INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

DETECCIÓN AUTOMÁTICA DE PLAGIO A TRAVÉS DE FORMACIÓN DE PASAJES

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA
ING. MIGUEL ÁNGEL SÁNCHEZ PÉREZ

DIRECTORES DE TESIS
DR. ALEXANDER GELBUKH
DR. GRIGORI SIDOROV

México, D. F., junio de 2014

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 13:00 horas del día 23 del mes de abril de 2014 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Detección automática de plagio a través de formación de pasajes"

Presentada por el alumno:

SÁNCHEZ	PÉREZ	MIGUEL ÁNGEL
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

A	1	2	0	4	2	5

aspirante de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de Tesis

Dr. Alexander Gelbukh

Dr. Grigori Sidorov

Dr. Sergio Suárez Guerra

Dr. Francisco Hiram Calvo Castro

Dr. Héctor Jiménez Salazar

Dr. Miguel Jesús Torres Ruiz

PRESIDENTE DEL COLEGIO DE PROFESORES

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION
Y COMPUTACION
DIRECCION

Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México, D.F. el día **6** del mes de **mayo** del año **2014**, el (la) que suscribe **Miguel Ángel Sánchez Pérez** alumno(a) del Programa de **Maestría en Ciencias de la Computación**, con número de registro **A120425**, adscrito(a) al **Centro de Investigación en Computación**, manifiesto(a) que es el (la) autor(a) intelectual del presente trabajo de Tesis bajo la dirección del (de la, de los) **Dr. Alexander Gelbukh y Dr. Grigori Sidorov,** y cede los derechos del trabajo titulado **Detección automática de plagio a través de formación de pasajes**, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del (de la) autor(a) y/o director(es) del trabajo. Este puede ser obtenido escribiendo a las siguientes direcciones **masp1988@hotmail.com, gelbukh@gelbukh.com y sidorov@cic.ipn.mx**. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Miguel Ángel Sánchez Pérez

Nombre y firma del alumno(a)

Resumen

En este trabajo se presenta un modelo computacional para la alineación de textos en la tarea de detección automática de plagio monolingüe. El modelo se basa en la comparación de oraciones en dos documentos buscando correspondencias entre fragmentos de textos. Usamos la representación en un espacio vectorial de las oraciones así como medidas estándares para calcular similitud entre vectores.

En el modelo se integran características usadas en otros modelos presentados en el estado del arte y aportaciones como:

- El uso de esquemas de ponderación basados en oraciones como unidades independientes en lugar de documentos.
- Dos métodos para eliminar correspondencias de fragmentos de textos que presentan solapamiento en alguno de sus fragmentos.

El modelo se entrenó en el corpus provisto por el 9º laboratorio de evaluación en detección de plagio, identificación de autoría y mal uso de software social (PAN 2013). El corpus del PAN 2013 se divide en cinco sub-corpus cubriendo una variedad de tipos de plagio, incluyendo copia exacta, ofuscación aleatoria, ofuscación a través de traducción cíclica y el uso de resúmenes de documentos. Luego de entrenar el modelo se experimentó en el corpus de prueba en el que se obtuvo una medida general *plagdet* de 86.9%. Los resultados obtenidos se compararon con los resultados obtenidos por los participantes en el 9º laboratorio de evaluación y presentados en el estado del arte, superando a dichos sistemas en más de 3%.

Actualmente se participa en el 11º laboratorio de evaluación en detección de plagio, identificación de autoría y mal uso de software social (PAN 2014) como parte del congreso CLEF que se llevará a cabo en Sheffield, UK, del 15 al 18 de septiembre de 2014.

Abstract

This work presents a computational model for word alignment in the task of monolingual automatic plagiarism detection. The model is based on sentences comparison in two documents searching for correspondences between text fragments. We use vector space representation of sentences as well as standards similarity measures between vectors.

The model integrates features used in other models presented in the state-of-the-art and original ideas like:

• The use of weighting schemes based on sentences as independent units instead of documents.

• Two methods for removing correspondences of text fragments that are overlapping in one of its fragments.

The model was trained in the corpus provided by the 9th evaluation lab on uncovering plagiarism, authorship, and social software misuse (PAN 2013). The PAN 2013 corpus is divided into five sub-corpus covering a variety of plagiarism types, including verbatim copy, random obfuscation, cyclic translation obfuscation and using summarization of documents. Once the model was trained we experimented in the test corpus in which a general measure plagdet of 86.9% was obtained. The results were compared with the results obtained by participants in the 9th evaluation lab and presented in the state-of-the-art, surpassing them by more than 3%.

We are currently participating in the 11º evaluation lab on uncovering plagiarism, authorship, and social software misuse (PAN 2014) as part of the CLEF conference to be hold in Sheffield, UK, on September 15-18, 2014.

Agradecimientos

Agradezco a mis asesores de tesis por la guía, formación y apoyo que me procuraron para la realización de este trabajo.

Agradezco a mis padres y hermano por su apoyo incondicional, consejos y exhortarme a ser mejor.

Agradezco a mi novia por sus palabras de aliento, compañía y paciencia.

Agradezco a mis amigos por su apoyo.

Agradezco al Centro de Investigación en Computación, al Instituto Politécnico Nacional y al CONACyT por las facilidades otorgadas para el desarrollo de este trabajo.

Índice

Índice de figuras

Índice de tablas

1 INTRODUCCIÓN

Existen varias definiciones de plagio. Algunas de estas definiciones son:

- "Copiar en lo sustancial obras ajenas, dándolas como propias" [1].
- Uso de cualquier fuente, publicada o no, sin el correcto reconocimiento a la fuente [2].
- Apropiación de las palabras y las ideas de otros [3].

Este se puede presentar en la música, imágenes, documentos escritos e incluso ideas. En este trabajo de tesis nos enfocamos en el plagio de documentos escritos.

La definición de plagio, sin embargo, es vaga, pues aplica a una vasta cantidad de situaciones en la que no todos los autores concuerdan. No obstante muchas de las situaciones más comunes coinciden. El sitio web www.plagiarism.org menciona algunas de estas situaciones comunes que son consideras plagio [4].

- Mostrar el trabajo de alguien más como propio.
- Copiar palabras o ideas de alguien más sin dar crédito (copia literal).
- No colocar una cita después de usar comillas.
- Cambiar palabras (paráfrasis) pero manteniendo la estructura de las oraciones de una fuente sin dar crédito.
- Copian tantas palabras o ideas de una fuente que integra la mayoría de un trabajo aunque se dé crédito.

En instituciones académicas la acusación de plagio a algún integrante de su comunidad resulta sumamente inmoral además que daña extraordinariamente su reputación y por lo mismo el plagio es un fenómeno combatido exhaustivamente. La gran mayoría de estas instituciones cuentan con reglamentos estrictos que contemplan el plagio como una falta grave e instruyen a su comunidad con el fin de evitar se presente, ya sea intencional o inconscientemente. Por ejemplo, en su documento sobre Integridad Académica del 2011, la Universidad de Princeton clasifica el plagio como una falta grave que puede resultar en un período de prueba, suspensión o expulsión.

No obstante, en los últimos años se ha dado un incremento considerable del plagio textual debido a la gran cantidad de información disponible en medios digitales como bases de datos, el internet o dispositivos de almacenamiento masivo, que han hecho de la detección de plagio manual una tarea prácticamente imposible. Esta problemática ha sido atacada a través de la misma tecnología con la creación de sistemas que asisten en la toma de decisiones acerca del posible plagio de un documento. Muchos de estos sistemas se basan en las mismas herramientas que se usan comúnmente para recuperar información de la cual se plagió, como buscadores, manejadores de bases de datos o acceso a documentos relacionados (trabajo de otros estudiantes o investigadores); que aunado a la gran capacidad de cómputo que existe hoy en día permite analizar y filtrar gran cantidad de información que tomaría mucho tiempo hacer manualmente.

1.1 Planteamiento del problema

El problema del análisis automático de plagio en documentos de textos se puede abordar de dos formas: El análisis intrínseco, el cual busca identificar potenciales fragmentos de texto plagiados analizando un documento sospechoso respecto a cambios en su estilo de escritura; y el análisis basado en corpus o usando documentos de referencia, el cual compara documentos sospechosos contra un conjunto de potenciales documentos originales [5].

Para fines de este trabajo de tesis nos enfocamos en el análisis de plagio usando documentos de referencia donde ya se cuenta con un conjunto de documentos de referencia reducido. Muchas de las técnicas propuestas en los sistemas para la detección de plagio que siguen este enfoque se basan en tres ideas principales [6]:

- Comparación de sub-cadenas: Busca identificar el número máximo de coincidencias en un par de cadenas, el cual luego es usado como indicador para la detección de plagio.
- Similitud de palabras claves: Busca extraer y ponderar palabras claves que identifiquen un tópico en un documento y compararlas con las palabras claves de otro documento.
- Análisis de huellas digitales textuales: Los documentos son divididos en secuencias de términos (fragmentos de textos), de los cuales se calcula un valor representativo y en conjunto generan la huella digital de un documento.

En el contexto del 9no laboratorio de evaluación en detección de plagio [7], identificación de autoría y mal uso de software social (PAN), la detección de plagio textual se divide dos tareas principales:

1 Recuperación de las fuentes: Dado un documento sospechoso y un motor de búsqueda, la tarea es recuperar todas las fuentes plagiadas mientras se minimiza el costo de la recuperación.
2 Alineación de textos: Dado un par de documentos, la tarea es identificar todos los pasajes contiguos de máxima longitud de texto reusado entre ellos.

En este trabajo de tesis nos enfocamos en la tarea de alineación de textos usando el corpus dado para la competencia del año 2013 como base para realizar experimentos, así como los resultados reportados de los sistemas que participaron en este con fines de comparación y como estado del arte en nuestro campo.

1.2 Objetivos

1.2.1 Objetivo general

Desarrollar un modelo computacional que encuentre la correspondencia entre fragmentos de textos entre dos documentos en la tarea de la detección automática de plagio.

1.2.2 Objetivos particulares

- Analizar la influencia del pre-procesamiento en la tarea de alineación de textos para la detección automática de plagio.
- Analizar la influencia de las características seleccionadas y su ponderación en la tarea de alineación de textos para la detección automática de plagio.
- Crear e implementar un algoritmo que minimice la granularidad de los pasajes plagiados detectados.
- Crear e implementar varias técnicas de post-procesamiento para mejorar los resultados.

1.3 Estructura del documento

Capítulo 2: Estado del arte respecto a la alineación de textos para la detección automática de plagio en el contexto del laboratorio de evaluación en detección de plagio textual, identificación de autoría y mal uso de software social.

Capítulo 3: Se presentan algunas de las técnicas del procesamiento de lenguaje natural usadas en el modelo propuesto.

Capítulo 4: Metodología seguida para la creación del modelo computacional divido en varias fases comunes como se muestra en el estado del arte.

Capítulo 5: Presentación y análisis de los resultados arrojados por el modelo propuesto en comparación con los modelos presentados en la competición del año 2013. Además se muestra la influencia que tiene cada sección del modelo propuesto en la metodología, así como la optimización de los parámetros que éste usa.

2 ESTADO DEL ARTE

En este capítulo se realiza un estudio de los principales modelos propuestos que abordan la temática de la detección automática de plagio en documentos de textos. El capítulo se divide en tres secciones que engloban las principales tareas en los modelos de detección de plagio basados en documentos de referencia.

2.1 Selección

Dado dos documentos, uno sospechoso y uno de referencia, las correspondencias entre estos documentos son identificadas a través de ciertas heurísticas. Muchas de estas heurísticas encuentran correspondencias exactas en el texto o crean correspondencias cambiando los textos a otros dominios o formas motivadas lingüísticamente [8].

2.1.1 Kong Leilei et al.

En [9] [10], después de eliminar caracteres especiales, convertir a minúsculas, eliminar palabras auxiliares y obtención de raíces; los documentos son segmentados en oraciones. Luego se calculan dos medidas de similitud entre ellas considerando correspondientes los pares que sobrepasen algún umbral. Las medidas utilizadas son la medida del coseno y una variación del coeficiente de Dice a la que llaman similitud de estructura [9]. La medida del coseno está dada por:

$$Sim(S,R) = \cos\theta = \frac{\sum_{k=1}^{n} w_{S_k} * w_{R_k}}{\sqrt{(\sum_{k=1}^{n} {w_{S_k}}^2)(\sum_{k=1}^{n} {w_{R_k}}^2)}} > t1 \quad (2.1)$$

Donde w_{S_k} y w_{R_k} son los pesos de S y R respectivamente. El valor usado para t1 es 0.42

La similitud de estructura está dada por:

$$T(S,R) = \frac{2 * \sum_{k=1}^{n} \min(w_{S_k}, w_{R_k})}{\sum_{k=1}^{n} w_{S_k} + \sum_{k=1}^{n} w_{R_k}} > t2 \quad (2.2)$$

Donde w_{S_k} y w_{R_k} son los pesos de S y R respectivamente. El valor usado para t2 es 0.32.

2.1.2 Rodríguez Torrejón et al.

En [11] [12], buscan correspondencias en los textos extrayendo n-gramas de contexto (CTnG) y algunas variaciones de estos a los que denominan n-gramas de contexto cercano (SCnG) y *odd-*

even n-gramas (OEnG). El objetivo es encontrar las mismas características en ambos documentos. Cabe mencionar que los SCnG y OEnG son una versión de los Salta-k n-gramas.

Los n-gramas de contexto para la tarea de detección de plagio se obtienen a través del modelado de los documentos en seis pasos [13]:

1. Conversión a minúsculas.
2. Eliminación de palabras auxiliares.
3. Eliminación de *tokens* de un solo símbolo.
4. Obtención de lexemas.
5. Ordenación alfabética interna de los *tokens* del n-grama.
6. Solapamiento de n-1 *tokens* entre n-gramas consecutivos.

Los n-gramas de contexto cercano (SCnG) se obtienen excluyendo el segundo elemento (SCnG derechos) o penúltimo elemento (SCnG izquierdos) de un grupo de n+1 elementos relevantes además de los n-gramas contextuales (SCnG directos) [14].

Mientras tanto, los *odd-even* n-gramas (OEnG) se obtienen eliminando el segundo y penúltimo elemento de un grupo de n+2 elementos relevantes.

Por ejemplo, para la oración:

"En un lugar de la mancha de cuyo nombre no quiero acordarme"

Que al ser modelado quedaría como:

"lugar mancha nombre querer acordar"

Los CT3G, SC3G y OE3G serían:

1.	lugar, mancha, nombre	→	lugar, mancha, nombre	1er SC3G directo o CT3G
2.	lugar, mancha, querer	→	lugar, mancha, querer	1er SC3G izquierdo
3.	lugar, nombre, querer	→	lugar, nombre, querer	1er SC3G derecho
4.	lugar, nombre, acordar	→	acordar, lugar, nombre	1er OE3G
5.	mancha, nombre, querer	→	mancha, nombre, querer	2do SC3G directo o CT3G
6.	mancha, nombre, acordar	→	acordar, manchar, nombre	2do SC3G izquierdo
7.	mancha, querer, acordar	→	acordar, mancha, querer	2do SC3G derecho
8.	nombre, querer, acordar	→	acordar, nombre, querer	3er SC3G directo o CT3G

En el sistema presentado en el concurso PAN 2012 [15] y PAN 2013 [8] usan 3-gramas.

2.1.3 Šimon Suchomel et al.

En [16] [17], extraen dos características principales:

- 5-gramas de palabras ordenadas alfabéticamente con al menos tres caracteres de longitud.

- 8-gramas de palabras auxiliares formados por las 50 palabras auxiliares más comunes del inglés reportadas por Stamatatos en [18], que se muestran en la tabla 2.1. Se ignoran los 8-gramas formados solamente por las seis palabras más comunes en el inglés (*the, of, and, a, in, to*) o la contracción *'s*.

Tabla 2.1 Lista de las 50 palabras auxiliares en el corpus BNC para el inglés

1. the	*11. with*	*21. are*	*31. or*	*41. her*
2. of	*12. he*	*22. not*	*32. an*	*42. n't*
3. and	*13. be*	*23. his*	*33. were*	*43. there*
4. a	*14. on*	*24. this*	*34. we*	*44. can*
5. in	*15. i*	*25. from*	*35. their*	*45. all*
6. to	*16. that*	*26. but*	*36. been*	*46. as*
7. is	*17. by*	*27. had*	*37. has*	*47. if*
8. was	*18. at*	*28. which*	*38. have*	*48. who*
9. it	*19. you*	*29. she*	*39. will*	*49. what*
10. for	*20. 's*	*30. they*	*40. would*	*50. said*

En [17], también prueban utilizando 4-gramas de palabas ordenadas alfabéticamente con al menos tres caracteres de longitud y 4-gramas de contexto cercano obtenidos a partir de 5-gramas eliminando el tercer elemento.

El objetivo es encontrar las características comunes en ambos documentos.

2.1.4 Prasha Shrestha et al.

En [19], se usan varios tipos de n-gramas.

- N-gramas de palabras auxiliares: N-gramas de solo palabras auxiliares presentes en el documento.
- N-gramas de entidades nombradas: N-gramas que contienen al menos una entidad nombrada.
- N-gramas de palabras: N-gramas formados por cualquier palabra en el documento.

En el proceso de selección:

Para extraer los n-gramas de palabras auxiliares se usa la lista de las 50 palabras auxiliares más comunes del inglés propuestas por Stamatatos en [18] y se buscan coincidencias exactas entre el documento sospechoso y el de referencia.

Para extraer los n-gramas de entidades nombradas se buscan todas las entidades nombradas presentes en el documento. Luego, se extraen todos los n-gramas de palabras que contengan dichas entidades nombradas, es decir, para una entidad nombrada 'x' se extraen los n-gramas donde 'x' es la primera palabra del n-grama hasta donde 'x' es la última palabra del n-grama. Luego, se comparan cada n-grama entre el documento sospechoso y el documento de referencia.

En este caso, se calcula la cantidad de palabras comunes entre los n-gramas y si sobrepasa un umbral dado se considera una correspondencia.

Por último se extraen los n-gramas de palabras y se usa el mismo método de comparación entre n-gramas que el usado en los n-gramas de entidades nombradas.

Los parámetros usados los calculan empíricamente midiendo el desempeño sobre el corpus de entrenamiento del PAN 2013.

2.1.5 Yurii Palkovskii

En [20] [21] primero pre-procesan los documentos convirtiendo a minúsculas, eliminando los números y obteniendo las raíces. Luego, extraen 5-gramas de palabras y le calculan una huella digital usando y las ordenan de acuerdo al algoritmo *alphasorting* propuesto en [22] [23].

2.1.6 Robin Küppers et al.

En [24] se pre-procesan los documentos cambiando los separadores (espacio, tabulador, nueva línea, retorno de carro) por espacio. Luego, se extraen fragmentos de textos de una longitud semi-fija de 250 caracteres. Se dice de longitud semi-fija porque a partir del fragmento de 250 caracteres, para evitar dividir un *token*, el algoritmo aumenta el fragmento en un carácter hasta encontrar un espacio.

Tabla 2.2 Optimización de parámetros (CS: Tamaño del fragmento, ST: Umbral, PD: *plagdet,* R: *Recall,* P: Precisión, G: Granularidad)

CS	ST	PD	R	P	G	CS	ST	PD	R	P	G
	0.20	0.12	0.29	0.10	1.09		0.20	0.29	0.37	0.57	1.33
	0.40	**0.28**	0.25	0.45	1.00		0.40	**0.40**	0.36	0.78	1.08
10000	0.60	0.22	0.20	0.33	1.00	1000	0.60	0.24	0.24	0.67	1.19
	0.80	0.17	0.17	0.17	1.00		0.80	0.17	0.17	0.33	1.18
	1.00	0.17	0.17	0.17	1.00		1.00	0.17	0.17	0.17	1.00
	0.20	0.20	0.30	0.32	1.08		0.20	0.22	0.39	0.47	1.77
	0.40	**0.31**	0.28	0.58	1.02		0.40	**0.38**	0.39	0.96	1.46
5000	0.60	0.23	0.21	0.33	1.00	500	0.60	0.22	0.24	0.66	1.38
	0.80	0.17	0.17	0.33	1.00		0.80	0.17	0.17	0.33	1.21
	1.00	0.17	0.17	0.17	1.00		1.00	0.17	0.17	0.17	1.00
	0.20	0.27	0.33	0.49	1.12		0.20	0.14	0.36	0.34	2.20
	0.40	**0.36**	0.33	0.71	1.02		0.40	**0.43**	0.42	0.97	1.27
2500	0.60	0.23	0.22	0.50	1.04	**250**	0.60	0.24	0.27	0.67	1.35
	0.80	0.17	0.17	0.33	1.00		0.80	0.18	0.18	0.33	1.20
	1.00	0.17	0.17	0.17	1.00		1.00	0.17	0.17	0.17	1.00

Tomando en cuenta que tenemos *n* fragmentos en el documento sospechoso y *m* fragmentos en el documento de referencia, se realiza una comparación *n x m* calculando la similitud entre cada par. Antes de calcular la similitud entre dos fragmentos se eliminan las palabras auxiliares.

Los fragmentos ajustados se tratan como bolsas de palabras entre las cuales se calcula el coeficiente de Dice como medida de similitud y se toman los pares que sobrepasan cierto umbral th. Los autores utilizan $th = 0.4$.

$$\frac{2 * |A \cap B|}{|A| + |B|} > th \qquad (2.3)$$

Los parámetros th y el tamaño de los fragmentos fueron obtenidos experimentalmente optimizando la medida *plagdet* propuesta en [25] sobre un sub-corpus de aproximadamente el 10% del tamaño del corpus del PAN2012. En la tabla 2.2 se muestran las pruebas realizadas donde se observa que el mejor resultada para *plagdet* se obtiene para CS=250 y ST=0.4.

2.1.7 Lee Gillam et al.

En [26] [27] los autores no dan detalles sobre las características usadas en su sistema debido a una patente pendiente pero mencionan algunas cosas que no hacen.

- No eliminan las palabras auxiliares sin ningún tipo de heurísticas ya que las consideran una parte importante del estilo de un documento.
- No usan métodos de encriptación o *hashing* con el objetivo de crear claves de longitud fija de los datos.
- No fragmentan los textos en grandes cantidades de pequeños n-gramas de caracteres o n-gramas de palabras.

Para la comparación de las características usan la medida del coseno con un umbral de 0.75

2.2 Integración

Después de obtener correspondencias entre las características extraídas de los documentos (n-gramas, oraciones, fragmentos de texto), se busca extender dichas características a pasajes de textos de máxima longitud reportadas como casos de plagio. El objetivo de la integración de las características es identificar un caso de plagio como un todo en lugar de solamente sus fragmentos [8].

2.2.1 Kong Leilei et al.

En [9] [10] los autores usan un algoritmo que llaman Algoritmo de Integración Bilateral Alternado (*Bilateral Alternating Merging Algorithm*). Sin embargo es en [10] donde describen su algoritmo como se muestra en el anexo A. Este algoritmo se basa en ir formando grupos de oraciones adyacentes alternado entre el documento sospechoso y el documento de referencia.

2.2.2 Rodríguez Torrejón et al.

En [11] [12] definen dos parámetros para la integración de las correspondencias encontradas. Estos parámetros son la longitud mínima de correspondencias adyacentes ($minLength$) y la distancia máxima entre correspondencias encontradas ($maxDist$). Estas medidas están dadas en n-gramas para el documento sospechoso y en caracteres para el documento de referencia.

$$doc.sospechoso \begin{cases} maxNgramDist = 2 \times chunkLength \\ minNgramLength = (monotony - 1.5) \times chunkLength \end{cases} \quad (2.4)$$

$$doc.\ referencia \begin{cases} maxCharDist = chunkLength \times wordLengthAverage \\ minCharLength = minNgramLength \times wordLengthAverage \end{cases} \quad (2.5)$$

Donde $wordLengthAverage$ es la longitud promedio en caracteres de las palabras. Los parámetros $chunkLength$ y $monotony$ son calculados experimentalmente. Para el corpus de entrenamiento en PAN2012 [15] se obtiene $chunkLength = 4\ ngramas$ y $monotony = 2\ chunks$, mientras que para el corpus de entrenamiento en PAN2013 [8] se obtiene $chunkLength = 8\ ngramas$ y $monotony = 2\ chunks$.

2.2.3 Šimon Suchomel et al.

En [16] los autores buscan integrar correspondencias en intervalos válidos. Un intervalo válido consisten de 4 características comunes en ambos documentos, con una distancia máxima entre característica de 4000 caracteres.

El algoritmo para obtener intervalos válidos es una modificación del algoritmo reportado en [28] para que pueda funcionar con más de un tipo de característica (5-grama de palabras y 8-grama de palabras auxiliares). La modificación consiste en usar los caracteres de inicio de las características en lugar de su posición en una lista ordenada.

La entrada del algoritmo es una lista de pares (Característica (ID) en documento 1 (D_1), Característica (ID) en documento 2 (D_2)). Si una característica en D_1 corresponde a más de una característica en D_2, la lista tiene más de una entrada para dicha característica. Los pasos del algoritmo planteado en [28] son los siguientes:

1. Establecer la variable local $depth$ a 0.
2. Ordenar la lista de pares por ID en D_1.
3. Dividir la lista hasta el "intervalo válido" más largo posible, ignorar cualquier par de IDs que no esté presente en algún intervalo válido.
4. Si solo existe un intervalo válido cubriendo toda la lista de entrada incrementar la variable $depth$ en 1.

5. Si la variable $depth$ es igual a 2, regresar todo el rango de IDs como el pasaje plagiado resultante.
6. Para cada intervalo válido, hacer lo siguiente:
    a. Crear una nueva lista de pares de IDs como ($ID\ en\ D_2$, $ID\ en\ D_1$), donde $ID\ en\ D_1$ pertenece al intervalo válido actual.
    b. Establecer la variable $depth$ a 1.
    c. Recorrer recursivamente el algoritmo, empezando desde el paso 2.

2.2.4 Prasha Shrestha et al.

En [19] dos correspondencias de características son integradas si la distancia entre ellas en ambos documentos es menor a seis palabras. Los autores primero ejecutan la integración de correspondencias usando los n-gramas de palabras auxiliares y luego para los n-gramas de entidades nombradas y los n-gramas de palabras. Luego integran todos los resultados usando las diferentes características, lo que les genera muchos pasajes solapados. Dichos pasajes solapados son tratados de la siguiente forma:

1. Si un pasaje es contenido completamente por otro tanto en el documento sospechoso como en el documento de referencia, eliminan completamente el pasaje más corto.
2. Si existe solapamiento entre dos pasajes en ambos documentos, integran ambos pasajes.
3. Si existe solapamiento en solo uno de los documentos, truncan el pasaje más corto (en términos de caracteres) con el fin de eliminar el solapamiento.

2.2.5 Yurii Palkovskii

En [20] [21] los autores se basan en un algoritmo genérico de agrupamiento basado en la distancia Euclidiana como se muestra en el anexo B; para realizar la integración de correspondencias entre características. Incluyen un conjunto de capas de filtrado en el paso 8 que se encarga de la fusión de los grupos.

Los autores descubren a través de la visualización de los resultados que la distribución exacta de las características comunes dentro de cualquier grupo se localiza en la diagonal del grupo.

2.2.6 Robin Küppers et al.

En [24], debido a que sus características son fragmentos de texto de 250 caracteres no solapados, nunca se encontrará solapamiento en las correspondencias entre características pero si se pueden encontrar correspondencias de características adyacentes. Los autores consideran dos correspondencias adyacentes si la distancia entre ellas es menor a 500 caracteres.

2.2.7 Lee Gillam et al.

En [26] [27] los autores establecen una distancia máxima para unir dos correspondencias entre características en 900 unidades. Como los autores no revelan información más detallada de su sistema no podemos decir que tipo de unidades son.

2.3 Post-procesamiento

Dado un conjunto de pasajes correspondientes, o alineados, estos son filtrados de acuerdo a ciertos criterios. La idea principal de esta fase es abordar los casos que presentan solapamiento y descartar pasajes extremadamente pequeños.

2.3.1 Kong Leilei et al.

En [9] [10] los autores eliminan los pasajes de textos de la fase de integración que no sobrepasan un umbral al aplicarle la similitud de estructura planteada en su fase de selección. Este umbral lo establecen en 0.3.

2.3.2 Rodríguez Torrejón et al.

En [11] [12] se unen las detecciones de la fase anterior en ambos documentos, tal que su distancia es menor o igual a 4000 caracteres.

2.3.3 Šimon Suchomel et al.

En [16] [17] primero eliminan el solapamiento. Si dos intervalos válidos solapados son menores a 300 caracteres, eliminan ambos. De otra forma mantienen la detección más larga en términos de la longitud en el documento sospechoso.

Luego, unen los intervalos válidos en una detección si algunos de los siguientes criterios se cumplen:

- La distancia entre los intervalos contienen al menos 4 características comunes y al menos contiene una característica por cada 10,000 caracteres.
- La distancia es menor a 30,000 caracteres y la suma del tamaño de los intervalos válidos adyacentes es al menos el doble de la distancia entre ellos.
- La distancia es menor a 30,000 caracteres y el número de características comunes por caracter en los intervalos adyacentes no es más de tres veces mayor que el número de características por caracter en el posible intervalo juntado.

Las distancias a que hacen referencia los puntos anteriores, son calculadas como el número de caracteres entre las detecciones en el documento de referencia más el número de caracteres en las detecciones en el documento sospechoso.

2.3.4 Prasha Shrestha et al.

En [19] los autores post-procesan los documentos con la idea que los pasajes vecinos a un pasaje plagiado son muy probables a ser plagiados también. En el documento sospechoso, toman las palabras que se encuentran en las 30 palabras vecinas en un pasaje detectado y obtienen los n-gramas de palabras. Luego comparan dichos n-gramas con los obtenidos de las 30 palabras vecinas del pasaje correspondiente en el documento de referencia.

También, para el caso del método basado en n-gramas de palabras, con el objetivo de aumentar la precisión, eliminan los pasajes menores a $n + \frac{2}{3}n$ detecciones consecutivas.

2.3.5 Yurii Palkovskii

En [20] [21] eliminan los grupos que tienen menos de 26 huellas digitales de las características extraídas y los grupos que tienen una longitud menor a 190 caracteres.

2.3.6 Robin Küppers et al.

En [24] no se aplica ninguna técnica de post-procesamiento.

2.3.7 Lee Gillam et al.

En [26] [27] eliminan los pasajes menores a 50 palabras que tienen una similitud menor a 0.75 usando la medida del coseno.

3 MARCO TEÓRICO

En este capítulo se presentan las bases necesarias para entender el estado del arte presentado en el capítulo 2 y modelo propuesto en el capítulo 4.

3.1 Características comunes en los sistemas de recuperación de información

Existen ciertas características que los sistemas de recuperación de información comparten. Muchas de estas características son usadas en determinado grado dependiendo de los problemas que se quieren resolver. Por ejemplo, en algunos sistemas las palabras auxiliares son eliminadas debido a que no aportan mucha información sobre el contenido de un documento, sin embargo, varios sistemas de detección de plagio hacen uso de ellas porque pueden representar el estilo de un autor.

3.1.1 Índice invertido

La mayoría de los sistemas de recuperación de información tiene como su estructura de dato primaria un *índice invertido*. Un índice invertido es una estructura de datos que enlista para cada palabra en una colección, todos los documentos que la contienen y la frecuencia de ocurrencia en cada documento. Un índice invertido hace fácil buscar coincidencias de una palabra consultada. Solo es cuestión de ir a la parte del índice invertido que corresponde a la palabra consultada y recuperar todos los documentos ahí enlistados [29].

3.1.2 Posición de la información

Una versión más sofisticada del índice invertido también contiene la información de la posición. En lugar de solo enlistar los documentos en los que una palabra ocurre, las posiciones de todas las ocurrencias en un documento también son enlistadas. La posición de una ocurrencia puede ser codificada como el valor desplazado desde el inicio de un documento. Un índice invertido con información de la posición nos permite buscar frases. Por ejemplo, para buscar *"seguro médico"*, se trabaja simultáneamente con las entradas de *"seguro"* y "*médico"* en el índice invertido. Primero, intersectamos los dos conjuntos de forma tal que tenemos documentos donde ambas palabras ocurren. Luego buscamos en la información de la posición y solo nos quedamos con aquellos resultados en los cuales la información de la posición indica que *"médico"* ocurre inmediatamente después de *"seguro"*. Esto es mucho más eficiente que leer y procesar todos los documentos de la colección secuencialmente. Sin embargo la noción de frase vista anteriormente es muy primitiva debido a que solo se pueden buscar frases fijas [29].

3.1.3 Palabras auxiliares

En algunos sistemas de recuperación de información, no todas las palabras son representadas en el índice invertido. Una lista de palabras auxiliares (en inglés *stopwords*) enlista aquellas palabras que son improbablemente útiles para la búsqueda [29].

El conjunto de palabras auxiliares está compuesto por pronombres, artículos, preposiciones o cualquier palabra porque su presencia en un documento no dice nada del documento como tal, su presencia es determinada por las características del lenguaje [30].

3.1.4 Obtención de raíces

Otra característica común de los sistemas de recuperación de información es la obtención de raíces. En recuperación de información, obtención de raíces usualmente hace referencia a una forma simplificada de análisis morfológico consistente en simplemente truncar una palabra. Por ejemplo de las palabras trabajar, trabajando, trabajador y trabajadores se obtiene la raíz trabaj-. Dos algoritmos comunes para la obtención de raíces en el idioma inglés son los de Lovins [31] y Porter [32], los cuales difieren en el algoritmo usado para determinar dónde truncar las palabras. Dos problemas con los algoritmos que truncan las palabras para obtener raíces son que estos combinan palabras semánticamente diferentes y que las raíces truncadas pueden ser ininteligibles para los usuarios [29].

3.2 Medidas de evaluación

Evaluación en recuperación de información hace uso frecuente de las nociones de precisión y exhaustividad (*recall* en inglés), y su uso se ha cruzado a la evaluación de los modelos del procesamiento estadístico de lenguaje natural. Para muchos problemas, tenemos un conjunto objetivo (por ejemplo, documentos relevantes u oraciones en las cuales una palabra tiene cierto sentido) contenido dentro de una colección más grande. Nuestro sistema luego decide por un conjunto seleccionado (documentos que piensa son importantes, u oraciones que cree contienen cierto sentido de una palabra, etc.). Esta situación se muestra e la figura 3.1. Los grupos seleccionado y objetivo pueden ser vistos como variables aleatorias indicadoras, y la distribución conjunta de las dos variables puede ser expresada como una matriz de contingencia (matriz de confusión) de 2x2 [33]:

	Actual	
Sistema	objetivo	NO objetivo
seleccionado	vp	fp
NO seleccionado	fn	vn

(3.1)

El número en cada celda muestra la frecuencia o cuenta del número de elementos en cada región del espacio. Los casos contados como vp (*verdaderos positivos*) y vn (*verdaderos negativos*) son los casos que nuestro sistema obtuvo correctamente. Los casos seleccionados incorrectamente en fp son llamados *falsos positivos*, *falsas aceptaciones* o *errores de tipo II*.

Los casos en fn que no fueron seleccionados son llamados *falsos negativos*, *falsos rechazos* o *errores de tipo I* [33].

La precisión está definida como una medida de la proporción de los elementos seleccionados que el sistema obtuvo correctamente:

$$precisión = \frac{vp}{vp + fp} \quad \textbf{(3.2)}$$

El *recall* (como denominaremos a la exhaustividad por su amplio uso en la literatura) está definida como la proporción de los elementos objetivos que el sistema seleccionó:

$$recall = \frac{vp}{vp + fn} \quad \textbf{(3.3)}$$

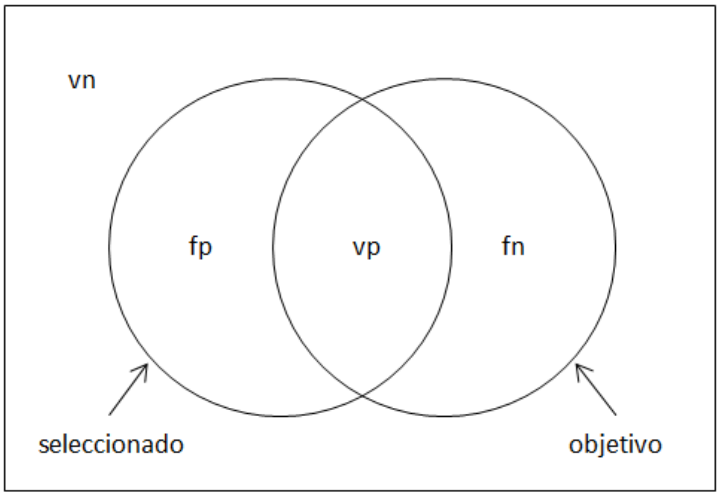


Figura 3.1 Diagrama motivando las medidas de precisión y exhaustividad

En aplicaciones como la recuperación de información, uno generalmente puede sacrificar precisión por *recall* o viceversa (por ejemplo, uno puede seleccionar cada documento en una colección y obtener un *recall* de 100% pero muy baja precisión, etc.) [33].

Por esta razón puede ser conveniente combinar precisión y *recall* en una sola medida de desempeño general. Una forma de hacer esto es a través de la medida F, una variante de la medida E introducida por van Rijsbergen en [34] como medida explícita de efectividad, donde $F = 1 - E$. La medida F se define como:

$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}} \quad \textbf{(3.4)}$$

Donde P es la precisión, R es el *recall* y α es el factor que determina el peso de la precisión y el *recall*. Un valor de $\alpha = 0.5$ es usualmente seleccionado para igual ponderación de P y R. Con este valor de α, la medida F se simplifica a $2PR/(R+P)$ [33].

Otras medidas usadas son la exactitud (*accuracy* en inglés) que reporta el porcentaje de elementos correctos, y el error que reporta el porcentaje de elementos incorrectos. Estas medidas están dadas por:

$$exactitud = \frac{vp + vn}{vp + vn + fp + fn} \quad \textbf{(3.5)}$$

$$error = \frac{fp + fn}{fp + fn + vp + vn} \quad \textbf{(3.6)}$$

Sin embargo, resulta que usualmente estas medidas de exactitud y error no son buenas porque en la mayor parte de los tipos de problemas, los elementos no objetivos y no seleccionados vn son muchos. Esto hace que la exactitud sea muy alta y que el error sea muy bajo para casi cualquier caso.

En la tabla 3.1 se muestran algunos ejemplos que ilustran como la exactitud se diferencia de la medida F (con $\alpha = 0.5$). En la serie (a) se muestra que el valor de F incrementa mientras que el valor de exactitud disminuye. En la serie (b) se muestra un valor igual para la exactitud mientras la medida F incrementa.

Tabla 3.1 Comportamiento de las medidas de evaluación

	vp	fp	fn	vn	Prec	Rec	F (α=0.5)	Exactitud	Error
(a)	25	0	125	99,850	1.000	0.167	0.286	0.9988	0.0013
	50	100	100	99,750	0.333	0.333	0.333	0.9980	0.0020
	75	150	75	99,700	0.333	0.500	0.400	0.9978	0.0023
	125	225	25	99,625	0.357	0.833	0.500	0.9975	0.0025
	150	275	0	99,575	0.353	1.000	0.522	0.9973	0.0028
(b)	50	0	100	99,850	1.000	0.333	0.500	0.9990	0.0010
	75	25	75	99,825	0.750	0.500	0.600	0.9990	0.0010
	100	50	50	99,800	0.667	0.667	0.667	0.9990	0.0010
	150	100	0	99,750	0.600	1.000	0.750	0.9990	0.0010

En este contexto, el uso de la precisión y el *recall* tiene tres ventajas [33]:

- Los valores de exactitud no son muy sensibles a los números de vp, fp y fn, mientras precisión y *recall* si lo son. Uno puede obtener valores extremadamente altos de exactitud seleccionando nada.
- Otra ventaja similar es que la medida *F* prefere resultados con mayor cantidad de verdaderos positivos, mientras que la exactitud solo es sensible al número de errores. Esto refleja nuestra intuición de: Estamos interesados en encontrar cosas, incluso a costa de regresar algo de basura.
- Al usar precisión y *recall*, uno puede dar un costo diferente a los elementos objetivos no seleccionados contra la basura seleccionada.

Una medida menos utilizada es la tasa de fallo (*fallout* en inglés) que representa la proporción de los elementos no objetivo que fueron erróneamente seleccionados.

$$fallout = \frac{fp}{fp + vn} \quad \textbf{(3.7)}$$

3.3 Modelo de espacio vectorial

El modelo de espacio vectorial es uno de los modelos más usados en la tarea de recuperación de información, principalmente por su simplicidad conceptual y el atractivo de usar la proximidad espacial como proximidad semántica. Los documentos y las consultas son representados en un espacio de alta dimensionalidad, en el cual cada dimensión del espacio corresponde a una palabra en la colección de documentos. Los documentos más relevantes para una consulta se espera sean los representados por los vectores más cercanos a dicha consulta, esto es, documentos que usan palabras similares a la consulta. En lugar de considerar la magnitud de los vectores, la cercanía se calcula usualmente por solo mirar a los ángulos y elegir los documentos que tienen los ángulos más pequeños con el vector de la consulta [35].

En la figura 3.2 se observa un espacio vectorial de dos dimensiones, correspondientes a las palabras *carro y seguro*. Las entidades representadas en el espacio son la consulta q representada por el vector (0.71, 0.71), y tres documentos d_1, d_2 y d_3 con las siguientes coordenadas: (0.13, 0.99), (0.8, 0.6), y (0.99, 0.13). Las coordenadas son derivadas del conteo de las ocurrencias de las palabras. Por ejemplo, *seguro* puede tener solo una ocurrencia en d_1 mientras hay muchas ocurrencias de *carro* [35].

En la figura, el documento d_2 tiene el menor ángulo con respecto a q por lo que será el documento con mayor rango en respuesta a la consulta *"seguro carro"* [35].

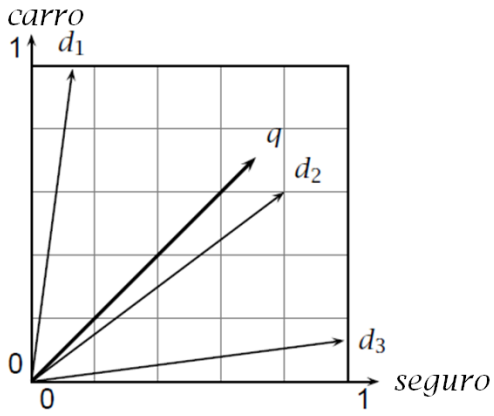


Figura 3.2 Espacio vectorial con dos dimensiones

3.3.1 Medidas de espacio vectorial

Una gran cantidad de medidas de similitud semántica se conceptualizan mejor como medidas de similitud entre vectores. Las figuras 3.3, 3.4 y 3.5 muestran ejemplos de espacios multidimensionales [36].

La matriz en la figura 3.3 representa palabras como vectores en un *espacio de documento*. El valor a_{ij} contiene el número de veces que la palabra *j* ocurre en el documento *i*. Las palabras se consideran similares en la medida que ocurran en los mismos documentos. En el *espacio de documento*, cosmonauta y astronauta no son similares dado que no comparten documentos; mientras que camión y carro son similares debido a que comparten documentos: ambas coocurren en d4 [36].

La matriz en la figura 3.4 representa palabras como vectores en un *espacio de palabras*. El valor b_{ij} contiene el número de veces que la palabra *j* coocurre con la palabra *i*. La concurrencia se puede definir respecto a documentos, párrafos u otra unidad. Las palabras son similares en la medida que ellas coocurran con las mismas palabras. Aquí, *cosmonauta* y *astronauta* son más similares que anteriormente dado que ambos coocurren con *luna*. La concurrencia en la figura 3.4 se ha definido respecto a los documentos en la figura 3.3 [36].

La matriz en la figura 3.5 representa los sustantivos (interpretados como cabeceras de sintagmas nominales) como vectores en un *espacio de modificadores*. El valor c_{ij} contiene el número de veces que una cabecera *j* es afectada por un modificador *i*. Las cabeceras son similares en la medida que ellas son afectadas por los mismos modificadores. Otra vez, *cosmonauta* y *astronauta* son similares. Pero curiosamente *luna* no es similar a *cosmonauta* y *astronauta* aquí, en contraste al *espacio de documentos* en la figura 3.3 y al *espacio de palabras* en la figura 3.4. Este contraste demuestra que distintos espacios obtienen diferentes tipos de similitud semántica. El tipo de información que concurre en los espacios de documentos y palabras captura la similitud de tópicos, es decir, palabras pertenecientes al mismo tópico. Mientras tanto, la información cabecera-modificador es más específica [36].

Las tres matrices también tienen una interpretación interesante si se observa la similitud entre filas en lugar de la similitud entre columnas. Al observar las matrices de esta forma, *A* define la similitud entre documentos. Esta es una forma estándar de definir la similitud entre documentos y entre documentos y consultas en recuperación de información. La matriz *C* define la similitud entre modificadores cuando se usa su transpuesta. Por ejemplo, *rojo* y *viejo* son similares (coocurren con *carro* y *camión*), sugiriendo que son usados para modificar sustantivos del mismo tipo. La matriz *B* es simétrica por lo que es lo mismo observar la similitud en filas como en columnas [36].

	cosmonauta	astronauta	luna	carro	camión
d1	1	0	1	1	0
d2	0	1	1	0	0
d3	1	0	0	0	0
d4	0	0	0	1	1
d5	0	0	0	1	0
d6	0	0	0	0	1

Figura 3.3 Una matriz *A* de documento-por-palabra

	cosmonauta	astronauta	luna	carro	camión
cosmonauta	2	0	1	1	0
astronauta	0	1	1	0	0
luna	1	1	2	1	0
carro	1	0	1	3	1
camión	0	0	0	1	2

Figura 3.4 Una matriz *B* de palabra-por-palabra

	cosmonauta	astronauta	luna	carro	camión
Soviético	1	0	0	1	1
Americano	0	1	0	1	1
caminata-espacial	1	1	0	0	0
rojo	0	0	0	1	1
llena	0	0	1	0	0
viejo	0	0	0	1	1

Figura 3.5 Una matriz *C* de modificador-por-cabecera

Hasta ahora la similitud entre vectores ha sido observada con una noción intuitiva. En la tabla 3.2 se definen varias medidas que han sido propuestas para hacer esta noción precisa. Primero, solo se consideran vectores binarios, esto es, vectores con entradas que son 0 o 1 [36].

Tabla 3.2 Medidas de similitud para vectores binarios

Medida de similitud	Definición
coeficiente de coincidencia	$\lvert X \cap Y \rvert$
coeficiente de Dice	$\frac{2\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert}$
coeficiente de Jaccard	$\frac{\lvert X \cap Y \rvert}{\lvert X \cup Y \rvert}$
coeficiente de solapamiento	$\frac{\lvert X \cap Y \rvert}{\min(\lvert X \rvert, \lvert Y \rvert)}$
coseno	$\frac{\lvert X \cap Y \rvert}{\sqrt{\lvert X \rvert \times \lvert Y \rvert}}$

La primera medida de similitud, el coeficiente de coincidencia, simplemente cuenta el número de dimensiones en las cuales ambos vectores son distintos a cero. En contraste a las otras medidas, ésta no toma en cuenta el tamaño de los vectores o el total de entradas distintas de cero en cada uno [36].

El coeficiente de Dice normaliza al dividir por el total de entradas distintas de cero. Se multiplica por 2 de forma tal que se obtiene una medida que varía entre 0.0 y 1.0 donde 1.0 indica vectores idénticos [36].

El coeficiente de Jaccard penaliza un número pequeño de entradas compartidas (como una proporción de todas las entradas distintas de cero) más que el coeficiente de Dice. Ambas medidas varían entre 0.0 (sin solapamiento) y 1.0 (perfectamente solapados), pero el coeficiente de Jaccard da valores más bajos a casos de bajo solapamiento. Por ejemplo, dos vectores con diez entradas distintas de cero y una entrada común obtienen un valor de Dice de $2 \times 1/(10 + 10) = 0.1$ y un valor de Jaccard de $1/(10 + 10 - 1) \approx 0.05$ [36].

El coeficiente de solapamiento se comporta como una medida de inclusión. Este tiene un valor de 1.0 si cada dimensión con un valor distinto de cero para el primer vector también es distinto de cero para el segundo vector y viceversa, en otras palabras, si $X \subseteq Y$ o $Y \subseteq X$ [36].

La medida coseno es idéntica al coeficiente de Dice para vectores con el mismo número de entradas distintas de cero, pero penaliza menos en casos donde el número de entradas distintas de cero son muy diferentes. Por ejemplo, si se compara un vector con una entrada distinta de cero y otro vector con 1000 entradas distintas de cero y si existe una entrada compartida, entonces se obtiene un coeficiente de Dice de $2 \times 1/(1 + 1000) \approx 0.002$ y un valor de coseno de $1/\sqrt{1000 \times 1} \approx 0.03$. Esta propiedad del coseno es importante en el procesamiento estadístico de

lenguaje natural dado que frecuentemente se comparan palabras u objetos que tienen diferentes cantidades de datos, pero que no queremos decir que son diferentes solo por eso [36].

Hasta ahora las medidas expuestas anteriormente trabajan sobre vectores binarios, pero los vectores binarios solo tienen una pequeña porción de información en cada dimensión. Una representación más poderosa para objetos lingüísticos es un espacio vectorial de valores reales [36].

Un vector de valores reales $\vec{x}$ de dimensionalidad n es una secuencia de n números reales, donde x_i denota el i^{th} componente de $\vec{x}$ (su valor en la dimensión *i*). Otra forma de escribir los vectores es $\vec{x} \in \mathbb{R}^n$ de forma tal que $\mathbb{R}^n$ representa un espacio vectorial de vectores de valores reales con n dimensiones. En un espacio vectorial Euclidiano, la longitud de un vector está definida como [36]:

$$|\vec{x}| = \sqrt{\sum_{i=1}^{n} x_i^2} \qquad \textbf{(3.8)}$$

Mientras tanto, el producto punto entre dos vectores está definido como:

$$\vec{\boldsymbol{x}} \cdot \vec{\boldsymbol{y}} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x_i}\, \boldsymbol{y_i} \qquad \textbf{(3.9)}$$

De las medidas de similitud introducidas anteriormente, el coseno es la más importante para vectores de valores reales. El coseno mide el coseno del ángulo entre dos vectores. Este varía desde 1.0 ($\cos(0°) = 1.0$) para vectores apuntando en la misma dirección a través de 0.0 para vectores ortogonales ($\cos(90°) = 0.0$) hasta -1.0 para vectores apuntando en direcciones opuestas ($\cos(180°) = -1.0$) [36].

Para el caso general de dos vectores de *n* dimensiones en u espacio de valores reales, la medida del coseno puede ser calculada como:

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \qquad \textbf{(3.10)}$$

Esta definición resalta otra interpretación del coseno, la interpretación como *el coeficiente de correlación normalizado*. Se calcula que tan bien x_i y y_i se correlacionan y luego se dividen por la longitud (Euclidiana) de ambos vectores para escalar por las magnitudes cada valor de x_i y y_i [36].

Se dice que un vector está normalizado si su longitud es unitaria de acuerdo a la norma Euclidiana:

$$|\vec{x}| = \sum_{i=1}^{n} x_i^2 = 1 \quad \textbf{(3.11)}$$

Para vectores normalizados, el coseno es simplemente el producto punto:

$$\cos(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y} \quad \textbf{(3.12)}$$

La distancia Euclidiana entre dos vectores mide que tan separados están en un espacio vectorial:

$$|\vec{x} - \vec{y}| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad \textbf{(3.13)}$$

Una propiedad interesante del coseno es que, si se aplica a vectores normalizados, este dará la misma clasificación de similitudes que la distancia Euclidiana. Esto es, si solo queremos saber cuál de dos objetos está más cercano a un tercero, entonces el coseno y la distancia Euclidiana darán la misma respuesta para vectores normalizados. La siguiente expresión muestra por qué la clasificación de acuerdo al coseno y la distancia Euclidiana resulta ser la misma [36]:

$$\begin{aligned} (|\vec{x} - \vec{y}|)^2 &= \sum_{i=1}^{n} (x_i - y_i)^2 \\ &= \sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} y_i^2 \\ &= 1 - 2\sum_{i=1}^{n} x_i y_i + 1 \\ &= 2(1 - \vec{x} \cdot \vec{y}) \end{aligned} \quad \textbf{(3.14)}$$

3.3.2 Ponderación de términos

La pregunta ahora es cómo ponderar las palabras en un modelo de espacio vectorial. Una posibilidad es usar el número de ocurrencias de una palabra en un documento como el peso de un término, pero existen métodos más efectivos de ponderación de términos. La información básica usada en la ponderación de términos es la frecuencia de término (*term frequency tf*), frecuencia de documento (*document frequency df*) y algunas veces frecuencia de colección (*collection frequency cf*), como se define en la tabla 3.3. Nótese que $df_t \leq cf_t$ y que $\sum_j tf_{i,j} = cf_i$. Es importante notar que la frecuencia de documento y la frecuencia de colección solo pueden ser usadas si existe una colección. Esta suposición no siempre es verdad, por ejemplo si las colecciones son creadas dinámicamente seleccionando varias bases de datos de un conjunto muy

grande (como puede ser el caso en alguno de los servicios de información en-línea), y luego unirlas en una colección temporal [35].

Tabla 3.3 Tres medidas que son usadas frecuentemente para ponderar términos en la recuperación de información

Medida	Símbolo	Definición
frecuencia de término	$tf_{i,j}$	Número de ocurrencias de w_i en d_j
frecuencia de documento	df_i	Número de documentos en la colección en donde ocurre w_i
frecuencia de colección	cf_i	Número total de ocurrencias de w_i en la colección

La información que se captura por la frecuencia de término es que tan saliente es una palabra en un documento dado. Mientras mayor frecuencia de término es más probable que la palabra sea una buena descripción del contenido del documento. La frecuencia de término usualmente es escalada por alguna función como $f(tf) = \sqrt{tf}$ o $f(tf) = 1 + \log(tf)\,, tf > 0$, porque a mayor ocurrencia de una palabra indica mayor importancia, pero no tanta importancia como la cantidad sin escalar sugiere. Por ejemplo, $\sqrt{3}$ o $1 + \log 3$ refleja mejor la importancia de una palabra con tres ocurrencias que la cantidad 3 por si sola. El documento es de alguna manera más importante que un documento con una ocurrencia, pero no tres veces más importante [35].

La segunda medida, la frecuencia de documento, puede ser interpretada como un indicador de contenido informativo. Una palabra enfocada semánticamente usualmente ocurre varias veces en un documento si la palabra no ocurre en todos los documentos. En cambio, una palabra no enfocada semánticamente se esparce homogéneamente en todos los documentos. Un ejemplo basado en un corpus de artículos de *The New York Times* es las palabras *aseguradora* e *intentar* en la tabla 3.4. Las dos palabras tiene aproximadamente la misma frecuencia de colección, es decir, el total de ocurrencias en la colección de documentos. Pero *aseguradora* ocurre solo la mitad de la cantidad de documentos que ocurre *intentar*. Esto es porque la palabra *intentar* puede ser usada cuando se habla de prácticamente cualquier tópico debido a uno puede *intentar* hacer algo en cualquier contexto. En contraste, *aseguradora* se refiere a un concepto estrechamente definido que solo es relevante a un pequeño conjunto de tópicos. Otra propiedad de las palabras enfocadas semánticamente es, si aparecen una vez en un documento, a menudo ocurren muchas veces. *Aseguradora* ocurre alrededor de tres veces por documento, tomando un promedio sobre documentos donde aparece al menos una vez. Esto es simplemente debido al hecho que la mayor parte de los artículos acerca de seguros médicos, seguros de autos o tópicos similares harán referencias múltiples veces al concepto aseguradora [35].

Tabla 3.4 Frecuencia de término y documento de dos palabras en un corpus de ejemplo

Palabra	Frecuencia de colección	Frecuencia de documento
aseguradora	10440	3997
intentar	10422	8760

Una forma de combinar la frecuencia de término $tf_{i,j}$ y la frecuencia de documento df_i de una palabra en un solo peso es de la siguiente forma:

$$peso(i,j) = \begin{cases} \left(1 + \log\left(tf_{i,j}\right)\right) \log \frac{N}{df_i}, & si\ tf_{i,j} \geq 1 \\ 0, & si\ tf_{i,j} = 0 \end{cases} \quad \textbf{(3.15)}$$

Donde N es el número total de documentos. La primera parte aplica para palabras que ocurren en un documento, mientras que para palabras que no aparecen ($tf_{i,j} = 0$), se asigna $peso(i,j) = 0$ [35].

La frecuencia de documento también es escalada logarítmicamente. La fórmula $\log \frac{N}{df_i} = \log N - \log df_i$ da el mayor peso a las palabras que ocurren en un documento ($\log N - \log df_i = \log N - \log 1 = \log N$). Una palabra que ocurra en todos los documentos obtendrá un peso de cero ($\log N - \log df_i = \log N - \log N = 0$) [35].

Esta forma de ponderar la frecuencia de documento es llamada a menudo frecuencia de inversa documento (*inverse document frequency*) o ponderación idf. Más generalmente, el esquema de ponderación en (3.8) es un ejemplo de una familia más amplia de esquemas de ponderación llamadas $tf.idf$. Cada uno de estos esquemas puede ser caracterizado por su ponderación de frecuencia de término, su ponderación de frecuencia de documento y su normalización. En la tabla 3.5 se muestran diferentes algunas posibilidades de ponderación, con una letra como código, que combinadas pueden generar un esquema de $tf.idf$. En (3.8) el esquema usado sería "ltn" [35].

Tabla 3.5 Componentes de algunos esquemas de ponderación tf-idf

Frecuencia de término		Frecuencia de documento		Normalización
n (natural)	$tf_{t,d}$	n (ninguna)	df_t	n (sin normalización)
l (logarítmica)	$1 + \log\left(tf_{t,d}\right)$	t	$\log \frac{N}{df_t}$	c (coseno)
a (aumentada)	$0.5 + \frac{0.5 \times tf_{t,d}}{\max_t\left(tf_{t,d}\right)}$			$\frac{1}{\sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}}$

La familia de esquemas de ponderación mostradas en la tabla 3.5 a veces es criticada como 'ad-hoc' porque no se derivan directamente de algún modelo matemático de distribuciones de términos o relevancia. Sin embargo, estos esquemas son efectivos en la práctica y funcionan robustamente en una amplia gama de aplicaciones [35].

3.3.3 N-gramas tradicionales como características en el modelo de espacio vectorial

Al utilizar solo palabras como características se pierde la información sobre las relaciones sintácticas entre las palabras. Las palabras se convierten en lo que se llama "bolsa de palabras" (en inglés, *bag of words*). Para muchas tareas esta pérdida de información es aceptable [30].

Una forma de caracterizar los textos, además de las palabras, son los n-gramas (tradicionales). Los n-gramas tradicionales son secuencias de elementos tal como aparecen en un documento. En este caso la letra N indica cuántos elementos debemos tomar en cuenta, es decir, la longitud de la secuencia o de n-grama. Por ejemplo, existen bigramas (2-gramas), trigramas (3-gramas), 4-gramas, 5-gramas, etc. De esa manera, si vamos a hablar de unigramas, es decir, de n-gramas construidos de un solo elemento, es lo mismo que hablar de palabras [30].

Otro grado de libertad es el tipo de elementos que constituyen los n-gramas. Pueden ser lemas o palabras, pero también pueden ser etiquetas de clases gramaticales (en inglés, *POS tags, part of speech tags*), tales como sustantivos, verbos, etc. Posiblemente las etiquetas pueden incluir características gramaticales más detalladas, por ejemplo, una etiqueta como VIP1S, podría significar "verbo, indicativo, presente, primera persona, singular". Podemos construir n-gramas utilizando ese tipo de etiquetas [30].

En los últimos años en algunas tareas se utilizan mucho los n-gramas de caracteres, es decir secuencias de caracteres tomadas de un texto. Curiosamente, para algunas tareas, como atribución de autoría, los n-gramas de caracteres dan buenos resultados. Su interpretación lingüística no está clara, y sigue siendo objeto de estudios [30].

Por ejemplo de la frase *"Juan lee un libro interesante"* podemos obtener los siguientes n-gramas:

2-gramas	*<Juan lee> , <lee un> , <un libro> , <libro interesante>*
3-gramas	*<Juan lee un> , <lee un libro> , <un libro interesante>*
4-gramas	*<Juan lee un libro> , <lee un libro interesante>*

También podemos sustituir cada palabra por su lema o por su clase gramatical y construir los n-gramas correspondientes.

Al utilizar los n-gramas como características, similar que en caso de las palabras (unigramas), usamos los esquemas de ponderación tf-idf. Nótese que las frecuencias de n-gramas usualmente son mucho más bajas que de las palabras, es decir los n-gramas aparecen mucho menos en los textos. Es lógico porque realmente estamos observando la aparición al mismo tiempo de las secuencias de dos o más palabras seguidas, lo que es un evento mucho menos probable que de una palabra sola.

3.3.4 Salta-k n-gramas como características en el modelo de espacio vectorial

Además de los n-gramas tradicionales existen otros tipos de construcción de n-gramas. Uno de ellos es la construcción de salta-k n-gramas (en inglés *skip-grams o k-skip-n-gram*).

Los salta-k n-gramas son una técnica muy usada en el campo del procesamiento de voz, en el cual se forman n-gramas (bigramas, trigramas, etc.) pero además de permitir secuencias de palabras (o elementos) se permiten "saltar" elementos. Mientras inicialmente se aplicó a fonemas en el habla humana, la misma técnica puede ser aplicada a palabras. Por ejemplo en la oración "Él conoció al valiente señor" hay tres trigramas: "Él conoció al", "conoció al valiente", "al valiente señor". Sin embargo uno podría argumentar que un trigrama igualmente importante sería "conoció al señor". Al usar salta-k n-gramas se permite omitir la palabra "valiente", obteniendo así este trigrama [37].

La definición formal de este tipo de construcción de n-gramas dada una oración $w_1 \dots w_m$ es el conjunto:

$$\left\{ w_{i_1}, w_{i_2}, \dots, w_{i_n} \middle| \sum_{j=1}^{n} i_j - i_{j-1} < k \right\} \quad \textbf{(3.16)}$$

Los salta-k n-gramas (o salta-gramas) reportados para un cierto valor de k permiten un total de k o menos saltos para construir el n-grama. Así, el resultado de un "salta-4 n-grama" incluye 4 saltos, 3 saltos, 2 saltos, 1 salto y 0 saltos (n-gramas tradicionales formados por elementos adyacentes) [37].

Tabla 3.6 Ejemplo de comparación entre n-gramas y salta-k n-gramas

2-gramas	salta-2 2-gramas	
insurgentes muertos	insurgentes muertos	muertos enfrentamientos
muertos en	insurgentes en	en continuos
en continuos	insurgentes continuos	en enfrentamientos
continuos enfrentamientos	muertos en	continuos enfrentamientos
	muertos continuos	

3-gramas	salta-2 3-gramas	
insurgentes muertos en	insurgentes muertos en	insurgentes continuos enfrentamientos
muertos en continuos	insurgentes muertos continuos	muertos en continuos
en continuos enfrentamientos	insurgentes muertos enfrentamientos	muertos en enfrentamientos
	insurgentes en continuos	muertos continuos enfrentamientos
	insurgentes en enfrentamientos	en enfrentamientos continuos

En la tabla 3.6 se muestran los salta-2 2-gramas y salta-2 3-gramas en comparación con los tradicionales 2-gramas y 3-gramas para la oración *"Insurgentes muertos en continuos enfrentamientos"* [37].

4 METODOLOGÍA

En este trabajo de tesis, el problema de la detección automática de plagio consiste en la obtención de correspondencias entre pasajes de textos de máxima longitud en dos documentos. Un caso de plagio va a estar dado por un pasaje en el documento sospechoso y otro correspondiente en el documento de referencia. El método propuesto es representado en cinco módulos principales cómo se muestra en la figura 4.1. El módulo de Pre-procesamiento es el módulo inicial mientras que el Generador de XML convierte los resultados al formato utilizado para la evaluación del rendimiento de los métodos. La línea discontinua en la figura representa que los módulos de Integración y Post-procesamiento pueden ser o no tomados en el método, esto con el objetivo de mostrar las mejoras obtenidas por los mismos.

En las siguientes secciones hacemos referencia también a algunos parámetros usados en nuestro modelo. Estos parámetros son:

- $RemSw$: Su valor define que tratamiento se le va a dar a las palabras auxiliares ($RemSw = [0, 1, 2]$). Ver sección 4.1.2.
- $MinSentLength$: Longitud mínima que debe tener una oración en número de palabras para no ser considerada pequeña ($MinSentLength \geq 1$). Ver sección 4.1.4.
- $MinSize$: Número mínimo de pares necesarios en un caso de plagio ($MinSize \geq 1$). Ver secciones 4.3.1 y 4.3.2.
- $MaxGap$: Número de oraciones intermedias que pueden existir entre dos oraciones consideradas adyacentes ($MaxGap \geq 0$). Ver sección 4.3.
- $t1, t2\ y\ t3$: Umbrales para las medidas de similitud ($0 \leq [t1, t2, t3] \leq 1$). Ver secciones 4.2.2, 4.2.3 y 4.4.1.
- $MinPlagLength$: Número mínimo de caracteres de cada parte de un caso de plagio ($MinPlagLength \geq 1$). Ver sección 4.4.3.

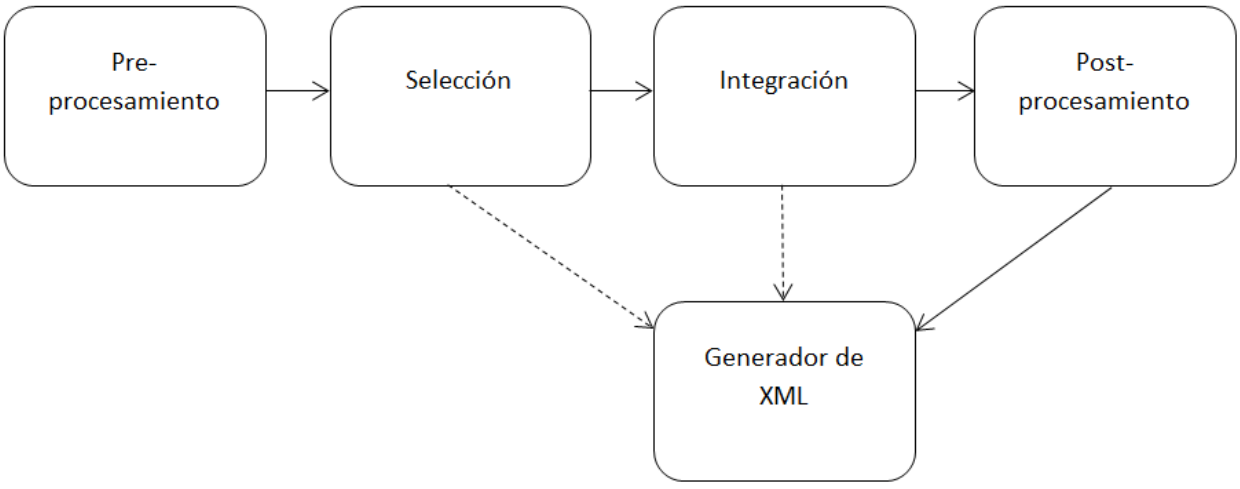


Figura 4.1 Método propuesto dividido en 5 módulos

Cada uno de los módulos anteriores consta de varios procesos. Dichos procesos se detallan en la implementación del método propuesto en la figura 4.2.

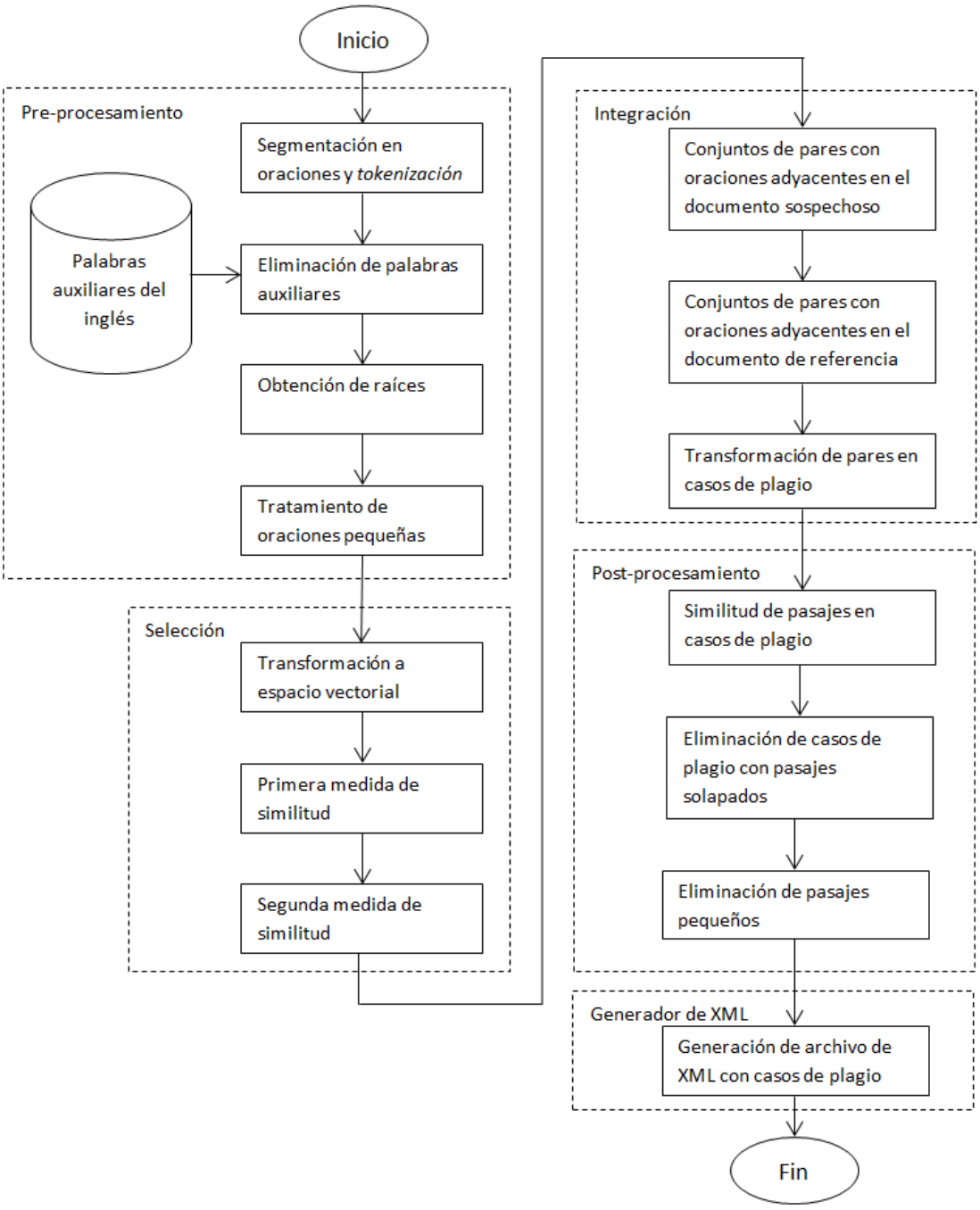


Figura 4.2 Implementación del método propuesto

4.1 Módulo de pre-procesamiento

Para cada documento a comparar se realizan cuatro tareas principales: segmentación por oraciones y *tokenización*, eliminación de palabras auxiliares, obtención de raíces y tratamiento de

oraciones pequeñas. Se asume que el corpus de entrada está formado por textos planos sin formato. También se realiza un filtrado de caracteres donde se eliminan los caracteres que no son letras, números o signos de puntación.

4.1.1 Segmentación en oraciones y *tokenización*

En el método propuesto segmentamos los documentos en oraciones ya que estas son la unidad mínima que representa una idea y consideramos que un caso de plagio va a estar compuesto por una o varias ideas. De cada una de las oraciones calculamos dos valores importantes:

1) Símbolo de inicio: Posición del primer símbolo de la oración dentro del texto.
2) Tamaño: Cantidad de caracteres de la oración.

La segmentación la realizamos utilizando el método propuesto en [38], pre-entrenado en el corpus de Penn Treebank (LDC) e implementado en la plataforma NLTK para Python.

Después de haber segmentado los documentos en oraciones es necesario dividirlas en palabras, proceso conocido como *tokenización*. Para esto usamos el método de Treebank basado en expresiones regulares que viene implementado en la plataforma NLTK para Python. Para nuestro modelo solo tomamos los *tokens* que empiezan con una letra o número.

4.1.2 Eliminación de palabras auxiliares

Las palabras auxiliares son aquellas que carecen de significado como pronombres, preposiciones, artículos, etc. Por ejemplo, en la tabla 4.1 se muestran las palabras auxiliares del idioma inglés contenidas en el paquete NLTK para Python.

Tabla 4.1 Lista de las palabras auxiliares en el paquete de NLTK para el inglés

i	*their*	*doing*	*below*	*more*
me	*theirs*	*a*	*to*	*most*
my	*themselves*	*an*	*from*	*other*
myself	*what*	*the*	*up*	*some*
we	*which*	*and*	*down*	*such*
our	*who*	*but*	*in*	*no*
ours	*whom*	*if*	*out*	*nor*
ourselves	*this*	*or*	*on*	*not*
you	*that*	*because*	*off*	*only*
your	*these*	*as*	*over*	*own*
yours	*those*	*until*	*under*	*same*
yourself	*am*	*while*	*again*	*so*
yourselves	*is*	*of*	*further*	*than*
he	*are*	*at*	*then*	*too*
him	*was*	*by*	*once*	*very*
his	*were*	*for*	*here*	*'s*
himself	*be*	*with*	*there*	*n't*

she	*been*	*about*	*when*	*can*
her	*being*	*against*	*where*	*will*
hers	*have*	*between*	*why*	*just*
herself	*has*	*into*	*how*	*don*
it	*had*	*through*	*all*	*should*
its	*having*	*during*	*any*	*now*
itself	*do*	*before*	*both*	
they	*does*	*after*	*each*	
them	*did*	*above*	*few*	

En esta fase se experimenta con tres opciones:

1) No se elimina ninguna palabra auxiliar ($RemSw = 0$).
2) Se eliminan las 50 palabras auxiliares más comunes para el inglés reportadas por Stamatatos en [18] ($RemSw = 2$).
3) Se eliminan las 127 palabras auxiliares del idioma inglés contenidas en el corpus del paquete NLTK para Python ($RemSw = 1$).

4.1.3 Obtención de raíces

La raíz es el conjunto de fonemas mínimo e irreductible que comparten las palabras de una misma familia [1]. Por ejemplo *am- en amado, amable, amigo, amor, etc*.

En lingüística computacional, sin embargo, los mejores algoritmos se basan en la eliminación de los afijos y el resultado no necesariamente es la raíz. En este trabajo de tesis se utiliza el algoritmo de Porter para el idioma inglés, el cual funciona mediante el tratamiento de sufijos complejos como compuestos formados por sufijos simples, y la eliminación de los sufijos simples en una serie de pasos. En cada paso se realiza la eliminación del sufijo dependiendo de la forma de la raíz restante, que usualmente involucra una medida de la longitud de sus sílabas [32].

4.1.4 Tratamiento de oraciones pequeñas

En esta fase lidiamos con el problema de oraciones muy pequeñas que pueden afectar en gran medida los resultados debido a la división de un pasaje en varios más pequeños. Su funcionamiento consiste en adjuntar las oraciones con menos palabras que el parámetro $MinSentLength$ a la siguiente oración. Este procedimiento se aplica de la siguiente forma:

$$vect(i+1) \leftarrow vect(i+1) + vect(i), \;\; si \sum_{j=1}^{n} vect_j(i) < MinSentLength \qquad (4.1)$$

Donde $vect$ representa el vector de una oración de n dimensiones. El valor de n está dado por el tamaño del diccionario del corpus. Los valores de las dimensiones son las frecuencias de cada palabra del diccionario en una oración.

4.2 Módulo de selección

El objetivo de este módulo es encontrar correspondencias entre las oraciones del documento sospechoso y el documento de referencia usando un modelo de espacio vectorial. La similitud entre cada vector se calcula en dos pasos como se detalla en las siguientes fases. Para esto usamos tres medidas de similitud:

- Coseno del ángulo entre los vectores.
- Coeficiente de Dice.
- Similitud de estructura propuesta en [9].

4.2.1 Transformación a espacio vectorial

Cada oración es tratada como una bolsa de palabras y representada como un vector en un espacio vectorial donde sus dimensiones son las palabras del vocabulario usado en ambos documentos. Se experimenta con tres valores o pesos de las dimensiones de cada vector:

1) Frecuencia de término sin normalización ($tf \cdot idf = 0$)

$$tf(t,d) = f(t,d) \quad \textbf{(4.2)}$$

Donde $f(t,d)$ es el número de veces que aparece el término t en documento d.

2) Frecuencia de término con normalización ($tf \cdot idf = 1$)

$$tf_{norm}(t,d) = \frac{f(t,d)}{\max(\{f(w,d): w \in d\})} \quad \textbf{(4.3)}$$

Donde $\max(f(w,d): w \in d)$: Frecuencia del término con mayor frecuencia

3) Frecuencia de término – frecuencia inversa de documento ($tf.idf = 2\ y\ 3$ dependiendo de la combinación con (1) o (2))

$$idf(t,D) = \log \frac{|D|}{1 + |\{d \in D: t \in d\}|} \quad \textbf{(4.4)}$$

$$tf \cdot idf(t,d,D) = tf_{norm}(t,d) \times idf(t,D) \quad \textbf{(4.5)}$$

Donde $|D|$ es el número de documentos en la colección, y $|\{d \in D: t \in d\}|$ es el número de documentos donde aparece el término t.

Es importante mencionar que para nuestro modelo los documentos a los que se hace referencia en las fórmulas anteriores son las oraciones y la colección es la unión de todas las oraciones del par de documentos a comparar.

4.2.2 Primera medida de similitud

Ya que hemos pasado las oraciones de ambos documentos a un espacio vectorial común procedemos a calcular la similitud entre cada oración del par de documentos $Sim_1(S,R)$. Para esto usamos la medida del coseno dada por la siguiente ecuación:

$$Sim_1(S,R) = CM(S,R) = \cos\theta = \frac{S \cdot R}{|S||R|} \quad \textbf{(4.6)}$$

Donde θ es el ángulo entre los vectores S y R. A partir de los valores calculados tomamos los pares de oraciones cuya similitud sobrepasan un umbral t1.

$$Sim_1(S,R) > t1 \quad \textbf{(4.7)}$$

4.2.3 Segunda medida de similitud

Después de obtener pares de oraciones como posibles casos de plagio a través de la primera medida de similitud procedemos a calcular la segunda medida de similitud $Sim_2(S,R)$ a dichos pares de oraciones usando alguna de las siguientes medidas: el coeficiente de Dice o la similitud de estructura. El cálculo se realiza de la siguiente forma:

Coeficiente de Dice:

$$Sim_2(S,R) = DC(S,R) = \frac{2(S_b \cdot R_b)}{|S_b|^2 + |R_b|^2} \quad \textbf{(4.8)}$$

Donde $S_b \; y \; R_b$ son los vectores binarios de $S \; y \; R$ respectivamente.

Similitud de estructura:

$$Sim_2(S,R) = TS(S,R) = \frac{2 * \sum_{k=1}^{n} \min(S_k, R_k)}{\sum_{k=1}^{n} S_k + \sum_{k=1}^{n} R_k} \quad \textbf{(4.9)}$$

De los valores calculados tomamos los pares de oraciones que su similitud sobrepasa un umbral t2.

$$Sim_2(S,R) > t2 \quad \textbf{(4.10)}$$

4.3 Módulo de integración de pasajes

Una de las medidas cardinales en el marco de evaluación utilizado, además de la precisión y la exhaustividad (*recall*), es la granularidad. Medida que penaliza a los sistemas que detectan un caso de plagio (figura 4.3 (a)) como varios (figura 4.3 (b)). Esta medida se explicará con más detalle en el capítulo de resultados.

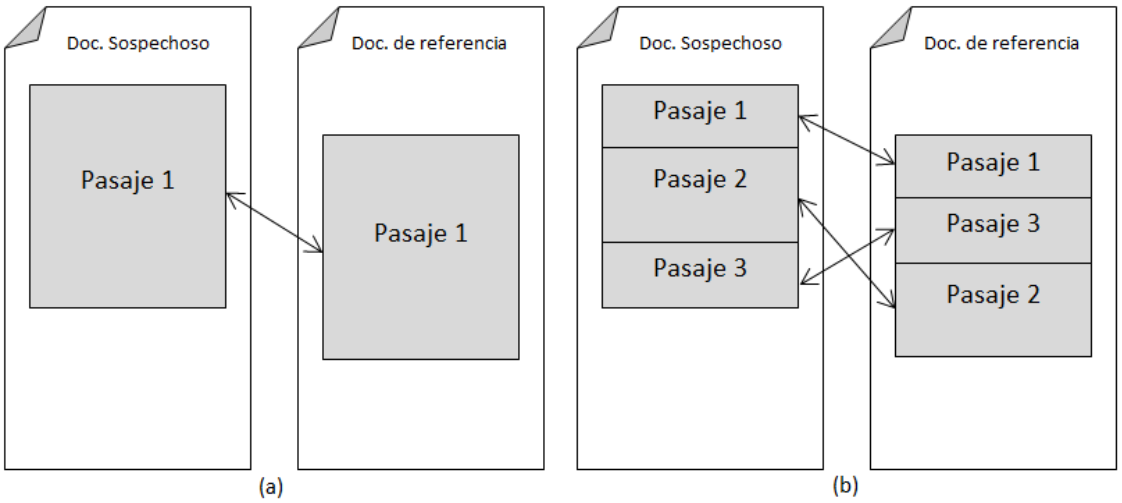


Figura 4.3 (a) Resultado preferido (b) Pasaje con granularidad

En este módulo proponemos un algoritmo para la integración de los pares de oraciones obtenidos en el módulo de selección con el fin de disminuir la granularidad, es decir, buscamos detectar un caso de plagio como un todo. Parte de este algoritmo también consiste en filtrar algunos casos dependiendo de algunos parámetros que veremos más adelante.

Antes de pasar a las siguientes etapas definimos dos conceptos importantes.

1) Los pares de oraciones obtenidos en el módulo anterior se representan como un par formado por la oración del documento sospechoso y la oración correspondiente en el documento de referencia.

$$PS = \{(c,r) \mid c \in SD \wedge r \in OD \wedge Sim_1(c,r) > t1 \wedge Sim_2(c,r) > t2)\} \quad \textbf{(4.11)}$$

Donde SD y OD es la enumeración de las oraciones en el documento sospechoso y de referencia respectivamente.

2) Consideramos que dos oraciones son adyacentes en función de la cantidad de oraciones que se encuentran entre ellas.

$$adj(s_i, s_j) = \begin{cases} V & Si\ |s_i - s_j| - MaxGap\ - 1 \leq 0 \\ F & Otro\ caso \end{cases} \quad \textbf{(4.12)}$$

Para el ejemplo de la figura 4.4, donde las flechas indican los pares de oraciones que cumplieron con los umbrales t1 y t2 en el módulo de selección, obtenemos el siguiente conjunto de pares de oraciones:

$$PS = \{(1,3), (2,4), (3,15), (10,15), (10,16), (21,20), (22,35), (23,34), (24,33)\}$$

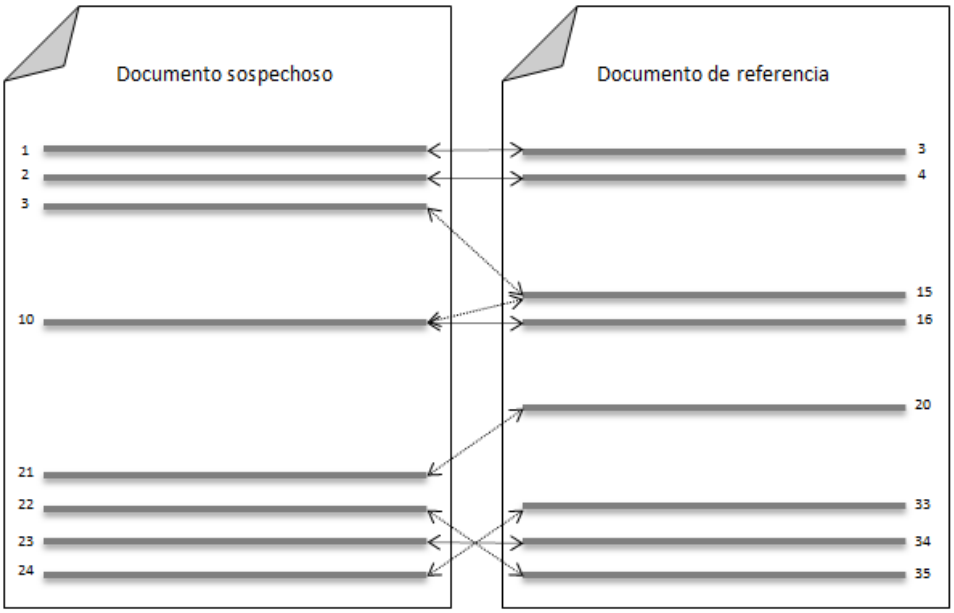


Figura 4.4 Pares de oraciones después de pre-selección

4.3.1 Conjuntos de pares con oraciones adyacentes en el documento sospechoso

En esta fase tomamos subconjuntos de PS donde los pares de oraciones que lo conforman son aquellos donde encontramos grupos de oraciones adyacentes del lado del documento sospechoso tomando en cuenta la definición de adyacencia entre oraciones y la cantidad de oraciones necesarias para formar un subconjunto, dada por el parámetro $MinSize$. El algoritmo para obtener estos subconjuntos PSS_i está dado de la siguiente forma:

1. Seleccionar un elemento del conjunto PS.
2. Obtener todos los elementos adyacentes al elemento seleccionado anteriormente respecto al número de oración en el documento sospechoso.
3. Seleccionar los elementos adyacentes a los elementos agregados.
4. Repetir el paso 2 y 3 hasta que no se obtengan más elementos
5. Conservar el conjunto obtenido si el número de elementos es mayor o igual a $MinSize$.
6. Repetir los pasos 1, 2, 3, 4 y 5 hasta seleccionar todos los elementos del conjunto PS.

En la figura 4.5 se muestra una implementación formal del algoritmo anterior usando teoría de conjuntos.

```
1  i = 1
2  mientras PS ≠ ∅ hacer
3      PSS_i ← {un elemento de PS}
4      PS ← PS − PSS_i
5      A ← {(a, b) ∈ PS | ∃(c, d) ∈ PSS_i : adj(a, c)}
6      mientras A ≠ ∅ hacer
7          PSS_i ← PSS_i ∪ A
8          PS ← PS − A
9          A ← {(a, b) ∈ PS | ∃(c, d) ∈ PSS_i : adj(a, c)}
10     fin
11     si |PSS_i| ≥ MinSize entonces
12         i ← i + 1
13     fin
14 fin
```

Figura 4.5 Algoritmo para obtención de pares de oraciones adyacentes en doc. sospechoso

Una forma más práctica y eficiente de implementar el algoritmo anterior es usando listas. Primero debemos partir de ordenar el conjunto de pares PS según el primer elemento de las duplas, es decir el número de oración en el documento sospechoso. Dado lo anterior el algoritmo consiste en verificar la adyacencia entre los elementos como se muestra en la figura 4.6.

Siguiendo con el ejemplo presentado en la figura 4.4, los subconjuntos PSS_i son los siguientes:

Para $MaxGap = 0$ y $MinSize = 1$:

$$PSS_1 = \{(1,3), (2,4), (3,15)\}$$
$$PSS_2 = \{(10,15), (10,16)\}$$
$$PSS_3 = \{(21,20), (22,35), (23,34), (24,33)\}$$

Para $MaxGap = 0$ y $MinSize = 2$:

$$PSS_1 = \{(1,3), (2,4), (3,15)\}$$
$$PSS_2 = \{(10,15), (10,16)\}$$
$$PSS_3 = \{(21,20), (22,35), (23,34), (24,33)\}$$

En la figura 4.6 se muestra un ejemplo gráfico para la obtención de los conjuntos de pares con oraciones adyacentes en el documento sospechoso para $MaxGap = 0$ y $MinSize = 1$.

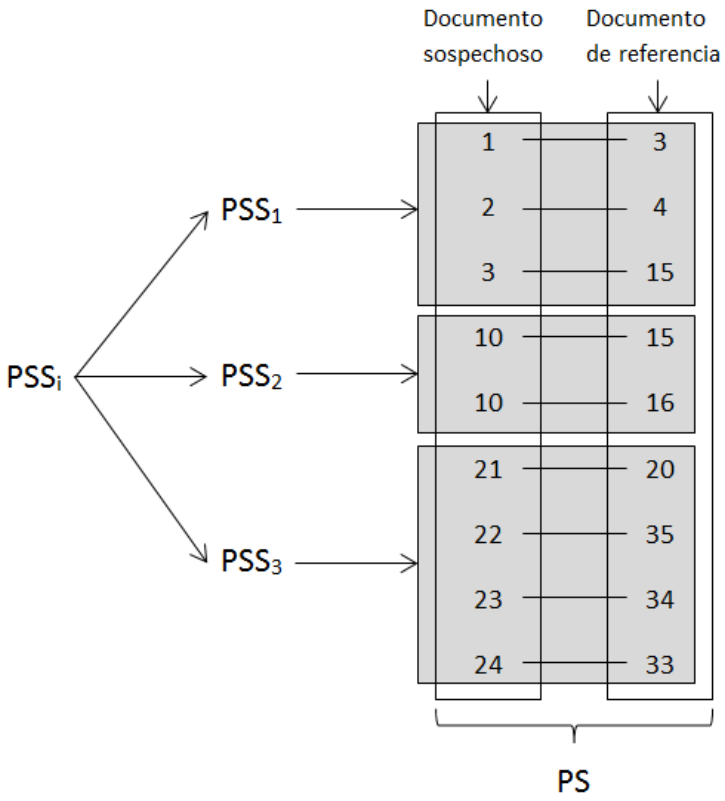


Figura 4.6 Obtención de conjuntos PSS

4.3.2 Conjuntos de pares con oraciones adyacentes en el documento de referencia

Similar a la fase anterior, para cada subconjunto PSS_i obtenido, buscamos subconjuntos de grupos de oraciones adyacentes, esta vez del lado del documento de referencia. El algoritmo para obtener estos subconjuntos $PSR_{i,j}$ está dado de la siguiente forma:

1. Seleccionar un elemento del conjunto PSS_i.
2. Obtener todos los elementos adyacentes al elemento seleccionado anteriormente respecto al número de oración en el documento de referencia.
3. Seleccionar los elementos adyacentes a los elementos agregados.
4. Repetir el paso 2 y 3 hasta que no se obtengan más elementos
5. Conservar el conjunto obtenido si el número de elementos es mayor o igual a $MinSize$.
6. Repetir los pasos 1, 2, 3, 4 y 5 hasta seleccionar todos los elementos del conjunto PSS_i.
7. Repetir los pasos anteriores para todos los conjuntos PSS_i.

La implementación formal usando teoría de conjuntos del algoritmo anterior se muestra en la figura 4.7.

```
1  para todo PSS_i hacer
2      j = 1
3      mientras PSS_i ≠ ∅ hacer
4          PSR_{i,j} ← {un elemento de PSS_i}
5          PSS_i ← PSS_i − PSR_{i,j}
6          A ← {(a, b) ∈ PSS_i | ∃(c, d) ∈ PSR_{i,j} : adj(a, c)}
7          mientras A ≠ ∅ hacer
8              PSR_{i,j} ← PSR_{i,j} ∪ A
9              PSS_i ← PSS_i − A
10             A ← {(a, b) ∈ PSS_i | ∃(c, d) ∈ PSR_{i,j} : adj(a, c)}
11         fin
12         si |PSR_{i,j}| ≥ MinSize entonces
13             j ← j + 1
14         fin
15     fin
16 fin
```

Figura 4.7 Algoritmo para obtención de pares de oraciones adyacentes en doc. de referencia

Al igual que el algoritmo de la sección anterior este módulo es más eficiente usando listas ordenadas. En este caso, debemos partir de ordenar de menor a mayor cada uno de los conjuntos PSS_i y luego verificar la adyacencia entre los elementos. En la figura 4.8 se muestra el resultado para un ejemplo dado.

Calculados los subconjuntos PSS_i en la sección anterior, los subconjuntos $PSR_{i,j}$ son los siguientes:

Para $MaxGap = 0$ y $MinSize = 1$:

$$PSR_{1,1} = \{(1,3), (2,4)\}$$
$$PSR_{1,2} = \{(3,15)\}$$
$$PSR_{2,1} = \{(10,15), (10,16)\}$$
$$PSR_{3,1} = \{(21,20)\}$$
$$PSR_{3,2} = \{(22,35), (23,34), (24,33)\}$$

Para $MaxGap = 0$ y $MinSize = 2$:

$$PSR_{1,1} = \{(1,3), (2,4)\}$$
$$PSR_{2,1} = \{(10,15), (10,16)\}$$
$$PSR_{2,1} = \{(10,15), (10,16)\}$$
$$PSR_{3,1} = \{(22,35), (23,34), (24,33)\}$$

En la figura 4.8 se muestra un ejemplo gráfico para la obtención de los conjuntos de pares con oraciones adyacentes en el documento de referencia a partir de los conjuntos PSS_i para $MaxGap = 0$ y $MinSize = 1$.

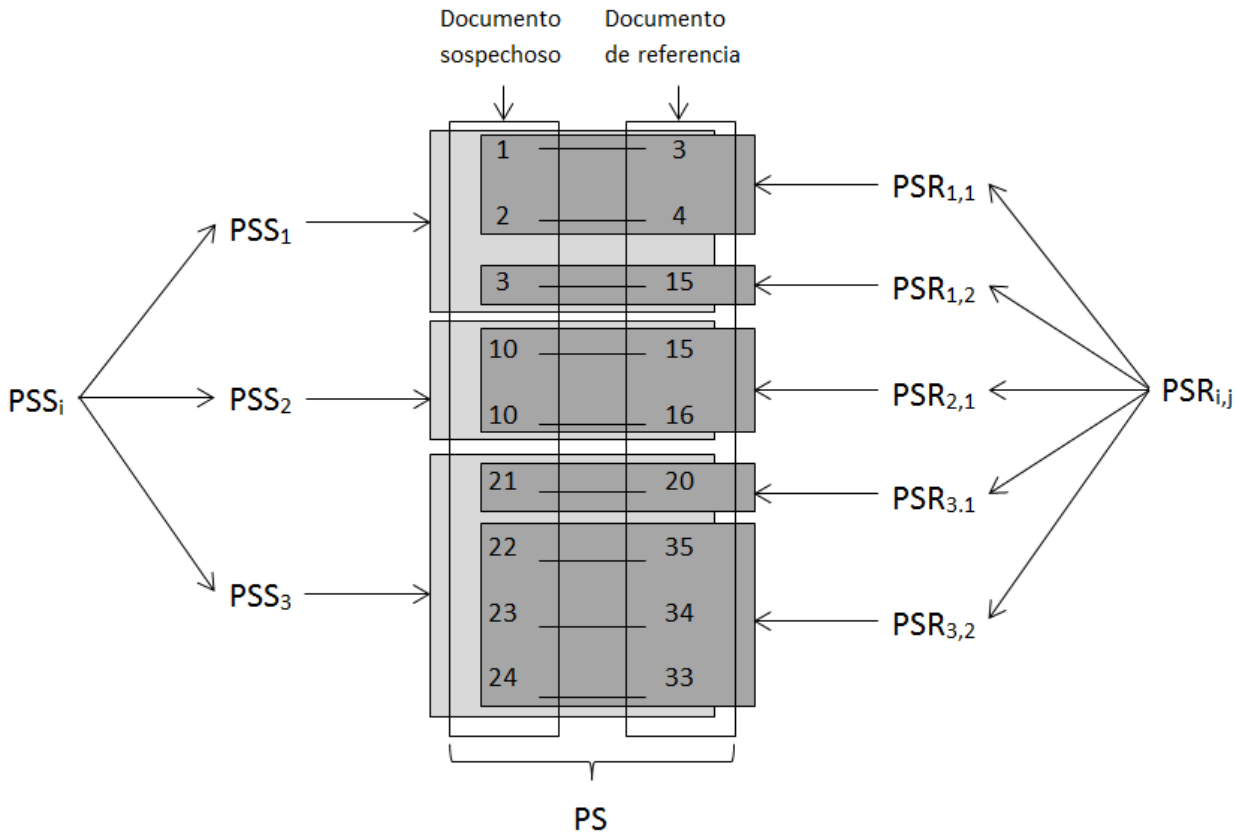


Figura 4.8 Obtención de conjuntos PSR

4.3.3 Transformación de pares en casos de plagio

Ya que hemos obtenido los subconjuntos $PSR_{i,j}$, los representamos como casos de plagios (PC) en función de la primera y última oración en el documento sospechoso y documento de referencia respectivamente. Esto lo hacemos de la siguiente forma:

$$C_{i,j} = \{c \mid \forall (c,r) \in PSR_{i,j}\} \quad \textbf{(4.13)}$$

$$R_{i,j} = \{r \mid \forall (c,r) \in PSR_{i,j}\} \quad \textbf{(4.14)}$$

$$PC = \{(\min(C_{i,j}), \max(C_{i,j}), \min(R_{i,j}), \max(R_{i,j})) \mid \forall C_{i,j}, R_{i,j}\} \quad \textbf{(4.15)}$$

Dado que un caso de plagio está dado por un pasaje en el documento sospechoso y un pasaje en el documento de referencia, la representación PC significa lo siguiente:

$Sea\ (a,b,c,d) \in PC,$
a: Primera oración del pasaje en el documento sospechoso.
b: Última oración del pasaje en el documento sospechoso.

c: Primera oración del pasaje en el documento de referencia.
d: Última oración del pasaje en el documento de referencia.

Siguiendo con el ejemplo de las secciones anteriores:

Para $MaxGap = 0$ y $MinSize = 1$:

$C_{1,1} = \{1,2\}$ $R_{1,1} = \{3,4\}$
$C_{1,2} = \{3\}$ $R_{1,2} = \{15\}$
$C_{2,1} = \{10\}$ $R_{2,1} = \{15,16\}$
$C_{3,1} = \{21\}$ $R_{3,1} = \{20\}$
$C_{3,2} = \{22,23,24\}$ $R_{3,2} = \{35,34,33\}$

$$PC = \{(1,2,3,4), (10,10,15,16), (22,24,33,35)\}$$

Para $MaxGap = 0$ y $MinSize = 2$:

$C_{1,1} = \{1,2\}$ $R_{1,1} = \{3,4\}$
$C_{2,1} = \{10\}$ $R_{2,1} = \{15,16\}$
$C_{3,1} = \{22,23,24\}$ $R_{3,1} = \{35,34,33\}$

$$PC = \{(1,2,3,4), (10,10,15,16), (22,24,33,35)\}$$

Esta forma de representar los casos de plagio nos permite incluir oraciones que no cumplieron con los umbrales de similitud y se encuentran entre pares que si cumplieron con dichos umbrales.

En la figura 4.9 se presenta un caso donde ocurre este fenómeno para $MaxGap = 2$, donde el caso de plagio incluye las oraciones 3 y 4 del documento sospechoso cuya similitud con las oraciones del documento de referencia no excedió los umbrales t1 o t2.

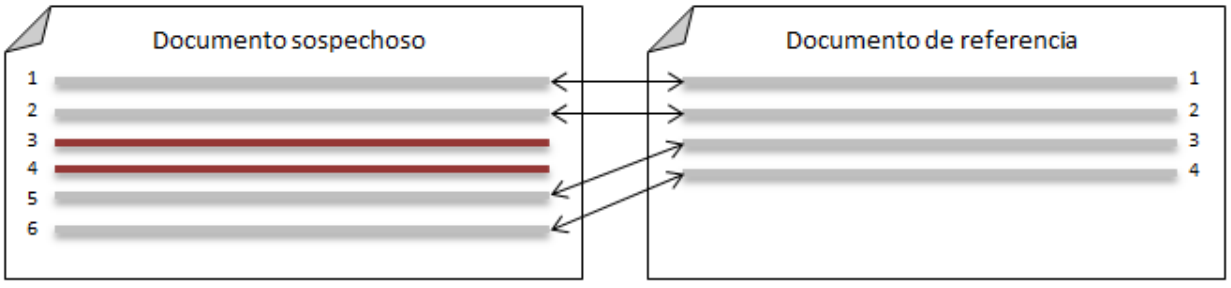


Figura 4.9 Pares de oraciones después de pre-selección

$$PSR_{1,1} = \{(1,1), (2,2), (5,3), (6,4)\}$$

4.4 Post-procesamiento

En este módulo aplicamos tres técnicas de post-procesamiento con el objetivo de incrementar la precisión de nuestro modelo.

1. Se eliminan aquellos casos de plagio donde la similitud entre sus pasajes, usando alguna de las medidas de similitud planteadas anteriormente, no sobrepasa un umbral t3.
2. Si existen casos de plagios con pasajes solapados, es decir, una o varias oraciones aparecen en diferentes pasajes; se elige el caso de plagio con mayor similitud entre sus dos pasajes y se elimina el resto.
3. Se eliminan los casos de plagio donde sus pasajes no sobrepasan una cierta cantidad de caracteres $MinPlagLength$.

4.4.1 Similitud de pasajes en casos de plagio

Para cada uno de los casos de plagio PC se calcula la similitud $Sim_3(S, R)$ entre un vector formado por las oraciones del pasaje en el documento sospechoso y otro formado por las oraciones del pasaje en el documento de referencia. Si dicha similitud sobrepasa un umbra t3 mantenemos el caso de plagio, en caso contrario, el caso de plagio es desechado. Dicha similitud se calcula usando alguna de las tres medidas propuestas anteriormente (medida del coseno del ángulo, coeficiente de dice o similitud de estructura).

$$PC_{post} = \left\{ (a,b,c,d) \in PC \mid Sim_3\left(\sum_{x=a}^{b} vectSD_x , \sum_{y=c}^{d} vectOD_y \right) > t3 \right\} \quad \textbf{(4.16)}$$

Donde $vectSD_x$ representa el vector de la oración x en el documento sospechoso SD y $vectOD_y$ representa el vector de la oración y en el documento de referencia OD.

4.4.2 Eliminación de casos de plagio con pasajes solapados

En esta sección buscamos eliminar casos de plagio que contienen pasajes solapados solo del lado del documento sospechoso. Esto debido a que consideramos que un pasaje en el documento sospechoso solo corresponde a un pasaje del documento de referencia, mientras que no ocurre lo mismo en la otra dirección, donde una misma fuente puede ser usada varias veces en un documento sospechoso. En el ejemplo representado en la figura 4.10, para el caso (a), de los pasajes solapados en el documento sospechoso se seleccionará aquel que cumpla cierta condición (la cual explicaremos más adelante); mientras en el caso (b) no se eliminará ningún pasaje.

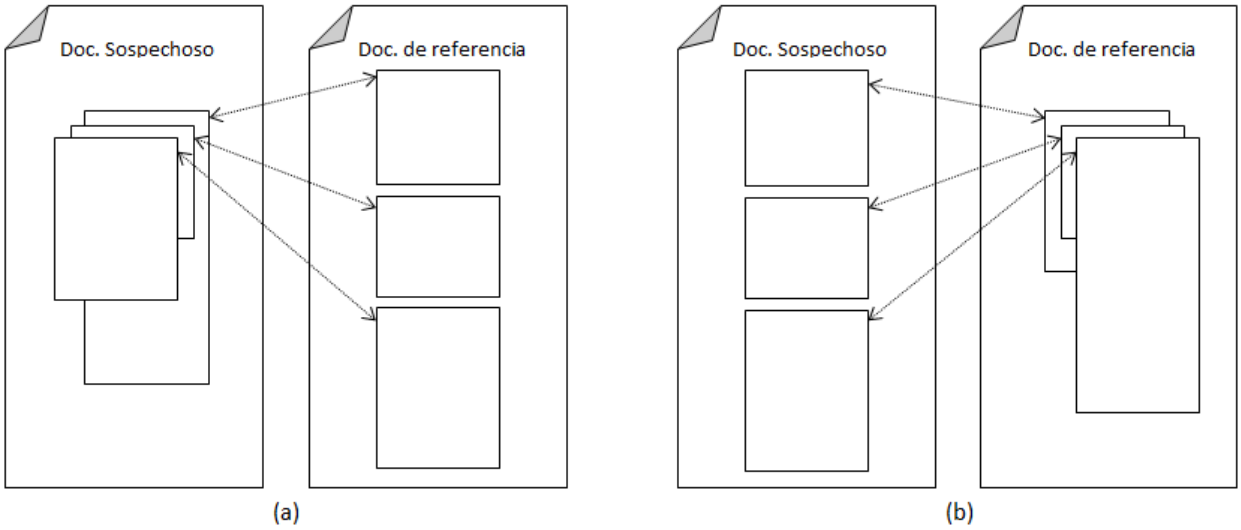


Figura 4.10 (a) Pasajes sospechosos solapados con diferentes fuentes (b) Pasajes sospechosos diferentes con fuentes solapadas

Para esta tarea planteamos dos definiciones de solapamiento con dos posibles criterios de selección para los casos de plagio a comparar.

Definimos solapamiento como:

1) Pasajes que empiezan en la misma oración del lado del documento sospechoso como se muestra en la figura 4.10a.
2) Pasajes que se interceptan en alguna oración del lado del documento sospechoso como se muestra en la figura 4.11.

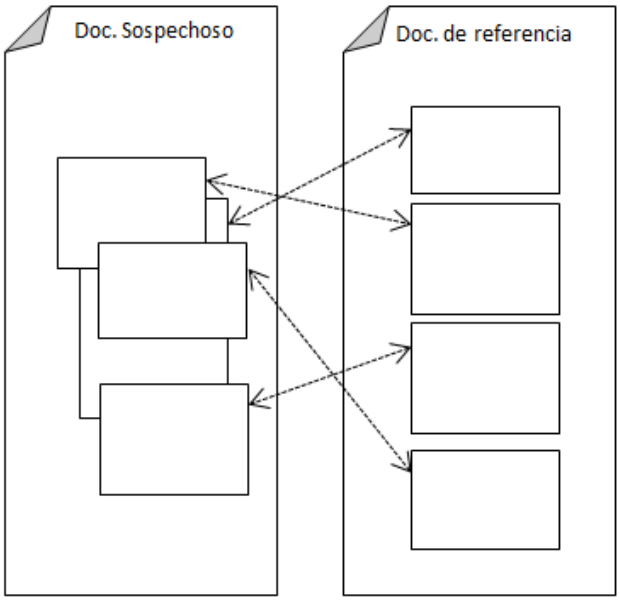


Figura 4.11 Casos de plagio con solapamiento

Los criterios de selección se basan en la comparación de dos casos de plagio. Estos criterios de comparación son los siguientes:

1) Se selecciona el caso de plagio que tenga mayor similitud entre sus vectores del documento sospechoso y del documento de referencia, a través de la siguiente fórmula:

$$\arg\max_{PC}(Sim(PC_1), Sim(PC_2)) \quad \textbf{(4.17)}$$

Donde,

$Sea\ (a, b, c, d) \in PC_i,$

$$Sim(PC_i) = \frac{\left(\sum_{x=a,\dots,b} \max_{y=c,\dots,d} CM\left(vectSD_x, vectOD_y\right)\right)}{|\{a, \dots, b\}|} \quad \textbf{(4.18)}$$

2) Se selecciona el caso de plagio con mayor similitud entre sus vectores del documento sospechoso y del documento de referencia distinguiendo la similitud entre las oraciones solapadas y las no solapadas, a través de la siguiente fórmula:

$$\arg\max_{PC}(Sim(PC_1|PC_2), Sim(PC_2|PC_1)) \quad \textbf{(4.19)}$$

En la ecuación 4.19 la similitud $Sim(PC_i|PC_j)$ se calcula de la siguiente forma:

$$Sim\left(PC_i\middle|PC_j\right) = SimOS\left(PC_i\middle|PC_j\right) + \left(1 - SimOS\left(PC_i\middle|PC_j\right)\right) \times SimNOS(PC_i|PC_j) \quad \textbf{(4.20)}$$

En la ecuación 4.20 la similitud $SimOS(PC_i|PC_j)$ representa la similitud entre los pasajes de PC_i tomando en cuenta solo las oraciones solapadas entre PC_i y PC_j del lado del documento sospechoso; y está dada por:

$Sean$: $(a, b, c, d) \in PC_i,$

$(e, f, g, h) \in PC_j,$

$OS = \{x | x \in \{a, \dots, b\} \wedge x \in \{e, \dots, f\}\}$ **(4.21)**

$$SimOS\left(PC_i|PC_j\right) = \frac{\left(\sum_{x \in OS} \max_{y=c,\dots,d} CM\left(vectSD_x, vectOD_y\right)\right)}{|OS|}$$

Mientras que la $SimNOS(PC_i|PC_j)$ representa la similitud entre los pasajes de PC_i tomando en cuenta solo las oraciones no solapadas entre PC_i y PC_j del lado del documento sospechoso; y está dada por:

$$\begin{aligned} &Sean: (a,b,c,d) \in PC_i, \\ &\quad (e,f,g,h) \in PC_j, \\ &\quad NOS = \{x | x \in \{a,\dots,b\} \wedge x \notin \{e,\dots,f\}\} \end{aligned} \qquad \textbf{(4.22)}$$

$$SimNOS(PC_i|PC_j) = \frac{\left(\sum_{x \in NOS} \max_{y=c,\dots,d} CM(vectSD_x, vectOD_y)\right)}{|NOS|}$$

4.4.3 Eliminación de pasajes pequeños

Esta fase de post-procesamiento está motivada por la idea que pequeños fragmentos de texto pueden representar frases famosas, conocimiento común o simplemente coincidencia entre varios textos. Decidir si este es un pasaje plagiado o no, puede ser difícil, riesgoso o incluso controversial.

El análisis de las características del corpus utilizado para la evaluación del método propuesto soporta esta idea. La figura 4.12, que representa el histograma de tamaños de pasajes por cada sub-corpus y cada documento, nos muestra que los pasajes se encuentran agrupados en cierta cantidad de longitudes lo que nos permite aplicar un filtro pasa altas, eliminando así los fragmentos de textos que consideramos muy pequeños.

El filtro pasa altas seleccionado se basa en un umbral $MinPlagLength$ aplicado en ambos lados de los casos de plagio, es decir, del lado del documento sospechoso y del documento de referencia. Los casos que no cumplen con este umbral son eliminados.

$$PC_{post} = \left\{ \begin{matrix} (a,b,c,d) \in PC | ini(b) - ini(a) + len(b) \geq MinPlagLength \\ \wedge\, ini(d) - ini(c) + len(d) \geq MinPlagLength \end{matrix} \right\} \qquad \textbf{(4.23)}$$

Donde $ini(x)$ regresa el símbolo de inicio de la oración x, y $len(x)$ regresa la longitud en caracteres de la oración x.

4.5 Generación de archivo XML con casos de plagio

Los casos de plagio son convertidos a otra notación en función del símbolo de inicio y el tamaño en caracteres del pasaje, características extraídas en la sección de segmentación. Todo esto con el objetivo de aplicar la métrica de evaluación propuesta en [25]. Esta representación es de la siguiente forma:

$$PC_{final} = \left\{ \begin{pmatrix} ini(a), ini(b) - ini(a) + len(b), \\ ini(c), ini(d) - ini(c) + len(d) \end{pmatrix} | \forall (a,b,c,d) \in PC_{post} \right\} \quad \textbf{(4.24)}$$

El archivo XML de la comparación de un par de documentos tendrá la siguiente forma dependiendo de cada $(x, y, z, w) \in PC_{final}$:

```
<document reference="suspicious-documentXYZ.txt">
<feature
name="detected-plagiarism"
this_offset="x"
this_length="y"
source_reference="source-documentABC.txt"
source_offset="z"
source_length="w"
/>
<feature ... />
...
</document>
```

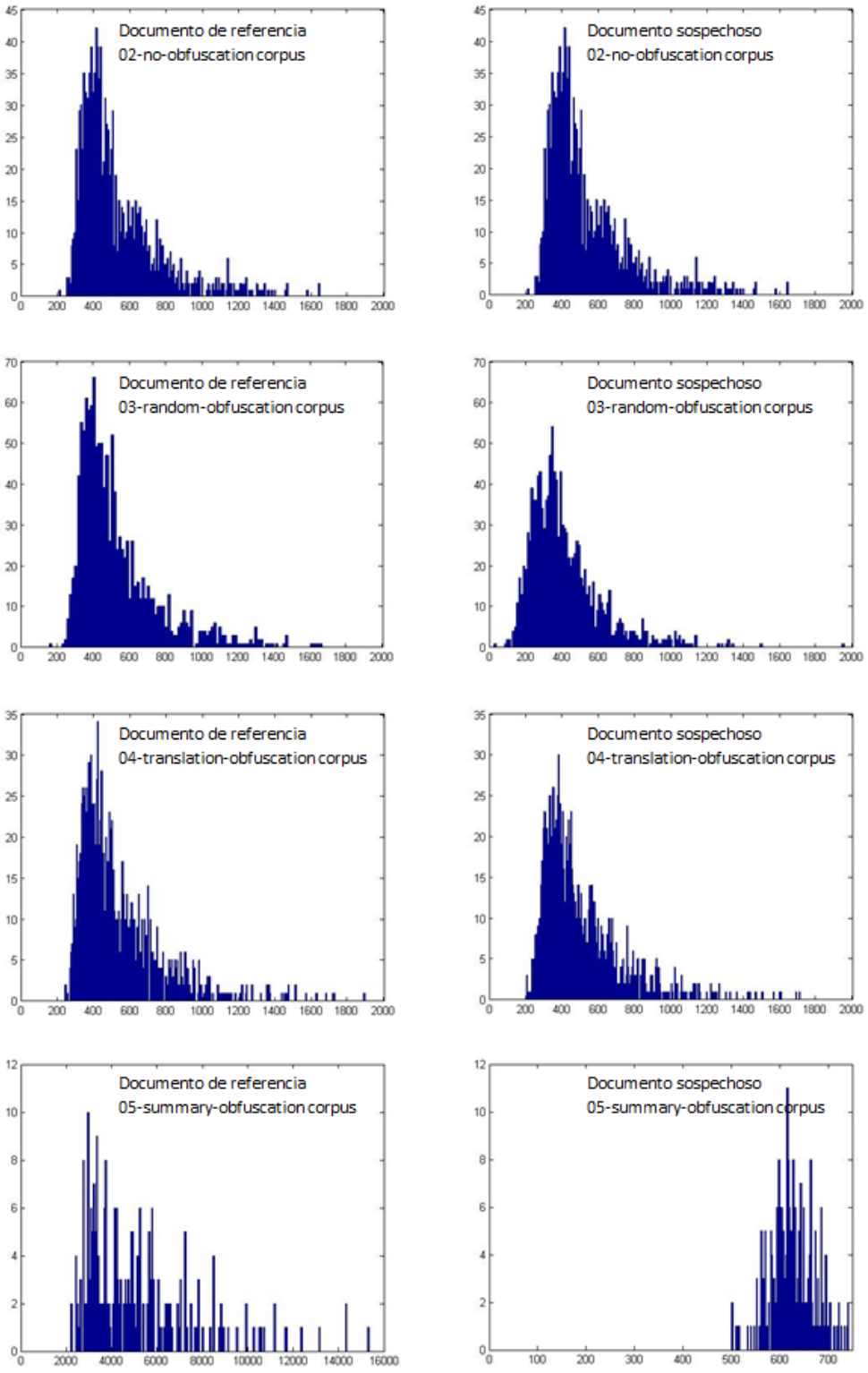


Figura 4.12 Histogramas de casos de plagio en *goldstandard*

5 RESULTADOS

En este capítulo describimos el corpus de entrenamiento y de prueba, las medidas de evaluación utilizadas, los experimentos realizados para mejorar los resultados y la comparación de nuestros resultados contra los métodos del estado del arte y otros métodos evaluados sobre el mismo corpus.

5.1 Corpus de evaluación

El corpus de evaluación es el del PAN 2013. Este corpus se basa en construir automáticamente pares de documentos que comprenden pasajes de texto reusados. El proceso de construcción y estrategias de ofuscación del corpus se presenta en [8]. Un par de documentos está formado por un documento sospechoso y un documento de referencia. El corpus contiene en total 3653 documentos sospechosos y 4774 documentos de referencia, los cuales son agrupados en 10000 pares, de forma tal que existen 2000 pares sin casos de plagio, 2000 pares que contienen casos de plagio sin alteraciones y 6000 pares que contienen alguna de las siguientes estrategias de alteración.

Alteración aleatoria *(random obfuscation)*: El uso de esta estrategia resulta en pasajes que no son legibles y no tienen significado. El propósito de esta estrategia es probar si los algoritmos de detección de plagio son capaces de identificar pasajes de texto reusado desde un punto de vista de un modelo de bolsa de palabra. La estrategia se basa en una secuencia de operaciones aleatorias como mezclar, agregar, eliminar o remplazar palabras o pequeñas frases de forma aleatoria. El remplazo de palabras se hace empleando una base de datos de sinónimos como WordNet, mientras que las frases son mezcladas manteniendo la secuencia de características gramaticales.

Alteración usando traducción de forma cíclica *(translation obfuscation)*: En esta estrategia un pasaje de texto plagiado es pasado por una secuencia de traducciones de forma tal que la salida de una traducción en la estrada de otra, mientras que el último lenguaje de la secuencia es el lenguaje original del pasaje. La idea de esta estrategia es explotar el hecho que traducir un texto implica inherentemente parafrasearlo. Se usan tres servicios de traducción web, Google Translate[1], Microsoft Translator[2] y MyMemory[3]. En toda secuencia de alteración se usan los tres servicios para evitar favorecer alguna traducción específica, ya que diferentes servicios de traducción se basan en diferentes modelos de entrenamiento.

El lenguaje inicial y final siempre es el inglés mientras que los lenguajes intermedios pueden ser de dos conjuntos de lenguajes: la familia de indo-europeos como francés, alemán, italiano, español y sueco; y mezcla de otras familias de lenguajes como arábigo, chino, hebreo, hindú y

[1] http://translate.google.com
[2] http://www.microsoft.com/translator
[3] http://mymemory.translated.net

japonés. Primero se elige uno de estos conjuntos y luego hasta tres lenguajes intermedios son empleados

Alteración usando resúmenes *(summary obfuscation)*: Esta estrategia se basa en el fundamento que incluir un resumen de un documento en un texto puede ser considerado un caso de plagio ya que mantiene las ideas principales de forma resumida. Se presume que un resumen no se basa en una simple concatenación de algunas oraciones del documento original.

En esta estrategia se usó el conjunto de documentos originales del corpus para generación de resúmenes del congreso DUC del 2001. Cada uno de estos documentos originales cuenta con dos resúmenes creados por asesores humanos. Para producir un caso de plagio estos resúmenos fueron plantados en los documentos del corpus para generación de resúmenes del congreso DUC del 2006.

El corpus de evaluación fue divido en dos partes (cada una con 5000 pares), una para el entrenamiento de los algoritmos que participaron en el taller y otro para probarlos. En este capítulo se presentan resultados que muestran la evolución del método propuesto en dependencia de los parámetros usados, así como una comparación con los algoritmos que participaron en el taller del PAN 2013.

5.2 Marco de evaluación

Un marco de evaluación en la tarea de la detección automática de plagio fue propuesto en [25] con el objetivo de cubrir la necesidad que existía en este campo. Fue creado en el contexto de los talleres de evaluación PAN de los años 2009 y 2010 [39] [40], y se ha convertido en la métrica estándar para las subsecuentes competiciones. En este marco de evaluación se propone una medida (*plagdet*) que está en función de la precisión, exhaustividad (*recall*) y granularidad como se presenta a continuación.

Denotemos d_{plg} como un documento que contiene plagio. Un caso de plagio en d_{plg} es una 4-tupla $s = \langle s_{plg}, d_{plg}, s_{src}, d_{src} \rangle$, donde s_{plg} es un pasaje plagiado en d_{plg}, y s_{src} es el pasaje original correspondiente en el documento de referencia d_{src}. De manera similar, un caso de plagio detectado se denota como $r = \langle r_{plg}, d_{plg}, r_{src}, d'_{src} \rangle$; donde r asocia un supuesto pasaje plagiado r_{plg} en d_{plg} con un pasaje r_{src} en d'_{src}. Decimos que $r\ detecta\ s$ si y solo si $r_{plg} \cap s_{plg} \neq \emptyset$, $r_{src} \cap s_{src} \neq \emptyset$ y $d'_{src} = d_{src}$.

Presentando una visión equivalente más concisa que simplificará las subsecuentes notaciones denotamos un documento d como un conjunto de referencias a sus caracteres $\mathrm{d} = \{(1, d), \ldots, (|d|, d)\}$, donde (i, d) refiere el i-ésimo caracter in d. De esta forma un caso de plagio s puede ser representado como $\mathrm{s} = \mathrm{s}_{plg} \cup \mathrm{s}_{src}$, donde $\mathrm{s}_{plg} \subseteq d_{plg}$ y $\mathrm{s}_{src} \subseteq d_{src}$. Los caracteres referenciados en s_{plg} y $\mathrm{s}_{\mathrm{src}}$ forman los pasajes s_{plg} y s_{src} en la visión anterior. De forma similar una detección r puede ser representada como $r = r_{plg} \cup r_{src}$. A partir de esto

podemos decir que $r\ detecta\ s$ si y solo si $r_{plg} \cap s_{plg} \neq \emptyset$ y $r_{src} \cap s_{src} \neq \emptyset$. Por último, S y R denotan conjuntos de casos de plagios y detecciones respectivamente. Basado en estas representaciones la precisión y la exhaustividad de R según S se definen como:

$$prec(S,R) = \frac{1}{|R|}\sum_{r\in R}\frac{|\bigcup_{s\in S}(s \sqcap r)|}{|r|} \quad \textbf{(5.1)}$$

$$rec(S,R) = \frac{1}{|S|}\sum_{s\in S}\frac{|\bigcup_{r\in R}(s \sqcap r)|}{|s|} \quad \textbf{(5.2)}$$

Donde

$$s \sqcap r = \begin{cases} s \cap r & si\ r\ detecta\ s \\ \emptyset & otro\ caso \end{cases} \quad \textbf{(5.3)}$$

Además de la precisión y la exhaustividad hay otro concepto que caracteriza el poder de un algoritmo de detección, es decir, si un caso de plagio fue detectado como uno solo o en varias partes. Para esto se define la granularidad de la detección de R según S:

$$gran(S,R) = \frac{1}{|S_R|}\sum_{s\in S_R}|R_s| \quad \textbf{(5.4)}$$

Donde $S_R \subseteq S$ son los casos *detectados* por las detecciones en R, y $R_s \subseteq R$ son las detecciones de un caso s dado.

$$S_R = \{s \mid s \in S \wedge \exists r \in R : r\ detecta\ s\} \quad \textbf{(5.5)}$$

$$R_s = \{r \mid r \in R \wedge r\ detecta\ s\} \quad \textbf{(5.6)}$$

El dominio de $gran(S,R)$ es $[1,|R|]$, con 1 indicando la correspondencia deseada uno-a-uno y $|R|$ indicando el peor de los casos, donde un solo caso $s \in S$ es detectado una y otra vez.

La precisión, exhaustividad y granularidad permiten un orden parcial entre los algoritmos de detección. Para obtener un orden absoluto estas medidas se combinan en una medida general de la siguiente forma:

$$plagdet(S,R) = \frac{F_\alpha}{\log_2(1+gran(S,R))} \quad \textbf{(5.7)}$$

Donde F_α denota la Medida-F. En las competiciones del PAN se usa $\alpha = 1$, es decir, la media armónica ponderada de la precisión y la exhaustividad ya que no hay indicación de que una sea más importante que la otra.

5.3 Optimización de parámetros

En esta sección buscamos optimizar los parámetros usados en nuestro modelo. Debido a que las combinaciones de los valores de cada parámetro son muchas asumimos que los parámetros son independientes entre sí en la mayoría de los casos. La optimización se realiza usando el corpus de entrenamiento y se plantea una hipótesis sobre el comportamiento de los resultados.

5.3.1 Parámetro RemSw

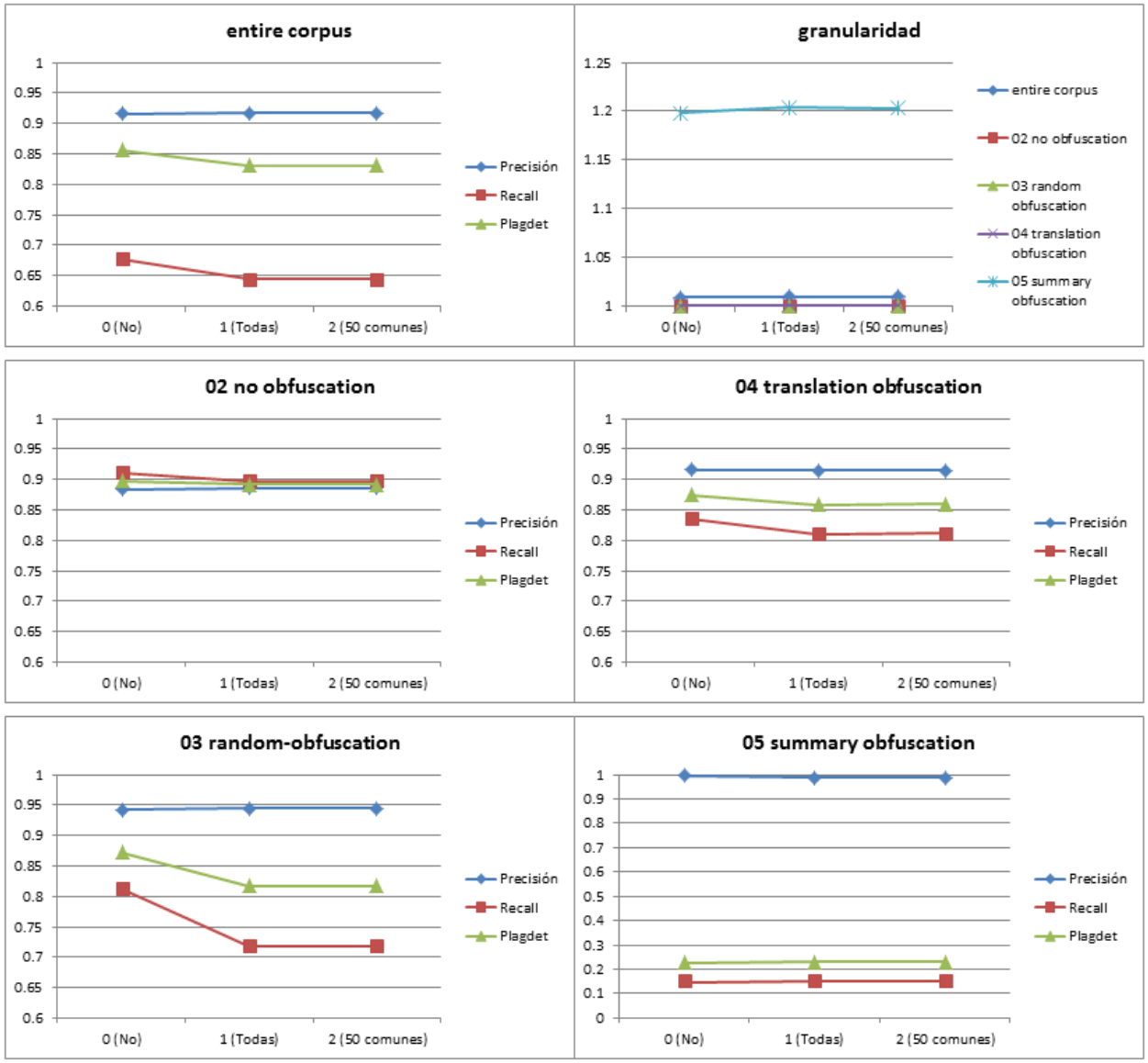


Figura 5.1 Precisión, *recall*, *plagdet* y granularidad respecto a RemSw

En esta sección mostramos el comportamiento del modelo en función del parámetro $RemSw$. Este parámetro define que procesamiento se le dará a las palabras auxiliares en el corpus. Como se describió en la sección 4.1.2 los posibles valores de este parámetro indican lo siguiente:

- $RemSw = 0$: No se eliminan las palabras auxiliares.
- $RemSw = 1$: Se eliminan todas las palabras auxiliares.
- $RemSw = 2$: Se eliminan las 50 palabras auxiliares más comunes.

La figura 5.1 muestra los resultados con la siguiente configuración:

Parámetro	**Valor**
$RemSw$	-
$MinSentLength$	5
$tf \cdot idf$	3
$t1$	0.33 (CM)
$t2$	0.33 (DC)
$MaxGap$	2
$MinSize$	2
$t3$	0.33 (DC)
$MinPlagLength$	150

Los resultados muestran que dejando las palabras auxiliares en el corpus permite mejorar el desempeño. Esto se debe a que el uso de palabras auxiliares puede definir un estilo de escritura. También, los resultados similares obtenidos sin eliminar las palabras auxiliares y eliminando las 50 palabras auxiliares más comunes, se debe a que un esquema tf-idf reduce la influencia de las palabras auxiliares comunes (muy similar a eliminar las 50 palabras más comunes), que tienden a repetirse más en un corpus, mientras que en aquellas que no se repiten mucho y pueden estar más relacionadas a la forma de escribir de un autor, se incrementa.

Cabe resaltar, que en estas etapas tempranas de nuestro modelo, es decir, pertenecientes al pre-procesamiento; buscamos el valor mayor para los parámetros con respecto a la medida del *recall*. Esto se debe al comportamiento de nuestro modelo, donde en las siguientes etapas los resultados se irán filtrando para mejorar la precisión.

5.3.2 Parámetro MinSentLength

En esta sección mostramos el comportamiento del modelo en función del parámetro $MinSentLength$. Este parámetro determina cuál es la longitud mínima de una oración. Si el total de las palabras en una oración es menor a $MinSentLength$ se anexa a la siguiente oración. En la figura 5.2 se muestra dicho comportamiento para valores de $MinSentLength$ de 0 a 20 con la siguiente configuración:

Parámetro	Valor
$RemSw$	0
$MinSentLength$	-
$tf \cdot idf$	3
$t1$	0.33 (CM)
$t2$	0.33 (DC)
$MaxGap$	2
$MinSize$	1
$t3$	0.33 (DC)
$MinPlagLength$	0

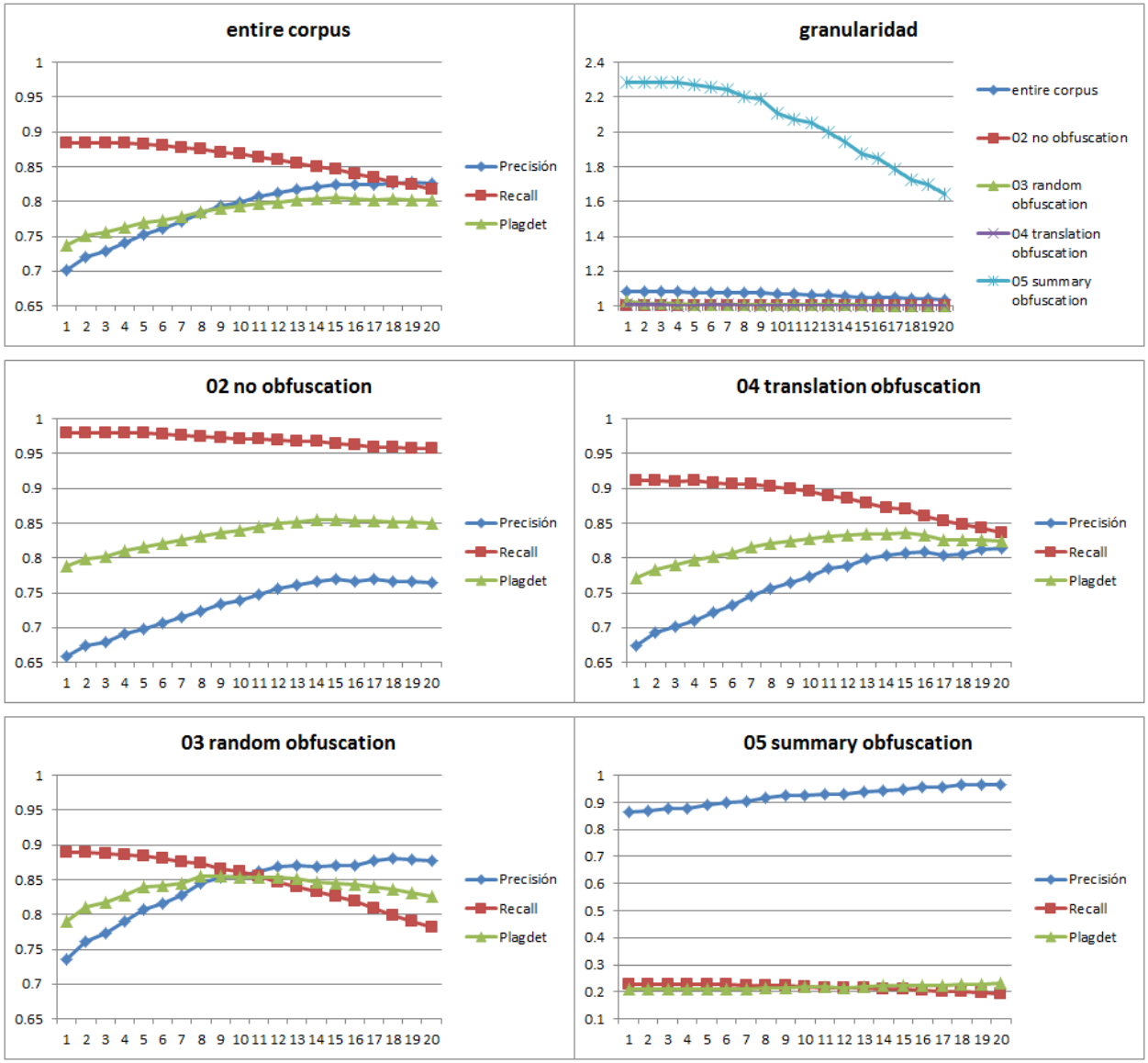


Figura 5.2 Precisión, *recall*, *plagdet* y granularidad respecto a MinSentLength (1)

En la figura 5.2 se puede observar que a pesar que la medida *plagdet* aumenta conforme aumenta el parámetro MinSentLength, la medida *recall* disminuye. Como mencionamos en la sección del parámetro $RemSw$, en estas etapas de pre-procesamiento preferimos los valores altos de *recall*.

Las gráficas muestran que este módulo no influye en los resultados satisfactoriamente para la configuración de parámetros utilizada. Debido a esto, decidimos realizar pruebas con otra configuración cambiando el valor del parámetro $MinPlagLength$ a 150. Este parámetro elimina los pasajes que tengan menos de 150 caracteres por lo que si nuestro modelo forma pasajes con oraciones muy pequeñas, estos serán eliminados. El funcionamiento del parámetro $MinPlagLength$, por lo tanto, afecta el funcionamiento de $MinSentLength$ como se muestra en la figura 5.3.

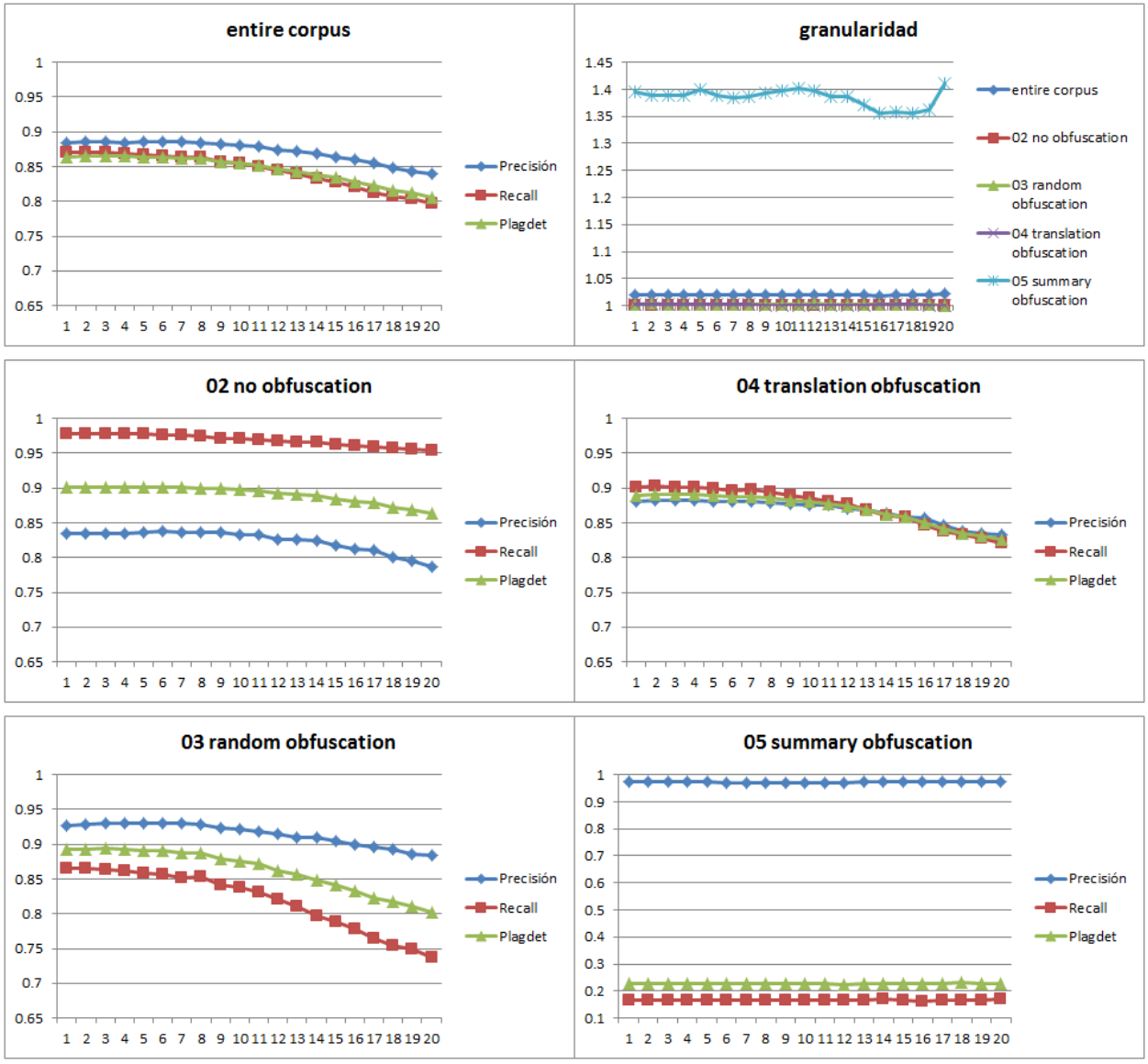


Figura 5.3 Precisión, *recall*, *plagdet* y granularidad respecto a MinSentLength (2)

A pesar de que la influencia de este módulo para aumentar los resultados es baja, su importancia radica en que actúa como un verificador del tamaño mínimo de una oración. De esta forma nuestro modelo puede ser implementado independientemente del tipo de segmentador de oraciones que se use.

La disminución paulatina de los resultados se debe a que se unen una gran cantidad de oraciones. Este comportamiento se debe a las características del corpus usado. En la figura 5.4 se muestra un histograma del número de palabras por oración en todo el corpus de entrenamiento, donde la media de palabras por oración es de 21. Este histograma muestra que la mayor parte de las oraciones contienen más de 3 palabras.

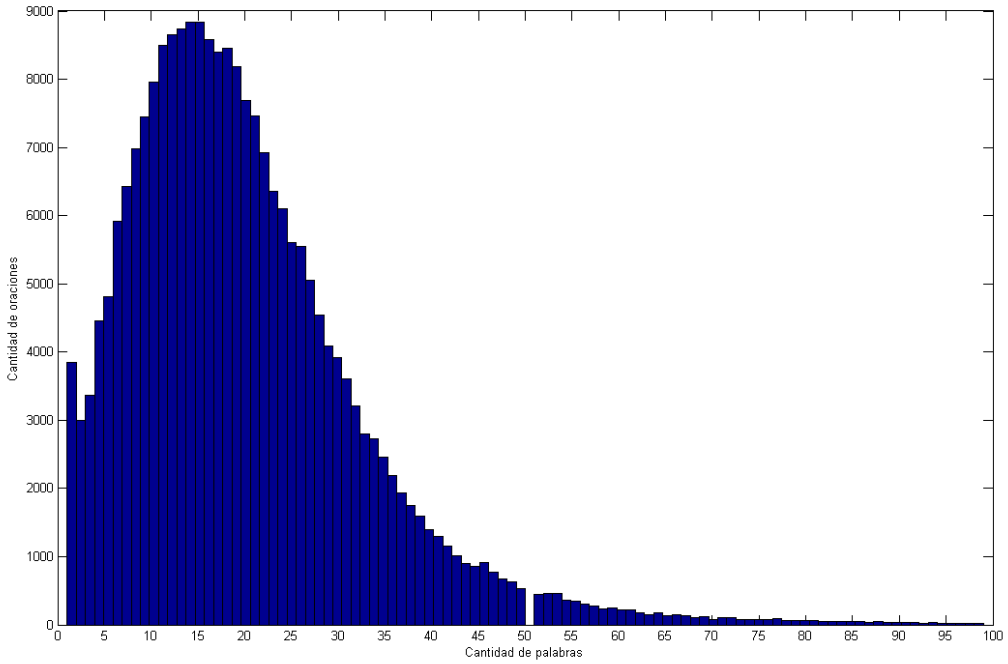


Figura 5.4 Histograma del número de palabras por oración en el corpus de entrenamiento

Analizando los resultados, el *recall* disminuye porque las oraciones plagiadas se unen a otras oraciones no plagiadas que evita que la oración resultante sea seleccionada (no sobrepasa los umbrales en la sección 4.2). La precisión disminuye por la misma razón aunque esta vez la oración resultante de la unión es seleccionada (sobrepasa los umbrales en la sección 4.2) pero contiene ruido, es decir, las oraciones que no fueron plagiadas pero sí fueron seleccionadas, afectan la precisión. El cambio en la granularidad es muy pequeño y para valores grandes de $MinSentLength$ la caída de la granularidad, todavía pequeña, se debe a que los casos de plagios seleccionados son muy grandes y por lo tanto con respecto al estándar de oro fueron detectados como un solo caso.

El mejor resultado para todo el corpus de entrenamiento, aunque prácticamente imperceptible en la gráfica, lo obtenemos para $MinSentLength = 3$.

5.3.3 Parámetro tf·idf

En esta sección mostramos el comportamiento del modelo en función de qué tipo de ponderación de palabras se usa. Como se explicó en la sección 4.2.1 los valores del parámetro $tf \cdot idf$ significan lo siguiente:

$tf \cdot idf = 0$	S usa frecuencia de términos solamente.
$tf \cdot idf = 1$	Se usa frecuencia de términos normalizada.
$tf \cdot idf = 2$	Se usa frecuencia de términos multiplicada por la frecuencia invertida de documento.
$tf \cdot idf = 3$	Se usa frecuencia de términos normalizada multiplicada por la frecuencia invertida de documento.

Los experimentos para el parámetro $tf \cdot idf$ se muestran en la figura 5.5 y se realizan con la siguiente configuración:

Parámetro	**Valor**
$RemSw$	0
$MinSentLength$	3
$tf \cdot idf$	-
$t1$	0.33 (CM)
$t2$	0.33 (DC)
$MaxGap$	2
$MinSize$	1
$t3$	0.33 (DC)
$MinPlagLength$	0

En la figura se observa claramente la ventaja de usar el esquema tf·idf propuesto para este tipo de tarea. Esta ventaja se debe a que se le da más importancia a las palabras que contienen mayor carga semántica, característica inherente de un esquema de ponderación tf·idf. También se puede observar que la normalización de la frecuencia de términos no afecta el resultado.

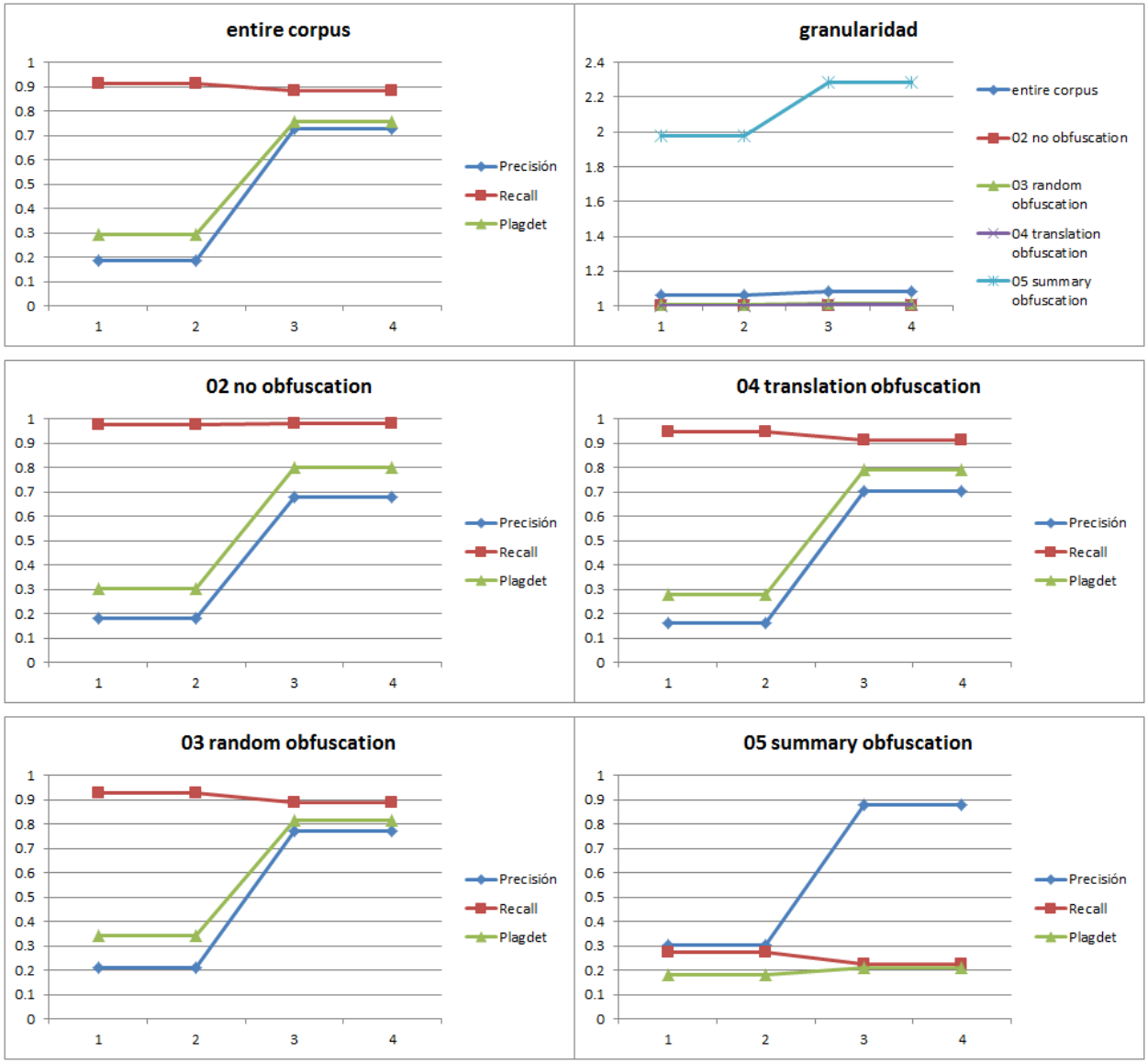


Figura 5.5 Precisión, *recall*, *plagdet* y granularidad respecto a tf-idf

5.3.4 Parámetro MinSize

En esta sección mostramos el comportamiento del modelo en función del parámetro $MinSize$. El parámetro $MinSize$ nos define el mínimo número de oraciones que un pasaje plagiado debe contener como mínimo. En la figura 5.4 se muestra dicho comportamiento para valores de $MinSize$ de 1 a 4 con la siguiente configuración:

Parámetro	Valor
$RemSw$	0
$MinSentLength$	3
$tf \cdot idf$	3
$t1$	0.33 (CM)
$t2$	0.33 (DC)

$MaxGap$	2
$MinSize$	-
$t3$	0.33 (DC)
$MinPlagLength$	0

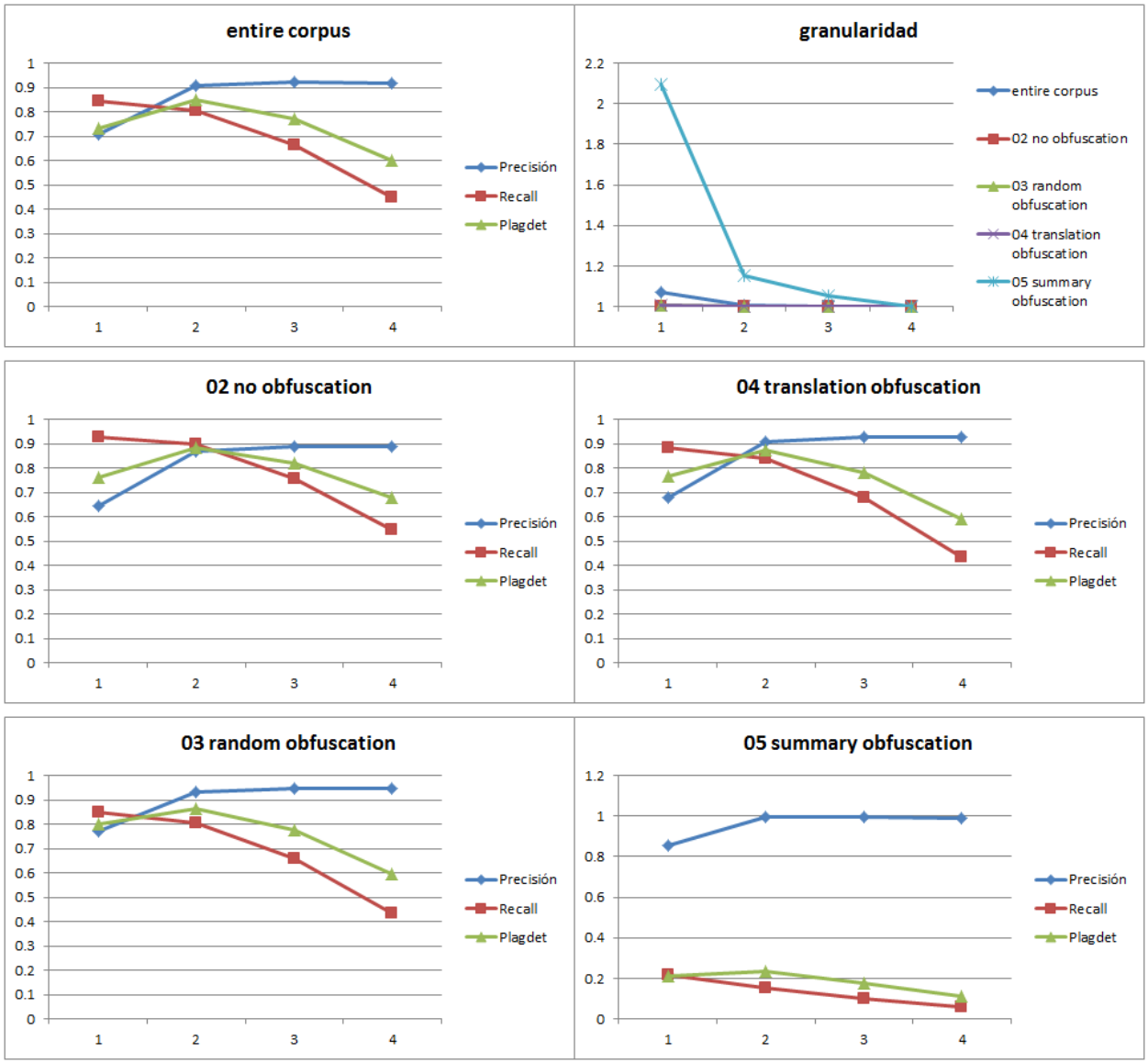


Figura 5.6 Precisión, *recall*, *plagdet* y granularidad respecto a MinSize (1)

La figura 5.4 muestra que la medida de evaluación general *plagdet* varía según un balance entre la precisión y *recall*. Es decir, se sacrifica *recall* por precisión. De acuerdo a la definición del parámetro $MinSize$ el *recall* disminuye porque se eliminan pasajes con cierta cantidad de oraciones (pasajes pequeños). Por otro lado, la precisión aumenta debido a que se eliminan falsos positivos producto de pasajes pequeños. Otra forma de verlo es: mientras mayor sea el pasaje

mayor la probabilidad de estar en lo correcto. Similarmente sucede con la granularidad: mientras mayor sea el pasaje mayor probabilidad de detectar un pasaje como uno solo.

Para esta configuración, el mejor resultado se obtiene para $MinSize$ igual a 2. Sin embargo, similar a la sección del parámetro $MinSentLength$, cambiamos el valor del parámetro $MinPlagLength$ a 150 en la configuración de los experimentos. Esto lo hacemos porque ambos parámetros ($MinSize$ y $MinSentLength$) están relacionados con pasajes pequeños principalmente. En la figura 5.7 se muestran los experimentos para esta nueva configuración.

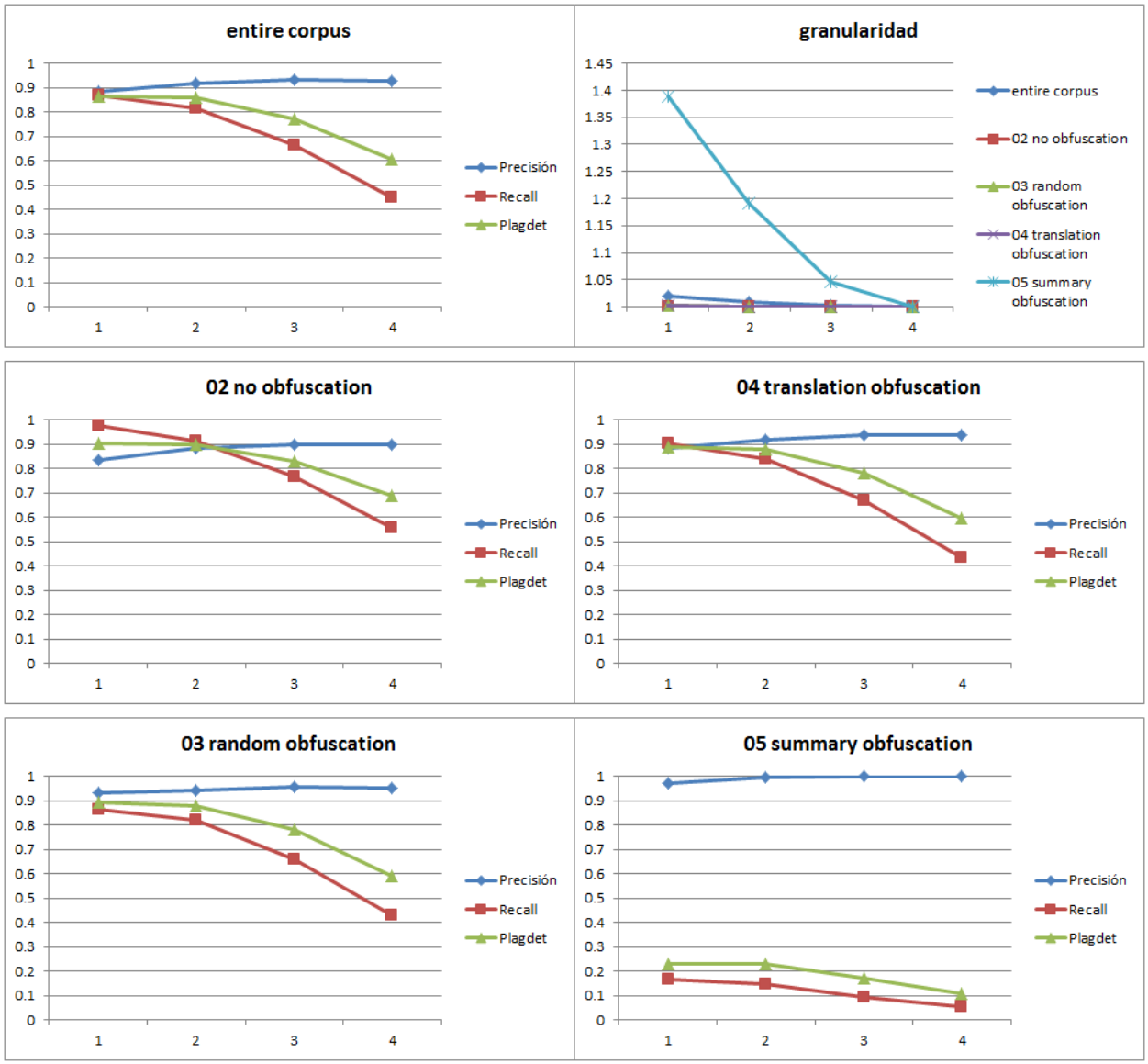


Figura 5.7 Precisión, *recall*, *plagdet* y granularidad respecto a MinSize (2)

En la gráfica para todo el corpus (*entire corpus*) se puede observar que el mejor resultado se obtiene para $MinSize = 1$, esto debido a que el parámetro $MinPlagLength = 150$ eliminó la mayor parte de los casos que causaban que el modelo fallara con la configuración anterior.

5.3.5 Parámetro MaxGap

En esta sección mostramos el comportamiento del modelo en función del parámetro $MaxGap$. Este parámetro nos define cuál es la separación máxima, en número de oraciones, que puede existir en dos oraciones seleccionadas para ser consideradas adyacentes. En la figura 5.8 se muestra dicho comportamiento para valores de $MaxGap$ de 0 a 19 con la siguiente configuración:

Parámetro	**Valor**
$RemSw$	0
$MinSentLength$	3
$tf \cdot idf$	3
$t1$	0.33 (CM)
$t2$	0.33 (DC)
$MaxGap$	-
$MinSize$	1
$t3$	0.33 (DC)
$MinPlagLength$	150

En las gráficas se puede observar que los resultados para todo el corpus mejoran al aumentar $MaxGap$ para los primeros valores. Sin embargo para el sub-corpus sin cambios (*02 no obfuscation*) los resultados decrecen. Esto se debe al hecho de que generalmente las oraciones que no fueron seleccionadas por las medidas de similitud pero fueron incluidas a los casos de plagio son falsos positivos. Mientras tanto para el sub-corpus basado en resúmenes (*05 summary obfuscation*) los resultados aumentan. Este comportamiento se debe a que se incluyen oraciones que no cumplieron con los umbrales de similitud pero que en muchos casos pertenecen al pasaje plagiado, lo que aumenta el *recall*; en el caso donde estas oraciones no pertenezcan al caso plagiado afecta negativamente la precisión.

La disminución paulatina de la granularidad se debe al hecho de que a mayor separación entre oraciones, mayor tamaño de los casos de plagio; y por lo tanto mayor probabilidad de detectar un caso de plagio como uno solo.

El mejor resultado obtenido en los experimentos es para $MaxGap = 4$.

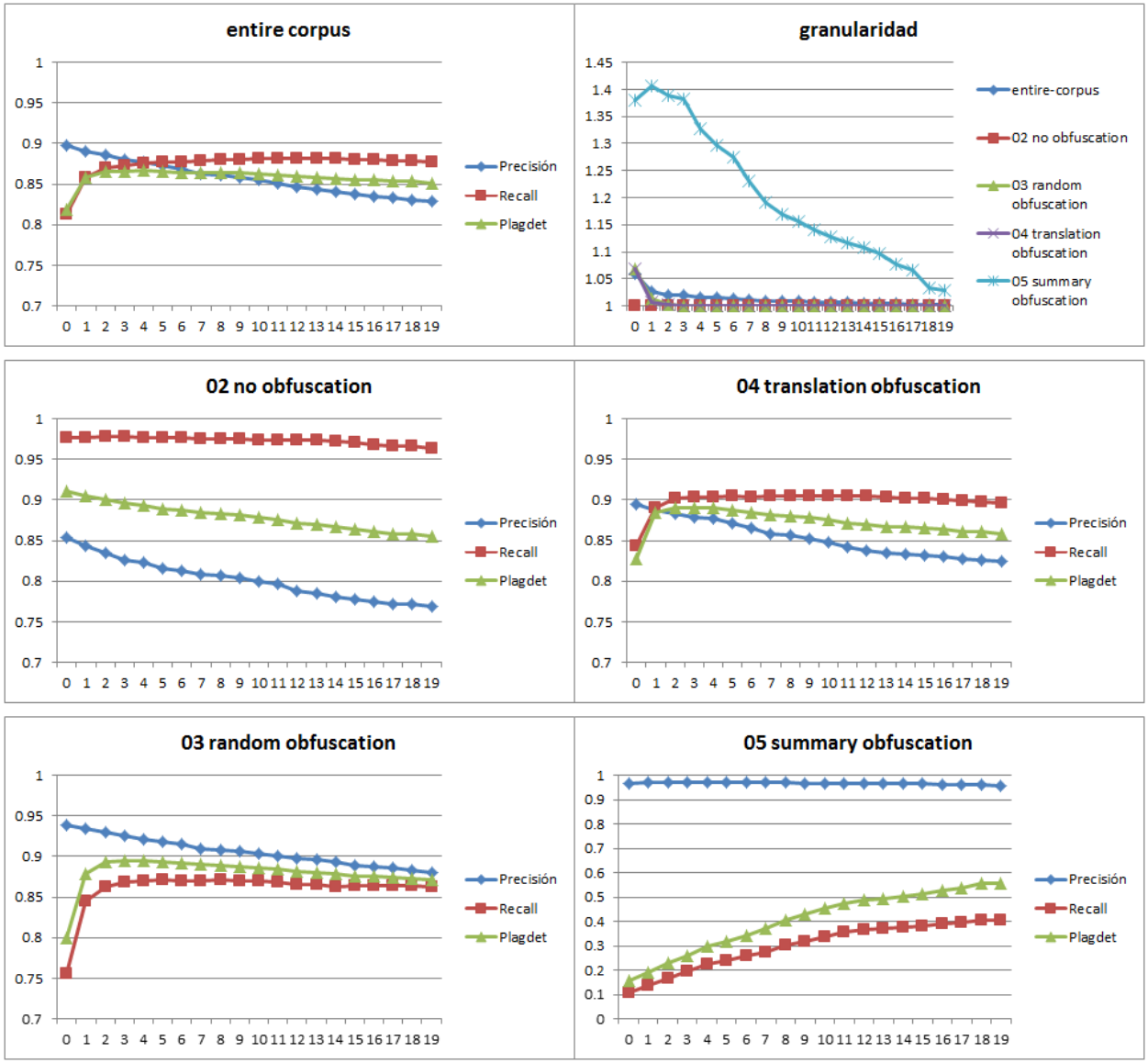


Figura 5.8 Precisión, *recall*, *plagdet* y granularidad respecto a MaxGap

5.3.6 Umbrales t1, t2 y t3

En esta sección mostramos el proceso de optimización de los parámetros t1, t2 y t3 correspondientes a la primera y segunda medida de similitud en el módulo de selección y a la similitud de pasajes en el módulo de post-procesamiento. Los experimentos se dividen en dos casos dependiendo de la medida de similitud usada como se muestra en la figura 5.9.

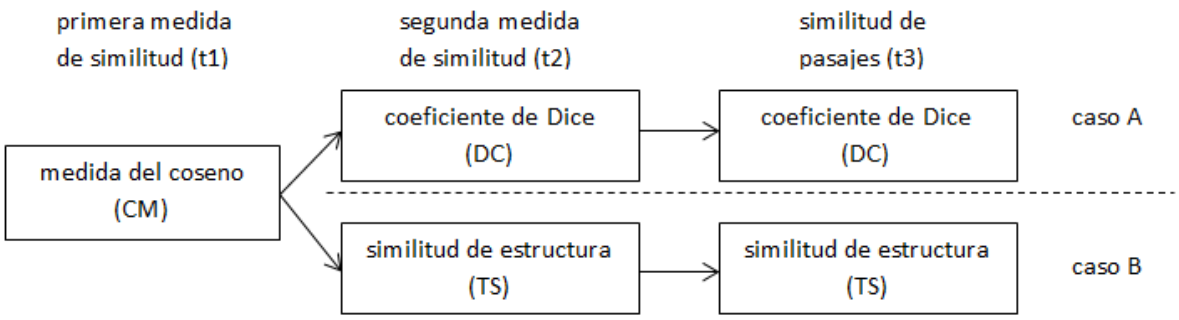


Figura 5.9 Casos para optimización de los parámetros t1, t2 y t3

Para los experimentos se usó la siguiente configuración:

Parámetro	**Valor**
$RemSw$	0
$MinSentLength$	3
$tf \cdot idf$	3
$t1$	- (CM)
$t2$	-
$MaxGap$	4
$MinSize$	1
$t3$	-
$MinPlagLength$	150

En la figura 5.10 se muestran los resultados optimizados para el parámetro t1. Para esto le asignamos -1 a los umbrales $t2$ y $t3$, equivalente a no usar los módulos de segunda medida de similitud y similitud de pasajes respectivamente. Por lo mismo, este resultado es el mismo para ambos casos, A y B.

Se obtiene el mejor resultado para $t1 = 0.38$, mismo que será usado junto a t2 para optimizar el umbral $t3$.

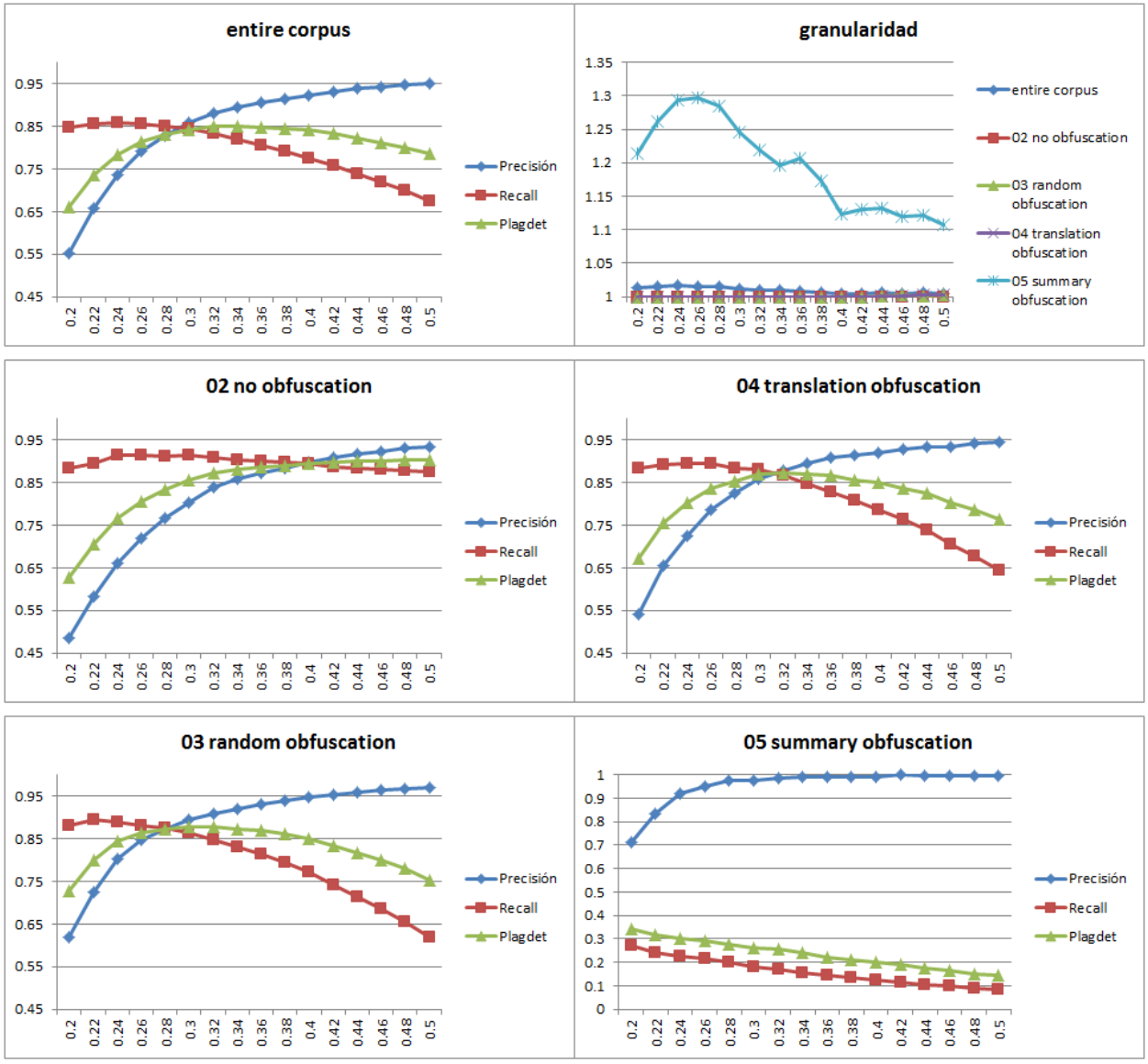


Figura 5.10 Resultados respecto a t1 para el caso A y B

Ahora optimizamos los resultados para el parámetro $t2$ asignándole -1 a los umbrales $t1$ y $t3$, como se muestra en la figura 5.11 para el caso A y en la figura 5.12 para el caso B.

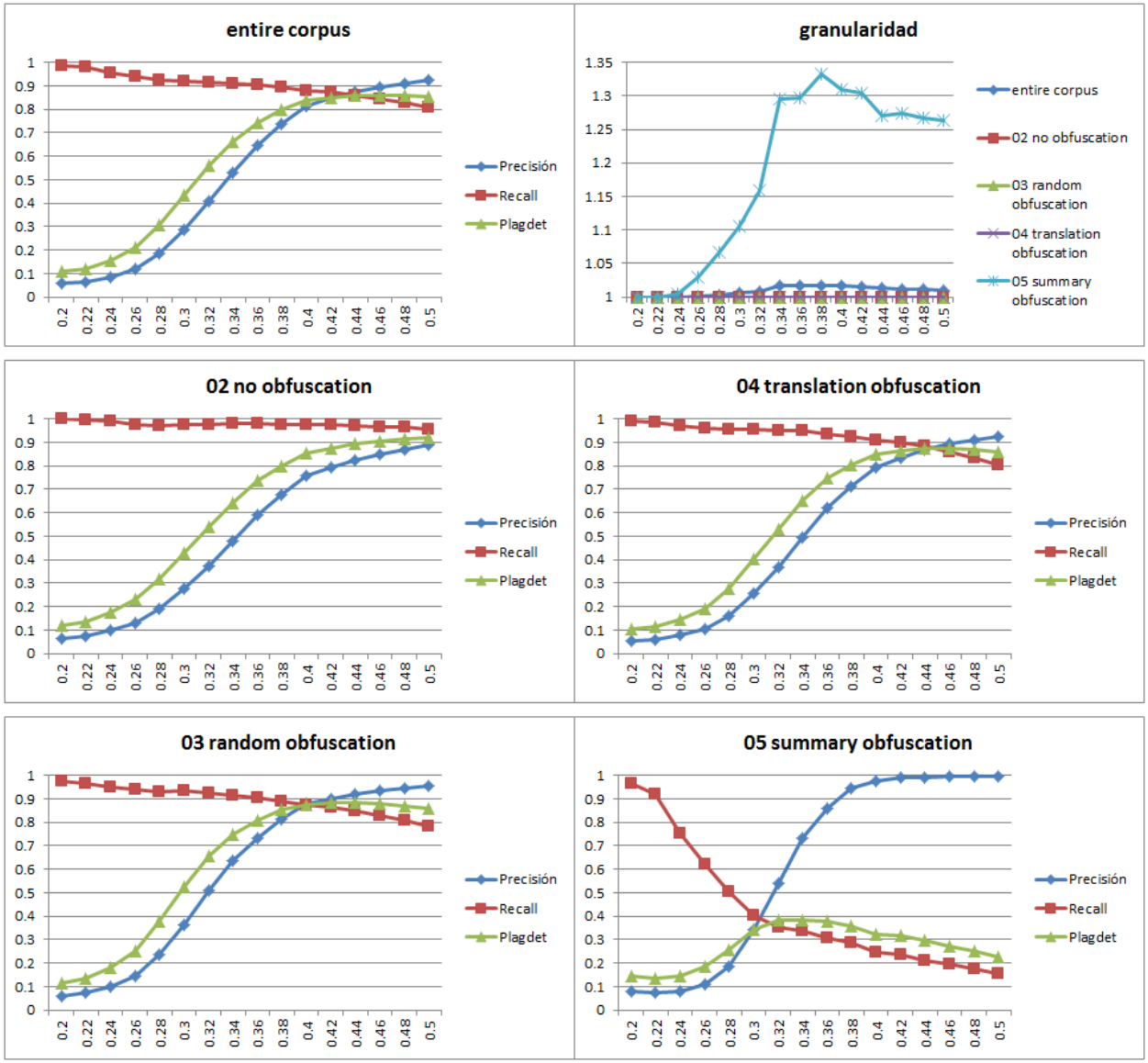


Figura 5.11 Resultados respecto a t2 para el caso A

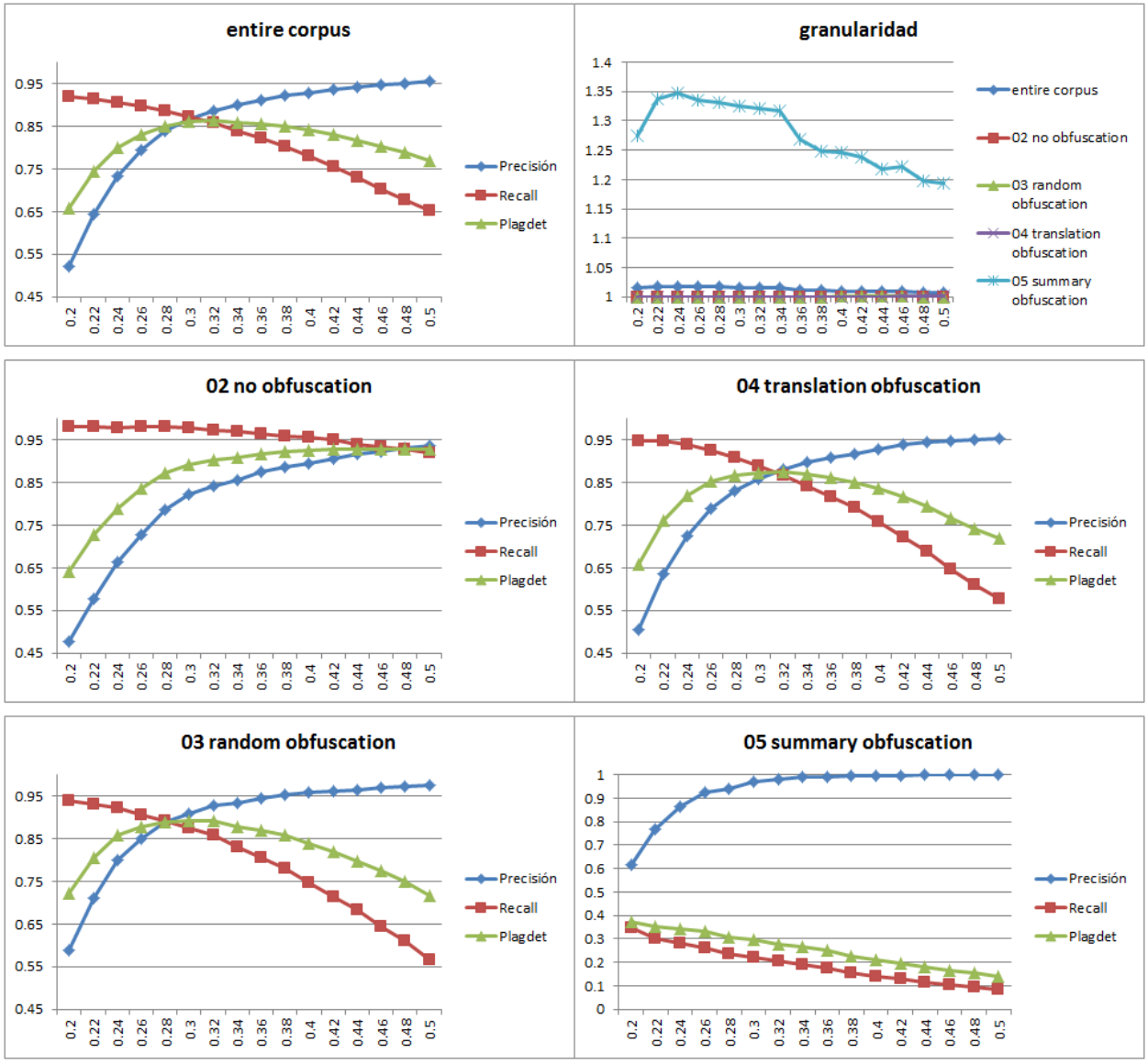


Figura 5.12 Resultados respecto a t2 para el caso B

Para el caso A el mejor resultado se obtiene para $t2 = 0.44$ mientras que para el caso B se obtiene $t2 = 0.32$. El resultado final del módulo de selección es la intersección de los resultados de la primera medida de similitud ($t1$) y la segunda medida de similitud ($t2$).

A continuación optimizamos los resultados para el parámetro t3 usando el mejor valor de la medida *plagdet* de $t1$ y $t2$ para sus respectivos casos, como se muestra en la figura 5.13 para el caso A y en la figura 5.14 para el caso B.

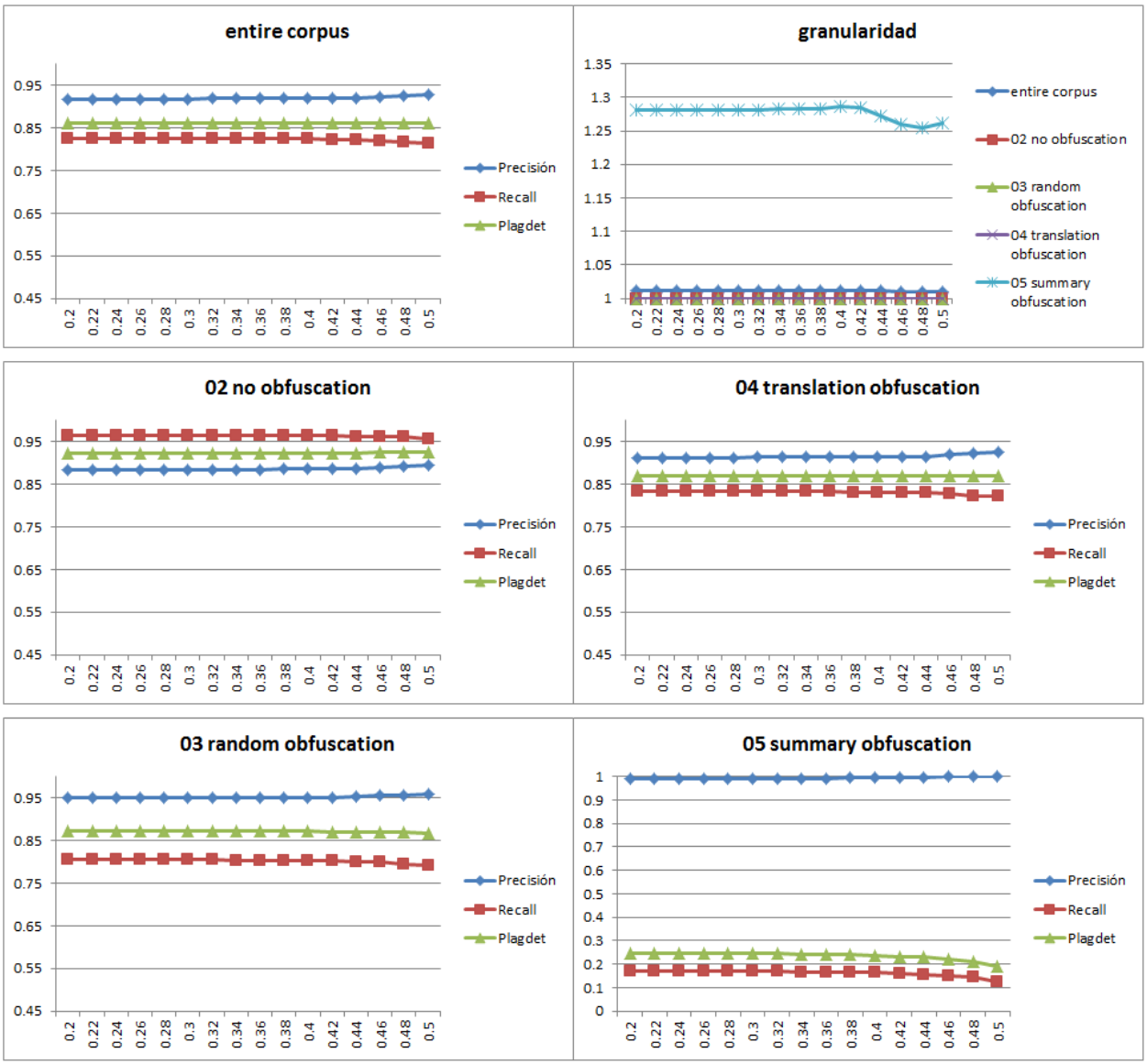


Figura 5.13 Resultados respecto a t3 para el caso A

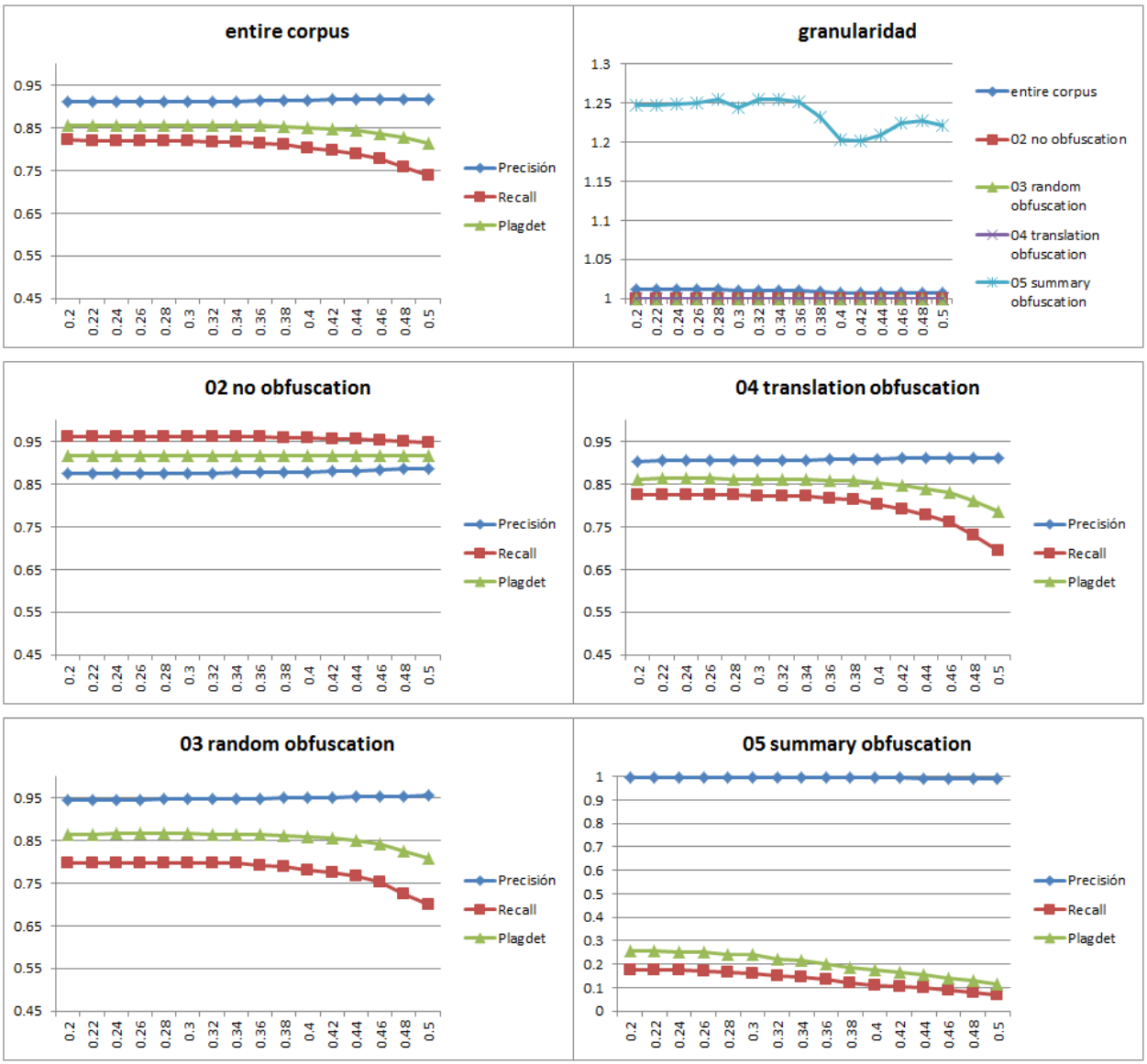


Figura 5.14 Resultados respecto a t3 para el caso B

El mejor resultado, aunque imperceptible en la gráfica, se obtiene para $t3 = 0.32$ para el caso A y $t3 = 0.24$ para el caso B. En los resultados generales el caso A superó al caso B. Esto nos dice que es mejor usar el coeficiente de Dice en lugar de la similitud de estructura para esta tarea.

Una observación importante es el hecho de que obtenemos mejores resultados con $t1 = 0.33, t2 = 0.33$ y $t3 = 0.33$ (valores elegidos hipotéticamente) como se muestra en la figura 5.8; que usando los valores obtenidos por la optimización de $t1$, $t2$ y $t3$. Esto se debe a que tomamos los mejores resultados de $t1$ y $t2$ con respecto a la medida *plagdet*, en lugar de tomar los mejores resultados o algún valor balanceado con respecto a la medida *recall* como lo hicimos en la optimización de $RemSw$ y $MinSentLength$. Luego de hacer lo anterior, la similitud de

pasajes no tiene mucho espacio para mejorar los resultados como se muestra en la figura 5.13 y en la figura 5.14.

5.4 Comparación contra los algoritmos que participaron en el PAN 2013

En esta sesión mostramos el resultado obtenido en el corpus de prueba del PAN 2013 en comparación con los sistemas que participaron en los concursos PAN 2012 y PAN 2013. Los resultados los dividimos en tablas de acuerdo a las medidas *plagdet*, precisión, *recall* y granularidad. A nuestro método lo nombramos SGSPLAG. Los resultados fueron extraídos de [8].

La configuración del modelo está en dependencia de la optimización de los parámetros de las secciones anteriores. Los parámetros para esta configuración son:

Parámetro	**Valor**
$RemSw$	0
$MinSentLength$	3
$tf \cdot idf$	3
$t1$	0.33 (CM)
$t2$	0.33 (DC)
$MaxGap$	4
$MinSize$	1
$t3$	0.33 (DC)
$MinPlagLength$	150

Tabla 5.1 Medida *plagdet* para los diferentes sub-corpus y el corpus entero

Equipo	Año	None	Random	Translation	Summary	Entire corpus
SGSPLAG	-	0.90418	0.88829	0.89133	0.29488	0.86902
Kong	2012	0.87249	0.83242	0.85212	0.43635	0.83679
Oberreuter	2012	0.9417	0.74955	0.84618	0.13208	0.82678
Torrejón	2013	0.92586	0.74711	0.85113	0.34131	0.8222
Kong	2013	0.8274	0.82281	0.85181	0.43399	0.81896
Palkovskii	2012	0.88161	0.79692	0.74032	0.27507	0.79155
Torrejón	2012	0.88222	0.70151	0.80112	0.44184	0.78767
Suchomel	2013	0.81761	0.75276	0.67544	0.61011	0.74482
Suchomel	2012	0.89848	0.65213	0.63088	0.50087	0.73224
Saremi	2013	0.84963	0.65668	0.70903	0.11116	0.69913
Shrestha	2013	0.89369	0.66714	0.62719	0.1186	0.69551
Kueppers	2012	0.81977	0.51602	0.56932	0.13848	0.62772
Palkovskii	2013	0.82431	0.49959	0.60694	0.09943	0.61523
Nourian	2013	0.90136	0.35076	0.43864	0.11535	0.57716
Sánchez-Vega	2012	0.52179	0.45598	0.44323	0.28807	0.45923
Baseline	2013	0.93404	0.07123	0.1063	0.04462	0.42191
Gillam	2012	0.87655	0.04723	0.01225	0.00218	0.41373
Gillam	2013	0.85884	0.04191	0.01224	0.00218	0.40059
Jayapal	2013	0.3878	0.18148	0.18181	0.0594	0.27081
Jayapal	2012	0.34758	0.12049	0.10504	0.04541	0.20169

Tabla 5.2 Precisión para los diferentes sub-corpus y el corpus entero

Equipo	Año	None	Random	Translation	Summary	Entire corpus
Nourian	2013	0.92921	0.96274	0.95856	0.99972	0.94707
Jayapal	2012	0.98542	0.95984	0.8959	0.83259	0.94507
Baseline	2013	0.88741	0.98101	0.97825	0.91147	0.92939
Torrejón	2013	0.9006	0.90996	0.89514	0.9075	0.89484
Oberreuter	2012	0.89037	0.87921	0.90328	0.98983	0.89443
Gillam	2012	0.88128	0.95572	0.97273	0.99591	0.88532
Gillam	2013	0.88088	0.95968	0.97273	0.99591	0.88487
SGSPLAG	-	0.83978	0.91042	0.88197	0.96363	0.88145
Jayapal	2013	0.91989	0.92314	0.85653	0.68832	0.87901
Shrestha	2013	0.80933	0.92335	0.88008	0.90455	0.87461
Kueppers	2012	0.83258	0.89889	0.89985	0.86239	0.86923
Saremi	2013	0.82676	0.9181	0.84819	0.946	0.86509
Kong	2012	0.80786	0.89367	0.85423	0.96399	0.85297
Suchomel	2012	0.81678	0.87581	0.85151	0.87478	0.84437
Kong	2013	0.76077	0.86224	0.85744	0.96384	0.82859
Torrejón	2012	0.81313	0.83881	0.81159	0.92666	0.8254
Palkovskii	2012	0.79219	0.84844	0.83218	0.94736	0.82371
Palkovskii	2013	0.79971	0.93137	0.82207	0.67604	0.81699
Suchomel	2013	0.69323	0.82973	0.68494	0.67088	0.72514
Sánchez-Vega	2012	0.4034	0.49524	0.373	0.45184	0.39857

Tabla 5.3 *Recall* para los diferentes sub-corpus y el corpus entero

Equipo	Año	None	Random	Translation	Summary	Entire corpus
SGSPLAG	-	0.97928	0.86825	0.90194	0.21190	0.87396
Kong	2012	0.94836	0.77903	0.85003	0.29892	0.82449
Kong	2013	0.90682	0.78682	0.84626	0.30017	0.81344
Saremi	2013	0.95416	0.68877	0.80473	0.10209	0.77123
Oberreuter	2012	0.99932	0.65322	0.79587	0.07076	0.76864
Suchomel	2013	0.99637	0.68886	0.66621	0.56296	0.76593
Torrejón	2013	0.95256	0.6337	0.81124	0.21593	0.7619
Palkovskii	2012	0.99379	0.7513	0.66672	0.16089	0.76181
Torrejón	2012	0.96414	0.60283	0.79092	0.29007	0.75324
Shrestha	2013	0.99902	0.71461	0.63618	0.09897	0.73814
Suchomel	2012	0.99835	0.51946	0.50106	0.35305	0.64667
Sánchez-Vega	2012	0.74452	0.43502	0.58133	0.22161	0.56225
Palkovskii	2013	0.85048	0.3642	0.49667	0.08082	0.53561
Kueppers	2012	0.83854	0.36865	0.42427	0.09265	0.51074
Nourian	2013	0.87626	0.23609	0.28568	0.07622	0.43381
Jayapal	2013	0.8604	0.18182	0.19411	0.07236	0.38187
Baseline	2013	0.9996	0.04181	0.08804	0.03649	0.34223
Gillam	2012	0.87187	0.02422	0.00616	0.00109	0.26994
Gillam	2013	0.83788	0.02142	0.00616	0.00109	0.2589
Jayapal	2012	0.51885	0.11148	0.09195	0.04574	0.22287

Tabla 5.4 Granularidad para los diferentes sub-corpus y el corpus entero

Equipo	Año	None	Random	Translation	Summary	Entire corpus
Gillam	2012	1	1	1	1	1
Gillam	2013	1	1	1	1	1
Oberreuter	2012	1	1	1	1	1
Palkovskii	2012	1	1	1	1	1
Torrejón	2012	1	1	1	1	1
Suchomel	2013	1	1	1	1.00476	1.00028
Suchomel	2012	1	1	1	1.0061	1.00032
Torrejón	2013	1	1	1	1.03086	1.00141
Kong	2012	1	1	1	1.06452	1.00282
Kong	2013	1	1	1	1.07742	1.00336
SGSPLAG	-	1.00000	1.00085	1.00080	1.26289	1.01388
Sánchez-Vega	2012	1.00394	1.022	1.03533	1.04523	1.02196
Kueppers	2012	1.02687	1.01847	1.01794	1.31061	1.03497
Nourian	2013	1.00092	1.11558	1.00485	1.34234	1.04343
Palkovskii	2013	1	1.06785	1.02825	1.73596	1.07295
Shrestha	2013	1.00083	1.30962	1.26184	1.83696	1.22084
Saremi	2013	1.06007	1.29511	1.24204	2.15556	1.2445
Baseline	2013	1.00912	1.18239	1.86726	1.97436	1.27473
Jayapal	2012	2.87916	2.1553	2.00578	2.75743	2.45403
Jayapal	2013	3.90017	2.19096	2.34218	3.60987	2.90698

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En este trabajo de tesis se propuso e implementó un modelo computacional para la detección automática de plagio, específicamente en la tarea de alineación de textos. El modelo propuesto consta de cuatro módulos donde se implementaron ideas originales como:

- El uso de un esquema de ponderación de características tf·idf donde las oraciones son consideradas documentos.
- Un método de integración de pares de oraciones plagiadas.
- Un método de eliminación de pasajes solapados.

Además se efectuaron pruebas exhaustivas para analizar el comportamiento de los módulos del método propuesto y su influencia en el resultado final. Los resultados del modelo superan a los algoritmos presentados en el estado del arte que han sido aplicados sobre el mismo corpus.

Como parte de este trabajo de tesis se participa actualmente en la tarea de Alineación de Texto como parte de la detección automática de plagio en el contexto del 11º laboratorio de evaluación en detección de plagio, identificación de autoría y mal uso de software social (PAN2014). Esta competencia se lleva a cabo como parte del congreso internacional CLEF in Sheffield, UK. Los resultados de esta competencia se esperan a finales de mayo.

6.2 Trabajo futuro

- No asumir independencia de los parámetros del método y efectuar experimentos donde se establecen dependencias entre ellos.
- Prueba del método propuesto en otros corpus.
- Obtención dinámica del parámetro $MaxGap$ a través de un algoritmo recursivo.
- Detección automática del posible tipo de plagio y ajuste de los parámetros del método de acuerdo a esta detección.
- Uso de características lingüísticamente motivadas como:
    - Uso de n-gramas de características gramaticales.
    - Uso de n-gramas sintácticos.
    - Separación de oraciones subordinadas.

7 Bibliografía

[1] Española, R. A. (s.f.). *Diccionario de la Lengua Española - Vigésima segunda edición*. Recuperado el 2013, de http://lema.rae.es/drae/

[2] University, P. (2011). *Academic Integrity at Princeton.* Recuperado el 2013, de http://www.princeton.edu/pr/pub/integrity/pages/academic-integrity-2011b.pdf

[3] Bouville, M. (2008). Plagiarism: Words and Ideas. *Science and Engineering Ethics* , 311-322.

[4] *What is plagiarism?* (s.f.). Recuperado el 2013, de http://www.plagiarism.org/plagiarism-101/what-is-plagiarism/

[5] Stein, B., Koppel, M., & Stamatatos, E. (2007). Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection. *SIGIR Forum.*

[6] Eissen, B. S. (2006). Near Similarity Search and Plagiarism Analysis. *Proceeding of 29th Annual Conference of the German Classification Society*, 430-437.

[7] *PAN Workshop and Competition: Uncovering Plagiarism, Authorship and Social Software Misuse*. (2013). Obtenido de http://pan.webis.de/

[8] Potthast, M., Hagen, M., Gollub, T., Tippmann, M., Kiesel, J., Rosso, P., y otros. (2013). Overview of the 5th International Competition on Plagiarism Detection. *CLEF 2013 Evaluation Labs and Workshop.* Valencia, Spain.

[9] Leilei, K., Haoliang, Q., Shuai, W., Cuixia, D., Suhong, W., & Yong, H. (2012). Approaches for Candidate Document Retrieval and Detailed Comparison of Plagiarism Detection. *Notebook for PAN at CLEF 2012.*

[10] LeiLei, K., Haoliang, Q., Cuixia, D., Mingxing, W., & Zhongyuan, H. (2013). Approaches for Source Retrieval and Text Alignment of Plagiarism Detection. *Notebook for PAN at CLEF 2013.*

[11] Rodríguez-Torrejón, D. A., & Martín-Ramos, J. M. (2012). Detailed Comparison Module in CoReMo 1.9 Plagiarism Detector. *Notebook for PAN at CLEF 2012.*

[12] Rodríguez-Torrejón, D. A., & Martín-Ramos, J. M. (2013). Text Alignment Module in CoReMo 2.1 Plagiarism Detector. *Notebook for PAN at CLEF 2013.*

[13] Rodríguez-Torrejón, D. A., & Martín-Ramos, J. M. (2010). Detección de plagio en documentos. Sistema externo monolingüe de altas prestaciones basado en n-gramas contextuales. *Procesamiento del Lenguaje Natural , 45*, 49-57.

[14] Rodriguez-Torrejon, D. A., & Martín-Ramos, J. M. (2012). N-gramas de Contexto Cercano para mejorar la Detección de Plagio. *II Congreso Español de Recuperación de Información.*

[15] Potthast, M., Gollub, T., Hagen, M., Graßegger, J., & Kiesel, J. (2012). Overview of the 4th International Competition on Plagiarism Detection. *CLEF 2012 Evaluation Labs and Workshop – Working Notes Papers.* Rome, Italy.

[16] Suchomel, Š., Kasprzak, J., & Brandejs, M. (2012). Three way search engine queries with multi-feature document comparison for plagiarism detection. *Notebook for PAN at CLEF 2012.*

[17] Suchomel, Š., Kasprzak, J., & Brandejs, M. (2013). Diverse Queries and Feature Type Selection for Plagiarism Discovery. *Notebook for PAN at CLEF 2013.*

[18] Stamatatos, E. (2011). Plagiarism Detection Using Stopword n-grams. *Journal of the American Society for Information Science and Technology* .

[19] Shrestha, P., & Solorio, T. (2013). Using Variety of n-Grams for the Detection of Different Kinds of Plagiarism. *Notebook for PAN at CLEF 2013.*

[20] Palkovskii, Y., & Belov, A. (2012). Applying Specific Clusterization and Fingerprint Density Distribution with Genetic Algorithm Overall Tuning in External Plagiarism Detection. *Notebook for PAN at CLEF 2012.*

[21] Palkovskii, Y., & Belov, A. (2013). Using Hybrid Similarity Methods for Plagiarism Detection. *Notebook for PAN at CLEF 2013.*

[22] Grozea, C., & Popescu, M. (2011). The Encoplot Similarity Measure for Automatic Detection of Plagiarism - Extended Technical Report. *http://brainsignals.de/encsimTR.pdf.*

[23] Grozea, C., & Popescu, M. (2010). Encoplot - Performance in the Second International Plagiarism Detection Challenge. *Lab Report for PAN at CLEF 2010.*

[24] Küppers, R., & Conrad, S. (2012). A Set-Based Approach to Plagiarism Detection. *Notebook for PAN at CLEF 2012.*

[25] Potthast, M., Stein, B., Barrón-Cedeño, A., & Rosso, P. An Evaluation Framework for Plagiarism Detection. *Proceedings of the 23rd International Conference on Computational Linguistics, COLING 2010.* Beijing, China.

[26] Gillam, L., Newbold, N., & Cooke, N. (2012). Educated guesses and equality judgements: using search engines and pairwise match for external plagiarism detection. *Notebook for PAN at CLEF 2012.*

[27] Gillam, L. (2013). Guess again and see if they line up: Surrey's runs at plagiarism detection. *Notebook for PAN at CLEF 2013.*

[28] Kasprzak, J., Brandejs, M., & Křipač, M. (2009). Finding Plagiarism by Evaluating Document Similarities. *The 25th edition of the Annual Conference of Spanish Society for Natural Language Processing.*

[29] Manning, C. D., & Schütze, H. (1999). Some Background on Information Retrieval. En C. D. Manning, & H. Schütze, *Foundations of Statistical Natural Language Processing.* Cambridge, MA: Massachusetts Institute of Technology Press.

[30] Sidorov, G. (2013). *Construcción no lineal de n-gramas en la lingüística.* Sociedad Mexicana de Inteligencia Artificial.

[31] Lovins, J. B. (1968). Development of a stemming algorithm. *Translation and Computational Linguistics , 11*, 22-31.

[32] Porter, M. (2006). An algorithm for suffix stripping. *Program: electronic library and information systems , 40*, 211-218.

[33] Manning, C. D., & Schütze, H. (1999). Evaluation Measures. En C. D. Manning, & H. Schütze, *Foundations of Statistical Natural Language Processing.* Cambridge, MA: Massachusetts Institute of Technology Press.

[34] Rijsbergen, C. J. (1979). Evaluation. En C. J. RIJSBERGEN, *Information Retrieval* (2nd ed.). London, GB: Butterworths.

[35] Manning, C. D., & Schütze, H. (1999). The Vector Space Model. En C. D. Manning, & H. Schütze, *Foundations of Statistical Natural Language Processing.* Cambridge, MA: Massachusetts Institute of Technology.

[36] Manning, C. D., & Schütze, H. (1999). Vector space measures. En C. D. Manning, & H. Schütze, *Foundations of Statistical Natural Language Processing.* Cambridge, MA: Massachusetts Institute of Technology Press.

[37] Guthrie, D., Allison, B., Liu, W., Guthrie, L., & Wilks, Y. (2006). A Closer Look at Skip-gram Modelling. *Proceedings of the Fifth international Conference on Language Resources and Evaluation LREC.* Genoa, Italy.

[38] Kiss, T., & Strunk, J. (2006). Unsupervised multilingual sentence boundary detection. *Computational Linguistics , 32*, 485-525.

[39] Potthast, M., Stein, B., Eiselt, A., Barrón-Cedeño, A., & Rosso, P. (2009). Overview of the 1st International Competition on Plagiarism Detection. *SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse*, 1-9. San Sebastian, Spain.

[40] Potthast, M., Stein, B., Eiselt, A., Barrón-Cedeño, A., & Rosso, P. (2010). Overview of the 2nd International Benchmarking Workshop on Plagiarism Detection. *Proceedings of PAN at CLEF 2010: Uncovering Plagiarism, Authorship, and Social Software Misuse.* Padua, Italy.

[41] Cedeño, L. A. (2008). *Detección automática de plagio en texto, Tesis de Maestría en Inteligencia artificial, reconocimiento de formas e imagen digital.* Valencia, España.

ANEXO A - Algoritmo de Integración Bilateral Alternado

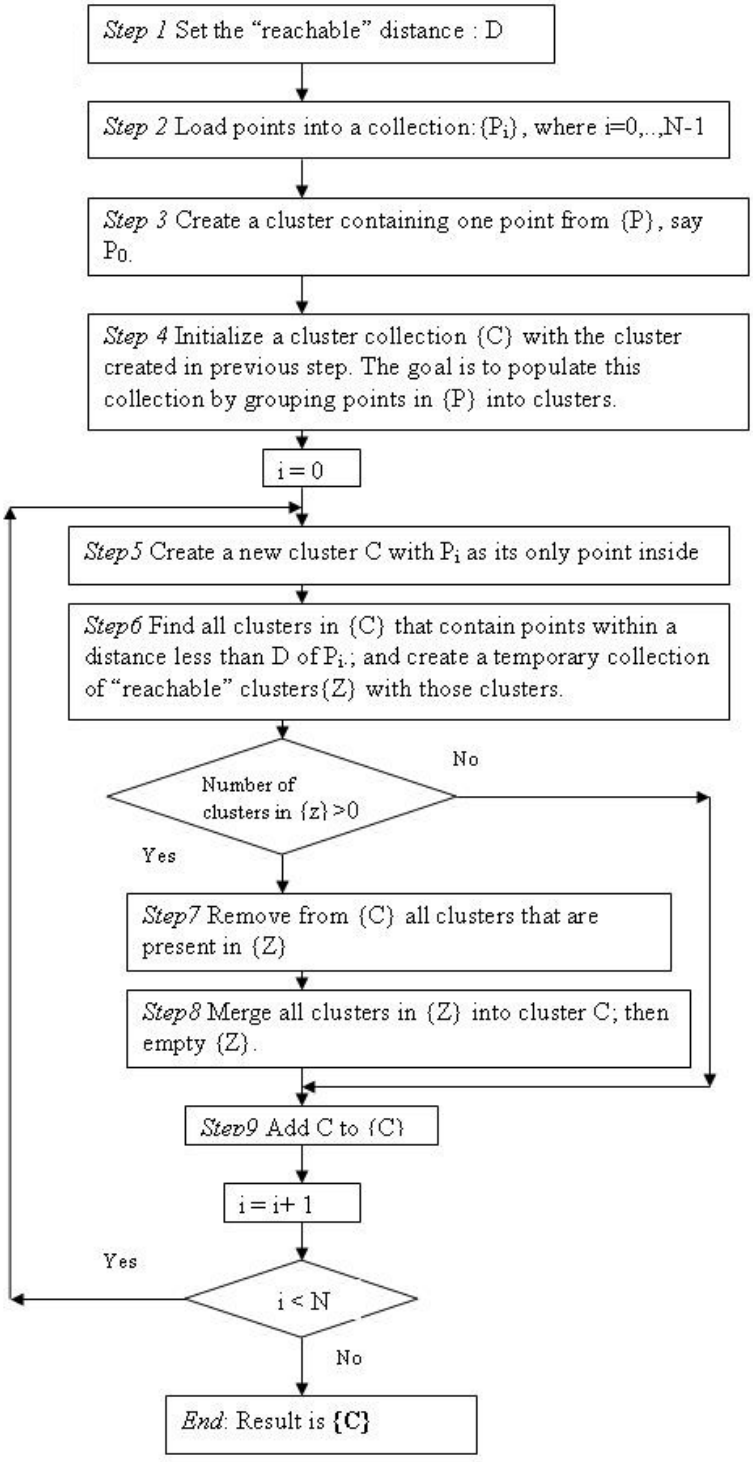
Bilateral Alternating Merging Algorithm

Input: Similar sentence list casesList

Output: The collection of the pairs(splg , ssrc) from the plagiarism fragment of the suspicious document splg and the plagiarism source document fragment ssrc

```
1 public void merger(List<String> casesList, int sign) {
2       if (sign = -1) {
3           leftSort(casesList);
4           if (end >= leftEndge && end <= rightEndge) {
5           resultList.add(dto);
6             } else {
7           for (i = 1..casesList.size()) {
8               if (|now – last| > dist) {
9                       checkAdjacent();
10                    merger();
11              }
12         }
13       } else {
14            rightSort(casesList);
15             if (endgeFirstInfs=endgeEndInfs && end >= leftEndge&& end <=
16                 rightEndge) {
17                   resultList.add(dto);
18            } else {
19                for (i = 1..casesList.size()) {
20                    if (|now – last| > dist) {
21                            checkAdjacent();
22                            merger();
23                      }
24                }
25           }
26      return resultList;
27      }
```

ANEXO B - Algoritmo genérico de agrupamiento basado en la distancia Euclidiana



ANEXO C - Código fuente del modelo propuesto en Python

```
#!/usr/bin/env python
""" Plagiarism detection system.
    Program for the PAN 2014 Plagiarism Detection task.
"""
__author__ = 'Miguel Angel Sanchez Perez'
__email__ = 'masp1988 at hotmail dot com'
__version__ = '1.0'

import os
import sys
import xml.dom.minidom
import codecs
import nltk
import re
import math
import time
import flist

def sum_vect(dic1,dic2):
    res=dic1
    for i in dic2.keys():
        if res.has_key(i):
            res[i]+=dic2[i]
        else:
            res[i]=dic2[i]
    return res

def ss_treat(list_dic,offsets,min_sentlen,rssent):
    if rssent=='no':
        i=0
        range_i=len(list_dic)-1
        while i<range_i:
            if sum(list_dic[i].values())<min_sentlen:
                list_dic[i+1]=sum_vect(list_dic[i+1],list_dic[i])
                del list_dic[i]
                offsets[i+1]=(offsets[i][0],offsets[i+1][1]+offsets[i][1])
                del offsets[i]
                range_i-=1
            else:
                i=i+1
    else:
        i=0
        range_i=len(list_dic)-1
        while i<range_i:
            if sum(list_dic[i].values())<min_sentlen:
                del list_dic[i]
                del offsets[i]
                range_i-=1
            else:
                i=i+1

def tf_idf(list_dic1,voc1,list_dic2,voc2):
    idf=sum_vect(voc1,voc2)
    td=len(list_dic1)+len(list_dic2)
```

```
    for i in range(len(list_dic1)):
        for j in list_dic1[i].keys():
            list_dic1[i][j]*=math.log(td/float(idf[j]))
    for i in range(len(list_dic2)):
        for j in list_dic2[i].keys():
            list_dic2[i][j]*=math.log(td/float(idf[j]))

def eucl_norm(d1):
        norm=0.0
        for val in d1.values():
            norm+=float(val*val)
        return math.sqrt(norm)

def cosine_measure(d1,d2):
    dot_prod=0.0
    det=eucl_norm(d1)*eucl_norm(d2)
    if det==0:
        return 0
    for word in d1.keys():
        if d2.has_key(word):
            dot_prod+=d1[word]*d2[word]
    return dot_prod/det

def dice_coeff(d1,d2):
    intj=0
    for i in d1.keys():
        if d2.has_key(i):
            intj+=1
    return 2*intj/float(len(d1)+len(d2))

def adjacent(a,b,th):
    if abs(a-b)-th-1<=0:
        return True
    else:
        return False

def integrate_cases(ps,src_gap,susp_gap,src_size,susp_size):
    ps.sort(key=lambda tup: tup[0])
    pss=[]
    sub_set=[]
    for pair in ps:
        if len(sub_set)==0:
            sub_set.append(pair)
        else:
            if adjacent(pair[0],sub_set[-1][0],susp_gap):
                sub_set.append(pair)
            else:
                if len(sub_set)>=susp_size:
                    pss.append(sub_set)
                sub_set=[pair]
    if len(sub_set)>=susp_size:
        pss.append(sub_set)
    psr=[]
    for pss_i in pss:
        pss_i.sort(key=lambda tup: tup[1])
        sub_set=[]
        for pair in pss_i:
```

```
            if len(sub_set)==0:
                sub_set.append(pair)
            else:
                if adjacent(pair[1],sub_set[-1][1],src_gap):
                    sub_set.append(pair)
                else:
                    if len(sub_set)>=src_size:
                        psr.append(sub_set)
                    sub_set=[pair]
        if len(sub_set)>=src_size:
            psr.append(sub_set)
    plags=[]
    for psr_i in psr:
        plags.append([(min([x[1] for x in psr_i]),max([x[1] for x in
psr_i])),(min([x[0] for x in psr_i]),max([x[0] for x in psr_i]))])
    return plags

def remove_overlap3(src_bow,susp_bow,plags):
    plags.sort(key=lambda tup: tup[1][0])
    print 'Before remove overlap (sorted)'
    for i in plags:
        print i
    print '\n'
    res=[]
    flag=0
    i=0
    while i<len(plags):
        print 'plags[i]',plags[i]
        cont_ol=0
        if flag==0:
            print 'Pivot',plags[i]
            for k in range(i+1,len(plags)):
                if plags[k][1][0]-plags[i][1][1]<=0:
                    cont_ol+=1
        else:
            print 'Pivot',res[-1]
            for k in range(i+1,len(plags)):
                if plags[k][1][0]-res[-1][1][1]<=0:
                    cont_ol+=1
        print 'cont_ol=',cont_ol,'   flag=',flag
        print 'To compare to'
        for h in plags[i+1:i+1+cont_ol]:
            print h
        print '\n'
        if cont_ol==0:
            if flag==0:
                res.append(plags[i])
                print 'res'
                for h in res:
                    print h
                print '\n'
            else:
                flag=0
            i+=1
        else:
            ind_max=i
            higher_sim=0.0
```

```
for j in range(1,cont_ol+1):
    if flag==0:
        sents_i=range(plags[i][1][0],plags[i][1][1]+1)
    else:
        sents_i=range(res[-1][1][0],res[-1][1][1]+1)
    sents_j=range(plags[i+j][1][0],plags[i+j][1][1]+1)
    print 'sents_i',sents_i
    print 'sents_j',sents_j
    sim_i_ol=0.0
    sim_j_ol=0.0
    sim_i_nol=0.0
    sim_j_nol=0.0
    cont_ol_sents=0
    cont_i_nol_sents=0
    cont_j_nol_sents=0
    for sent in sents_i:
        sim_max=0.0
        for k in range(plags[i][0][0],plags[i][0][1]+1):
            sim=cosine_measure(susp_bow[sent],src_bow[k])
            if sim>sim_max:
                sim_max=sim
        if sent in sents_j:
            sim_i_ol+=sim_max
            cont_ol_sents+=1
        else:
            sim_i_nol+=sim_max
            cont_i_nol_sents+=1
    for sent in sents_j:
        sim_max=0.0
        for k in range(plags[i+j][0][0],plags[i+j][0][1]+1):
            sim=cosine_measure(susp_bow[sent],src_bow[k])
            if sim>sim_max:
                sim_max=sim
        if sent in sents_i:
            sim_j_ol+=sim_max
        else:
            sim_j_nol+=sim_max
            cont_j_nol_sents+=1
    sim_i=sim_i_ol/cont_ol_sents
    if cont_i_nol_sents!=0:
        sim_i=sim_i+(1-sim_i)*sim_i_nol/float(cont_i_nol_sents)
    sim_j=sim_j_ol/cont_ol_sents
    if cont_j_nol_sents!=0:
        sim_j=sim_j+(1-sim_j)*sim_j_nol/float(cont_j_nol_sents)
    if sim_i>0.99 and sim_j>0.99:
        if len(sents_j)>len(sents_i):
            if sim_j>higher_sim:
                ind_max=i+j
                higher_sim=sim_j
    elif sim_j>sim_i:
        if sim_j>higher_sim:
            ind_max=i+j
            higher_sim=sim_j
if flag==0:
    res.append(plags[ind_max])
    print 'res flag=0'
    for h in res:
```

```
                    print h
                print '\n'
            elif ind_max!=i:
                del res[-1]
                res.append(plags[ind_max])
                print 'res flag=1'
                for h in res:
                    print h
                print '\n'
            i=i+cont_ol
            flag=1
    return res

def tokenize(text,voc={},offsets=[],sents=[],rem_sw='no'):
    """
    INPUT:  text: Text to be pre-processed
            voc: vocabulary used in the text with idf
            offsets: start index and length of each sentence
            sents: sentences of the text without tokenization
    OUTPUT: Returns a list of lists representing each sentence divided in
tokens
    """
    #text=text.replace(chr(0), ' ')
    sent_detector = nltk.data.load('tokenizers/punkt/english.pickle')
    sents.extend(sent_detector.tokenize(text))
    offsets.extend([(a,b-a) for (a,b) in sent_detector.span_tokenize(text)])
    sent_tokens=[nltk.TreebankWordTokenizer().tokenize(sent) for sent in
sents]
    stemmer=nltk.stem.porter.PorterStemmer()
    sent_tokens_pp=[]
    stopwords=flist.flist().words()
    cont=0
    for tokens in sent_tokens:
        if rem_sw=='no':
            temp={}
            for i in [stemmer.stem(word.lower()) for word in tokens if
re.match(r'([a-zA-Z]|[0-9])(.)*',word)]:
                if temp.has_key(i):
                    temp[i]+=1
                else:
                    temp[i]=1
        elif rem_sw=='50':
            stopwords=flist.flist().words50()
            temp={}
            for i in [stemmer.stem(word.lower()) for word in tokens if
re.match(r'([a-zA-Z]|[0-9])(.)*',word) and word.lower() not in stopwords]:
                if temp.has_key(i):
                    temp[i]+=1
                else:
                    temp[i]=1
        else:
            temp={}
            for i in [stemmer.stem(word.lower()) for word in tokens if
re.match(r'([a-zA-Z]|[0-9])(.)*',word) and word.lower() not in stopwords]:
                if temp.has_key(i):
                    temp[i]+=1
                else:
```

```
                    temp[i]=1
            if len(temp)>0:
                sent_tokens_pp.append(temp)
                for i in temp.keys():
                    if voc.has_key(i):
                        voc[i]+=1
                    else:
                        voc[i]=1
                cont=cont+1
            else:
                del offsets[cont]
        return sent_tokens_pp

def serialize_features(susp, src, features, outdir):
    """ Serialze a feature list into a xml file.
    The xml is structured as described in
    http://www.webis.de/research/corpora/pan-pc-12/pan12/readme.txt
    The filename will follow the naming scheme {susp}-{src}.xml and is
located
    in the current directory.  Existing files will be overwritten.

    Keyword arguments:
    susp     -- the filename of the suspicious document
    src      -- the filename of the source document
    features -- a list containing feature-tuples of the form
                ((start_pos_susp, end_pos_susp),
                 (start_pos_src, end_pos_src))
    """
    impl = xml.dom.minidom.getDOMImplementation()
    doc = impl.createDocument(None, 'document', None)
    root = doc.documentElement
    root.setAttribute('reference', susp)
    doc.createElement('feature')

    for f in features:
        feature = doc.createElement('feature')
        feature.setAttribute('name', 'detected-plagiarism')
        feature.setAttribute('this_offset', str(f[1][0]))
        feature.setAttribute('this_length', str(f[1][1] - f[1][0]))
        feature.setAttribute('source_reference', src)
        feature.setAttribute('source_offset', str(f[0][0]))
        feature.setAttribute('source_length', str(f[0][1] - f[0][0]))
        root.appendChild(feature)

    doc.writexml(open(outdir + susp.split('.')[0] + '-'
                      + src.split('.')[0] + '.xml', 'w'),
                 encoding='utf-8')


# Plagiarism pipeline
# ===================

""" The following class implement a very basic baseline comparison, which
aims at near duplicate plagiarism. It is only intended to show a simple
pipeline your plagiarism detector can follow.
Replace the single steps with your implementation to get started.
"""
```

```
class SGSPLAG:
    def __init__(self, susp, src, outdir):
        """ Parameters. """
        self.th1=0.33
        self.th2=0.33
        self.th3=0.33
        self.src_gap=2
        self.susp_gap=2
        self.src_size=2
        self.susp_size=2
        self.min_sentlen=5
        self.min_plaglen=150
        self.rssent='no'
        self.tf_idf_p='yes'
        self.rem_sw='no'

        self.susp = susp
        self.src = src
        self.susp_file = os.path.split(susp)[1]
        self.src_file = os.path.split(src)[1]
        self.susp_id = os.path.splitext(susp)[0]
        self.src_id = os.path.splitext(src)[0]
        self.src_voc={}
        self.susp_voc={}
        self.src_offsets=[]
        self.susp_offsets=[]
        self.src_sents=[]
        self.susp_sents=[]
        self.output = self.susp_id + '-' + self.src_id + '.xml'
        self.detections = None
        self.outdir=outdir

    def process(self):
        """ Process the plagiarism pipeline. """
        self.preprocess()
        self.detections = self.compare()
        self.postprocess()

    def preprocess(self):
        """ Preprocess the suspicious and source document. """
        src_fp = codecs.open(self.src, 'r', 'utf-8')
        self.src_text = src_fp.read()
        src_fp.close()

self.src_bow=tokenize(self.src_text,self.src_voc,self.src_offsets,self.src_se
nts,self.rem_sw)
        ss_treat(self.src_bow,self.src_offsets,self.min_sentlen,self.rssent)

        susp_fp = codecs.open(self.susp, 'r', 'utf-8')
        self.susp_text = susp_fp.read()
        susp_fp.close()

self.susp_bow=tokenize(self.susp_text,self.susp_voc,self.susp_offsets,self.su
sp_sents,self.rem_sw)

ss_treat(self.susp_bow,self.susp_offsets,self.min_sentlen,self.rssent)
```

```
        if self.tf_idf_p=='yes':
            tf_idf(self.src_bow,self.src_voc,self.susp_bow,self.susp_voc)

    def compare(self):
        """ Test a suspicious document for near-duplicate plagiarism with
regards to
        a source document and return a feature list.
        """
        ps=[]
        detections=[]
        for c in range(len(self.susp_bow)):
            for r in range(len(self.src_bow)):
                if cosine_measure(self.susp_bow[c],self.src_bow[r])>self.th1
and dice_coeff(self.susp_bow[c],self.src_bow[r])>self.th2:
                    ps.append((c,r))

plags=integrate_cases(ps,self.src_gap,self.susp_gap,self.src_size,self.susp_s
ize)
        i=0
        range_i=len(plags)
        while i<range_i:
            src_d={}
            for j in range(plags[i][0][0],plags[i][0][1]+1):
                src_d=sum_vect(src_d,self.src_bow[j])
            susp_d={}
            for j in range(plags[i][1][0],plags[i][1][1]+1):
                susp_d=sum_vect(susp_d,self.susp_bow[j])
            if dice_coeff(src_d,susp_d)<=self.th3:
                del plags[i]
                range_i-=1
            else:
                i+=1
        print 'After DICE plags'
        for i in plags:
            print i
        print '\n'
        plags=remove_overlap3(self.src_bow,self.susp_bow,plags)
        print 'After Remove'
        for i in plags:
            print i
        for plag in plags:

arg1=(self.src_offsets[plag[0][0]][0],self.src_offsets[plag[0][1]][0]+self.sr
c_offsets[plag[0][1]][1])

arg2=(self.susp_offsets[plag[1][0]][0],self.susp_offsets[plag[1][1]][0]+self.
susp_offsets[plag[1][1]][1])
            if arg1[1]-arg1[0]>=self.min_plaglen and arg2[1]-
arg2[0]>=self.min_plaglen:
                detections.append([arg1,arg2])
        print '\n'
        print 'Final results'
        for i in detections:
            print 'offset:',str(i[0][0]).rjust(6),'len:',str(i[0][1]-
i[0][0]).rjust(6),'\toffset:',str(i[1][0]).rjust(6),'len:',str(i[1][1]-
i[1][0]).rjust(6)
```

```
        return detections

    def postprocess(self):
        """ Postprocess the results. """
        serialize_features(self.susp_file, self.src_file, self.detections,
self.outdir)

# Main
# ====

if __name__ == "__main__":
    """ Process the commandline arguments. We expect three arguments: The
path
    pointing to the pairs file and the paths pointing to the directories
where
    the actual source and suspicious documents are located.
    """
    if len(sys.argv) == 5:
        t1=time.time()
        srcdir = sys.argv[2]
        suspdir = sys.argv[3]
        outdir = sys.argv[4]
        if outdir[-1] != "/":
            outdir+="/"
        lines = open(sys.argv[1], 'r').readlines()
        cont=0
        for line in lines:
            if cont in range(516):
                cont+=1
                continue
            print line
            susp, src = line.split()
            sgsplag_obj = SGSPLAG(os.path.join(suspdir, susp),
                                  os.path.join(srcdir, src), outdir)
            sgsplag_obj.process()
            break
        t2=time.time()
        print t2-t1
    else:
        print('\n'.join(["Unexpected number of commandline arguments.",
                         "Usage:  ./pan13-plagiarism-text-alignment-example.py
{pairs} {src-dir} {susp-dir} {out-dir}"]))
```