veinte caracteres (éste nombre se encuentra almacenado en la base de datos relacionada al archivo). Se debe de hacer la preparación de dicho archivo, en primer lugar se tiene que hacer la conversión del archivo a un formato de texto plano (TXT), esto es con el lenguaje de programación PHP en segundo plano, posteriormente éste archivo ya en texto plano debe de procederse a la eliminación de los espacios en blanco mayores a uno, convertir todo a minúsculas, eliminación de caracteres especiales.

	Tarea de ingeniería
Número: 5	**Número de historia:** 6
Nombre tarea: Segunda fase: Comparación preliminar contra el repositorio	
Tipo de tarea: Desarrollo	**Puntos estimados:** 4
Fecha inicio: 03-11-2016	**Fecha fin:** 15-12-2016
Programador responsable: Elvis A. Aliaga Payehuanca	

Descripción:

Esta es la fase crítica de toda la plataforma de software, y por cuestiones de optimización de recursos de tiempo, hardware se tiene que hacer su implementación con el lenguaje de programación C++.

Lo que se tiene que hacer es tomar el archivo ya preparado (TXT) que es el que el usuario solicitó para su verificación y realizar la comparación ***preliminar*** contra todo el repositorio local de la Universidad.

Para realizar éste proceso ***no se requiere*** demasiados recursos, puesto que solo es para ver la posibilidad si un documento podría tener indicios de ser copia de otro.

	Tarea de ingeniería
Número: 6	**Número de historia:** 7
Nombre tarea: Segunda fase: Comparación definitiva contra el repositorio	

Tipo de tarea: Desarrollo	**Puntos estimados:** 4
Fecha inicio: 03-11-2016	**Fecha fin:** 23-12-2016
Programador responsable: Elvis A. Aliaga Payehuanca	

Descripción:

Esta es la fase crítica de toda la plataforma de software, y por cuestiones de optimización de recursos de tiempo, hardware se tiene que hacer su implementación con el lenguaje de programación C++.

Lo que se tiene que hacer es tomar el archivo ya preparado (TXT) que es el que el usuario solicitó para su verificación y realizar la comparación ***definitiva*** contra todo el repositorio local de la Universidad.

Para realizar éste proceso ***si se requiere*** demasiados recursos, por lo tanto es más costoso, pero, ya en la primera fase (preliminar) se ha desacartado gran parte del repositorio.

Tarea de ingeniería	
Número: 7	**Número de historia:** 8
Nombre tarea: Tercera fase: Determinación del plagio	
Tipo de tarea: Desarrollo	**Puntos estimados:** 4
Fecha inicio: 15-12-2016	**Fecha fin:** 26-12-2016
Programador responsable: Elvis A. Aliaga Payehuanca	

Descripción:

La determinación del plagio debe de realizarse a través de porcentajes, indicando el documento solicitado para su verificación es plagio de tal documento en tal porcentaje, para esto se debe de utilizar diversos criterios y no confundir con plagio cuando un texto es correctamente citado.

Tarea de ingeniería	
Número: 8	**Número de historia:** 9 y 11
Nombre tarea: Mostrar e imprimir resultados de análisis	
Tipo de tarea: Desarrollo	**Puntos estimados:** 4
Fecha inicio: 26-12-2016	**Fecha fin:** 26-12-2016
Programador responsable: Elvis A. Aliaga Payehuanca	

Descripción:

Un módulo que permita imprimir resultados de las comparaciones de los archivos enviados para su análisis respectivo contra los repositorios locales, en la misma impresión debe de mostrarse los resultados de dichas comparaciones, indicando que partes del documento son copias e indicando también los orígenes de donde son copiados.

Para mostrar éstos resultados, se debe de utilizar vistas que resalten en

colores las partes de documento copiados.

	Tarea de ingeniería
Número: 9	**Número de historia:** 10
Nombre tarea: Preparación del repositorio local	
Tipo de tarea: Desarrollo	**Puntos estimados:** 4
Fecha inicio: 15-12-2016	**Fecha fin:** 26-12-2016
Programador responsable: Elvis A. Aliaga Payehuanca	

Descripción:

Un módulo que permita realizar la preparación de los archivos del repositorio local de la Universidad Nacional del Altiplano de Puno, puesto que los trabajos de investigación se encuentran digitalizados, pero el formato que utilizan es PDF, éstos archivos deben de ser preparados para que las comparaciones se realicen directamente.

Los procesos de preparación de los archivos del repositorio tienen que hacerse en el lenguaje de programación PHP corriendo en segundo plano.

ANEXO 3: MATRIZ DE TRAZADO DE HISTORIAS DE USUARIO Y TAREAS DE INGENIERÍA

Tareas / Historias	T - 1	T - 2	T - 3	T - 4	T - 5	T - 6	T - 7	T - 8	T - 9
H - 1	X								
H - 2	X								
H - 3		X							
H - 4			X						
H - 5				X					
H - 6					X				
H - 7						X			
H - 8							X		
H - 9								X	
H - 10									X
H - 11								X	

ANEXO 4: PRIORIZACIÓN DE LAS HISTORIAS DE USUARIO

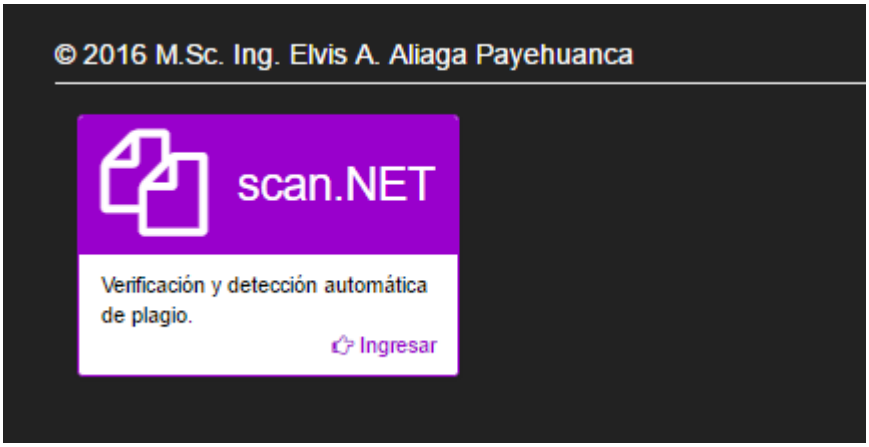
Nombre de historia	Prioridad	Esfuerzo
Historia 1: Identificación y Autorización del Usuario	Alta	3
Historia 2: Administración de Usuarios	Baja	4
Historia 3: Gestión de archivos maestros	Baja	3
Historia 4: Subir archivo solicitado para la comparación	Alta	3
Historia 5: Primera fase: Preparación del archivo	Alta	4
Historia 6: Segunda fase: Comparación preliminar contra el repositorio	Alta	5
Historia 7: Tercera fase: Comparación definitiva contra el repositorio	Alta	5
Historia 8: Tercera fase: Determinación del plagio	Alta	4
Historia 9: Mostrar documentos analizados	Media	4
Historia 10: Preparación del repositorio local	Media	4
Historia 11: Imprimir resultados	Baja	2

ANEXO 5: MANUAL DE USUARIO DEL SOFTWARE

SOFTWARE PARA DETECCIÓN AUTOMÁTICA DE SIMILITUD EN DOCUMENTOS DE INVESTIGACIÓN

El Software para la detección de similitudes en documentos de investigación posee una interfaz web para subir el archivo en formato PDF a ser evaluado y los algoritmos de verificación de similitudes se han implementado en el lenguaje de programación C++ (que es un lenguaje más robusto) que corre en el servidor.

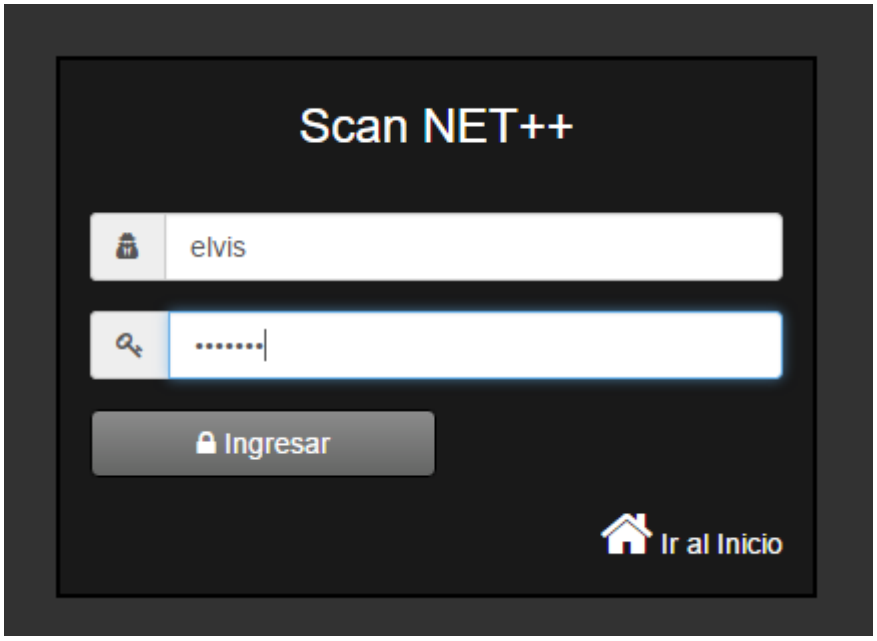
Icono de acceso directo a la plataforma



La plataforma de software para la detección de plagio en documentos de investigación, es una aplicación web, dicha aplicación se encuentra disponible vía internet y puede ser utilizado por cualquier usuario a través de la web.

Se recomienda la utilización del navegador web Chrome de Google, aunque funciona utilizado cualquier otro navegador.

Ventana de control de acceso al software



Todos los usuarios que hacen uso de la plataforma de software deben de estar previamente registrados. En primer lugar tienen de identificarse a través de la ventana anterior y una vez que en esta etapa se tenga éxito la plataforma procederá a autorizarle el acceso a los recursos del mismo que tiene permiso acceder.

La aplicación tiene sus opciones de recordar contraseña, esto con la finalidad de restablecer la contraseña a los usuarios en caso de olvido de las mismas, el procedimiento utilizado es a través del envío de un correo electrónico a sus cuentas previamente ingresadas al momento de su registro inicial.

Se tiene limitado los intentos de ingreso a la aplicación, es decir solo le es permitido cinco veces, si no se tiene éxito en esta cantidad de veces la cuenta automáticamente es bloqueada.

Interfaz para seleccionar un archivo para su análisis

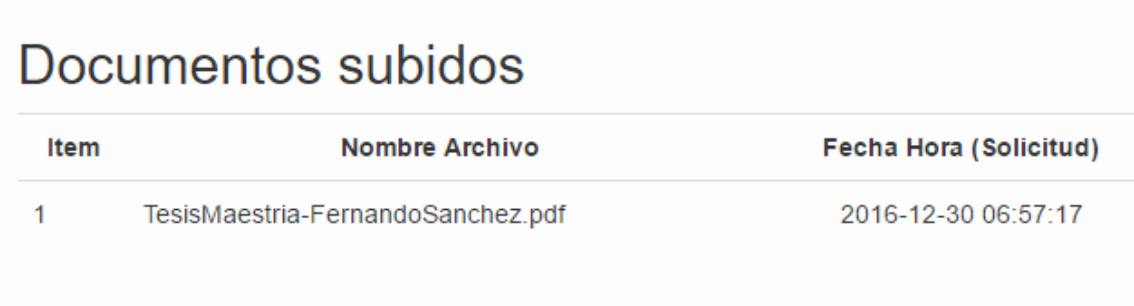
A través de ésta interfaz se va a proceder seleccionar un archivo desde los dispositivos de almacenamiento. En ésta oportunidad se está limitando en cuanto al tipo de archivo aceptando solo archivos del tipo TXT, DOC y PDF. También estamos limitando (pero solo es cuestión de configurar los servidores web para eliminar tal restricción) el tamaño de los archivos a analizar a 2MB (Mega Bytes).

Interfaz para enviar un archivo para su análisis

La anterior figura nos muestra la interfaz una vez ya seleccionado el archivo, en la que se el nombre del archivo y el tamaño del mismo. Ya se encuentra listo para proceder a enviar el archivo que el usuario solicita para su verificación.

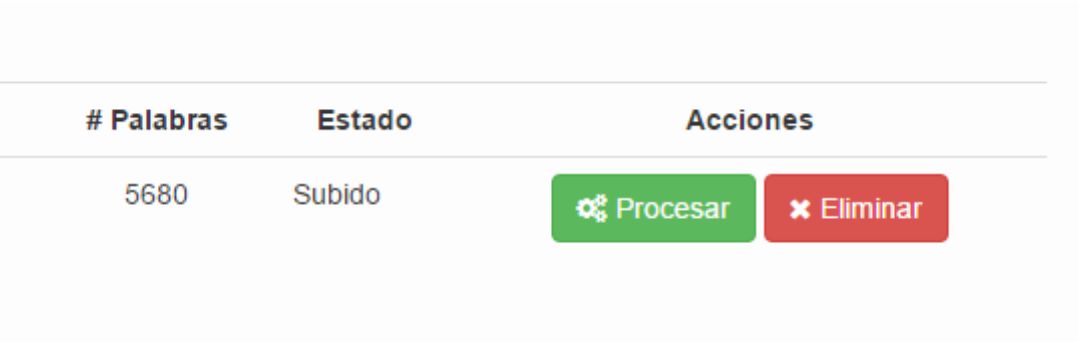
Una vez seleccionado el archivo, se procede a subir a los servidor dicho archivo, tarea de demorará según el tamaño del mismo y las características de la conexión a internet del usuario, se indica con una barra de progreso el estado de dicho proceso y una vez culminado se habilita de manera automática para que se pueda envirar otros archivos para ser analizados.

Interfaz para mostrar los documentos enviados



Todos los archivos enviados para análisis se irán acumulando y lo mostramos indicando el nombre del documento con su respectiva extensión, la fecha que ha sido enviado para su análisis, el número de palabras de dicho documento y también tenemos como se muestra en la figura de abajo dos opciones adicionales que el mismo usuario podrá realizar: Iniciar proceso de verificación o la eliminación del documento.

Interfaz para enviar un archivo para su análisis



Interfaz para mostrar los documentos en proceso de análisis

En la figura anterior se muestran todos los archivos que ya iniciaron el procesamiento de verificación de coincidencias, éste proceso va a tardar dependiendo de la carga del procesador y del tamaño del archivo solicitado y de la cantidad de nuestros repositorios locales, en la vista se muestran la fecha de inicio de procesamiento, el número de palabras, el estado y tenemos una opción de actualización de la vista, aunque ésta actualización de realiza de manera automática a partir del servidor una vez haya terminado algún proceso, la tecnología usada para el manejo de los eventos desde el servidor es Node.JS.

La implementación del proceso de verificación, en sus fases de análisis preliminar y análisis definitivo se ha implementado con el lenguaje de programación de C++, estos procesos corren en segundo plano y son invocados desde PHP una vez que se ha hecho la preparación de los archivos.

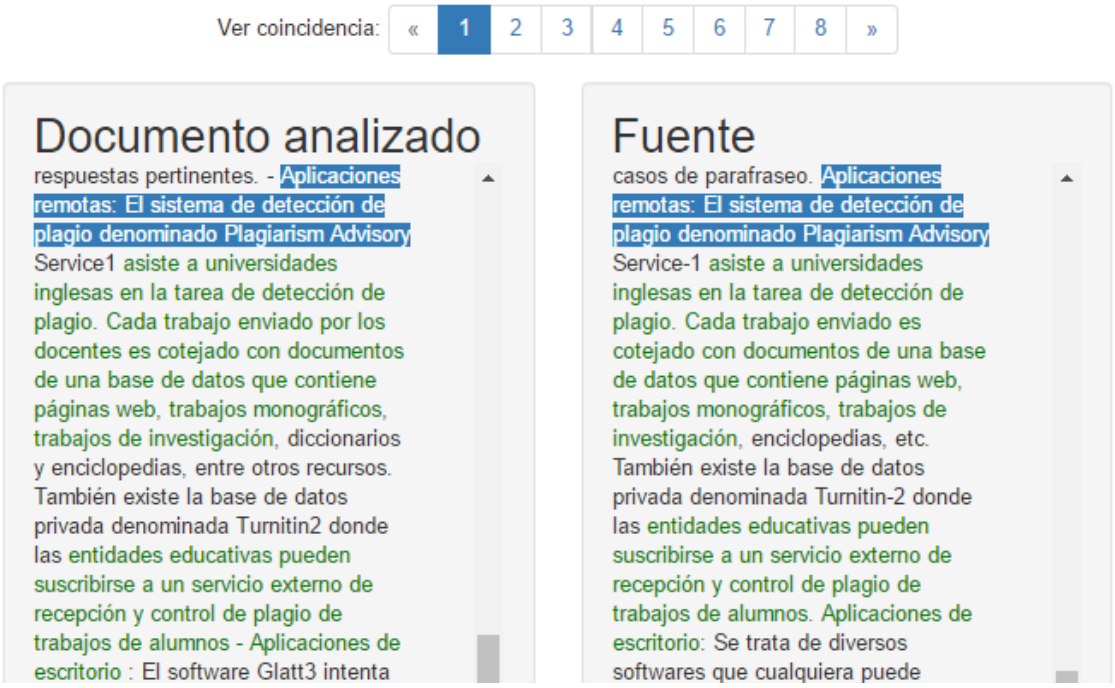
Interfaz para mostrar los documentos analizados

Elaboración: Del Ejecutor

En la figura anterior mostramos la interfaz que nos permite mostrar todos los documentos analizados, es decir documentos que ya se tiene resultados de las verificaciones de coincidencias contra los repositorios de vicerrectorado de investigación de la Universidad. Se muestran el nombre del documento, fecha y hora de solicitud de verificación, fecha y hora de finalización, el porcentaje de plagio de manera general, estado y finalmente tenemos una opción de ***Revisar,*** que a través de ésta opción se puede ver los documentos fuentes de nuestros repositorios de donde han sido copiados.

Interfaz para enviar un archivo para su análisis

Vista de las coincidencias con documentos fuentes del repositorio



En la imagen anterior se muestra la vista que nos permite mostrar a detalle cada una de las coincidencias de que se han detectado, en ella se indica todos los documentos fuentes de las que han sido copiados y podemos explorar archivo por archivo y en ella indicar el número de coincidencias e indicar los párrafos o conjunto de palaras que han sido copiadas.

Las coincidencias lo resaltamos de colores para que nos podamos guiar muy rápidamente en ésta parte del análisis de los resultados y también tenemos las opciones de impresión de los mismos.

ANEXO 6: CÓDIGO FUENTE PARA SUBIR EL ARCHIVE A ANALIZAR (PHP)

```php
<?php
    session_start();
    if(!isset($_SESSION['idusuario'])){
        header("Location: login.php");
    }
    $idusuario = $_SESSION['idusuario'];

    include 'cn.php';

    function generarCodigo($longitud) {
        $key = '';
        $pattern = '1234567890abcdefghijklmnopqrstuvwxyz';
        $max = strlen($pattern)-1;
        for($i=0;$i < $longitud;$i++) $key .=
$pattern{mt_rand(0,$max)};
        return $key;
    }

    $postdata = json_decode(file_get_contents("php://input"));
    $clave = generarCodigo(20);
    $file = basename($ FILES['file']['name']);
    $extension = end(explode(".", $_FILES['file']['name']));

    $sql = "insert into documento (archivo, idtipo, fsolicitud,
npalabras, idestado, clave, idusuario) ";
    $sql.= "values
('$file','$extension',CURRENT_TIMESTAMP,'200','1','$clave','$idusuario
') ";
    $cn->query($sql);

    $sql = "select max(iddoc) as maxid from documento ";
    $result = $cn->query($sql);
    $row = $result->fetch_array();
    $newid = str_pad($row['maxid'], 8, '0', STR_PAD_LEFT);


    # Guardamos


    $subir = move_uploaded_file($_FILES["file"]["tmp_name"],
"../files/".$clave.'.'.$extension);

?>
```

ANEXO 7: CÓDIGO FUENTE PARA CONVERTIR A TXT (PHP)

```php
<?php
    include 'cn.php';
    include 'Convertir.php';
    $postdata = json_decode(file_get_contents("php://input"));

    $id  = $postdata->id;

    $sql = "select clave from documento where iddoc = $id";
    $resultado = $cn->query($sql);
    $fila = $resultado->fetch_array();

    # convertir de PDF a TXT
    echo $nuevonombre =
Convertir::PDFaTXT('../files/'.$fila['clave'].'.pdf');

    $sql = "update documento set idestado='2',
fproceso=CURRENT_TIMESTAMP where iddoc='$id'";
    $cn->query($sql);
?>
```

ANEXO 8: CÓDIGO PRINCIPAL DE LA APLICACIÓN (C++)

```
double similarity[maxN];

int main(){

    SZ = 3; /// TRI_GRAMAS
    /// 01. READ THE CURRENT DOCUMENT
    freopen("current.txt","r",stdin);

    string current;
    unordered_map<HASH,long long> h_current;
    read1(h_current,current);
    current = normalize(current);


    /// 02. READ THE GLOBAL DOCUMENT
    freopen("global.txt","r",stdin);

    unordered_map<HASH,long long> h_global;
    read2(h_global);

    /// 03. READ ALL FILES TO ANALIZE EN REPOSITORY

    int cant_files = 1;
    for(int i = 0; i < cant_files; ++i){
        freopen("files1.txt","r",stdin);

        unordered_map<HASH, long long > h_file;
        read2(h_file);

        similarity[i] = coseno_inver_index(h_current , h_file ,
h_global);
    }

    //cout << current <<endl;
    vector<pair<int,int> > current_pos_words = get_idx_ini(current);
    CONTROL current_control(current_pos_words.size());
    HASHING current_hash(current.size());
    current_hash.preprocess(current);



    /// 04. ANALISIS DETALLADO
    cant_files = 1;
    for(int i = 0; i < cant_files; ++i){
        freopen("files1.txt","r",stdin);
        cout << i+1 <<"  = "<<similarity[i] << endl;

        ///if(similarity[i] > 0.001)

        {
            vector< vector<pair<int,int> > > info;
            vector<int> positions , len_positions;

            string other;
            read_simplest(other);
            other = normalize(other);
```

```
            vector<pair<int,int> > other_pos_words =
get_idx_ini(other);
            CONTROL other_control(other_pos_words.size());
            HASHING other_hash(other.size());
            other_hash.preprocess(other);
            //cout << other << endl;

            for(int tam = LEN_WORDS_MAXI; tam >= LEN_WORDS_MINI; tam--
){

generate_matches(current_pos_words,other_pos_words,current_hash,other_
hash,tam,current_control,other_control,info,positions, len_positions);
            }

            int len = positions.size();
            cout << len << endl;
            for(int k = 0; k < len; ++k){
                int x = current_pos_words[positions[k]].first;
                int y = current_pos_words[positions[k] +
len_positions[k] - 1].second;
                cout << current.substr(x,y-x+1)<<":" << endl;

              //  cout << x<<"-"<<y<<": "<<endl;
                for(auto it : info[k]){
                    x = other_pos_words[it.first].first;
                    y = other_pos_words[it.first + it.second -
1].second;
                   // cout << x<<"-"<<y<<", ";
                    cout << other.substr(x,y-x+1)<<" , ";
                   // cout << it.first <<" "<<it.second<<" - ";
                }
                cout << endl << endl;
            }
        }

    }

    /// 04. AGREGO AL REPOSITORIO , OUTPUT UN TXT ACTUALIZADO

    return 0;
}
```

ANEXO 9: CLASES PRINCIPALES (C++)

```
class HASHING{
    int n;
    vector<HASH> pot;
    vector<HASH> H;
    public:
        HASHING(int _n){
            n = _n;
            pot.assign(n + 1,0);
            H.assign(n + 1,0);
        }

        int N(){return n; }

        void preprocess(string &s){
            pot[0] = 1;
            for(int i = 1; i <= n; ++i){
                pot[i] = pot[i - 1] * B;
            }
            for(int i = 1; i <= n; ++i){
                H[i] = H[i-1] * B + s[i-1]-'a'+1;
            }
        }
        HASH read(int a,int b){ return H[b] - H[a]*pot[b-a]; }
};


double coseno_simplest(unordered_map<HASH,long long> &mapeoa,
unordered_map<HASH,long long> &mapeob){

    double up = 0;
    double low1 = 0;
    for(auto it : mapeoa){
        up += it.second * mapeob[it.first];
        low1 += it.second * it.second;
    }
    double low2 = 0;
    for(auto it : mapeob){
        low2 += it.second * it.second;
    }
    low1 = sqrt(low1);
    low2 = sqrt(low2);

    if(fabs(low1) < EPS || fabs(low2) < EPS){
        return 0;
    }

    return up / (low1 * low2);
}
void generate_matches(vector<pair<int,int> >
&current_pos_words,vector<pair<int,int> > &other_pos_words, HASHING
&current_hash,HASHING &other_hash,int tam,CONTROL
&current_control,CONTROL &other_control,vector<vector<pair<int,int > >
> &info,vector<int> &positions,vector<int> &len_positions){

    other_control.preprocess();
    int len_other = other_pos_words.size();
    unordered_map<HASH,vector<int> > mapeo_other;
```

```
    vector<bool> auxiliar = other_control.used;

    //cout << tam << endl;
    for(int i = 0; i + tam <= len_other; ++i){
        if(other_control.get_sum(i , i + tam - 1) == 0){
            int len = other_pos_words[i + tam - 1].second -
other_pos_words[i].first + 1;
          // cout << other pos words[i].first <<"
"<<other_pos_words[i].first + len - 1 <<" , ";

mapeo_other[other_hash.read(other_pos_words[i].first,other_pos_words[i
].first + len)].push_back(i);
        }
    }
    //cout << endl;

    current_control.preprocess();
    int len_current = current_pos_words.size();

    unordered_map<HASH,int> inside;
    for(int i = 0; i + tam <= len_current; ++i){
        if(current_control.get_sum(i , i + tam - 1) == 0){
                int len = current_pos_words[i + tam - 1].second -
current_pos_words[i].first + 1;
                HASH tmp =
current_hash.read(current_pos_words[i].first
,current_pos_words[i].first + len);

                if(inside.find(tmp) != inside.end()){
                    info.push_back(info[inside[tmp]]);
                    positions.push_back(i);
                    len_positions.push_back(tam);
                    continue;
                }

                if(mapeo_other.find(tmp) != mapeo_other.end()){
                    vector<int> where = mapeo_other[tmp];

                    int idx = info.size();
                    bool valid = false;
                    len = where.size();
                    for(int k = 0; k < len; ++k){
                        if(auxiliar[where[k]] == false){
                            if(!valid){
                                info.push_back(vector<pair<int,int>
>());

                                positions.push_back(i);
                                len_positions.push_back(tam);
                            }
                            valid = true;
                            info[idx].push_back({where[k],tam});
                            for(int pos = max(0 , where[k] - tam + 1);
pos < where[k] + tam; pos++){
                                    auxiliar[pos] = true;
                            }
                            for(int pos = where[k]; pos < where[k] +
tam; pos++){

                                other_control.on(pos);
                            }
                        }
                    }
```

```
                        if(valid){
                            for(int pos = i; pos < i + tam; ++pos){
                                current_control.on(pos);
                            }
                            i += tam - 1;
                            inside[tmp] = idx;
                        }
                    }
                }
            }
        }
```