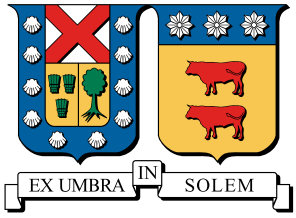
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



ANÁLISIS Y PROPUESTA DE MEJORAS A ALGORITMOS DE DETECCIÓN INTRÍNSECA DE PLAGIO

PABLO LUIS REYES FERNÁNDEZ

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

PROFESOR GUÍA	:	Dr. CLAUDIO TORRES
PROFESOR CORREFERENTE	:	Dr. MARCELO MENDOZA
PROFESOR CORREFERENTE	:	Dr. JUAN PABLO CÁRDENAS

Noviembre - 2016

Dedico este trabajo a mi familia, en especial a mis padres,
cuyo apoyo incondicional y arduo sacrificio me ha permitido llegar a donde estoy el día de hoy.

Resumen

El plagio es un problema que en los últimos años a tomado mayor fuerza debido a la gran cantidad documentos electrónicos presentes en la Internet, haciendo que la tarea de detectar plagio de manera manual sea prácticamente imposible. Debido a esto ha existido un aumento en investigación de detección automática de plagio, la cual en lo que a texto se refiere se ha desarrollado en dos grandes ramas:

- **Detección de plagio con referencia**
  Consiste en analizar los fragmentos de un documento y compararlo con un corpus de documentos originales, buscando si alguno de los fragmentos del documento en análisis está contenido en alguno de los documentos del corpus.
- **Detección intrínseca de plagio**
  Consiste en analizar los fragmentos de un documento, y sin la necesidad de contar con un corpus de documentos originales, determinar los fragmentos que puedan no haber sido escritos por el autor a partir del análisis de características estilográficas dentro del documento.

Este trabajo presenta una investigación de las técnicas actuales de detección automática de plagio en texto, para luego enfocarse en la *detección intrínseca de plagio*. Dentro del ámbito de la *detección intrínseca de plagio*, se revisan algunos algoritmos, analizando con mayor profundidad el algoritmo *DOCODE*, ganador de la competencia internacional de detección de plagio PAN el año 2011 en la categoría de detección intrínseca de plagio. Finalmente se proponen dos mejoras al algoritmo *DOCODE*, llamadas *DOCODE Normalizado* y *DOCODE Normalizado por Segmento*, las cuales son probadas utilizando el corpus de prueba puesto a disposición por la competencia PAN del año 2011 y evaluadas usando estándares fijados por dicha competencia. Los resultados muestran que las mejoras propuestas a *DOCODE* logran un mejor desempeño en la detección de posibles fragmentos plagiados.

Palabras Clave: Plagio, Detección intrínseca de plagio

Abstract

Plagiarism is a problem that in the last years has grown due to the many electronic documents on the Internet, making the task of detecting plagiarism manually is practically impossible. Because of this there has been an increase in research of automatic detection of plagiarism. The research of automatic plagiarism detection on text has developed into two main branches:

- **Plagiarism detection with reference**
  It is analyzing the fragments of a document and compare it with a corpus of original documents, searching if some fragment of the document is contained in any of the documents in the corpus
- **Intrinsic plagiarism detection**
  It is analyzing the fragments of a document, without the need for a corpus of original documents, looking the fragments that may not have been written by the author, after to study the characteristics of document style.

This work presents an investigation of current techniques on automatic plagiarism detection on text, for then focus on the *intrinsic plagiarism detection*. Within the scope of the *intrinsic plagiarism detection*, some algorithms are reviewed, more deeply the *DOCODE* algorithm, winner of the international competition plagiarism detection PAN 2011 in the category of intrinsic plagiarism detection. Finally, two improvements are proposed to the *DOCODE* algorithm, called *DOCODE Normalizado* and *DOCODE Normalizado por Segmento*, both are tested using the corpus made available by the PAN competition of 2011 and evaluated using standards set by such competition. The results show that the proposed improvements to *DOCODE* algorithm achieve better performance in detecting possible plagiarized fragments.

Keywords: Plagiarism, Intrinsic plagiarism detection

Índice General

Índice de Figuras

Índice de tablas

Introducción

La Internet que conocemos hoy en día ha tenido un gran impacto en la vida de las personas. Ha creado nuevas formas de comunicación, ha conectado a los que están lejos, y por sobre todo se ha transformado en una fuente casi inagotable de datos. La Internet de hoy a permitido el acceso a todo tipo de información y se ha transformado en un gran aliado para estudiantes, académicos, científicos, investigadores y personas en general. Ahora, este acceso casi ilimitado a la información ha traído consigo algunos problemas, como por ejemplo, el aumento de los casos de plagio en documentos escritos.

La Real Academia Española [7] define la acción de plagiar como:

"Copiar en lo sustancial obras ajenas, dándolas como propias."

Debido a la gran cantidad de recursos existentes en Internet el problema del plagio es casi imposible solucionar de manera manual, y por lo mismo ha existido un aumento de la investigación de detección automática de plagio en textos digitales, siendo incluso la motivación de eventos mundiales como es el caso de la competencia PAN [11].

En este trabajo nos enfocaremos en la detección de plagio escrito en textos digitales, entendiéndose el acto de plagiar como la incorporación de fragmentos de otro documento de otro autor sin darle a este el crédito correspondiente.

Identificación del Problema

El plagio en textos digitales es un problema que ha ido en aumento principalmente debido al fácil acceso a documentos electrónicos. Es un problema actual, presente en ámbito académico, laboral y científico.

A nivel de texto, la acción de plagiar puede ocurrir de varias formas. Una de las más comunes consiste en copiar un fragmento textual sin incluir su fuente. Otra forma un poco más elaborada consiste en copiar un fragmento cambiando el orden de las palabras y/o reemplazando palabras por sinónimos. Una tercera forma es la inclusión de fragmentos traducidos desde otros documentos en otros idiomas. Finalmente existe el llamado plagio por referencia, el cuál se da cuando una referencia está en un documento y se incluye en otro sin haber leído el origen; Esta forma de plagio es muy difícil de detectar ya que si una referencia está bien hecha es casi imposible comprobar si el autor del documento leyó o no el origen de la referencia.

La tarea de la detección de plagio consiste en identificar los fragmentos de un documento que puedan no haber sido escritos por el autor del mismo. En este ámbito Barrón Cedeño en [1] agrupa los métodos de detección de plagio en dos grandes ramas: *detección intrínseca de plagio*, donde no es necesaria la comparación de un documento con otros, y la *detección de plagio con referencia* donde si existe la comparación entre documentos.

Este trabajo se enfocará en la *detección intrínseca de plagio*, capturando el estilo de escritura del autor del documento y determinando posibles fragmentos plagiados a partir de los cambios de estilo a lo largo del documento.

Objetivos

En el marco de esta Memoria, se ha definido una serie de objetivos generales y específicos, los cuales se presentan a continuación:

1. **Objetivo General**
   Analizar, mejorar e implementar algoritmos de detección de anomalías de estilo en textos digitales.

2. **Objetivos Específicos**

   - Analizar y conocer métodos y algoritmos que permiten detectar similitud entre textos digitales.
   - Adaptar y/o modificar métodos y algoritmos que permiten detectar similitud entre textos digitales en base al objetivo planteado.
   - Definir indicadores y métricas que permitan determinar los niveles y tipos de similitud entre textos digitales.
   - Formar una base de conocimiento técnico y teórico necesario para construir una herramienta de software que permita determinar de forma automática similitudes entre textos científicos.

Estructura de la Memoria

Este trabajo presenta en primera instancia una investigación de los métodos actuales de detección automática de plagio en textos digitales, para luego analizar con mayor profundidad algoritmos y métricas de detección intrínseca de plagio, es decir, detección de plagio sin necesidad de comparación con otros documentos. Finalmente, se propone mejoras para algunos de los algoritmos analizados y se verifica experimentalmente si las mejoras propuestas cumplen o no su objetivo, todo con el fin de contar con una base de conocimiento en pos del desarrollo futuro de una herramienta de detección de niveles de similitud entre textos científicos.

La Memoria ha sido estructurada en los siguientes capítulos:

Capítulo 1 - Estado del Arte: Da un recorrido por las técnicas, métricas, algoritmos y herramientas utilizadas en la detección de plagio, conocimientos que fueron aplicados en el desarrollo del presente trabajo.

Capítulo 2 - Análisis de Algoritmos de Detección de Plagio: Muestra un análisis más profundo a algunos de los algoritmos presentados en trabajos realizados por otros autores en el ámbito de la detección intrínseca de plagio.

Capítulo 3 - Propuesta de Mejoras al Algoritmo Analizado: Presenta mejoras propuestas al algoritmo DOCODE, algoritmo de detección intrínseca de plagio en textos digitales.

Capítulo 5 - Experimentos y Análisis de Resultados: Presenta los experimentos realizados y resultados obtenidos de la ejecución del algoritmo DOCODE y las mejoras propuestas a un corpus de documentos.

Capítulo 6 - Conclusiones: Finalmente, se entregan las conclusiones obtenidas luego de la realización de este trabajo.

Capítulo 1

Estado del Arte

La detección de plagio en textos digitales puede ser abordada desde varios aspectos y características del documento. Por ejemplo, en el ámbito científico, dos o más textos científicos pueden ser similares en algunos meta-datos como la disciplina en que se desarrolla la investigación, las palabras claves que definen los autores, entre otros atributos propios del documento y que manejan en su mayoría los repositorios de revistas científicas indexadas más importantes del mundo como la Web Of Science de Thomson Reuters [19].

Por otro lado, también es posible determinar secciones de un documento que puedan no haber sido escritas por el autor del mismo a partir del análisis del contenido del texto y/o determinar niveles de similitud entre documentos desde el análisis y comparación del texto contenido en uno o más documentos en sí.

En este capítulo describiremos los principales avances y métodos existentes hasta hoy sobre detección automática de plagio en texto, métodos que van en la misma linea del objetivo del presente trabajo.

1.1. Características fundamentales de un texto como indicadoras de plagio

Antes de analizar cualquier técnica o método de detección de plagio en texto, es necesario seleccionar ciertas características fundamentales que nos permitirán diferenciar entre textos plagiados y textos originales. En este ámbito, Clough [4] definió características de un texto que ayudan a determinar si este puede ser un caso de plagio, estas características, si bien fueron determinadas dentro del ámbito académico, son consideradas hasta hoy la base a explotar por cualquier método de detección automática de plagio. Las características descritas por Clough dentro del ámbito académico son:

1. **Uso de Vocabulario.** Analizar el vocabulario utilizado en alguna tarea con respecto a documentos escritos previamente por el mismo estudiante. La existencia de una alta cantidad de vocabulario nuevo podría ayudar a determinar si un estudiante realmente escribió un texto o no.
2. **Cambios de vocabulario.** Si el vocabulario utilizado en un texto cambia significativamente a través de un documento.
3. **Texto incoherente.** Si un texto fluye de manera inconsistente o confusa.
4. **Puntuación.** Es muy poco probable que dos autores utilicen los signos de puntuación exactamente de la misma manera.

5. **Cantidad de texto común entre documentos.** Es poco frecuente que dos documentos escritos de manera independiente compartan grandes cantidades de texto.

6. **Errores en común.** Resulta muy improbable que dos textos independientes tengan los mismos errores de escritura.

7. **Distribución de las palabras.** Es poco frecuente que la distribución en el uso de las palabras a través de textos escritos independientemente sea la misma.

8. **Estructura sintáctica del texto.** Resulta poco probable que dos textos compartan una estructura sintáctica común.

9. **Largas secuencias de texto en común.** Es poco probable que dos textos independientes (incluso cuando traten el mismo tema), compartan largas secuencias de caracteres o palabras consecutivas.

10. **Orden de similitud entre textos.** Si existe un conjunto significativo de palabras o frases comunes en dos textos, puede haber un caso de plagio.

11. **Dependencia entre ciertas palabras y frases.** Un autor tiene preferencias sobre el uso de ciertas palabras y frases. Encontrarlas en un trabajo realizado por otro, debe ser considerado sospechoso.

12. **Frecuencia de palabras.** Es poco común que las palabras halladas en dos textos independientes sean usadas con la misma frecuencia.

13. **Preferencia por el uso de sentencias cortas o largas.** Los autores pueden tener una marcada preferencia sobre la longitud de las sentencias. Dicha longitud podría ser poco usual en compañía de otras características.

14. **Legibilidad del texto.** Resulta improbable que dos autores compartan las mismas medidas de legibilidad, tales como los índices de Gunning, Flesch o SMOG.

15. **Referencias incongruentes.** La aparición de referencias en el texto que no se encuentran en la bibliografía o viceversa son disparadores de un posible caso de plagio.

Tomando como base estas características de un texto, podemos comenzar a interiorizarnos en métodos de detección de plagio propiamente tales. Barrón Cedeño [1] agrupa los métodos de detección en dos grandes ramas: los métodos de *análisis intrínseco de plagio*, donde no es necesaria la comparación de un documento con otros, y los métodos de *detección de plagio con referencia* donde si existe la comparación entre documentos. Utilizaremos esta misma distribución para presentar los distintos métodos de detección de plagio.

1.2. Análisis intrínseco de plagio

Este tipo de métodos de detección de plagio consiste en analizar un documento sin realizar una comparación de este con otros documentos.

La idea principal es capturar el estilo y la complejidad a través de un documento sospechoso con el afán de encontrar fragmentos inusuales que sean candidatos a ser casos de plagio. Esto se logra analizando las palabras más usadas por el autor, conectores y forma de escribir, entendiéndose que cualquier excepción a la norma sea un posible caso de plagio.

Meyer zu Eissen y Kulig [5] realizaron un trabajo abordando esta linea investigativa de la detección de plagio en texto. En este trabajo indican que la detección intrínseca de plagio puede

ser trabajada dividiendo un documento en partes más pequeñas como oraciones, párrafos o secciones y a partir de estas cuantificar el estilo de escritura del autor del documento. Cada autor desarrolla un estilo individual de escritura, usando consciente o inconscientemente patrones de construcción de oraciones y uso individual de vocabulario, generándose características de aspectos estilométricos que pueden ser cuantificados mediante la construcción de fórmulas y que permiten discriminar entre textos que pertenecen o no al autor.

Meyer zu Eissen et al. indican que la mayoría de las características estilométricas se basan en las siguientes características semióticas[1]:

1. Estadísticas de texto, que operan a nivel de tipo.

   Ejemplos: número de comas, signos de interrogación, longitud de palabra.

2. Características sintácticas, que miden el estilo de escritura a nivel de la oración.

   Ejemplos: longitudes de oraciones, uso de palabras funcionales.

3. Características Part-of-speech para cuantificar el uso de las clases de palabras.

   Ejemplos: número de adjetivos o pronombres.

4. Número promedio de palabras de paro.

   Ejemplos: número de stopwords, palabras en otro idioma, palabras complejas

5. Características estructurales, que reflejan la organización del texto.

   Ejemplos: longitud de párrafos, longitud de capítulos

Meyer zu Eissen et al. [5] presentan a partir de esto una serie de índices, los cuales complementa Luis Barrón Cedeño [1] completándose un total de 9 parámetros que permiten cuantificar el estilo y complejidad de escritura de un autor, los cuales son calculados considerando el texto completo del documento sospechoso, y luego para cada párrafo. Con los resultados obtenidos, se realiza una comparación para buscar las variaciones que puedan ser reflejo de fragmentos que no fueron escritos por el mismo autor, es decir, candidatos a estar plagiados.

Sea s un documento sospechoso, los parámetros a considerar son:

1. **Promedio de clases de palabras basado en la frecuencia.**

   Cada palabra $w \in s$ es asignada a una clase denominada $c(w)$. La clase asignada a cada palabra depende de su frecuencia en el documento. La palabra (o conjunto de palabras) que presente la máxima frecuencia de aparición en s es denominada w^* y se asocia a la clase c_0 . El resto de palabras en s es asignado a la clase cuyo subíndice se determina por $\left\lfloor \frac{\log_2(f(w^*)}{f(w)} \right\rfloor$, en donde $\lfloor \cdot \rfloor$ es la función piso. Esta medida refleja la complejidad y el tamaño del vocabulario de un documento. Se utiliza debido a que suele ser bastante estable cuando se analiza un documento original, escrito por un único autor, sin importar su longitud.

2. **Longitud de las oraciones.**

   El promedio de la longitud de oraciones suele ser relativamente uniforme a través de un documento escrito por un autor.

[1] La semiosis es cualquier forma de actividad, conducta o proceso que involucre signos. Incluyendo la creación de un significado. Es un proceso que se desarrolla en la mente del intérprete; se inicia con la percepción del signo y finaliza con la presencia en su mente del objeto del signo.[21]

3. **Partes de la oración.**

   Las categorías gramaticales utilizadas hablan del estilo de escritura de un autor.

4. **Número promedio de palabras de paro.**

   El uso de determinados artículos, preposiciones y otras palabras que no aportan demasiado significado a un texto puede ser completamente diferente de un autor a otro.

5. **índice de confusión de Gunning.**

   Esta medida ha sido diseñada para determinar que tan comprensible es un texto escrito (particularmente en inglés). El valor obtenido por dicha medida es una aproximación al número de años de educación formal que una persona requiere para comprender un texto leyéndolo una sola vez.

   Dicho índice se calcula tomando una muestra de texto de alrededor de 100 palabras por medio de la siguiente ecuación:

$$I_G = 0.4 \left(\frac{\mid palabras \mid}{\mid oraciones \mid} + 100 \frac{\mid palabras\ complejas \mid}{\mid palabras \mid} \right)$$

   Donde $\mid \cdot \mid$ es el número de elementos $\cdot$ en la muestra de texto. Se considera que una palabra compleja contiene al menos tres sílabas (a excepción de los nombres propios, palabras compuestas o con sufijos como *es*, *ed* o *ing*).

6. **índice de Flesch-Kincaid.**

   Este índice es muy parecido al anterior y de nuevo intenta calcular los años de educación necesarios para comprender un documento.

   Su cálculo se hace por medio de la siguiente ecuación:

$$I_{FK} = 1.599\,\lambda - 1.1015\,\beta - 31.517$$

   donde λ es el número promedio de palabras de una sílaba por cada cien palabras y β es la longitud promedio de las oraciones en número de palabras.

7. **índice de Dale-Chall.**

   Este índice fue diseñado en los años cuarenta para determinar de nuevo los años de estudio necesarios para leer un texto.

   La fórmula para su cálculo es:

$$I_{DC} = 0.0496\,\beta + 0.1579\,\phi + 3.6365 \tag{1.1}$$

   donde ϕ es el porcentaje de ”palabras complejas” en el texto (es necesario definir previamente un vocabulario con estas palabras) y β es la longitud promedio de las oraciones en número de palabras.

8. **Función R.**

   Esta medida intenta capturar la variedad en el vocabulario de un autor.

   Se calcula por medio de la siguiente ecuación:

$$R = \frac{100\,\log(M)}{M^2} \tag{1.2}$$

   donde M es el número de palabras en el texto analizado.

9. **Función K.**

   Esta función es en realidad una alternativa para el cálculo de la riqueza de vocabulario obtenida por medio de la función R.

   Se calcula de la siguiente manera:

$$K = \frac{10^4 \left(\sum_{i=1}^{\infty} i^2 V_i - M \right)}{M^2} \tag{1.3}$$

   donde V_i es el número de palabras que aparece i veces en el texto. M es el número de palabras en el texto analizado.

1.3. Detección de plagio con referencia

La detección de plagio con referencia consiste en comparar un documento sospechoso con un conjunto de documentos originales.

Luis Barrón Cedeño [1] define formalmente esta rama de detección de plagio en texto como:

"Dado un corpus[2] D conformado por un conjunto de documentos originales y un documento sospechoso s, la tarea de detección de plagio puede reducirse a realizar una comparación exhaustiva del texto en s sobre el corpus D para responder a la pregunta: ¿Existe algún fragmento s_i perteneciente a s que esté incluido en algún documento de D?"

A continuación se mostrará una serie de métodos presentados por distintos autores en esta linea de detección de plagio.

Análisis a nivel de documentos

Uno de los primeros métodos de detección automática de plagio que existen es el Stanford Copy Analysis Mechanism, SCAM [16].

SCAM es capaz de detectar relaciones de plagio, subconjunto, copia y relación, los cuales significan que un documento contiene algunas partes de otro, está contenido en otro, es una copia de él o está fuertemente relacionado. Esto lo logra implementando los que los autores definen como *modelo de frecuencia relativa*, basado principalmente en una adaptación de similitud de coseno.

SCAM trabaja a nivel de documento, pero no siempre es necesario que un documento sea un duplicado de otro para que exista plagio, basta con que un fragmento sea extraído por otro y o realizado pequeñas modificaciones de forma para que exista un caso de plagio, caso que SCAM no es posible de detectar.

Hashing

El hashing o fingerprinting es una de las técnica más antiguas y más utilizada. Consiste en seleccionar partes de un texto y calcular el hash de esa porción, utilizando una función de hash criptográfica (MD5 generalmente) con la cual se procesan tanto los documentos del corpus de referencia como el texto sospechoso y luego se realiza una comparación de hashes. El problema de esta técnica es el denominado efecto avalancha: si cambia un bit del input, cambia drásticamente el output. Entonces, si una de las partes seleccionada difiere muy poco, el hash de esa parte va a diferir completamente, evitando que esa similitud sea detectada. Esto

[2]Conjunto de datos, textos u otros materiales sobre determinada materia que pueden servir de base para una investigación o trabajo

hace que la longitud de las partes a *hashear* no pueda ser muy grande: si se tomaran oraciones por ejemplo, se podría eludir la detección modificando sólo una palabra. Al mismo tiempo, cuanto menor sea la longitud de las partes, más costoso será el análisis (mayor tiempo de procesamiento, mayor espacio necesario para almacenar los hashes).

Modelo de espacio vectorial

Mario Zechner en [22], presenta un método basado en un modelo de espacio vectorial el cual trata el tema de la detección de plagio como la busqueda del vecino más cercano en un espacio vectorial de alta dimensión. El método propuesto consta de tres etapas:

- Vectorización de los pasajes de cada documento del corpus de referencia y separación del espacio vectorial corpus de referencia.
- Vectorización de los pasajes de un documento sospechoso y la búsqueda de vecino más cercano para cada pasaje en el espacio vectorial corpus de referencia. La detección de plagio para cada documento sospechoso se basa en su lista de vecinos más cercanos a través de umbrales de similitud.
- El procesamiento posterior de los pasajes plagiados detectados, la fusión de pasajes plagiados posteriores a un solo bloque.

Para realizar la vectorización, cada término único en el corpus de referencia se representa como una dimensión en el espacio vectorial $\mathbb{R}^d$, donde d es el número de palabras únicas. En lugar de crear un único vector de un documento completo se crean vectores para cada oración en un documento que se desea detectar plagio a nivel de oraciones.

Para cada oración, en cada documento corpus de referencia se construye un vector de frecuencias basado en palabras en minúscula, sin considerar los "stopwords".

Finalmente se utiliza el sistema de similitud de coseno para evaluar la similitud entre oraciones dependiendo de la distancia de los vectores, dado por:

$$SimilitudCoseno(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\| \vec{x} \|_2 \| \vec{y} \|_2}$$

donde $\vec{x}, \vec{y} \in \mathbb{R}^d$.

El problema de este tipo de métodos radica en el cambio de palabras por sinómimos de estas o la traducción desde otro idioma de un texto, por lo que se hace necesario combinar estas técnicas de detección de plagio con otras como, por ejemplo, las basadas en el análisis de n-gramas que definiremos más adelante.

Máxima subsecuencia común

Otro método que vale la pena destacar es el presentado por Victoria Elizalde en su trabajo *Estudio y desarrollo de nuevos algoritmos de detección de plagio* [6]. Este tipo de métodos consiste en encontrar la máxima subsecuencia de palabras en común. Se eligen cadenas largas, ya que a mayor longitud de las cadenas, mayor es la probabilidad de que el fragmento sea una copia y no una coincidencia casual.

Este método tiene la desventaja de no ser tolerante al cambio de orden de las palabras, cosa que es importante a la hora de detectar plagio. En la figura 1.2 podemos ver como bajo el método de máxima subsecuencia común la parte de la frase *la obra de otro* no es considerada, pero si ha sido copiada.

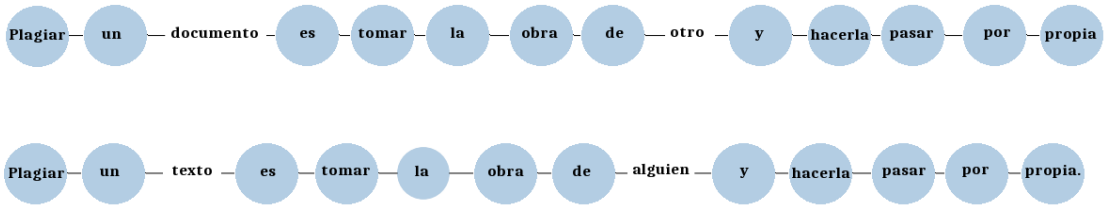


Figura 1.1: Máxima subsecuencia común de las mismas frases [6].

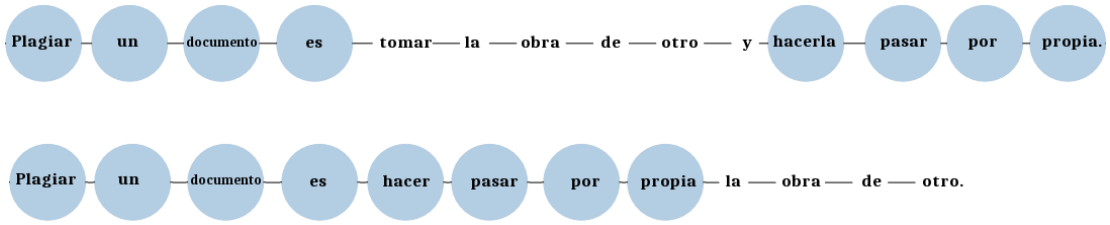


Figura 1.2: Máxima subsecuencia común de las mismas frases, sensible a cambio de orden de palabras [6].

A partir de este método nace el algoritmo Longest Common Subsequence (LCS) que devuelve el total de palabras coincidentes en cada oración de los textos. Este algoritmo es utilizado para la comparación de textos mediante la herramienta Diff de Unix [20], que básicamente comprueba las diferencias entre dos versiones de un mismo archivo, muy común en las herramientas de versionado.

Análisis basado en comparación de n-gramas

Uno de los métodos más trabajados en la literatura actual son los basados en comparación de n-gramas [3]. Autores como Barron Cedeño [1] [2], Stamatatos [18], Chiara Basile [3], Clough entre otros han aportado en el area de la detección de plagio haciendo uso de n-gramas.

La justificación en el uso de n-gramas es que dos textos tienen un nivel muy bajo de n-gramas en común para un $n > 1$. Los n-gramas varían en forma y tamaño según el autor, algunos toman n-gramas de caracteres, otros de palabras y en algunos casos n-gramas formados por el número de caracteres de cada palabra de una oración.

Por otro lado, el procedimiento en los métodos basados en n-gramas es bastante parecido en todos los autores. Primero se toman n-gramas del documento, luego se calcula en algunos casos la frecuencia [3], en otros sólo el conjunto de n-gramas, y finalmente se calcula la distancia con alguna función.

Un punto importante es el valor de n, ya que a medida que aumenta su valor los n-gramas tienen a volverse únicos, por lo que baja el número de coincidencias detectadas. Autores como Cedeño[1] recomiendan el uso de trigramas, ya que con estos se han obtenido los mejores resultados de sus investigaciones.

1.4. Herramientas actuales de detección de plagio en textos digitales

En la actualidad existe una serie de herramientas, principalmente web, que permiten analizar textos y determinar plagio dentro de estos. La gran mayoría de las herramientas

[3] Un n-grama es una subsecuencia de n elementos de una secuencia dada

actuales tiene un enfoque académico y de las características que más se repite es la búsqueda de similitudes y comparación del texto analizado con recursos web como wikipedia, google scholar, entre otros.

A continuación se presenta un análisis a algunas de las herramientas actuales disponibles.

PlagiarismDetect

PlagiarismDetect (http://es.plagiarismdetect.org/) es un servicio web que ofrece detección de plagio en textos escritos en inglés y español. El algoritmo utilizado escanea los textos (con caracteres desde la A a Z y 0 a 9) y busca similitudes en recursos en línea abiertos.

PlagiarismDetect es un sistema de pago, que a la fecha 30-10-2016 contaba con dos tipos de licencia. Una licencia Estándar, de 5 centavos de dolar por cada 275 palabras analizadas, que da acceso a un análisis básico de detección de plagio donde se compara el documento del usuario con recursos web y se entrega un informe final con las principales fuentes de posibles similitudes encontradas. Por otro lado, existe una licencia Premium, de 25 centavos de dolar por cada 275 palabras analizadas, que ofrece además de lo ofrecido en el plan Estándar, un análisis inteligente del texto con un algoritmo de exploración de múltiples capas que permite una detección de plagio más precisa.

¿Qué hace?

- Compara sus documentos con fuentes en línea abiertos (es decir, las páginas web que no están cerrados por indexación) en busca de similitudes.
- Entrega un informe con el porcentaje de similitud total y pone de relieve las partes que son potencialmente plagio.
- Entrega enlaces a las fuentes de donde se encontraron las similitudes.

¿Qué no hace?

- No comprueba textos en idiomas distintos al inglés o español. Si se ingresa un texto en idioma distinto igualmente se procesa y cobra por el procesamiento, pero no se asegura un buen resultado.
- No reconoce caracteres especiales matemáticos o de otro tipo. Los únicos caracteres que el sistema reconoce son las letras “A” a la “Z” y los números “0” a “9”.
- No comprueba la calidad de las citas. Se solicita enviar documentos sin la sección de referencias, ya que esto sólo aumentará el precio que pagas.
- No comprueba gramática, puntuación ni estilos de escritura.

Plagscan

Plagscan (https://www.plagscan.com/) es un servicio web de detección de plagio, permite detección mediante comparación con fuentes web, documentos propios y base de datos interna del sistema (a la cual se puede aportar). Permite el análisis de archivos formato docx y pdf, además de la opción de ingresar texto plano directamente.

En su página web, se definen el servicio como sigue:

“PlagScan es un servicio web basado en navegador, que comprueba la originalidad de sus textos. Usted puede subir documentos en todos los formatos convencionales (MS Word, PDF, etc.) o puede

pegar directamente el texto y realizar su comprobación. Nuestro servicio utiliza un algoritmo innovador específico, basado en los conocimientos más recientes de lenguaje de programación."[12]

Informe de resultados

El informe presentado por Plagscan presenta las siguientes características:

- PlagLevel
  El "PlagLevel" le ofrece en forma de porcentaje una estimación del nivel de plagio, y si es necesario por los encuentros realizados de PlagScan, realizar una revisión más detallada de los documentos subidos. Importante aquí es: en esta estimación todavía no se diferencia entre plagio y cita.
- Lista de coincidencias
  En esta son listadas todas las coincidencias de su documento subido junto a sus respectivas fuentes. Así puede, reconocer de manera más rápida, si se han encontrado coincidencias en una determinada fuente y con ello potenciales plagios.
- Informe en navegador interactivo con contraposición
  El informe en navegador interactivo basado en la lista de coincidencias, muestra los aciertos encontrados en el texto. También ofrece el marcado directo en texto de posibles coincidencias y según su tipo las muestra en uno de tres colores distintos. Con las flechas de su teclado puede saltar de una coincidencia a la siguiente y las fuentes correspondientes son marcadas en negrita encima del texto. A la vez, son marcadas las posibles coincidencias directamente en el texto. Estas dependiendo de su significado son mostradas en distintos colores. En este informe usted obtiene las ventajas que le ofrece la barra de menú azul que encuentra entre texto y fuentes: Con esta barra de menú usted puede quitar lo marcado por PlagScan en el texto, marcar una parte como cita si esta no fue marcada como tal o marcar como plagio una parte del texto identificada por usted como tal.
- Informe formato documento word con comentarios Documento Word, idéntico al subido por el usuario, pero en el que se encuentran marcados los posibles plagios y provisto de comentarios. Estos comentarios no solo marcan los posibles plagios, sino que también muestran las fuentes correspondientes. Este tipo de informe ofrece una solución elegante y confortable, puesto que el formato del documento original se mantiene y recibe un documento idéntico provisto con comentarios.

DOCODE

DOcument **CO**py **DE**tector (*DOCODE*) es una herramienta de detección de plagio desarrollado en la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile, diseñada por ingenieros de la institución y que permite detectar el plagio en los trabajos de los alumnos. Fue el ganador competencia PAN @ CLEF 2011 en la categoría de Detección Intrínseca de Plagio.

DOCODE permite comparar un documento contra millones de archivos alojados en la Web. Además, de forma opcional, también permite comparar el documento contra un repositorio de archivos creados previamente en una institución. De esta forma, *DOCODE* aumenta su rango de búsqueda y ofrece un mayor nivel de eficacia de sus resultados.

Actualmente funciona con un sistema de cuentas ya sea gratuita o de pago según el volumen de texto a analizar. La cuenta gratuita permite el análisis diario de un máximo de 2 documentos doc, docx, pdf, txt o rtf, llegando a una revisión de hasta 5.000 palabras en total.

Capítulo 2

Análisis de Algoritmos de Detección de Plagio

En el marco de este trabajo nos enfocaremos en el análisis intrínseco de plagio. A continuación se analizará algunos de los algoritmos y técnicas propuestas por distintos autores en esta área.

En el área de la detección intrínseca de plagio, la mayoría de los trabajos se basan en la detección o captura del estilo de escritura del autor del documento, aunque no es el único método, pues existen autores que proponen el análisis de la complejidad de los segmentos respecto al resto de los segmentos del documento.

En general, independiente de tipo de análisis, los algoritmos y técnicas presentes en la literatura coinciden en la estructura en que desarrollan este análisis, siendo este divida en tres grandes fases:

La primera fase consiste en la segmentación del texto mediante algún criterio definido para cada uno de los algoritmos. Aquí podemos encontrar autores que realizan una segmentación por capítulos, secciones, párrafos o cantidad de palabra. En esta etapa también es donde se realiza en algunos casos ciertos preprocesamientos al texto, por ejemplo, eliminando *stopwords* o quitando símbolos y caracteres que no pertenezcan al conjunto [a-z].

La segunda fase, consiste en aplicar el algoritmo propiamente tal a los segmentos, buscando determinar el estilo del autor o la complejidad de estos. Aquí podemos encontrar autores como Efstathios Stamatatos que propone en [17] considerar un documento como una bolsa de n-gramas de palabras, creando, para un n definido, un vector de frecuencia de n-gramas normalizado al que llama *Perfil del Texto*, repitiendo la operación para cada uno de los segmentos. Por otro lado, Dario G. Funez y Marcelo L. Errecalde proponen en [8] segmentar el texto para luego a cada uno de los segmentos calcular uno a uno los índices estilométricos presentadas en la sección 1.2, generando una cantidad de vectores igual al número de segmentos, donde cada vector es de dimensión d, dependiendo de la cantidad de índices considerados.

La tercera y última fase consiste en definir un criterio que permita clasificar cada uno de los segmentos definiendo cuales de estos se consideran escritos por el autor del texto y cuales como un posible caso de plagio. Aquí la tendencia es comparar los valores obtenidos en la fase anterior, poniendo especial atención en los los *outlier* que se puedan presentar.

A continuación se analizará en profundidad el algoritmo *DOCODE*, el cuál utiliza el criterio de captura del estilo de escritura del autor del documento, considerando un documento como una secuencia de palabras.

2.1. DOCODE

DOCODE es un algoritmo desarrollado por ingenieros de la Universidad de Chile, ganador del concurso PAN @CLEF del 2011 en la categoría detección intrínseca de plagio.

En la sección 1.4 se mencionó *DOCODE* como una de las herramientas actuales disponibles para la detección de plagio. En esta sección analizaremos el algoritmo propuesto por sus autores en el trabajo titulado "Diseño e implementación de una técnica para la detección intrínseca de plagio en documentos digitales"[9].

El objetivo de *DOCODE* es determinar el estilo de escritura del autor, basado en el uso de las palabras por parte de estos. Esto lo hace desde el principio de que diferentes autores tienden a utilizar diferentes palabras al escribir sus ideas sea o no sobre un mismo tema. De lo anterior surge la siguiente hipótesis:

"si alguna de las palabras usadas en un documentos son específicas de un autor, entonces se puede pensar que estas palabras se concentran en los párrafos o segmentos que dicho autor escribió"

Para lograr el objetivo, el algoritmo primero busca caracterizar el estilo de escritura del autor y luego utilizar alguna técnica para clasificar los distintos pasajes del documento, marcando aquellos que presenten diferencias significativas con el estilo general.

¿Cómo funciona?

DOCODE es un algoritmo que considera las palabras como unidad fundamental, por lo que analiza todas las palabras sin remover las palabras de paro del documento. Debido a esto es que antes de analizar un documento es necesario realizar un preprocesamiento al mismo, quitando todo carácter que no pertenezca al conjunto $[a-z]$.

Con el texto preprocesado, se procede a generar un vector V, vector de frecuencias de palabras no normalizado del documento completo. Luego el documento es divido en segmentos de m palabras formando un conjunto de segmentos C. Finalmente para cada segmento $c \in C$ se genera un vector de frecuencia v_c no normalizado con las palabra del segmento.

Ahora se busca determinar el estilo de escritura dentro del documento. Para esto, por cada segmento $c \in C$, se toma el vector de frecuencia de palabras v_c, para luego, por cada palabra $w \in v_c$ calcular la diferencia respecto a esta misma palabra en el vector de frecuencias del documento completo V, obteniendo el valor d_c que caracteriza la homogeneidad de la escritura del segmento como se expresa a continuación:

$$d_c = \sum_{w \in v_c} \frac{|FREQ(w,V) - FREQ(w,v_c)|}{|FREQ(w,V) + FREQ(w,v_c)|} \tag{2.1}$$

Tomando todos los valores d_c calculados, se determina el estilo de escritura del documento de la siguiente manera:

$$Estilo = \frac{1}{|C|} \sum_{c \in C} d_c \tag{2.2}$$

Finalmente es necesario definir un criterio que permita a partir de los valores d_c y *Estilo* determinar que segmentos del conjunto C son considerados como escritos por el autor y cuáles como un posible caso de plagio. En el caso de este algoritmo, los autores definen para esto un *Umbral* que corresponde a la diferencia entre el *Estilo* y un δ, donde cada segmento con un valor d_c inferior al *Umbral* será considerado como posible segmento plagiado. De manera experimental los autores determinaron que el valor $\delta = 0.075$ para segmentos de largo $m = 400$, ver [9]

Algorithm 1 DOCODE

Require: C, V, m, δ
1: **for** $c \in C$ **do**
2: $d_c = 0$
3: construir v_c usando los términos del segmento c
4: **for** $w \in v_c$ **do**
5: $d_c = d_c + \frac{|FREQ(w,V) - FREQ(w,v_c)|}{|FREQ(w,V) + FREQ(w,v_c)|}$
6: **end for**
7: **end for**
8: $EstiloDocumento = \frac{1}{|C|} \sum_{c \in C} d_c$
9: **for** $c \in C$ **do**
10: **if** $d_c < Estilo - \delta$ **then**
11: Marcar segmento c como posible caso de plagio
12: **end if**
13: **end for**

Análisis del algoritmo

Como se aprecia en la ecuación (2.1) *DOCODE* calcula d_c, como sumatoria de la diferencia de magnitud de la componente w en los vectores V y v_c (vectores de frecuencia de palabras no normalizados del documento completo y segmento c respectivamente), dividido la suma de ambas magnitudes. Analizando los casos extremos se puede apreciar lo siguiente:

- En el caso de las palabras más utilizadas en el documento, el cálculo de la diferencia de magnitudes de las componentes tiende a ser la mayoría de las veces cercana a 1.
- En el caso de las palabras menos utilizadas en el documento, el calculo de la diferencia de magnitudes de las componentes tiende a ser pequeña, siendo cero en los casos en que w aparece solo una vez en el documento, o más de una vez, pero concentrada en el segmento c.

Para obtener una mejor visualización de los casos mencionados se toma como ejemplo el análisis realizado a una carta de opinión enviada a *The New York Times*, publicada el día 15 de Agosto del 2016, titulada *Genetics and Running* [10]. La figura 2.1 permite apreciar la carta, la cual preprocesada, es decir, traspasar todo a minúscula y eliminar todo carácter que no pertenezca al grupo $[a - z]$, tiene una longitud de 122 palabras.

Para el análisis se utilizan como parámetros un largo de segmentos $m = 40$ palabras y un valor $\delta = 0.075$ utilizado para determinar el valor de $Umbral$. Una vez procesada la carta con *DOCODE*, el resultado es un vector de frecuencia de palabras V con 90 componentes y 4 segmentos donde los primeros 3 tienen un largo de 40 palabras y el último solo 2 palabras. La figura 2.2 muestra en azul el estilo de escritura de la carta, en rojo el valor de los d_c respectivos de cada segmento y en naranjo el valor $Umbral$. Las figuras 2.3, 2.4, 2.5 y 2.6 presentan en naranjo la frecuencia de cada palabra $w \in v_c$ de un segmento específico y en azul la frecuencia de la misma palabra w en el vector V.

Para ejemplificar el cálculo de los factores d_c de los segmentos respectivos consideremos el primer segmento de la carta y calculemos el valor de d_1, tal como se expresa en la ecuación 2.1, de esta manera d_1 se obtiene como sigue:

$$d_1 = \sum_{w \in v_1} \frac{|FREQ(w,V) - FREQ(w,v_1)|}{|FREQ(w,V) + FREQ(w,v_1)|}$$

Genetics and Running

AUG. 15, 2016

To the Editor:

Re "The Secret of Jamaica's Runners" (Sunday Review, Aug. 14):

Orlando Patterson may be a bit too hasty in dismissing a genetic explanation for the success of Jamaicans at track. It may be true that we have few outstanding West African sprinters today, but contemporary West Africans are not the relevant population.

Perhaps that subset of West Africans who wound up in Jamaica years ago was a population greatly enriched for genes that contribute to sprinting skills and an inclination toward running. People harboring such genes would delight in exercising their skills and might well establish a culture in which track was the favorite sport and the one to which many young people aspired.

DAVID S. HODES

Dobbs Ferry, N.Y.

Figura 2.1: Genetics and Running - The Opinion Pages - The New York Times

$$
\begin{aligned}
d_1 = {} & \frac{|1-1|}{|2|} + \frac{|3-1|}{|4|} + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} + \frac{|2-1|}{|3|} + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} \\
& + \frac{|2-2|}{|4|} + \frac{|1-1|}{|2|} + \frac{|3-2|}{|5|} + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} \\
& + \frac{|4-2|}{|6|} + \frac{|3-1|}{|4|} + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} + \frac{|4-1|}{|5|} \\
& + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} + \frac{|2-2|}{|4|} + \frac{|2-1|}{|3|} + \frac{|1-1|}{|2|} + \frac{|6-3|}{|9|}
\end{aligned}
$$

$$
\begin{aligned}
d_1 = {} & 0 + \frac{2}{4} + 0 + 0 + 0 + 0 + \frac{1}{3} + 0 + 0 + 0 + 0 + \frac{1}{5} + 0 + 0 + 0 + 0 + 0 + 0 + \frac{2}{6} + \frac{2}{4} \\
& + 0 + 0 + 0 + 0 + 0 + 0 + \frac{3}{5} + 0 + 0 + 0 + 0 + \frac{1}{3} + 0 + \frac{3}{9}
\end{aligned}
$$

$$d_1 = 3.13$$

Finalmente y sólo con el objetivo de presentar los valores graficamente en el intervalo acotado $[0.0 - 1.0]$, normalizamos el valor de d_1, quedando:

$$d_1 = 0.079$$

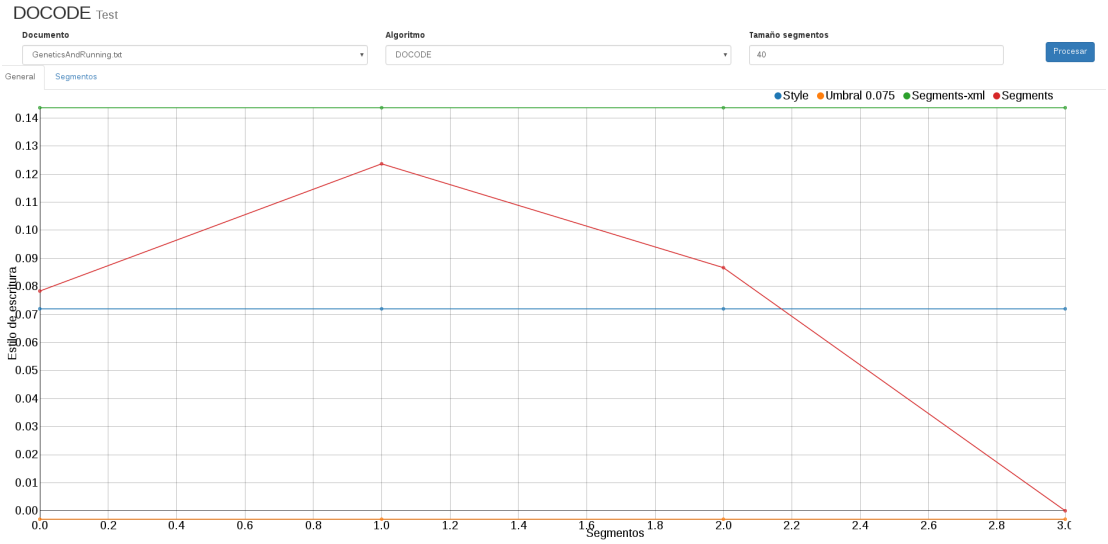


Figura 2.2: DOCODE - Genetics and Running - The Opinion Pages - The New York Times

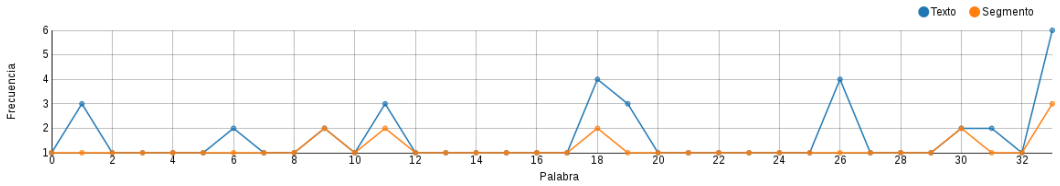


Figura 2.3: DOCODE - Segmento 1 - Genetics and Running - The Opinion Pages - The New York Times

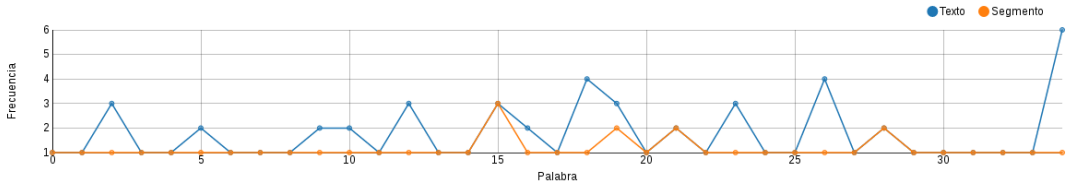


Figura 2.4: DOCODE - Segmento 2 - Genetics and Running - The Opinion Pages - The New York Times

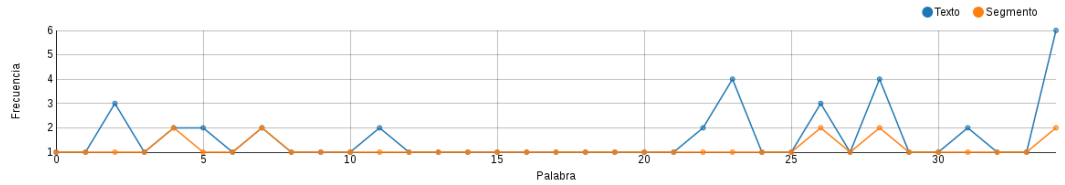


Figura 2.5: DOCODE - Segmento 3 - Genetics and Running - The Opinion Pages- The New York Times

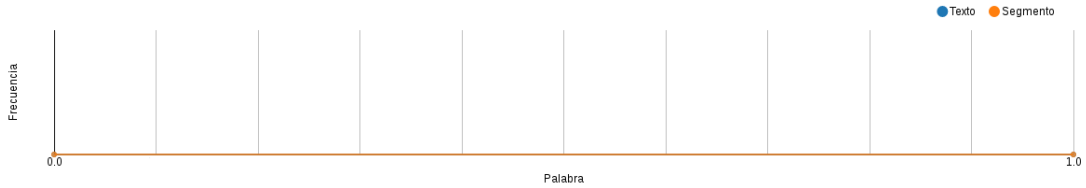


Figura 2.6: DOCODE - Segmento 4 - Genetics and Running - The Opinion Pages- The New York Times

Como se puede apreciar, debido a la longitud del texto, la mayoría de las palabras tiene una frecuencia 1, por lo que al momento de calcular el d_c del segmento respectivo, estas palabras de frecuencia 1 sumarán 0 al total de d_c observándose claramente lo descrito para el caso de las palabras menos utilizadas. El mejor ejemplo de esta situación es el el segmento 4, el cual tiene solo dos palabras y que aparecen solo una vez en el texto, por lo que el cálculo del factor d_4 queda de la forma:

$$d_4 = \frac{|1-1|}{|2|} + \frac{|1-1|}{|2|} \tag{2.3}$$

$$d_4 = 0 \tag{2.4}$$

Por otro lado, si observamos el segmento 3, la última componente del vector v_3 corresponde a la palabra más utilizada por el autor en el texto, con frecuencia 6, mientras que en el segmento solo aparece 1 vez, esto hace que al momento de calcular el valor d_3, en la sumaria la diferencia de esta componente en los vectores V y v_3 tenga un valor $0,71$ siendo la mayor de todas las diferencias y la más cercana a 1, cumpliendo lo descrito anteriormente para el caso de las palabras más utilizadas.

En conclusión, el procesamiento realizado por *DOCODE* consiste en asignar un mayor valor a la aparición de las palabras más utilizadas por el autor y por contraparte penalizar la aparición de las palabras menos utilizadas, entendiéndose que la concentración en un segmento específico de palabras poco utilizadas en documento implicará la sospecha que ese segmento no haya sido escrito por el autor, siendo detectado por el algoritmo debido al bajo valor obtenido en el calculo de d_c para el segmento c.

A continuación se presenta de manera gráfica la aplicación del algoritmo *DOCODE* a tres textos, *Litte Woman* de Louisa May Alcott, *Harry Potter and the Sorcerer's Stone* de J.K. Rowling y una versión de la Biblia que incluye antiguo y nuevo testamento titulada *The King James Bible*. En cada gráfico de lineas se puede apreciar el estilo de escritura del documento en color azul, en rojo la diferencia (d_c) de cada segmento respecto al vector de frecuencia V y finalmente el valor *Umbral* del documento en naranjo.

Al analizar las gráficas se aprecia en la figura 2.8 que la diferencia de los segmentos calculada (d_c) en ningún momento es inferiores al valor *Umbral*, por lo que podemos concluir,

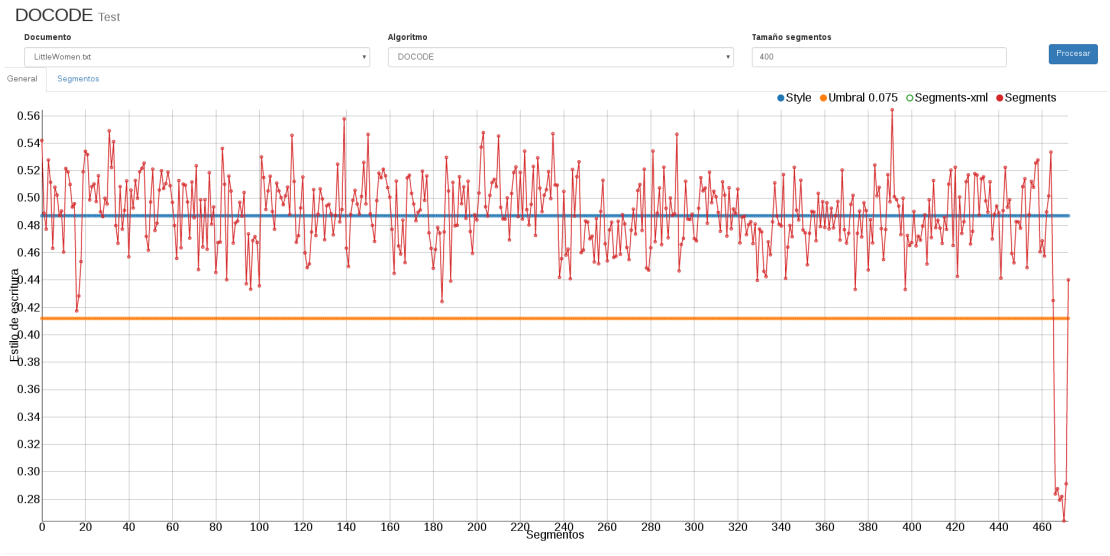


Figura 2.7: Algoritmo DOCODE - Libro: Little Women - Autor: Louisa May Alcott.

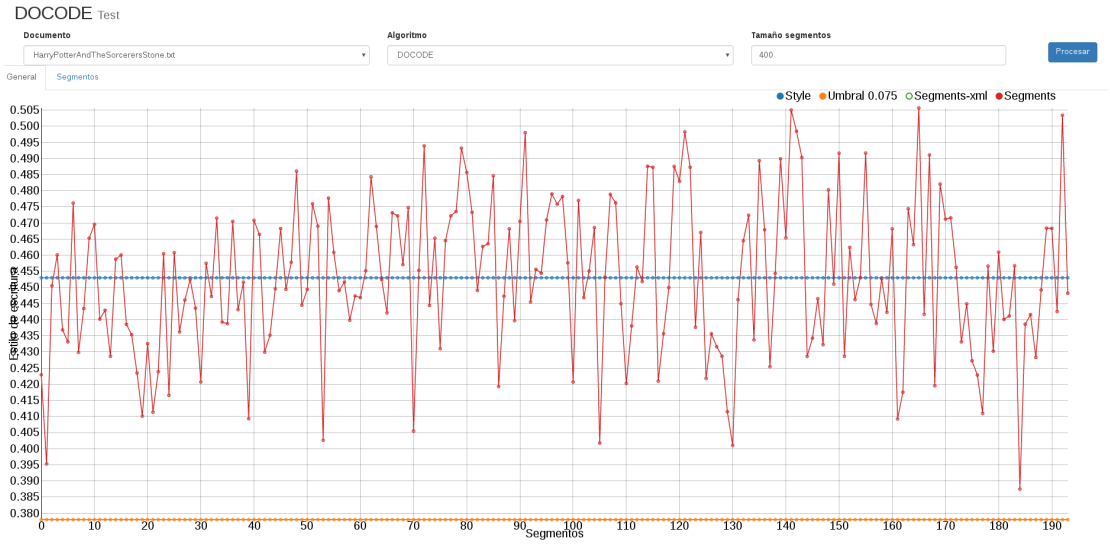


Figura 2.8: Algoritmo DOCODE - Libro: Harry Potter and the Sorcerer's Stone - Autor: J.K. Rowling.

bajo la aplicación del algoritmo, que el texto a sido escrito en su totalidad por un mismo autor. Algo similar sucede en la figura 2.7 aunque en este caso ocurre un cambio drástico de estilo en los segmentos finales del documento, esto se debe a que el documento fue adquirido para su análisis desde el repositorio de libros disponibles de manera gratuita bajo el Proyecto Gutenberg [15], el cual al final de cada libro disponible anexa una licencia y políticas de uso, por lo tanto se entiende que este cambio brusco de estilo se debe a que no fue el autor el que escribió los segmentos finales del documento y por ende se puede verificar la efectividad del algoritmo utilizado para el análisis del mismo. Finalmente la figura 2.9 muestra el análisis a una versión de la Biblia. Es sabido que la Biblia no fue escrita por una sola persona y esto se puede ver reflejado claramente en los resultados obtenidos desde la aplicación de *DOCODE*,

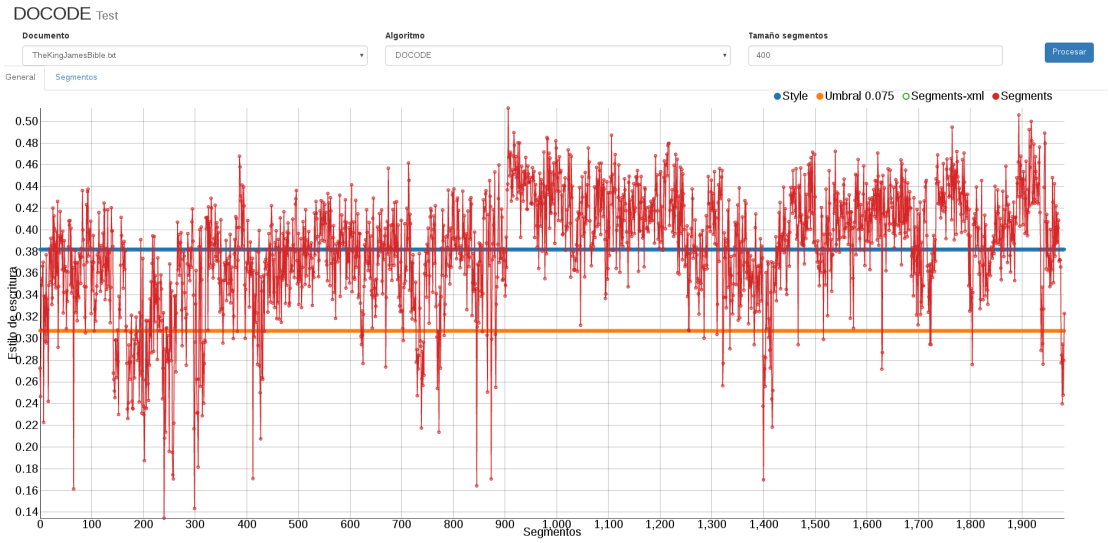


Figura 2.9: Algoritmo DOCODE - Libro: The King James Bible

es más, se puede ver un punto de inflexión en la mitad de la gráfica, que nos indica el cambio de estilo de escritura que ocurre entre el antiguo y nuevo testamento.

Capítulo 3

Propuesta de Mejoras al Algoritmo Analizado

Como se mencionó en la sección anterior, *DOCODE* determina el estilo de escritura a lo largo del documento como el promedio de los d_c obtenidos para cada segmento del documento. Por su parte, cada d_c se calcula como la sumatoria de la diferencia de magnitud de la componente w en los vectores V y v_c dividido la suma de ambas magnitudes, donde v y v_c son los vectores de frecuencia de palabras no normalizados del documento completo y segmento c respectivamente y w es cada componente del vector v_c. Este enfoque es bueno ya que si el autor mantiene un estilo de escritura constante se espera que los d_c calculados tengan valores muy similares entre sí debido a que, al mantener un estilo constante, las palabras más utilizadas por el autor se presentarán casi en la misma proporción a lo largo de los segmento y las diferencias de frecuencia de estas palabras entre los vectores v_c y V tendrá valores similares en todos los segmentos, haciendo por contraparte que la aparición de un pequeño porcentaje de palabara poco utilizada por el autor en un segmento específico no tenga un gran efecto en el valor d_c del segmento en cuestión. Lo anterior se espera para la mayoría de los segmentos, siendo entonces la concentración de un alto porcentaje de palabras poco utilizadas en un segmento en particular el que indicará que este segmento rompe con el estilo general y puede no haber sido escrito por el autor.

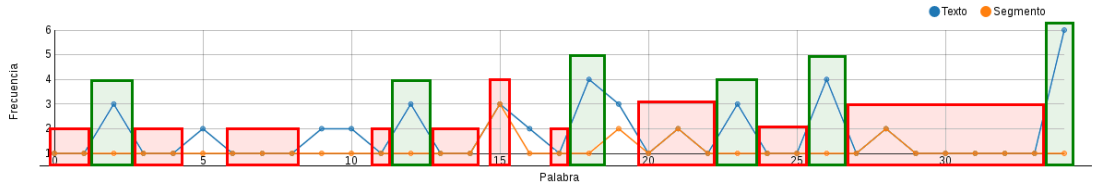


Figura 3.1: DOCODE - Segmento 2 Remarcado - Genetics and Running - The Opinion Pages - The New York Times

El problema de este enfoque se presenta cuando un segmento n se forma a partir de una concentración de palabras que son poco utilizadas en el documento junto a una variedad de las palabras más utilizadas en el mismo, pero con una baja frecuencia de aparición en el segmento, esto provoca en el cálculo de d_n, que la baja frecuencia de aparición de las palabras más utilizadas resulte en diferencias siempre muy cercana a 1 y aumenten rápidamente el bajo valor obtenido del calculo de las diferencias en las palabras menos utilizadas en el documento, llegando finalmente a un valor para d_n que se mantiene dentro del promedio de los d_c de los

demás segmentos del texto y no sea detectado como un posible caso plagio. Un ejemplo claro de esta situación se puede visualizar en el segundo segmento resultante del análisis realizado a la carta enviada a The New York Times (figura 2.4) y que ha sido marcada para su mejor entendimiento en la figura 3.1. Esta figura representa la frecuencia de palabras del segmento número 2 del documento (naranjo) versus la frecuencia de las mismas en el texto completo (azul). Como se puede apreciar, aproximadamente el 50 % de las palabras aparece solo 1 vez en el documento y segmento (puntos encerrados en rojo), pero el otro 50 % de las palabras del segmento (encerradas en verde) son aquellas con mayor frecuencia en el documento y que elevan el resultado obtenido en el calculo de d_2.

Para hacer frente a esta situación se propone trabajar con **vectores normalizados**.

3.1. DOCODE Normalizado

El uso de vectores normalizados, cambia un poco el enfoque que tiene *DOCODE* en su versión original, haciendo que ahora el foco no esté en la cantidad de aparición de una palabra en un segmento determinado, sino que en la distribución de esta palabra en el y los segmentos, es decir, manteniendo la hipótesis que expresan los autores *"si alguna de las palabras usadas en un documentos son específicas de un autor, entonces se puede pensar que estas palabras se concentran en los párrafos o segmentos que dicho autor escribió"*, ahora esperamos que no solo se concentren en dichos segmentos, sino mantengan una proporción en un rango constante.

Efectos del uso de vectores normalizados

Para visualizar los efectos del uso de vectores normalizados consideremos el siguiente texto formado de un vocabulario de dos palabras, el cual someteremos a procesamiento bajo *DOCODE* y *DOCODE Normalizado*:

palabra1 palabra2 palabra1 palabra2 palabra1 palabra2 palabra1 palabra1 palabra2 palabra1 palabra1 palabra2 palabra1 palabra1 palabra2 palabra1 palabra1

Para el procesamiento, se divide el texto anterior en segmentos de 6 palabras obteniéndose:

Segmento1 = *palabra1 palabra2 palabra1 palabra2 palabra1 palabra2*
Segmento2 = *palabra1 palabra1 palabra2 palabra1 palabra1 palabra2*
Segmento3 = *palabra1 palabra1 palabra2 palabra1 palabra1*

Usando *DOCODE*, generamos los vectores V y v_c respectivos:

$$V(palabra1, palabra2) = (11\,,\,6)$$
$$v_{segmento1}(palabra1, palabra2) = (3\,,\,3)$$
$$v_{segmento2}(palabra1, palabra2) = (4\,,\,2)$$
$$v_{segmento3}(palabra1, palabra2) = (4\,,\,1)$$

Como se puede observar, los vectores tienen dos componentes, por lo que es posible llevarlos a un plano cartesiano de ejes $x = FRECUENCIA\ palabra1$, $y = FRECUENCIA\ palabra2$. La figura 3.2 grafica los puntos mencionados anteriormente, junto al valor de la diferencia d_c calculadas mediante la ecuación 2.1. *DOCODE* considera que los segmentos siguen el estilo de escritura del texto siempre que su diferencia respecto al estilo se mantenga dentro de un *Umbral* definido. Es fácil notar que esta diferencia viene dada por la distancia que existe desde el vector que representa al segmento al vector que representa la totalidad del texto. En la figura

podemos ver gráficamente que bajo *DOCODE* el segmento1 es el segmento más cercano al vector V, seguido del segmento2 y finalmente más lejano está el segmento3.

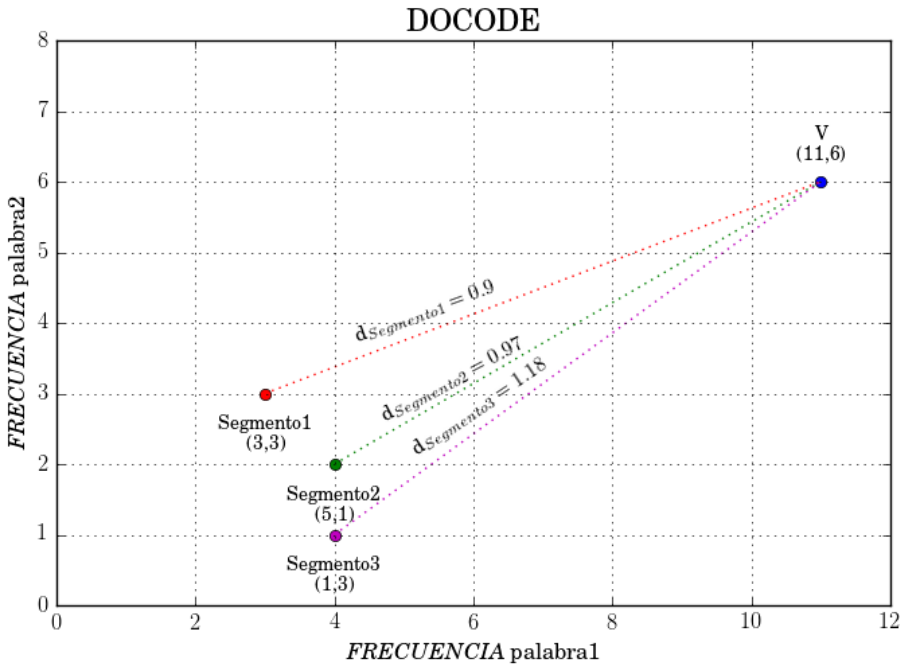


Figura 3.2: DOCODE - Gráfico puntos y diferencias V, v_c

Al utilizar vectores normalizados, los vectores $\hat{V}$ y $\hat{v}_c$ toman los siguientes valores:

$$\hat{V}(palabra1, palabra2) = (0.65\,,\,0.35)$$
$$\hat{v}_{segmento1}(palabra1, palabra2) = (0.5\,,\,0.5)$$
$$\hat{v}_{segmento2}(palabra1, palabra2) = (0.67\,,\,0.33)$$
$$\hat{v}_{segmento3}(palabra1, palabra2) = (0.8\,,\,0.2)$$

Haciendo el mismo ejercicio anterior, en la figura 3.3 se presenta el plano cartesiano esta vez con los puntos que representan los vectores normalizados junto a la diferencia calculada para cada uno de estos. Al comparar las figuras 3.2 y 3.3 se puede observar que al normalizar los vectores, es ahora el *segmento*2 el segmento más cercano al vector que representa el texto completo, esto expresa claramente el efecto de normalizar. El segmento2 es el segmento que tiene la distribución mas parecida a la distribución que tienen las palabras en el texto completo y por ende se entiende que debe ser considerado como uno de los segmentos que mantiene en mayor medida el estilo del documento, algo que no es así para la versión original de *DOCODE*.

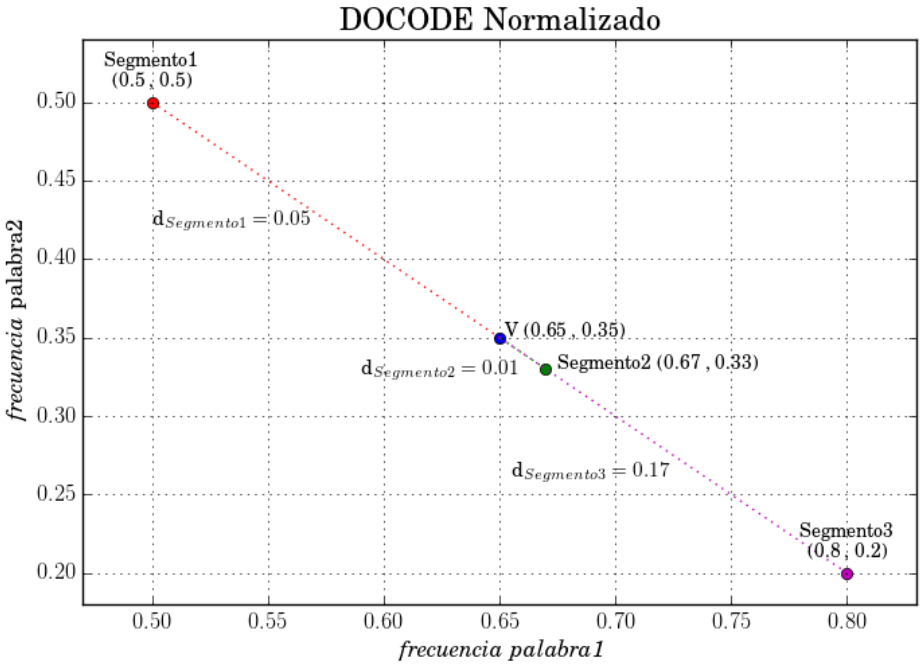


Figura 3.3: DOCODE Normalizado - Gráfico puntos y diferencias V, v_c

De lo anterior, queda claro que normalizar no cambia simplemente las magnitudes de los vectores, sino que es algo mucho más profundo. Analizando la linea 5 de la versión original de *DOCODE*(algortimo 1), el cálculo de los d_c se realiza de la siguiente manera:

$$d_c = d_c + \frac{|FREQ(w,V) - FREQ(w,v_c)|}{|FREQ(w,V) + FREQ(w,v_c)|} \tag{3.1}$$

Donde $FREQ(w,V)$ es la frecuencia de la palabra w en el vector V y $FREQ(w,v_c)$ la frecuencia de la palabra w en v_c. $FREQ(w,V)$ y $FREQ(w,v_c)$ pueden ser expresados en función de las frecuencias normalizadas y las cardinalidades de los vectores V y v_c como sigue:

$$FREQ(w,V) = \#V \cdot freq(w,V) \tag{3.2}$$

$$FREQ(w,v_c) = \#v_c \cdot freq(w,v_c) \tag{3.3}$$

Donde $\#V$ y $\#v_c$ son las cardinalidades de los vectores V y v_c y $freq(w,V)$ y $freq(w,v_c)$ la frecuencia normalizada de la componente w de los vectores vector V y v_c respectivamente.

Reemplazando las equivalencias 3.2 y 3.3 en la ecuación 3.1 se obtiene:

$$d_c = d_c + \frac{|\#V \cdot freq(w,V) - \#v_c \cdot freq(w,v_c)|}{|\#V \cdot freq(w,V) + \#v_c \cdot freq(w,v_c)|}$$

Como $\#V$ es siempre positivo, se puede sacar del valor absoluto:

$$d_c = d_c + \frac{\#V \left| freq(w,V) - \frac{\#v_c}{\#V} \cdot freq(w,v_c) \right|}{\#V \left| freq(w,V) + \frac{\#v_c}{\#V} \cdot freq(w,v_c) \right|}$$

Simplificando términos finalmente queda para el cálculo de d_c la ecuación:

$$d_c = d_c + \frac{\left| freq(w,V) - \frac{\#v_c}{\#V} \cdot freq(w,v_c) \right|}{\left| freq(w,V) + \frac{\#v_c}{\#V} \cdot freq(w,v_c) \right|}$$

Como se puede observar, *DOCODE*, al momento de calcular las diferencias d_c de los segmentos multiplica la frecuencia normalizada del vector v_c por la constante $\frac{\#v_c}{\#V}$ la cual puede tomar valores en el rango $[0,1]$. Considerando que para un correcto funcionamiento de *DOCODE* es ideal dividir el documento en varios segmentos, se puede deducir a simple vista que si $\#v_c \ll \#V$ la constante $\frac{\#v_c}{\#V}$ será muy pequeña. Lo anterior muestra que *DOCODE* en cierta medida no considera el real aporte de la frecuencia de las palabras en los segmentos en que se divide el documento.

Al trabajar con vectores normalizados, lo que se hace es quitar este factor que disminuye o quita peso a la frecuencia de las palabras en los segmentos, considerando de esta manera la siguiente ecuación en el cálculo de las diferencias d_c de los respectivos segmentos:

$$d_c = d_c + \frac{|freq(w,V) - freq(w,v_c)|}{|freq(w,V) + fre(w,v_c)|} \tag{3.4}$$

El algoritmo 2 presenta en color rojo las modificaciones realizadas al algoritmo 1.Los cambios realizados respecto a la versión original son la normalización de los vectores V y v_c que se utilizan en el cálculo del estilo del documento, la ecuación para el cálculo de los factores d_c y además un cambio más radical correspondiente a la linea 11, el cual indica que desde ahora un segmento será considerado como posible plagio cuando el valor de d_c sobrepase el valor *Umbral* por arriba. Para entender este cambio debemos analizar como se comporta el algoritmo en los casos extremos:

- En el caso de las palabras más utilizadas, se espera que el rango de aparición de una palabra en los segmentos sea cercano al que representa en el documento completo, es decir, si la frecuencia de una palabra representa el 30 % de del total del documento, se espera que en cada segmento del documento tenga un porcentaje similar de aparición. Con esto, las diferencias calculadas entre las componentes de los vectores para las palabras más utilizadas por el autor tenderán a 0, por contraparte,
- En el caso de las palabra menos utilizadas por el autor, al calcular las diferencias de magnitud de las componentes referidas a estas palabras, los valores tenderán a ser muy elevados, cercanos a 1, esto se debe a que se compara el pequeño porcentaje que representa la palabra en el total de palabras que tiene un documento completo versus el porcentaje mucho mayor que puede representar la misma en un segmento con un número acotado de palabras.

Con esto, se puede apreciar que al existir una concentración de palabras poco utilizadas por el autor en un segmento del documento, el valor de d_c calculado para dicho segmento tenderá a ser mayor que el de los demás segmentos, concluyéndose que bajo este tipo de análisis, **si un**

Algorithm 2 DOCODE Normalizado

Require: C, V, m, δ

Normalizar V
for $c \in C$ **do**
  $d_c = 0$
  construir v_c **normalizado** usando los términos del segmento c
  for $w \in v_c$ **do**
    $d_c = d_c + \frac{|freq(w,V) - freq(w,v_c)|}{|freq(w,V) + fre(w,v_c)|}$
  end for
end for
$EstiloDocumento = \frac{1}{|C|} \sum_{c \in C} d_c$
for $c \in C$ **do**
  if $d_c > Estilo + \delta$ **then**
    Marcar segmento c como posible caso de plagio
  end if
end for

segmento tiene un valor d_c superior a un *Umbral* definido entonces puede ser considerado un posible segmento no escrito por el autor del documento.

En las figuras 3.4, 3.5 y 3.6 se presenta un análisis gráfico de los resultados obtenidos de la aplicación de *DOCODE Normalizado* a los libros de ejemplo usados en la sección 2.1, *Litte Woman* de Louisa May Alcott, *Harry Potter and the Sorcerer's Stone* de J.K. Rowling y *The King James Bible*. En cada gráfico de lineas se puede apreciar el estilo de escritura del documento en color azul, en rojo la diferencia (d_c) de cada segmento respecto al vector de frecuencia normalizado V y finalmente el valor *Umbral* del documento en naranjo.

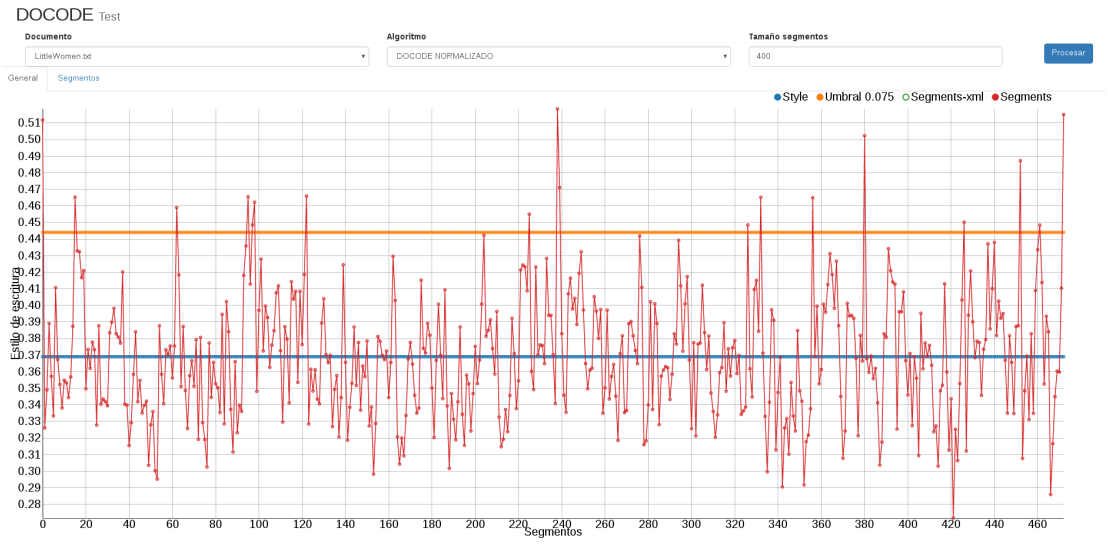


Figura 3.4: Algoritmo DOCODE Normalizado - Libro: Little Women - Autor: Louisa May Alcott.

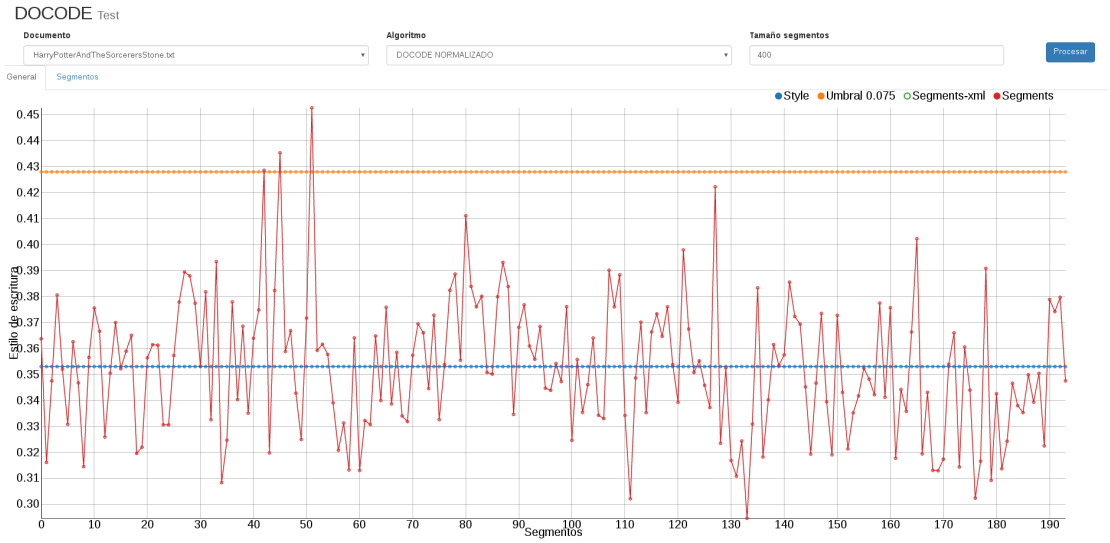


Figura 3.5: Algoritmo DOCODE Normalizado - Libro: Harry Potter and the Sorcerer's Stone - Autor: J.K. Rowling.

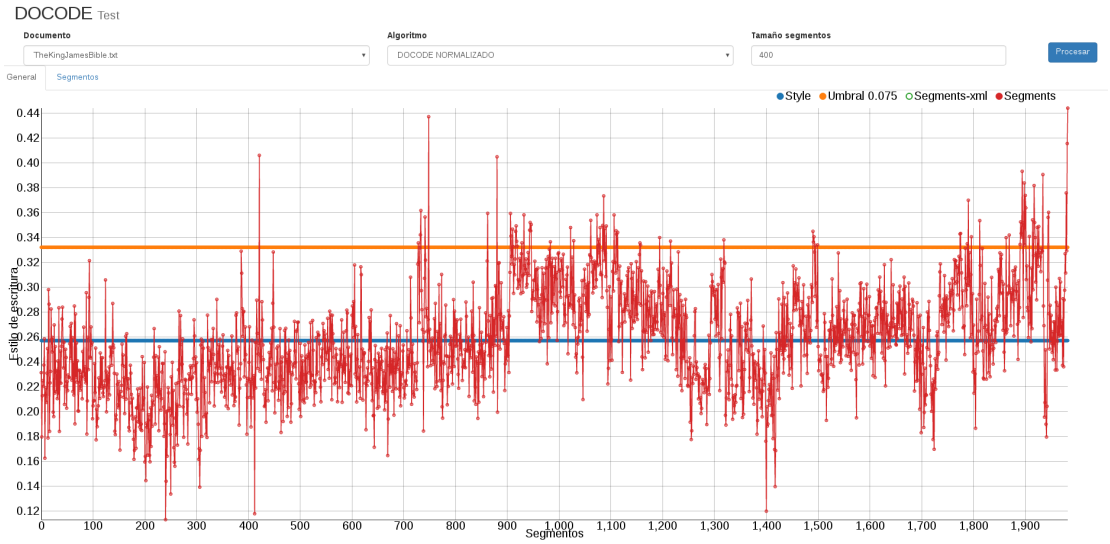


Figura 3.6: Algoritmo DOCODE Normalizado- Libro: The King James Bible

3.2. DOCODE Normalizado por Segmento

En la sección 3.1 se propone normalizar con el objetivo de analizar la distribución de las palabras a lo largo de los segmentos y si esta guarda relación con la distribución de las mismas en el documento completo. Esta propuesta, quita el factor $\frac{\#v_c}{\#V}$ que reduce el peso de la frecuencia de palabras en el segmento y calcula las diferencias a partir de frecuencias normalizadas. El problema de esta propuesta se presenta en el cálculo de diferencias de frecuencias normalizadas de palabras que aparecen poco en el documento. Una palabra con baja frecuencia en el documento al ser normalizada respecto a la cantidad total de palabras

tendrá un valor muy pequeño, en cambio, el valor normalizado de la frecuencia de la palabra en un segmento será mucho mayor debido a la cantidad menor de palabras que tiene un segmento. Para ejemplificar está situación, suponga un documento de 2000 palabras, donde la *palabraX* aparece solo una vez en todo el documento. Al normalizar, la componente *palabraX* del vector V tendrá un valor $freq(palabraX, V) = 0.0005$. Ahora, suponga el documento fue divido en segmentos de $m = 400$ palabras, la frecuencia normalizada de la *palabraX* en el vector v_c del segmento en que aparece tendrá un valor $freq(palabraX, v_c) = 0.0025$. El efecto de esta situación se ve reflejado en el cálculo del valor d_c del segmento respectivo, el cual aumentará mucho su valor si cuenta con palabras de muy baja frecuencia dentro del documento.

Para hacer frente a este problema, se propone trabajar con vectores normalizados, pero construyendo un nuevo vector normalizado $V|_{v_c}$ a partir del vector V restringido a las componentes del vector v_c respectivo, es decir, para cada segmento se toma desde el vector no normalizado V sólo las palabras contenidas en el vector v_c, esto hace que se realice una comparación de la distribución del las palabras sin considerar el ruido que producen las palabras que no están contenidas en v_c pero sí en V y que afectan los valores normalizados que de cada palabra contenida en V. El cambio mencionado, nuevamente se ve reflejado en el cálculo de los factores d_c en el algoritmo, obteniéndose para este cálculo la siguiente ecuación:

$$d_c = d_c + \frac{|freq(w, V|_{v_c}) - freq(w, v_c)|}{|freq(w, V|_{v_c}) + fre(w, v_c)|} \tag{3.5}$$

Para entender de mejor manera el efecto de normalizar acotado a las componentes del segmento consideremos el siguiente texto, y segmentos de 6 palabras:

palabra1 palabra2 palabra1 palabra2 palabra1 palabra2 palabra1 palabra1 palabra2 palabra1 palabra1 palabra2 palabra1 palabra1 palabra2 palabra1 palabra1 palabra3

Al dividir el texto en segmentos se obtiene:

Segmento1 = *palabra1 palabra2 palabra1 palabra2 palabra1 palabra2*
Segmento2 = *palabra1 palabra1 palabra2 palabra1 palabra1 palabra2*
Segmento3 = *palabra1 palabra1 palabra2 palabra1 palabra1 palabra3*

Bajo *DOCODE*, los vectores V y v_c resultantes son:

$$V(palabra1, palabra2, palabra3) = (11\,,\,6\,,\,1)$$
$$v_{segmento1}(palabra1, palabra2, palabra3) = (3\,,\,3\,,\,0)$$
$$v_{segmento2}(palabra1, palabra2, palabra3) = (4\,,\,2\,,\,0)$$
$$v_{segmento3}(palabra1, palabra2, palabra3) = (4\,,\,1\,,\,1)$$

Si se utiliza *DOCODE Normalizado* quedan de la forma:

$$V(palabra1, palabra2, palabra3) = (0.62\,,\,0.33\,,\,0.05)$$
$$v_{segmento1}(palabra1, palabra2) = (0.5\,,\,0.5)$$
$$v_{segmento2}(palabra1, palabra2) = (0.67\,,\,0.33)$$
$$v_{segmento3}(palabra1, palabra2, palabra3) = (0.66\,,\,0.16\,,\,0.16)$$

En cambio, al usar esta nueva mejora, *DOCODE Normalizado por Segmento*, el vector V se acota a las componentes del segmento con el que será comparado, por lo tanto para cada segmento existirá un $V|_{v_c}$ distinto tal como se expresa a continuación:

$$V(palabra1, palabra2, palabra3) = (0.62\,,\,0.33\,,\,0.05)$$

$$V|_{v_{segmento1}}(palabra1, palabra2) = (0.65\,,\,0.35)$$
$$v_{segmento1}(palabra1, palabra2) = (0.5\,,\,0.5)$$

$$V|_{v_{segmento2}}(palabra1, palabra2) = (0.65\,,\,0.35)$$
$$v_{segmento2}(palabra1, palabra2) = (0.67\,,\,0.33)$$

$$V|_{v_{segmento3}}(palabra1, palabra2, palabra3) = (0.62\,,\,0.33\,,\,0.05)$$
$$v_{segmento3}(palabra1, palabra2, palabra3) = (0.66\,,\,0.16\,,\,0.16)$$

De esta manera, al momento de realizar el cálculo de los factores d_c para los segmentos 1 y 2, el ruido que producía la palabra 3 ha sido eliminado.

A continuación se presenta una nueva versión del algoritmo, remarcando en rojo los cambios respecto a las versiones anteriores.

Algorithm 3 DOCODE Normalizado por Segmento

Require: C, V, m, δ
1: **for** $c \in C$ **do**
2: $d_c = 0$
3: construir v_c **normalizado** usando los términos del segmento c
4: **contruir** $V|_{v_c}$ **tomando desde** V **solo las palabras contenidas en** v_c
5: **Normalizar** V_{auxc}
6: **for** $w \in v_c$ **do**
7: $d_c = d_c + \frac{|freq(w,V|_{v_c}) - freq(w,v_c)|}{|freq(w,V|_{v_c}) + fre(w,v_c)|}$
8: **end for**
9: **end for**
10: $EstiloDocumento = \frac{1}{|C|}\sum_{c \in C} d_c$
11: **for** $c \in C$ **do**
12: **if** $d_c > Estilo + \delta$ **then**
13: Marcar segmento c como posible caso de plagio
14: **end if**
15: **end for**

La razón de esta mejora, nace de la necesidad de eliminar el ruido provocado por las palabras que no están contenidas en los vectores v_c a la hora de comparar y realizar el calculo de las diferencias d_c de cada segmento respecto al vector de frecuencias V del documento completo. Al igual que en la propuesta anterior, aquí un segmento es considerado posible plagio cuando su valor calculado de d_c sobrepasa por arriba el *Umbral* definido. Las figuras 3.7, 3.8 y 3.9 reflejan de manera gráfica el funcionamiento del algoritmo *DOCODE Normalizado por Segmentos* en los tres libros utilizados para mostrar el funcionamiento de los algoritmos anteriores.

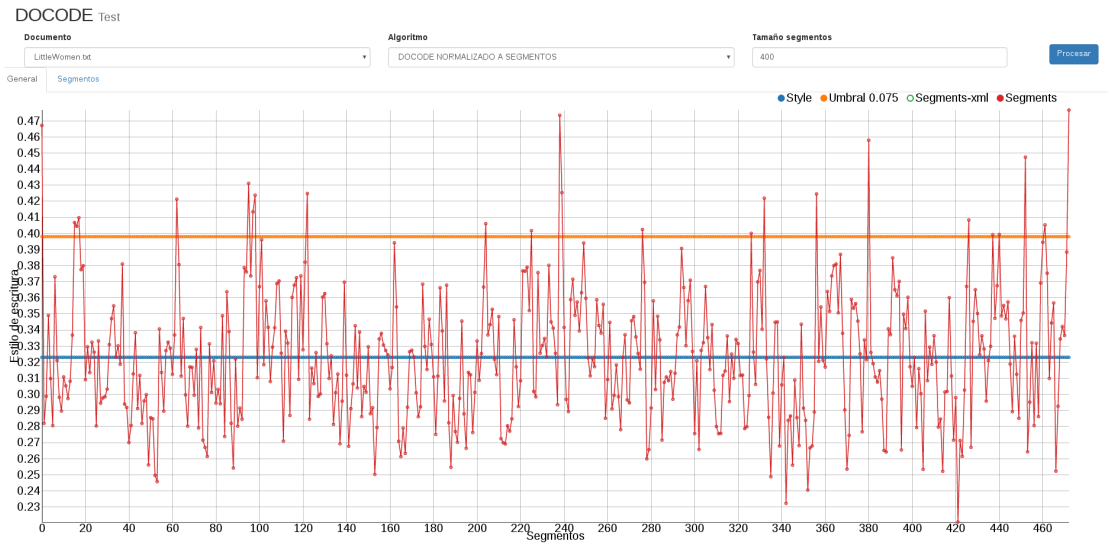


Figura 3.7: Algoritmo DOCODE Normalizado por Segmento - Libro: Little Women - Autor: Louisa May Alcott.

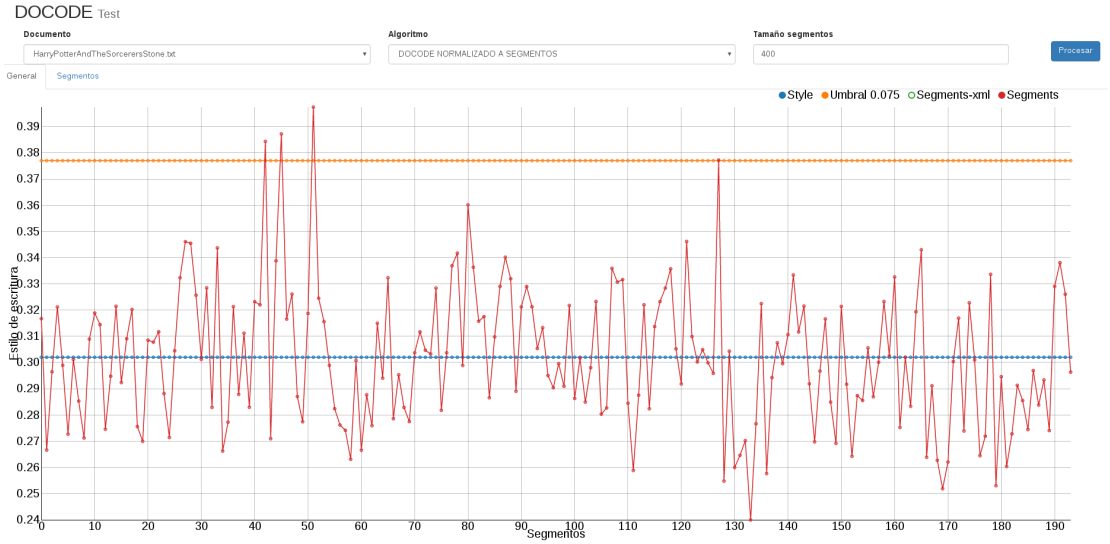


Figura 3.8: Algoritmo DOCODE Normalizado por Segmento - Libro: Harry Potter and the Sorcerer's Stone - Autor: J.K. Rowling.

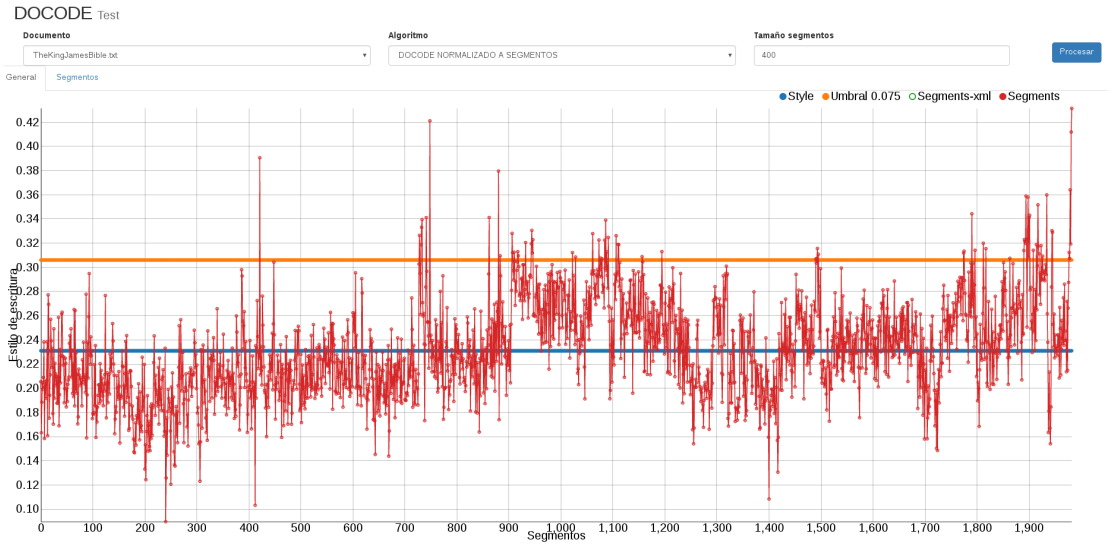


Figura 3.9: Algoritmo DOCODE Normalizado por Segmento - Libro: The King James Bible

Capítulo 4

Entorno de Desarrollo, Scripts y Herramienta de Visualización

En el transcurso de este trabajo se desarrollaron una serie de scripts que permitieron llevar a cabo los experimentos descritos más adelante en la sección 5.3 además de una herramientas de visualización del funcionamiento de los algoritmos *DOCODE*, *DOCODE Normalizado* y *DOCODE Normalizado por Segmento*.

A continuación se presentan detalles del entorno de desarrollo, librerías y recursos utilizado en el desarrollo y detalles de los scripts y herramienta de visualización desarrollada.

4.1. Entorno de desarrollo

El entorno de desarrollo y herramientas utilizadas durante el desarrollo de esta memoria se describe a continuación:

Características de Hardware

Equipo	Procesador	Intel core i7 M620
	Memoria RAM	8 Gb

Características de Software

Sistema Operativo	Fedora Linux 24
Lenguajes de Programación	Python 2.7
	Javascript
Frameworks	Micro-Framework Python Flask
Librerías Importantes	JQuery versión 2.2.
	D3.js versión 2 y 3

4.2. Scripts

Para llevar a cabo los experimentos se desarrollo una serie de scripts en Python, agrupados en el directorio DOCODEX3 disponible en los recursos anexos de este trabajo y en el repositorio github `https://github.com/plreyes/Memoria-DOCODEx3`. A continuación se presenta la estructura del directorio y una breve explicación de cada uno de los scripts desarrollados.

- **Source/**
  Directorio que contiene las carpetas y archivos del grupo *suspicious-documents* del corpus PAN 2010

- **Annotations/**
  Directorio donde se almacena una copia de los archivos XML originales que indican los segmentos plagiados de los documentos del grupo *suspicious-documents* del corpus PAN 2010, agrupado en carpetas *part1, part2, ..., part32* tal como se presenta en el corpus y además agrupados por nivel de plagio en las carpetas *poco, medio, mucho*.

- **Annotations2/**
  Directorio donde se almacena los archivos XML modificados que indican los segmentos plagiados de los documentos del grupo *suspicious-documents* del corpus PAN 2010, agrupado en carpetas *part1/, part2/, ..., part32/* tal como se presenta en el corpus y además agrupados por nivel de plagio en las carpetas *poco/, medio/, mucho/*.

- **Detections/**
  Directorio donde se almacena los archivos XML con el detalle de las detecciones de plagio realizadas.

- **Results/**
  Directorio que contiene los archivos TXT con los resultados de los experimentos realizados.

- **createDirectories.py**
  Script que crea todos los directorios y subdirectorios necesarios y requeridos por los otros scripts desarrollados. Se debe especificar la ruta raíz del directorio donde se encuentra **DOCODEX3**.

- **generateAnnotations2.py**
  Script que crea archivos XML modificados a partir de los archivos XML originales del corpus PAN 2010, pero indicando segmentos plagiados según tamaño de segmentos definido por el usuario.

- **clasificador.py**
  Script que clasifica los archivos XML originales y modificados del grupo *suspicious-documents* del corpus PAN 2010 según nivel de plagio.

- **docodex3Processor.py**
  Script que procesa los documentos de texto de la sub-carpeta del directorio **Sources/** indicada por el usuario utilizando los algoritmos *DOCODE*, *DOCODE Normalizado* y *DOCODE Normalizado por Segmento* para distintos valores de lambda en el rango $[0.1 - 3.0]$, generando XML de detecciones y guardando dichos XML en el directorio **Detections/**.

- **searchLambdaF1.py**
  Script que determina el valor de $\lambda_{F_1\,optimo}$ que obtiene mejor F_1 a partir de las detecciones registradas en los archivos XML del directorio **Detections** y los XML originales o modificados que indican donde existía el plagio realmente (**Annotations/** o **Annotations2/** según indique el usuario). Entrega las métricas *Puntaje Total*, *Recall*, *Precision*, F_1 y $\lambda_{F_1\,optimo}$ para los algoritmos *DOCODE*, *DOCODE Normalizado* y *DOCODE Normalizado por Segmento*.

- **searchLambdaPS.py**
  Script que determina el valor de $\lambda_{PT\,optimo}$ que obtiene mejor *Puntaje Total* a partir de las detecciones registradas en los archivos XML del directorio **Detections/** y los XML originales o modificados que indican donde existía el plagio realmente (**Annotations/** o **Annotations2/** según indique el usuario). Entrega las métricas *Puntaje Total*, *Recall*, *Precision*, F_1 y $\lambda_{PT\,optimo}$ para los algoritmos *DOCODE*, *DOCODE Normalizado* y *DOCODE Normalizado por Segmento*.

- **resultsBestLambdaF1.py**
  Script que entrega las métricas *Puntaje Total*, *Recall*, *Precision*, F_1 y $\lambda_{F_1\,optimo}$ para los algoritmos *DOCODE*, *DOCODE Normalizado* y *DOCODE Normalizado por Segmento* utilizando los valores de $\lambda_{F_1\,optimo}$ indicados por el usuario.

- **resultsBestLambdaPS.py**
  Script que entrega las métricas *Puntaje Total*, *Recall*, *Precision*, F_1 y $\lambda_{PT\,optimo}$ para los algoritmos *DOCODE*, *DOCODE Normalizado* y *DOCODE Normalizado por Segmento* utilizando los valores de $\lambda_{PT\,optimo}$ indicados por el usuario.

- **textClass.py**
  Clase que realiza el preprocesamiento al texto a analizar y contiene los métodos *docode*, *docode_normalizado* y *docode_normalizado_segmento*.

Todos estos scripts fueron desarrollados con el fin de realizar los experimentos descritos en la sección 5.3. Es posible replicar los experimentos simplemente ejecutando cada uno de los scripts descritos en el orden en que aparecen en el listado.

4.3. Herramienta de Visualización

Con el objetivo de visualizar gráficamente el procesamiento realizado a un documento por los algoritmos *DOCODE*, *DOCODE Normalizado* y *DOCODE Normalizado por Segmento*, se desarrollo una herramienta web utilizando Flask, un micro-framework web desarrollado en python.

Esta herramienta permite al usuario seleccionar un documento y procesarlo utilizando alguno de los tres algoritmos desarrollados. Una vez procesado el documento, la herramienta entrega al usuario información básica como la cantidad de palabras, cantidad de segmentos en los que se dividió el texto y el valor calculado del estilo de escritura del documento. Junto a esto también entrega al usuario un gráfico de lineas múltiple con cuatro lineas de diferentes colores que representan:

- **Linea Azul**, valor calculado del estilo de escritura del documento.

- **Linea Naranja**, valor *Umbral* que define el punto de corte del valor de estilo para considerar un segmento como plagio o no plagio. Bajo *DOCODE* los segmentos con valor de estilo inferior al valor *Umbral* son considerados segmentos plagiados, en cambio utilizando *DOCODE Normalizado* y *DOCODE Normalizado por Segmento* los segmentos con valor de estilo superior al *Umbral* son considerados segmentos plagiados.

- **Linea Roja**, valor del estilo de escritura de cada segmento en que se dividió el documento.

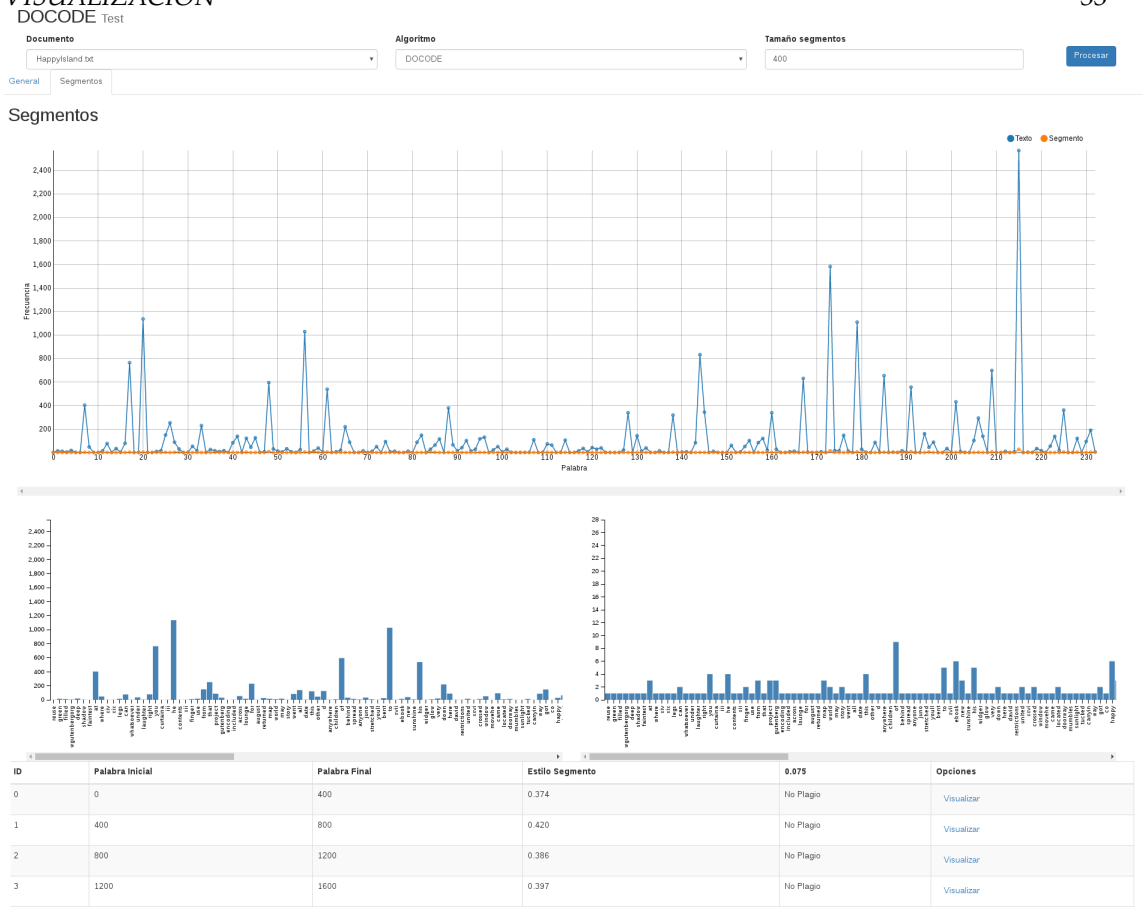
- **Linea Verde**, indica los segmentos que realmente contienen plagio tomando valor 0 o 1 según sea el caso para cada algoritmo. Utilizando *DOCODE* los segmentos con valor 1 son segmentos no plagiados y los segmentos con valor 0 son segmentos plagiados. Bajo

procesamiento utilizando *DOCODE Normalizado* y *DOCODE Normalizado por Segmento* la situación es al revéz, es decir, 0 indica segmento sin plagio y 1 indica segmento plagiado.

La figura 4.1 presenta un ejemplo de las visualizaciones generadas por la herramienta desarrollada.

Figura 4.1: Herramienta Web de visualización de algoritmos - Visualización de estilos

Finalmente, la herramienta entrega al usuario un desglose de cada uno de los segmentos en que se dividió el texto, permitiendo visualizar y comparar a partir de un gráfico de lineas y gráfico de barras la frecuencia de las palabras en el documento completo y cada uno de los segmentos. Figura 4.2.



ID	Palabra Inicial	Palabra Final	Estilo Segmento	0.075	Opciones
0	0	400	0.374	No Plagio	Visualizar
1	400	800	0.420	No Plagio	Visualizar
2	800	1200	0.386	No Plagio	Visualizar
3	1200	1600	0.397	No Plagio	Visualizar

Figura 4.2: Herramienta Web de visualización de algoritmos - Visualización de frecuencias

Capítulo 5

Experimentos y Análisis de Resultados

PAN, como indica en su web [11], es una serie de eventos científicos y tareas compartidas en medicina forense de texto digitales. A lo largo de los años PAN se ha integrado a la conferencia CLEF y es en esta donde año a año desde el 2009 se presentan los resultados de los desafíos propuestos en temas sobre descubrimiento de plagio, autoría, y mal uso de software social. En el marco del desarrollo de las tareas y desafíos propuesto por este evento, PAN @CLEF pone a disposición de los participantes una serie recursos para probar y entrenar los algoritmos en desarrollo. El año 2011 el evento se centró en dos grandes desafíos, la detección de autoría y la detección de plagio bajo la modalidad de *detección intrínseca de plagio* y *detección de plagio con referencia*. En los experimentos realizados se utilizó el corpus **PAN Plagiarism Corpus 2010**[13], el cual fue puesto a disposición en el evento PAN @CLEF del año 2011, competencia y corpus con los que ganó ese año en la categoría de *detección intrínseca de plagio* el algoritmo *DOCODE* analizado en el presente documento. El uso de este set de datos nos permitirá contar con un conjunto de textos de prueba a los que DOCODE ya fue sometido y permitirá reflejar de mejor manera el resultado de las mejoras propuestas al algoritmo en comparación a la versión original.

5.1. PAN Plagiarism Corpus 2010

PAN Plagiarism Corpus 2010[13] es un corpus basado en 22000 libros en inglés, 520 libros en Alemán y 210 libros en español obtenidos desde el Proyecto Gutenberg[15].

El corpus viene formado de dos grandes grupos de documentos, *source-documents* compuesto de 11148 textos originales y *suspicious-documents* el cual está compuesto por 15925 textos formados a partir del primer grupo, a los cuales se les ha insertado pasajes de otros documentos de manera artificial creando así documentos contenedores de plagio.

Cada documento del corpus viene acompañado de un archivo en formato xml del mismo nombre, que contiene información como el nombre del libro y autor en el caso de los textos originales, y la información de los pasajes que han sido plagiados junto a la fuente desde la cual han sido extraídos para los documentos con plagio artificial.

Para los experimentos realizados se hizo uso sólo del grupo *suspicious-documents*, ya que debido a que los algoritmos trabajados apuntan a la detección intrínseca de plagio, no es necesario contar con las fuentes desde donde fueron extraídos los pasajes plagiados, sino solo es necesario analizar el documento he intentar determinar que pasajes no fueron escritos por el autor.

5.2. Criterios de evaluación

A continuación se describe una serie de métricas que serán utilizadas para medir la efectividad de *DOCODE* y las mejoras propuestas en la sección 3

Matriz de Confusión

Una matriz de confusión es una herramienta que permite la visualización del desempeño de un algoritmo. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real.

	Predicción Negativa	Predicción Positiva
Real Negativo	TN	FP
Real Positivo	FN	TP

Tabla 5.1: Matriz de confusión

A partir de la matriz de confusión existe una serie de métricas que permiten analizar el desempeño del algoritmo, algunas de estas utilizadas en los experimentos son:

- Precision, indica el grado o frecuencia en el que un par de pasajes identificados como un caso de plagio es correcto. En otras palabras, responde a la pregunta: cuando el algoritmo predice positivo, ¿con qué frecuencia es correcto?

$$precision = \frac{TP}{TP + FP}$$

- Recall, establece el porcentaje de pasajes plagiados que el algoritmo clasifica correctamente

$$recall = \frac{TP}{TP + FN}$$

- F_1, media armónica entre *precision* y *recall*

$$F_1 = 2\,\frac{precision\,recall}{precision + recall}$$

Granularidad

La granularidad es una métrica introducida por Potthast et al.[14] para evaluar la usabilidad de un algoritmo. Bajo esta métrica, cada caso real de plagio debe ser informado sólo una vez, por lo que si el reporte de este caso se produce de manera fragmentada la granularidad aumenta. Cabe destacar que el valor deseable de granularidad es 1.

$$Granularidad = \frac{1}{|S_R|} \sum_{s \in S_R} |C_s|$$

S_R corresponden a los casos detectados correctamente, y $|CS|$ corresponde al número de detecciones para el caso del s.

Para entender mejor esta métrica, considere un documento de 1000 palabras que contiene plagio en un segmento que va desde la palabra 100 a la 200, es decir existe sólo un caso de plagio en el documento. Ahora suponga que se procesa dicho documento utilizando algún

algoritmo de detección de plagio como *DOCODE* o las mejoras que se proponen en este trabajo y como resultado de este procesamiento el algoritmo indica justamente que existe un posible caso de plagio desde la palabra 100 a la 200. En la situación anterior el valor de granularidad será 1, ya que existe sólo un caso de plagio ($|S_R| = 1$) y fue detectado en un sólo bloque ($C_s = 1$), por lo tanto.

$$Granularidad = \frac{1}{|S_R|} \sum_{s \in S_R} |C_s| = \frac{1}{|1|} \sum_{s \in S_R} |1| = 1$$

Ahora, bajo la misma situación anterior, pero esta vez considerando que el algoritmo elegido para buscar plagio nos dice que existen dos segmentos que pueden ser plagio, desde la palabra 100 a la 150 y desde la 151 hasta la 200, es decir, detectó correctamente el segmento plagiado, pero lo hizo en dos instancias o bloques. Este pequeño cambio hace que la granularidad aumente su valor a dos, pues aún existe sólo un caso de plagio, pero fue detectado en 2 bloques ($C_s = 2$), por lo tanto:

$$Granularidad = \frac{1}{|S_R|} \sum_{s \in S_R} |C_s| = \frac{1}{|1|} \sum_{s \in S_R} |2| = 2$$

Así, al aumentar los casos de plagio en el documento, la granularidad se encargará de decir que tan eficiente es el algoritmo en detectar dichos casos de la manera menos fragmentada posible.

Puntaje Total

A partir de las métricas descritas en los apartados anteriores 5.2 y 5.2, finalmente el *Puntaje Total* será calculado bajo la siguiente ecuación

$$Puntaje\,Total = \frac{F_1}{\log_2(1 + Granularidad)}$$

5.3. Experimentos

Antes de analizar cuál de los tres algoritmos obtiene los mejores resultados es necesario definir el valor *Umbral* que determinará que segmentos de los documentos serán considerados como posibles casos de plagio. En el caso *DOCODE*, los autores determinan de manera experimental que el $Umbral = Estilo - 0.075$ obtiene los mejores resultados para segmentos de $m = 400$ palabras.

La figura 5.1 muestra 3 histogramas de los valores de d_c obtenidos por cada uno de los 3 algoritmos luego de procesar la Biblia (figuras 2.9, 3.6 y 3.9) y en rojo muestra los segmentos que fueron considerados como plagio por *DOCODE* por estar bajo el valor *Umbral*.

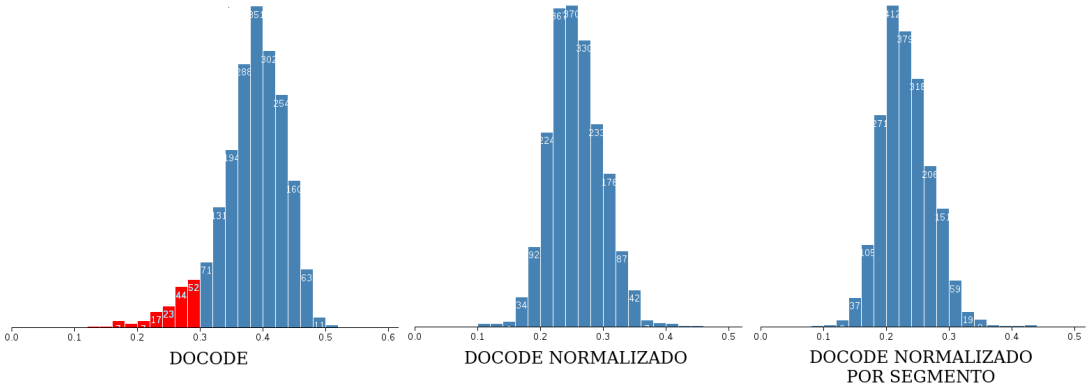


Figura 5.1: Histogramas valores d_c de segmentos procesados por las diferentes versiones de DOCODE. En rojo valores d_c de segmentos califican como plagio.

Para los algoritmos *DOCODE Normalizado* y *DOCODE Normalizado por Segmento* aún es necesario definir el valor *Umbral* que entregue los mejores resultados. Para esto se propone determinar si existe una relación entre el valor del *Estilo* del documento y la desviación estándar (σ) de los d_c calculados. La relación buscada viene definida por la ecuación:

$$Umbral = Estilo + \lambda \cdot \sigma$$

Entonces el objetivo se reduce a buscar el valor de λ que permita obtener los mejores resultados al utilizar los algoritmos *DOCODE Normalizado* y *DOCODE Normalizado por Segmento*. Para esto se trabajará en dos ramas paralelas:

- Por un lado de buscará el valor de λ que obtiene el mejor F_1, al que llamaremos $\lambda_{F_1\,optimo}$. La razón se debe a que F_1 permite observar que tan bueno son los resultados de los algoritmos en cuanto a precisión o cantidad de casos detectados correctamente. Se considera F_1 y no el valor de la métrica *precision* directamente pues F_1 considera para su cálculo el valor de *recall* y por ende evita que siempre los mejores resultados sea marcando todo como si fuese plagio.
- Por otro lado se buscará el valor de λ que obtiene el mejor *Puntaje Total*, al que llamaremos $\lambda_{PT\,optimo}$. La métrica *Puntaje Total* añade a lo anterior la medida de granularidad, por lo tanto analiza además de la precisión la eficiencia en la detección.

Para determinar el valor de $\lambda_{F_1\,optimo}$ y $\lambda_{PT\,optimo}$ se procesará cada uno de los textos contenidos en el grupo de documentos *suspicious-documents* del corpus PAN Plagiarism Corpus 2010 con ambos algoritmos utilizando valores de $\lambda \in [0.1 - 3.0]$ aumentando en 0.1 en cada iteración. Esto se llevará a cabo en dos modalidades. La primera, analizando los documentos del corpus siguiendo el orden de carpetas en que viene presentado el corpus, es decir, se procesará los documentos contenidos en las carpetas $part1, part2, ..., part32$ por separado y se registrará los valores $\lambda_{F_1\,optimo}$ y $\lambda_{PT\,optimo}$ para cada una de estas carpetas. La segunda modalidad consiste en agrupar los documentos del corpus según el nivel de plagio contenido en estos, quedando de esta manera la siguiente clasificación:

Clasificación	Nivel de Plagio (%)	Cantidad Documentos
Poco	[0 - 10]	10673
Medio	]10 - 25]	2169
Mucho	]25 - 100]	3083

Tabla 5.2: Clasificación de documentos según nivel de plagio

Una vez obtenidos los valores $\lambda_{F_1\,optimo}$ y $\lambda_{PT\,optimo}$, se procesará nuevamente cada uno de los documentos del corpus siguiendo las mismas dos modalidades definidas (por carpeta y nivel de plagio), esta vez utilizando el valor de los λ obtenidos para cada una de las ramas y finalmente se compararán resultados determinando cúal de los tres algoritmos es más eficiente en la detección de plagio.

5.4. Resultados

Para llevar a cabo los experimentos y debido a que los algoritmos desarrollados presentan sus resultados en base a segmentos de m palabras es que se debió modificar los archivos XML que detallan la posición y largo de los segmentos plagiados para cada documento del corpus, adaptando estos archivos XML al formato de salida de los algoritmos. El procesamiento realizado consistió en tomar el documento de texto y dividirlo en segmentos de m palabras. Luego se analizó el XML original del corpus y se determinó cuáles de los segmentos en que se dividió el documentos contenían a lo menos una palabra marcada como plagio dentro del XML original, marcando todos estos como segmentos plagiados dentro del XML modificado. La modificación de los XML originales permite que se pueda realizar la comparación de los pasajes marcados como plagio por los algoritmos con los pasajes originalmente plagiados usando la figura de segmentos.

A continuación se presentan los resultados obtenidos en cada una de las ramas de experimentos definidas anteriormente.

Mejor F_1

Para obtener el valor de $\lambda_{F_1\,optimo}$ que obtiene mejor F_1 en ambas mejoras, cada una de las carpetas que componen el corpus y las carpetas por nivel de plagio fueron procesadas utilizando distintos valores de λ en el rango $[0.1 - 3.0]$ aumentando en 0.1 en cada iteración. Para cada carpeta se registró los valores de *Puntaje Total*, *Recall*, *Precision*, F_1 y λ que obtenía el mejor resultado, resultados que se pueden observar en las tablas A.1, A.2 y A.3 de la sección Anexos al final del documento.

Del ejercicio anterior, se obtuvo el $\lambda_{F_1\,optimo}$ para una de las carpetas por algoritmo. De los $\lambda_{F_1\,optimo}$ obtenidos para cada algoritmo se calculó la mediada y se deteminó este valor como el $\lambda_{F_1\,optimo}$ del algoritmo en cuestión. A continuación se presenta el valor $\lambda_{F_1\,optimo}$ para cada algoritmo, siempre trabajando con segmentos de tamaño $m = 400$ palabras:

DOCODE Normalizado	$\lambda_{F_1\,optimo} = 0.2$
DOCODE Normalizado por Segmento	$\lambda_{F_1\,optimo} = 0.8$

Las tablas 5.3 y 5.4 presentan el resumen de resultados obtenidos del procesamiento de los documentos del corpus utilizando los valores de $\lambda_{F_1\,optimo}$ tanto de *DOCODE Normalizado* como de *DOCODE Normalizado por Segmento*.

Carpeta	**DOCODE**		**DOCODE Normalizado** $\lambda_{F_1\,optimo} = 0.2$		**DOCODE Normalizado por Segmento** $\lambda_{F_1\,optimo} = 0.8$	
	F_1	Puntaje Total	F_1	Puntaje Total	F_1	Puntaje Total
part1	0.606	0.238	0.63	0.113	**0.634**	0.142
part2	0.563	0.223	0.618	0.111	**0.627**	0.137
part3	**0.613**	0.23	0.609	0.108	**0.613**	0.137
part4	0.552	0.236	0.614	0.112	**0.615**	0.138
part5	0.575	0.215	0.569	0.098	**0.577**	0.12
part6	0.561	0.237	0.627	0.11	**0.63**	0.138
part7	0.679	0.223	0.678	0.108	**0.687**	0.131
part8	**0.63**	0.259	0.625	0.112	0.629	0.14
part9	**0.56**	0.215	0.541	0.1	0.532	0.124
part10	**0.605**	0.198	0.595	0.101	0.6	0.122
part11	**0.644**	0.235	0.633	0.106	0.642	0.132
part12	0.57	0.215	0.629	0.109	**0.631**	0.135
part13	**0.637**	0.213	0.605	0.105	0.614	0.129
part14	**0.635**	0.225	0.595	0.105	0.601	0.131
part15	**0.673**	0.233	0.617	0.107	0.62	0.132
part16	0.576	0.246	0.61	0.115	**0.613**	0.145
part17	0.598	0.201	0.638	0.108	**0.65**	0.135
part18	0.584	0.244	0.627	0.111	**0.633**	0.14
part19	**0.599**	0.241	0.579	0.103	0.592	0.128
part20	**0.659**	0.245	0.592	0.103	0.591	0.129
part21	0.662	0.21	0.662	0.111	**0.672**	0.137
part22	0.568	0.232	0.618	0.115	**0.621**	0.146
part23	0.583	0.205	**0.594**	0.105	0.591	0.13
part24	0.546	0.252	0.567	0.107	**0.579**	0.136
part25	0.547	0.207	0.596	0.109	**0.599**	0.136
part26	0.629	0.248	0.639	0.113	**0.655**	0.143
part27	**0.655**	0.23	0.612	0.105	0.619	0.129
part28	**0.625**	0.242	0.604	0.114	0.607	0.145
part29	**0.633**	0.224	0.588	0.098	0.598	0.118
part30	0.554	0.19	0.575	0.103	**0.579**	0.13
part31	**0.659**	0.208	0.628	0.109	0.631	0.135
part32	0.57	0.197	**0.604**	0.109	**0.604**	0.137

Tabla 5.3: F_1. Procesamiento por carpeta del corpus usando $\lambda_{F_1\,optimo}$ en cada mejora propuesta.

Carpeta	**DOCODE**		**DOCODE Normalizado** $\lambda_{F_1\,optimo} = 0.2$		**DOCODE Normalizado por Segmento** $\lambda_{F_1\,optimo} = 0.8$	
	F_1	Puntaje Total	F_1	Puntaje Total	F_1	Puntaje Total
poco	**0.434**	0.184	0.328	0.06	0.334	0.076
medio	0.689	0.275	0.923	0.166	**0.926**	0.208
mucho	0.705	0.25	**0.981**	0.17	0.957	0.203

Tabla 5.4: F_1. Procesamiento bajo clasificación por nivel de plagio usando $\lambda_{F_1\,optimo}$ en cada mejora propuesta.

Mejor Puntaje Total

Para obtener el valor de $\lambda_{PT\,optimo}$ que obtiene mejor *Puntaje Total* en ambas mejoras, cada una de las carpetas que componen el corpus y las carpetas por nivel de plagio fueron procesadas utilizando distintos valores de λ en el rango $[0.1 - 3.0]$ aumentando en 0.1 en cada iteración. Para cada carpeta se registró los valores de *Puntaje Total*, *Recall*, *Precision*, F_1 y λ que obtenía el mejor resultado, resultados que se pueden observar en las tablas A.1, A.2 y A.3 de la sección Anexos al final del documento.

Del ejercicio anterior, se obtuvo el $\lambda_{PT\,optimo}$ para una de las carpetas por algoritmo. De los $\lambda_{PT\,optimo}$ obtenidos para cada algoritmo se calculó la mediada y se determinó este valor como el $\lambda_{F_1\,optimo}$ del algoritmo en cuestión. A continuación se presenta el valor $\lambda_{PT\,optimo}$ para cada algoritmo, siempre trabajando con segmentos de tamaño $m = 400$ palabras:

DOCODE Normalizado	$\lambda_{PT\,optimo} = 2.3$
DOCODE Normalizado por Segmento	$\lambda_{PT\,optimo} = 2.5$

La tabla 5.5 y 5.4 presentan el resumen de resultados obtenidos del procesamiento de los documentos del corpus utilizando los valores de $\lambda_{PT\,optimo}$ tanto de *DOCODE Normalizado* como de *DOCODE Normalizado por Segmento*.

Carpeta	**DOCODE**		**DOCODE Normalizado** $\lambda_{PT\,optimo} = 2.3$		**DOCODE Normalizado por Segmento** $\lambda_{PT\,optimo} = 2.5$	
	Puntaje Total	F_1	Puntaje Total	F_1	Puntaje Total	F_1
part1	0.238	0.606	**0.295**	0.44	0.281	0.445
part2	0.223	0.563	**0.302**	0.412	0.278	0.412
part3	0.23	0.613	0.275	0.368	**0.283**	0.425
part4	0.236	0.552	0.254	0.355	**0.258**	0.388
part5	0.215	0.575	0.235	0.428	**0.247**	0.42
part6	0.237	0.561	**0.276**	0.394	0.259	0.415
part7	0.223	0.679	**0.272**	0.437	0.265	0.425
part8	0.259	0.63	**0.285**	0.381	0.277	0.399
part9	0.215	0.56	**0.289**	0.366	0.26	0.401
part10	0.198	0.605	**0.285**	0.426	0.245	0.422
part11	0.235	0.644	**0.297**	0.459	0.243	0.439
part12	0.215	0.57	**0.309**	0.417	0.263	0.404
part13	0.213	0.637	**0.309**	0.418	0.286	0.439
part14	0.225	0.635	**0.295**	0.417	0.261	0.44
part15	0.233	0.673	0.251	0.384	**0.259**	0.41
part16	0.246	0.576	0.285	0.377	**0.287**	0.406
part17	0.201	0.598	**0.33**	0.422	0.312	0.48
part18	0.244	0.584	**0.301**	0.388	0.28	0.423
part19	0.241	0.599	**0.288**	0.398	0.272	0.431
part20	0.245	0.659	**0.278**	0.396	0.245	0.412
part21	0.21	0.662	**0.313**	0.404	0.28	0.423
part22	0.232	0.568	**0.295**	0.389	0.272	0.404
part23	0.205	0.583	**0.284**	0.392	0.278	0.417
part24	0.252	0.546	**0.285**	0.364	0.276	0.391
part25	0.207	0.547	**0.275**	0.358	0.272	0.379
part26	0.248	0.629	**0.306**	0.409	0.266	0.415
part27	0.23	0.655	**0.309**	0.4	0.278	0.426
part28	0.242	0.625	**0.29**	0.378	0.284	0.435
part29	0.224	0.633	**0.289**	0.406	0.246	0.366
part30	0.19	0.554	**0.27**	0.377	0.258	0.416
part31	0.208	0.659	0.275	0.384	**0.282**	0.424
part32	0.197	0.57	**0.307**	0.396	0.274	0.432

Tabla 5.5: Puntaje Total. Procesamiento por carpeta del corpus usando $\lambda_{PT\,optimo}$ en cada mejora propuesta.

Carpeta	**DOCODE**		**DOCODE Normalizado** $\lambda_{PT\,optimo} = 2.3$		**DOCODE Normalizado por Segmento** $\lambda_{PT\,optimo} = 2.5$	
	Puntaje Total	F_1	Puntaje Total	F_1	Puntaje Total	F_1
poco	0.184	0.434	**0.203**	0.31	0.192	0.364
medio	0.275	0.689	**0.381**	0.593	0.358	0.642
mucho	0.25	0.705	**0.368**	0.49	0.35	0.494

Tabla 5.6: Puntaje Total. Procesamiento bajo clasificación por nivel de plagio usando $\lambda_{PT\,optimo}$ en cada mejora propuesta.

Capítulo 6

Conclusiones

6.1. Conclusiones Generales

Las mejoras propuestas al algoritmo DOCODE demostraron obtener mejores resultados en cada uno de los enfoques en los que se realizaron los experimentos, siendo *DOCODE Normalizado* el que obtuvo los mejores resultados generales de *Puntaje Total*, y *DOCODE Normalizado por Segmento* el que obtuvo los mejores resultados generales al maximizar F_1. Claramente el *Puntaje Total* es una medida mucho más robusta y con esto *DOCODE Normalizado* se presenta como la mejor alternativa a DOCODE en su versión original, a pesar de esto, *DOCODE Normalizado por Segmentos* de igual manera superó los resultados de DOCODE versión original en la búsqueda del mejor *Puntaje Total*, manteniendose siempre muy cercano a los resultados obtenidos por *DOCODE Normalizado*, por lo que demostró ser una mejora viable considerando que tiene resultados levemente menores a su par y/o supera en la métrica F_1.

Además de obtener mejores resultados, fue posible determinar una relación entre la desviación estándar de las diferencias de estilo de los segmentos, el estilo general del documento y un factor lambda, que permiten definir un *Umbral* acorde a cómo varía el estilo de escritura dentro del documento, y por ende, acorde a cada documento.

Cabe destacar que las mejoras fueron realizadas sobre una implementación realizada por el autor del presente documento ya que no se contaba con el código fuente de DOCODE. A pesar de esto, el código se implementa a partir del pseudo-código de DOCODE presentado por los creadores del algoritmo en [9].

Conceptualmente tanto DOCODE como las mejoras propuestas funcionan mejor en documentos con un bajo nivel de plagio, ya que en documentos con un alto nivel no es posible aislar el estilo de escritura del autor. No obstante, los resultados de los experimentos al clasificar los documentos según el nivel de plagio dieron mejores resultados al procesar documentos con mucho plagio, algo claramente contrario a lo expresado en las lineas anteriores. El motivo de esta situación es que al existir un alto nivel de plagio, cualquier segmento que los algoritmos marquen como segmento posiblemente no escrito por el autor tendra altas probabilidades de si ser plagio, y es debido a esto los mejores resultados en documentos con un alto nivel de plagio. Aún así, el estilo de escritura del autor de un documento será mejor capturado en documentos con un bajo nivel de plagio.

6.2. Cumplimiento de Objetivos

En la sección se definieron los objetivos específicos que se buscaban cumplir con este trabajo. A continuación se reflexionará sobre el cumplimiento de cada uno de ellos.

- **Analizar y conocer métodos y algoritmos que permiten detectar similitud entre textos.**
  Este objetivo se puede ver reflejado en el capítulo 1, donde se presentan varias de las técnicas actualmente utilizadas en los tópicos de detección de plagio en textos digitales, tanto en el área de *detección de plagio con referencia* como *detección intrínseca de plagio*, además de presentar servicios web de empresas que se dedican al rubro de la detección de plagio. Junto a lo anterior, en el capitulo 2 se analiza con mayor profundidad técnicas y algoritmos de *detección intrínseca de plagio*.

- **Adaptar y/o modificar métodos y algoritmos que permiten detectar similitud entre textos en base al objetivo planteado.**
  En el capítulo 3 se proponen mejoras al algoritmo DOCODE, algoritmo que cuenta con el título de ganador del concurso PAN @CLEF del año 2011 en la categoría de *detección de plagio con referencia*. El resultado fue exitoso y se terminó con dos propuestas que mejoran los resultados obtenidos por DOCODE en su versión original. Cabe destacar que las mejoras fueron realizadas sobre una implementación hecha por el autor del presente trabajo a partír del pseudo-código presentado por los creadores de DOCODE.

- **Definir indicadores y métricas que permitan determinar los niveles y tipos de similitud entre textos científicos.**
  El capítulo 5 presenta una serie de métricas utilizadas por gran parte de los trabajos presentes en la literatura en el área de detección de plagio. Dichas métricas fueron tomadas e implementadas en el análisis de resultados de las mejoras propuestas al algoritmo DOCODE para medir la precisión y efectividad de estas respecto a la versión original.

- **Formar una base de conocimiento técnico y teórico necesario para construir una herramienta de software que permita determinar de forma automática similitudes entre textos científicos.**
  Terminado este trabajo, el resultado es la mejora del algoritmo DOCODE en dos algoritmos, los cuales demostraron obtener mejores resultados que los obtenidos con la versión original del algoritmo. Dichos algoritmos, junto al conocimiento adquirido a lo largo del desarrollo del presente trabajo se transforman en la base buscada que permitirá en un futuro implementar una herramienta de software de detección de similitudes entre textos científicos.

6.3. Trabajo Futuro

Las mejoras propuestas a DOCODE en el presente trabajo resultaron ser muy positivas, obteniéndose con estas mejores resultados en tanto en el valor del *Puntaje Total* como en el valor de F_1 según sea el enfoque utilizado.

Uno de los primeros puntos a mejorar es la performance de los algoritmo, aplicando programación paralela al procesar textos de gran tamaño o grandes volúmenes de documentos.

Por otro lado, se propone el estudio del comportamiento tanto de DOCODE como de las mejoras propuestas a tamaño de segmentos mayores. En este trabajo el tamaño de los segmentos utilizado es $m = 400$ palabras, principalmente con el objetivo de comparar los resultados con los obtenidos y presentados por los creadores de DOCODE en [9]. Este estudio permitirá observar si se mantiene la relación $Umbral = Estilo + \sigma\lambda$ con los lambdas obtenidos en este trabajo o si el valor de los $\lambda_{optimos}$ de cada mejora(*DOCODE Normalizado* y *DOCODE Normalizado por Segmento*) varía si aumenta o disminuye el tamaño de los segmentos.

En el presente trabajo, todo el procesamiento se realizó utilizando como unidad básica una palabra, es decir uni-gramas. Un estudio interesante es considerar el uso de bi-gramas o tri-gramas tanto en DOCODE como en las mejoras.

Finalmente, el trabajo realizado se centra en capturar el estilo de escritura del autor del documento en análisis. Rescatando de este análisis el valor del estilo de escritura del autor junto con el vector de frecuencia de palabras sería posible adaptar DOCODE y las mejoras a la resolución de un nuevo desafío, que es la detección de autores.

Bibliografía

[1] Luis Alberto Barrón Cedeño. «Detección automática de plagio en texto». Tesis desarrollada dentro del Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital. Tesis de mtría. Valencia, España: Universidad Politécnica de Valencia, nov. de 2008.

[2] Alberto Barrón-Cedeño y Paolo Rosso. «On Automatic Plagiarism Detection Based on n-Grams Comparison». English. En: *Advances in Information Retrieval*. Ed. por Mohand Boughanem y col. Vol. 5478. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, págs. 696-700. ISBN: 978-3-642-00957-0. DOI: 10.1007/978-3-642-00958-7_69. URL: http://dx.doi.org/10.1007/978-3-642-00958-7_69.

[3] Chiara Basile y col. «Caglioti E.: A plagiarism detection procedure in three steps: selection, matches and 'squares». En: *SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN 09*. 2009, págs. 1-9.

[4] Paul Clough. *Plagiarism in natural and programming languages: an overview of current tools and technologies*. 2000.

[5] Sven Meyer Zu Eissen, Benno Stein y Marion Kulig. «Plagiarism detection without reference collections». En: *in Advances in Data Analysis, 2007*. 2007, págs. 359-366.

[6] Victoria Elizalde. «Estudio y desarrollo de nuevos algoritmos de detección de plagio». Tesis de mtría. Buenos Aires, Argentina: Universidad de Buenos Aires, jun. de 2011.

[7] Real Academia Española. *plagiar*. Online; accedido 29-Octubre-2016. 2016. URL: http://dle.rae.es/?id=TIZy4Xb.

[8] Dario G. Funez y Marcelo L. Errecalde. «Detección de plagio intrínseco usando la segmentación de texto». En: (2011).

[9] Gabriel Ignacio León Oberreuter Gallardo. «Diseño e Implementación de una Técnica para la Detección Intrínseca de Plagio en Documentos Digitales». Tesis para optar al grado de magíster en gestión de operaciones. Tesis de mtría. Santiago, Chile: Universidad de Chile, 2013.

[10] DAVID S. HODES. *Genetics and Running*. Online; accedido 17-Agosto-2016. 2016. URL: http://www.nytimes.com/2016/08/16/opinion/genetics-and-running.html?ref=opinion.

[11] *PAN*. http://pan.webis.de/. Accedido: 2016-09-12. Sep. de 2016.

[12] Plagscan. *PlagScan - ¿Por qué PlagScan?* Online; accedido 03-Julio-2016. 2016. URL: https://www.plagscan.com/es/hechos-sobre-control-de-plagio.

[13] Martin Potthast y col. «An Evaluation Framework for Plagiarism Detection». En: *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*. Beijing, China: Association for Computational Linguistics, ago. de 2010.

[14] Martin Potthast y col. «P.: Overview of the 1st International Competition on Plagiarism Detection». En: *In: SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09), CEUR-WS.org*. 2009, págs. 1-9.

[15] *Project Gutenberg*. http://www.gutenberg.org/wiki/Main_Page. Digital Library. URL: http://www.gutenberg.org/wiki/Main%5C_Page.

[16] Narayanan Shivakumar y Hector Garcia-molina. «SCAM: A Copy Detection Mechanism for Digital Documents». En: *In Proceedings of the Second Annual Conference on the Theory and Practice of Digital Libraries*. 1995.

[17] Efstathios Stamatatos. «Intrinsic Plagiarism Detection Using Character n-gram Profiles». En: *CEUR-WS* 502.8 (2009), págs. 1613-0073. ISSN: 1532-2890. DOI: 10.1002/asi.21630. URL: http://ceur-ws.org/Vol-502/paper8.pdf.

[18] Efstathios Stamatatos. «Plagiarism detection using stopword n-grams». En: *Journal of the American Society for Information Science and Technology* 62.12 (2011), págs. 2512-2527. ISSN: 1532-2890. DOI: 10.1002/asi.21630. URL: http://dx.doi.org/10.1002/asi.21630.

[19] *Web of Science*. http://thomsonreuters.com/en/products-services/scholarly-scientific-research/scholarly-search-and-discovery/web-of-science.html. Accessed: 2016-10-27.

[20] Wikipedia. *diff — Wikipedia, The Free Encyclopedia*. Online; accedido 27-Octubre-2016. 2016. URL: https://es.wikipedia.org/wiki/Diff.

[21] Wikipedia. *Semiosos — Wikipedia, The Free Encyclopedia*. Online; accedido 02-Julio-2015. 2015. URL: https://es.wikipedia.org/wiki/Semiosis.

[22] Mario Zechner y col. «External and Intrinsic Plagiarism Detection Using Vector Space Models». En: *In Stein et*. 2009.

Anexo A

Resultados experimentos

Aquí se presenta el detalle de los resultados de los experimentos realizados en la búsqueda de $\lambda_{F_1\,optimo}$ y $\lambda_{PT\,optimo}$ que obtienen los mejores valores de F_1 y *Puntaje Total* respectivamente.

A.1. Aplicación de algoritmos al corpus de prueba.

Resultados obtenidos de la aplicación de DOCODE, DOCODE Normalizado y DOCODE Normalizado por Segmento a cada carpeta contenida en el corpus de prueba.

La tabla A.1 presenta los resultados de la aplicación de DOCODE a cada carpeta del corpus de prueba. Las tablas A.2 y A.3 presentan los resultados de la aplicación de DOCODE Normalizado y DOCODE Normalizado a cada carpeta del corpus de prueba, registrando el valor de $\lambda_{F_1\,optimo}$ que obtuvo el mejor F_1. Finalmente las tablas A.4 y A.4 presentan los resultados de la aplicación de DOCODE Normalizado y DOCODE Normalizado por Segmento a cada carpeta del corpus de prueba, registrando el valor de $\lambda_{PT\,optimo}$ que obtuvo el mejor *Puntaje Total*.

Carpeta	Puntaje	Recall	Precision	Granularidad	F_1
part1	0.238	0.526	0.714	4.839	0.606
part2	0.223	0.46	0.726	4.764	0.563
part3	0.23	0.542	0.706	5.348	0.613
part4	0.236	0.472	0.664	4.058	0.552
part5	0.215	0.533	0.625	5.39	0.575
part6	0.237	0.474	0.688	4.169	0.561
part7	0.223	0.616	0.756	7.254	0.679
part8	0.259	0.573	0.7	4.411	0.63
part9	0.215	0.481	0.671	5.103	0.56
part10	0.198	0.545	0.679	7.336	0.605
part11	0.235	0.574	0.734	5.679	0.644
part12	0.215	0.478	0.705	5.28	0.57
part13	0.213	0.572	0.719	6.917	0.637
part14	0.225	0.563	0.728	6.088	0.635
part15	0.233	0.617	0.74	6.432	0.673
part16	0.246	0.483	0.712	4.064	0.576
part17	0.201	0.527	0.69	6.848	0.598
part18	0.244	0.521	0.664	4.236	0.584
part19	0.241	0.541	0.67	4.588	0.599
part20	0.245	0.62	0.703	5.433	0.659
part21	0.21	0.584	0.765	7.889	0.662
part22	0.232	0.487	0.682	4.478	0.568
part23	0.205	0.511	0.679	6.166	0.583
part24	0.252	0.473	0.646	3.487	0.546
part25	0.207	0.484	0.628	5.243	0.547
part26	0.248	0.568	0.705	4.796	0.629
part27	0.23	0.622	0.692	6.216	0.655
part28	0.242	0.547	0.729	4.982	0.625
part29	0.224	0.573	0.707	6.092	0.633
part30	0.19	0.47	0.675	6.549	0.554
part31	0.208	0.586	0.752	8.037	0.659
part32	0.197	0.491	0.679	6.423	0.57

Tabla A.1: Resultados aplicación de DOCODE a cada carpeta del corpus de prueba

A.2. λ_{optimo} DOCODE Normalizado

Carpeta	**Puntaje**	**Recall**	Precisión	**Granularidad**	F_1	$\lambda_{F_1\,optimo}$
part1	0.113	0.975	0.465	45.96	0.63	0.2
part2	0.114	0.968	0.456	42.806	0.62	0.3
part3	0.112	0.976	0.444	43.348	0.61	0.3
part4	0.112	0.971	0.449	44.045	0.614	0.2
part5	0.098	0.977	0.401	55.156	0.569	0.2
part6	0.108	0.978	0.461	55.351	0.627	0.1
part7	0.108	0.982	0.518	76.851	0.678	0.2
part8	0.112	0.977	0.459	46.215	0.625	0.2
part9	0.098	0.97	0.376	45.964	0.542	0.1
part10	0.099	0.98	0.43	64.05	0.598	0.1
part11	0.106	0.98	0.467	60.428	0.633	0.2
part12	0.106	0.983	0.463	59.619	0.63	0.1
part13	0.105	0.977	0.438	53.63	0.605	0.2
part14	0.103	0.975	0.43	54.591	0.597	0.1
part15	0.104	0.98	0.451	59.665	0.618	0.1
part16	0.112	0.978	0.444	42.492	0.611	0.1
part17	0.115	0.966	0.481	47.714	0.642	0.4
part18	0.111	0.973	0.462	49.424	0.627	0.2
part19	0.156	0.868	0.439	12.286	0.583	1.2
part20	0.103	0.974	0.425	51.591	0.592	0.2
part21	0.117	0.971	0.505	50.6	0.664	0.4
part22	0.113	0.977	0.454	43.596	0.62	0.1
part23	0.103	0.979	0.426	53.93	0.594	0.1
part24	0.107	0.971	0.4	38.316	0.567	0.2
part25	0.107	0.976	0.429	47.371	0.596	0.1
part26	0.129	0.959	0.486	31.213	0.645	0.6
part27	0.102	0.982	0.447	62.787	0.614	0.1
part28	0.111	0.975	0.438	42.99	0.604	0.1
part29	0.109	0.964	0.427	42.281	0.592	0.6
part30	0.113	0.96	0.411	33.349	0.576	0.5
part31	0.109	0.982	0.462	54.061	0.628	0.2
part32	0.107	0.975	0.44	50.41	0.606	0.1

Tabla A.2: Resultados aplicación de DOCODE Normalizado a cada carpeta del corpus de prueba, registrando $\lambda_{F_1\,optimo}$ que obtiene mejor F_1

Carpeta	**Puntaje**	**Recall**	Precisión	**Granularidad**	F_1	$\lambda_{F_1\,optimo}$
part1	0.165	0.903	0.497	13.869	0.641	1.1
part2	0.143	0.918	0.478	20.158	0.629	0.9
part3	0.137	0.942	0.454	21.164	0.613	0.8
part4	0.169	0.848	0.486	11.678	0.618	1.2
part5	0.12	0.932	0.418	26.894	0.577	0.8
part6	0.144	0.921	0.481	19.868	0.632	0.9
part7	0.136	0.938	0.544	32.193	0.689	0.9
part8	0.146	0.922	0.479	19.074	0.63	0.9
part9	0.169	0.802	0.408	8.212	0.541	1.4
part10	0.113	0.946	0.439	37.912	0.6	0.6
part11	0.157	0.884	0.51	16.415	0.647	1.2
part12	0.121	0.961	0.473	36.442	0.634	0.5
part13	0.149	0.9	0.472	16.835	0.619	1.1
part14	0.136	0.903	0.453	20.468	0.603	0.9
part15	0.115	0.96	0.462	42.178	0.624	0.4
part16	0.135	0.946	0.458	22.95	0.617	0.6
part17	0.129	0.94	0.498	31.565	0.651	0.7
part18	0.147	0.916	0.485	18.971	0.634	0.9
part19	0.128	0.924	0.435	23.753	0.592	0.8
part20	0.124	0.939	0.432	26.309	0.592	0.7
part21	0.124	0.963	0.517	42.557	0.673	0.5
part22	0.135	0.947	0.468	23.747	0.626	0.6
part23	0.178	0.821	0.475	9.472	0.602	1.4
part24	0.136	0.925	0.421	18.15	0.579	0.8
part25	0.122	0.946	0.439	29.355	0.6	0.5
part26	0.137	0.945	0.501	26.589	0.655	0.7
part27	0.124	0.946	0.461	31.157	0.62	0.7
part28	0.145	0.922	0.452	17.29	0.607	0.8
part29	0.129	0.921	0.446	24.171	0.601	1.0
part30	0.142	0.911	0.43	16.315	0.584	1.0
part31	0.135	0.939	0.475	24.719	0.631	0.8
part32	0.119	0.953	0.45	34.039	0.611	0.4

Tabla A.3: Resultados aplicación de DOCODE Normalizado por Segmento a cada carpeta del corpus de prueba, registrando $\lambda_{F_1\,optimo}$ que obtiene mejor F_1

Carpeta	**Puntaje**	**Recall**	**Precision**	**Granularidad**	F_1	$\lambda_{PT\,optimo}$
part1	0.312	0.456	0.511	1.915	0.482	2.2
part2	0.305	0.322	0.486	1.411	0.387	2.4
part3	0.281	0.399	0.416	1.735	0.407	2.2
part4	0.269	0.418	0.474	2.135	0.444	2.1
part5	0.291	0.339	0.437	1.481	0.382	2.5
part6	0.287	0.239	0.482	1.161	0.32	2.7
part7	0.287	0.337	0.494	1.628	0.401	2.4
part8	0.294	0.395	0.449	1.692	0.42	2.2
part9	0.299	0.279	0.418	1.176	0.335	2.5
part10	0.285	0.403	0.452	1.816	0.426	2.3
part11	0.304	0.287	0.502	1.303	0.365	2.6
part12	0.309	0.369	0.48	1.551	0.417	2.3
part13	0.309	0.404	0.434	1.557	0.418	2.3
part14	0.295	0.383	0.458	1.665	0.417	2.3
part15	0.272	0.259	0.425	1.273	0.322	2.5
part16	0.302	0.379	0.464	1.607	0.417	2.2
part17	0.33	0.362	0.507	1.427	0.422	2.3
part18	0.302	0.382	0.45	1.577	0.413	2.2
part19	0.288	0.363	0.441	1.608	0.398	2.3
part20	0.289	0.436	0.448	1.88	0.442	2.2
part21	0.313	0.355	0.469	1.451	0.404	2.3
part22	0.305	0.388	0.494	1.684	0.435	2.2
part23	0.288	0.3	0.428	1.339	0.353	2.5
part24	0.3	0.445	0.404	1.659	0.424	2.1
part25	0.279	0.357	0.415	1.592	0.384	2.2
part26	0.312	0.397	0.481	1.624	0.435	2.2
part27	0.309	0.369	0.436	1.452	0.4	2.3
part28	0.304	0.275	0.519	1.272	0.36	2.5
part29	0.299	0.336	0.454	1.452	0.386	2.4
part30	0.289	0.407	0.422	1.701	0.414	2.2
part31	0.287	0.312	0.45	1.437	0.369	2.4
part32	0.307	0.348	0.459	1.441	0.396	2.3

Tabla A.4: Resultados aplicación de DOCODE Normalizado a cada carpeta del corpus de prueba, registrando $\lambda_{PT\,optimo}$ que obtiene mejor Puntaje Total

Carpeta	**Puntaje**	**Recall**	Precision	**Granularidad**	F_1	$\lambda_{PT\,optimo}$
part1	0.285	0.446	0.514	2.186	0.478	2.4
part2	0.282	0.333	0.497	1.667	0.399	2.6
part3	0.292	0.28	0.434	1.24	0.34	2.9
part4	0.263	0.471	0.491	2.55	0.481	2.2
part5	0.257	0.38	0.431	1.975	0.404	2.6
part6	0.27	0.261	0.479	1.385	0.338	3.0
part7	0.292	0.464	0.507	2.163	0.485	2.4
part8	0.277	0.501	0.478	2.404	0.489	2.2
part9	0.276	0.295	0.429	1.408	0.35	2.8
part10	0.274	0.319	0.463	1.6	0.378	2.8
part11	0.259	0.238	0.498	1.363	0.322	3.0
part12	0.269	0.408	0.494	2.16	0.447	2.4
part13	0.293	0.389	0.462	1.718	0.422	2.6
part14	0.286	0.306	0.452	1.424	0.365	2.9
part15	0.264	0.236	0.456	1.262	0.311	3.0
part16	0.287	0.359	0.467	1.663	0.406	2.5
part17	0.314	0.394	0.527	1.705	0.451	2.6
part18	0.28	0.378	0.481	1.849	0.423	2.5
part19	0.272	0.404	0.463	2.001	0.431	2.5
part20	0.253	0.335	0.444	1.852	0.382	2.7
part21	0.285	0.441	0.504	2.141	0.47	2.3
part22	0.287	0.455	0.5	2.16	0.476	2.3
part23	0.278	0.379	0.464	1.835	0.417	2.5
part24	0.279	0.409	0.421	1.803	0.415	2.4
part25	0.275	0.391	0.434	1.825	0.411	2.4
part26	0.272	0.47	0.517	2.511	0.492	2.3
part27	0.29	0.304	0.461	1.403	0.366	2.8
part28	0.297	0.319	0.519	1.513	0.395	2.7
part29	0.277	0.444	0.448	2.053	0.446	2.3
part30	0.266	0.371	0.451	1.888	0.407	2.6
part31	0.282	0.384	0.473	1.84	0.424	2.5
part32	0.283	0.358	0.489	1.755	0.413	2.6

Tabla A.5: Resultados aplicación de DOCODE Normalizado por Segmento a cada carpeta del corpus de prueba, registrando $\lambda_{PT\,optimo}$ que obtiene mejor Puntaje Total

A.3. Aplicación de algoritmos al corpus de prueba clasificado por nivel de plagio.

Resultados obtenidos de la aplicación de DOCODE, DOCODE Normalizado y DOCODE Normalizado por Segmento al corpus de prueba, clasificando los documentos del corpus según su nivel de plagio contenido.

La tabla A.6 presenta detalles de la clasificación por nivel de plagio contenido en los documentos realizada al corpus de prueba. La tabla A.7 presenta los resultados de la aplicación de DOCODE al corpus clasificado por nivel de plagio. Las tablas A.8 y A.9 presentan los resultados de la aplicación de DOCODE Normalizado y DOCODE Normalizado por Segmento al corpus clasificado por nivel de plagio, registrando el valor de $\lambda_{F_1\,optimo}$ que obtuvo el mejor F_1. Finalmente las tablas A.10 y A.11 presentan los resultados de la aplicación de DOCODE Normalizado y DOCODE Normalizado por Segmento al corpus clasificado por nivel de plagio,

registrando el valor de $\lambda_{PT\,optimo}$ que obtuvo el mejor *Puntaje Total*.

Clasificación	Nivel de Plagio (%)	Cantidad Documentos
Poco	[0 - 10]	10673
Medio	]10 - 25]	2169
Mucho	]25 - 100]	3083

Tabla A.6: Clasificación de documentos según nivel de plagio

Carpeta	**Puntaje**	**Recall**	**Precision**	**Granularidad**	F_1
poco	0.184	0.503	0.381	4.124	0.434
medio	0.275	0.532	0.979	4.676	0.689
mucho	0.25	0.545	0.997	6.052	0.705

Tabla A.7: Resultados aplicación de DOCODE al corpus de prueba clasificado por nivel de plagio.

Carpeta	**Puntaje**	**Recall**	**Precision**	**Granularidad**	F_1	$\lambda_{F_1\,optimo}$
poco	0.115	0.776	0.213	6.438	0.334	1.5
medio	0.162	0.982	0.871	50.926	0.923	0.1
mucho	0.167	0.979	0.988	58.583	0.983	0.1

Tabla A.8: Resultados aplicación de DOCODE Normalizado al corpus de prueba clasificado por nivel de plagio, registrando $\lambda_{F_1\,optimo}$ que obtiene mejor F_1

Carpeta	**Puntaje**	**Recall**	**Precision**	**Granularidad**	F_1	$\lambda_{F_1\,optimo}$
poco	0.154	0.702	0.257	4.413	0.376	2.0
medio	0.172	0.98	0.884	41.042	0.93	0.2
mucho	0.169	0.974	0.989	55.48	0.981	0.1

Tabla A.9: Resultados aplicación de DOCODE Normalizado por Segmento al corpus de prueba clasificado por nivel de plagio, registrando $\lambda_{F_1\,optimo}$ que obtiene mejor F_1

Carpeta	**Puntaje**	**Recall**	**Precision**	**Granularidad**	F_1	$\lambda_{PT\,optimo}$
poco	0.225	0.295	0.246	1.29	0.268	2.8
medio	0.381	0.433	0.942	1.942	0.593	2.3
mucho	0.381	0.422	0.988	1.928	0.591	2.1

Tabla A.10: Resultados aplicación de DOCODE Normalizado al corpus de prueba clasificado por nivel de plagio, registrando $\lambda_{PT\,optimo}$ que obtiene mejor Puntaje Total

Carpeta	**Puntaje**	**Recall**	**Precision**	**Granularidad**	F_1	$\lambda_{PT\,optimo}$
poco	0.21	0.381	0.264	1.796	0.312	3.0
medio	0.358	0.483	0.959	2.466	0.642	2.5
mucho	0.36	0.416	0.989	2.087	0.586	2.3

Tabla A.11: Resultados aplicación de DOCODE Normalizado por Segmento al corpus de prueba clasificado por nivel de plagio, registrando $\lambda_{PT\,optimo}$ que obtiene mejor Puntaje Total