

Tesis de Licenciatura en Ciencias de la Computación

Estudio y desarrollo de nuevos algoritmos de detección de plagio

Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

Junio 2011

Tesista: Victoria Elizalde

LU 452/06

Director: Hugo D. Scolnik

Resumen

El plagio es presentar como propias las ideas o trabajos de otro. Es un tema de mucho interés actualmente ya que si bien siempre existió, hoy se ve potenciado por el uso de Internet, que habilita un acceso más fácil a la información. Es un problema que se presenta en los más variados ámbitos: educativos, científicos, profesionales (hasta en concursos literarios).

Los hashings perceptuales son funciones de hash pensadas para representar contenido multimedia (textos, imágenes, audio) por medio de características esenciales del documento y no a través de los bits que ocupa el archivo en el disco. Su objetivo no es el control de integridad como en los hashings tradicionales, sino la identificación de contenido.

En este trabajo investigamos a fondo los distintos tipos de detección de plagio, enfocándonos en la detección con referencia, en la cual se compara un texto sospechoso con una base de datos de documentos. En la segunda parte de la tesis aplicamos la idea de los hashings perceptuales a uno de estos métodos, obteniendo resultados iguales en términos de precisión y cobertura pero con un ahorro de memoria de hasta 30%.

Abstract

Plagiarism is to take someone else's idea or work and pose it as your own. This problem has always existed, but nowadays Internet makes it very easy to copy, making it more widespread than ever. Cases have appeared in many environments: educational, scientific, professional (even in literary contests).

Perceptual hashings are hash functions used to represent multimedia content such as texts, images or sounds. The aim is to capture the perceptual characteristics of this content - those perceived as essential by human beings - as opposed to the bits representation in the hard drive. These are not meant as integrity control like cryptographic hash functions, but as content identification.

In this work, we investigate the different types of plagiarism detection, particularly external detection, in which a suspicious text is compared against a document corpus. In the second part of this thesis we design new metrics based on perceptual hashings. We found a new n-gram technique, which has the same recall and precision than word n-grams (the standard metric in n-gram approaches) but saves up to 30% of disk space.

Índice general

Resumen	II
Abstract	III
Índice de cuadros	VII
Índice de Figuras	VIII
Agradecimientos	x
1. Introducción	1
1.1. Motivación	1
1.2. Objetivos de la tesis	3
1.3. Organización de la tesis	3
1.4. Hashings Perceptuales	4
1.4.1. Definición y aplicaciones	4
1.4.2. Propiedades deseables	6
1.4.3. Estructura	7
2. Estado del arte de la detección de plagio	8

2.1.	Problemas relacionados con la detección de plagio	9
2.1.1.	Detección de plagio en distintos escenarios	9
2.1.2.	Detección con o sin referencia	10
2.1.3.	Determinación del plagiador	11
2.1.4.	Detección en distintos idiomas	12
2.2.	Metodologías para detección con referencia	16
2.2.1.	Taxonomía	16
2.2.1.1.	Métodos sintácticos	17
2.2.1.2.	Métodos semánticos o lingüísticos	35
2.2.2.	Otros aspectos	40
2.2.2.1.	Unidad de comparación	40
2.2.2.2.	Funciones de distancia	41
2.2.3.	Más allá del corpus de referencia: búsqueda de documentos	52
2.3.	Corpus disponibles	54
2.3.1.	DejaVu	54
2.3.2.	METER	56
2.3.3.	PAN	59
2.3.4.	Clough09	61
2.4.	Medidas para comparar enfoques	63
3.	Implementación de software de detección de plagio	67
3.1.	Análisis preliminar	68
3.1.1.	N-gramas	69
3.1.2.	Funciones de distancia	72

3.2. Análisis detallado	76
3.3. El programa	77
3.3.1. Tecnologías utilizadas	78
4. Experimentos	80
4.1. Análisis preliminar	80
4.1.1. Experimento 1	80
4.1.2. Experimento 2	87
4.1.2.1. Funciones de distancia	89
4.1.2.2. Estrategias	90
4.2. Análisis detallado y postprocesamiento	93
5. Conclusiones y trabajo futuro	97
5.1. Detección de plagio	97
5.2. Detección mediante n-gramas	98
5.3. Perceptual Hashing	99
Bibliografía	100

Índice de cuadros

2.1. Relaciones semánticas en WordNet	38
2.2. Funciones de distancia sobre vectores	41
2.3. Funciones de distancia sobre conjuntos	44
2.4. Tabla comparativa de los distintos corpus disponibles.	58
4.1. Tabla con las combinaciones con f.measure de 0.8 o más	92
4.2. Resultados del análisis detallado del primer documento sospechoso	95
4.3. Resultados del análisis detallado del primer documento sospechoso con distintos parámetros	96

Índice de figuras

2.1. Los problemas asociados con la detección de plagio.	9
2.2. Distintos enfoques para la detección de plagio en distintos idiomas. . . .	15
2.3. Una molécula de ARN vista como una cadena de caracteres. Fuente: Wikipedia	26
2.4. Un ejemplo de Máxima subcadena común en dos frases muy similares. . .	27
2.5. La máxima subsecuencia común de las mismas frases.	28
2.6. Un ejemplo de que la máxima subsecuencia común es sensible al orden de las palabras. En realidad el sufijo “la” de la palabra “hacerla” no pertenece a la subsecuencia, pero decidimos no reflejarlo para no complicar el dibujo.	29
2.7. El ejemplo anterior analizado con GST. Los distintos colores representan los distintos tiles que retorna el algoritmo.	31
2.8. El mismo ejemplo de frases plagiadas alterando el orden, representado con un dotplot (se reemplazó las posiciones de las palabras por las mismas palabras).	34
2.9. La taxonomía de métodos de detección de plagio con referencia.	39
2.10. Representación de distintos coeficientes al aumentar la intersección de dos conjuntos.	46

2.11. Scatter plot de distintos coeficientes al aumentar la intersección de dos conjuntos.	47
2.12. Representación de distintos coeficientes al aumentar el tamaño de uno de los dos conjuntos.	48
2.13. Scatter plot de distintos coeficientes al aumentar el tamaño de uno de los dos conjuntos.	49
2.14. Representación de distintos coeficientes al aumentar el tamaño del conjunto más pequeño.	50
2.15. Scatter plot de distintos coeficientes al aumentar el tamaño del conjunto más pequeño.	51
2.16. Casos detectados y casos de plagio.	64
3.1. Los temas investigados, con las distintas propuestas.	68
3.2. El primer trigramo de la frase “Plagiar un documento es” con las distintas estrategias de selección.	72
3.3. El vector con las frecuencias de los n-gramas, transformado a rangos.	76
4.1. Dos ejemplos distintos de como varían precisión (en azul) y cobertura (en verde).	82
4.2. F-measure de los n-gramas de palabras con distintas longitudes y funciones de distancia.	84
4.3. F-measure de las estrategias originales con distintas longitudes y funciones de distancia.	85
4.4. Las mejores quince combinaciones con sus respectivos valores de f-measure, memoria y tiempo de ejecución.	88

Agradecimientos

A Hugo, por aceptar ser mi director de tesis, proponerme un tema interesante y guiarme en la investigación. Aprendí mucho, y lo disfruté más. Espero que sigamos trabajando juntos en otras cosas.

A Juan Cruz, que estuvo a mi lado todo el tiempo, ayudándome tanto técnica como afectivamente. Gracias por estar en todo.

A Maxi y al Abu, que lamentablemente no están para verme ahora. Mi abuelo fue el primer graduado universitario en la familia, nos dio esa posibilidad al resto, y nunca paró de enseñarnos lo importante que es estudiar para poder crecer. Maxi siempre se esforzó para transmitir su pasión por la Ciencia, y al menos conmigo, lo logró.

A Lau y a mi vieja, que pacientemente ayudaron a revisar este texto, pese a no entender páginas enteras, y se lo tomaron con mucho humor.

A Fede, los Gonzas, Santi, Herman, Matías, Rodrigo, Christian, Berna, Andrew, Sergio, Page, Emi, Metegol, Ayelén y Marta que me acompañaron en distintos momentos de la carrera.

A Mariana, que siempre encuentra una nueva forma de malcriarme y llama después de cada examen. A la Ama, que también se interesa en todo, y ante los nuevos pregunta “¿Y por qué no diez?”. A Mateo y a mi viejo.

A todos los increíbles docentes que encontré a lo largo de la carrera, son realmente muchos para nombrarlos a todos.

Como señalaron Fede y Gonza en su tesis: a todos los argentinos, que financian a la UBA con sus impuestos.

Capítulo 1

Introducción

1.1. Motivación

Detectar la similitud de textos tiene varias aplicaciones: desde encontrar réplicas cercanas de documentos en la Web o documentos repetidos en una PC, hasta detectar plagio de documentos en los más variados ambientes.

Shivakumar y Garcia-Molina [1] hablan de la importancia de encontrar réplicas (near-replicas) en la web. Las réplicas en este contexto son copias de un mismo texto, que por algún motivo se diferencian en algún aspecto no esencial: como estar en distinto formato o que la página contenga botones, links o publicidades. De tratarse de copias exactas, este problema sería trivial.

Encontrar estas réplicas es importante para los Web Crawlers y para mejorar los algoritmos de ranking en buscadores. Puede resultar útil en el caso de manuales, los que podrían ser reemplazados por otra copia en el caso de no encontrar la buscada.

El plagio es un problema mucho más complicado. Plagiar un documento es tomar

la obra de otro y hacerla pasar por propia. Abarca un espectro mucho más amplio de posibilidades: desde la copia burda y textual, hasta la reescritura de un texto manteniendo el contenido. Todo el documento puede ser producto de plagio o sólo una pequeña parte; un documento podría contener pasajes plagiados de más de una fuente.

El plagio puede ser intencional, en cuyo caso probablemente el plagiador trate de disimularlo tanto como le sea posible para evitar ser detectado. Puede ser también producto de la ignorancia: citar indebidamente las fuentes de un documento, o no usar comillas cuando se cita textualmente también constituye una forma de plagio.

Es un problema cada vez más recurrente en los ámbitos académicos, profesionales y políticos. En el sitio [plagiarism.org](http://www.plagiarism.org)¹ se citan varios estudios en los que un gran porcentaje de alumnos admite haber plagiado alguna vez o haber prestado su trabajo para que otro lo copie. En algunos el porcentaje llega al ochenta por ciento. Se menciona además que pocas universidades están dispuestas a sancionar a los plagiadores.

Los casos de plagio llegan a extremos tan increíbles, que incluso recientemente un Diputado de la Nación presentó un proyecto para penalizar el plagio [2] en el cual plagió el artículo sobre plagio de la Wikipedia².

En este contexto, ¿cómo puede ser de ayuda el software? Un programa de detección automática de plagio no reemplaza a una persona, sino que la asiste en la detección. La computadora ayuda en cuanto puede revisar miles de documentos, cosa imposible para un ser humano. El software sólo señala pasajes sospechosos pero es la persona quien toma la decisión final sobre si se trata de plagio o no: por ejemplo, varias frases copiadas textualmente entre comillas y con la debida cita no constituyen plagio, pero sólo una

¹http://www.plagiarism.org/plag_facts.html

²<http://es.wikipedia.org/wiki/Plagio>

persona puede apreciar esta diferencia.

Hay un límite en la calidad de plagio que una computadora puede detectar: cuanto mayor sea el nivel de reelaboración del que plagia, menores son las chances de que sea detectado. Por ejemplo, exponer una idea como propia (pero reescrita) es algo que no va a ser detectado automáticamente. De todas formas, la detección automática de plagio es un campo que seguramente será de ayuda para descubrir muchos más casos de plagio que los detectados manualmente.

1.2. Objetivos de la tesis

Los objetivos de la tesis son:

1. Revisión de la literatura existente y de los algoritmos correspondientes (métodos de espacios vectoriales, estadísticos, lingüísticos, de longitud de subsecuencias compartidas, Greedy String Tiling, etc)
2. Diseño de métricas eficientes, nuevos algoritmos, y testeo de los mismos
3. Implementación de software capaz de comparar textos escritos en Word, PDF, txt, etc.

1.3. Organización de la tesis

En el primer capítulo hablaremos de un tipo de hash conocido como perceptual hash. Explicaremos sus aplicaciones, su estructura y propiedades.

En el segundo capítulo haremos una revisión del estado del arte de la detección automática de plagio:

1. Describiremos los distintos problemas relacionados con la detección de plagio: con o sin colección de referencia, en distintos idiomas, dirección del plagio, etc.
2. Presentaremos una taxonomía de métodos de detección de plagio con referencia y su descripción en detalle.
3. Explicaremos porqué no es posible utilizar casos reales, y analizaremos los distintos corpus disponibles en la Web.
4. Expondremos las medidas de performance que permiten comparar los distintos métodos.

En el tercer capítulo describiremos los nuevos algoritmos que desarrollamos, producto de aplicar el concepto de perceptual hash al método de detección de plagio mediante n-gramas. Encontramos una métrica con cobertura y precisión equivalentes a los n-gramas de palabras, la estrategia de selección de n-gramas más utilizada en este tipo de sistemas.

1.4. Hashings Perceptuales

1.4.1. Definición y aplicaciones

Las funciones criptográficas de hash son funciones booleanas que dado un input variable producen un output de longitud fija. Su característica fundamental es que ante la mínima variación en el input el resultado del hash es completamente diferente. Esto se conoce como “efecto avalancha”. Se usan principalmente para el control de integridad y, junto a una clave para encriptación, para firma digital.

Los hashings perceptuales, en cambio, son funciones pensadas para archivos multimedia (textos, imágenes, audio, video), y como tales se basan en la percepción humana, no en los bits de un archivo. Se busca que dos archivos distintos que representan lo mismo sean vistos como iguales. Por ejemplo, dos copias de una misma imagen, una en formato vectorial y otra en formato bitmap. El contenido de los archivos va a ser totalmente distinto, sin embargo para el ojo humano la imagen es la misma. Lo mismo sucede con una canción guardada en dos archivos con distinta calidad. Este concepto también aplica a textos: un mismo libro procesado por distintos OCRs es el mismo libro, pero probablemente cada OCR se haya equivocado en algunos caracteres (distintos). Ésto se podría extender aún más si pensamos que reescribir frases de un texto cambiando algunas palabras por sinónimos o alterando el orden de las palabras no altera el contenido perceptual del archivo.

Es por eso que se los llama perceptuales: si se altera un archivo cambiando información no perceptual el hash será el mismo o muy parecido. Desaparece entonces el efecto avalancha. Esta propiedad se conoce como robustez, por lo que los hashings perceptuales son también denominados en la literatura hashings robustos.

A diferencia de las funciones de hash tradicionales, estos hashings perceptuales son usados para identificación y autenticación de contenido digital. Por ejemplo, se pueden usar para identificar canciones que suenan en la radio desde un teléfono celular (estableciendo comparaciones contra una base de datos con hashings perceptuales de distintas canciones) o videos subidos a páginas de Internet [3] (y así detectar violaciones de copyright). En el caso de textos, los hashings perceptuales pueden ser utilizados para detectar plagio de documentos.

1.4.2. Propiedades deseables

Estas aplicaciones nos llevan a otras propiedades deseables además de la robustez [4, 6]. Por un lado, el hash debe ser único: inputs perceptualmente distintos tienen hashings diferentes. Por otro lado, deben ser seguros (impredecibles): debería ser difícil encontrar un par de archivos con distinto contenido perceptual y el mismo hash, para evitar un ataque al esquema. Estas tres propiedades compiten entre sí: para ser impredecible y evitar colisiones, el output debería ser lo mas aleatorio posible, lo cual va en contra de la robustez.

Además de estas propiedades, hay otras comunes a los sistemas de encriptación, propuestas por Shannon [7] : confusión y difusión. Coskun y Memon [5] proponen una extensión a estos conceptos para hashings perceptuales. Confusión es la complejidad de la relación entre la clave y el valor del hash. Dicho de otra manera, un hash tiene buena confusión si dados el input X y las claves secretas K y K' , $H = h(K, X)$ y $H' = h(K', X)$ es computacionalmente difícil encontrar la relación entre los hashings H y H' con K y K' difiriendo en por lo menos un bit. Esta definición es la que se usa para los hashings criptográficos tradicionales, pero es también válida para los hashings perceptuales.

Difusión es la complejidad de la relación entre la información entre el texto plano y el texto cifrado, en el caso de un hash entre el input y el valor del hash. En otras palabras, dados los inputs X y X' , y la clave K , $H = h(X, K)$ y $H' = h(X', K)$ es computacionalmente difícil encontrar la relación entre H y H' , donde X y X' difieren en al menos un bit. Buena difusión en los hashings tradicionales, implica el efecto avalancha anteriormente mencionado. Como ya dijimos, esta definición no se ajusta a los hashings perceptuales. La definición de difusión que dan Coskun y Meemon en [5] es “la irrelevancia

o relación compleja entre la información perceptual del input y el valor del hash”. La propiedad de seguridad/impredecibilidad es una reformulación del concepto de difusión.

1.4.3. Estructura

En general, un hashing perceptual está compuesto de, al menos, dos partes fundamentales: la parte de extracción de características (feature extraction) y la de clustering o compresión. En la primera etapa se toman las características esenciales del archivo de input, produciendo un hash intermedio. Es en esta etapa donde se debe lograr la robustez y la unicidad del hash.

En la segunda etapa se debe lograr difusión: el hash intermedio suele tener mucha relación entre el input y el hash, y hay que evitar que esto suceda.

Finalmente, el hash podría tener o no una clave secreta. La clave secreta se usa para encriptar el hash, proveyendo así confusión; de esta forma se evitan ataques del tipo texto plano elegido en el esquema. Si no hubiera clave un atacante podría jugar con el input, variándolo levemente hasta encontrar un hash igual para un input perceptualmente distinto. Podría también alterar el input sin alterar el hash, encontrando así una colisión.

Capítulo 2

Estado del arte de la detección de plagio

La detección automática de plagio es un área de investigación que abarca distintos problemas:

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

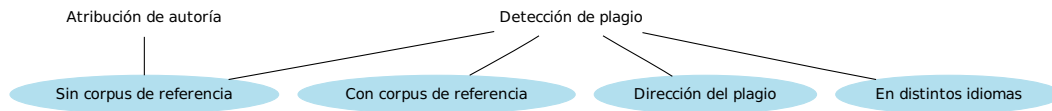


Figura 2.1: Los problemas asociados con la detección de plagio.

Un sistema de detección de plagio debe tener ciertas características, independientemente del tipo de plagio que detecte o la metodología que utilice. Schleimer, Wilkerson y Aiken [28] proponen las siguientes propiedades:

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

2.1. Problemas relacionados con la detección de plagio

2.1.1. Detección de plagio en distintos escenarios

El plagio puede ocurrir en distintos escenarios: ámbitos educativos, académicos o profesionales. Los documentos plagiados pueden ser informes, tareas de la escuela o la

universidad, artículos periodísticos o códigos fuentes.

En principio, los métodos para detectar plagio sólo se diferencian entre aquellos que analizan texto y aquellos que analizan código. Mientras los métodos para detectar plagio en texto analizan caracteres, frases, estilo del texto y distintas estructuras del lenguaje, los que detectan plagio en código revisan la estructura del mismo. Las métricas de estos sistemas [8] son específicas: cantidad de loops, cantidad de ramas de decisión (sentencias if), cantidad de variables (usadas o declaradas), etc.

Estos sistemas [9] ignoran los nombres de variables, constantes y nombres de parámetros. Omiten también comentarios y espacios en blanco. Otra utilidad de este tipo de software es la de encontrar código duplicado en grandes sistemas.

2.1.2. Detección con o sin referencia

En general cuando uno piensa en detección de plagio, lo más intuitivo es pensar que se compara un documento con un conjunto de otros documentos, y se buscan coincidencias. Esto es lo que se conoce como detección con referencia, porque se utiliza un corpus de documentos como base para comparar contra el texto sospechado de plagio. Esta es una técnica sobre la cual se ha escrito mucho, y la mayoría de los sistemas de detección de plagio se han desarrollado siguiendo este esquema. Sin embargo, posee una desventaja: si la fuente de la cual se plagió no está en la base de datos de documentos de referencia, el plagio no será detectado. La fortaleza de este esquema reside entonces no solamente en el algoritmo usado para comparar documentos, sino también en tener una base de datos de referencia muy grande. Este problema es el más estudiado en el campo y será analizado en detalle en la sección 2.2.

Existe otra técnica llamada detección intrínseca de plagio (sin corpus de referencia) que se basa en detectar el plagio mirando sólo el documento sospechado. La idea de esta técnica es detectar cambios drásticos en el estilo o la estructura de un documento. Por ejemplo, si un alumno secundario copia fragmentos de un texto, seguramente la redacción de las partes copiadas será más sofisticada (frases más complejas, sin errores de ortografía, etc.) que la del resto del texto. Un docente entrenado puede notar esto fácilmente.

Para detectar plagio de esta forma se procesa el texto analizando la sintaxis, la estructura de las distintas frases y párrafos: longitud de frases y de párrafos, cantidad de signos de pregunta y de comas, secuencias de palabras [12], etc. Otro método es usar el Averaged Word Frequency Class [10]: se dividen las palabras en clases según su frecuencia de aparición en textos, estimada con la ayuda de un corpus de documentos del idioma a analizar. Luego se calcula el promedio de todas las clases de palabras que aparecen en el texto, pudiendo así determinar la riqueza del vocabulario del escritor. Así, si un párrafo tiene palabras mucho menos frecuentes que las del resto del documento se sospecha que hubo plagio. Según Rosso et al. [11], la ventaja de esta medida con respecto a otras es que es estable en documentos cortos, mientras que en general las técnicas utilizadas para medir la riqueza de vocabulario no reportan resultados confiables en esas condiciones, algo fundamental en detección de plagio.

2.1.3. Determinación del plagiador

Una vez que se encontró un plagio entre dos documentos, aparece otro problema muy interesante: determinar quién plagió a quién.

Según [13], este problema está relacionado con el de detección de plagio sin referencia:

en este caso también es útil analizar el estilo de los textos. Al encontrar un fragmento repetido en ambos textos, se analizan éstos para descubrir cual tiene un estilo similar al del fragmento repetido, y así saber quien es el autor original (siempre y cuando no se trate de dos textos que plagiaron el mismo pasaje de otra fuente).

En Grozea y Popescu [13] se presenta una forma de determinar la dirección del plagio entre dos documentos. Se parte de la base de que los fragmentos plagiados ya se conocen y se analizan los n-gramas de tamaño $n = 8$ (un tamaño intermedio) en común. Se grafican estos n-gramas en común con respecto a la posición relativa en cada uno de los textos y se buscan asimetrías. Cuando el documento original se encuentra en el eje x, aparecen en los gráficos unos grupos de puntos en sentido horizontal. Los autores interpretan que esto es porque los pasajes en común son más parecidos a los del texto en el eje x y por eso esos n-gramas aparecen más veces en ese texto. En el caso de encontrarse el texto original en el eje y, el sentido de esos grupos de puntos va a ser vertical.

Para que el método de detección sea automático, resuelven el problema de ver en que sentido están los grupos de puntos como un problema de Visión en Computación. Obtienen buenos resultados: 75.41 de precisión global, probado con el corpus del PAN (del cual se hará mención más adelante en la sección 2.3.3).

2.1.4. Detección en distintos idiomas

Hasta ahora en la discusión sobre los distintos problemas en el campo de la detección de plagio no se hizo mención de un tema muy importante: el idioma. Hoy en día es usual encontrar autores que traducen una publicación de un idioma (generalmente del inglés) a su idioma original, a veces inclusive utilizando traductores automáticos. Este es un

problema que ha sido estudiado en menor medida en la literatura.

Fragmentos plagiados con traducción automática los hace excelentes candidatos para las técnicas de detección sin referencia previamente mencionadas como la detección de cambios en el estilo, ya que la redacción de las frases plagiadas seguramente será inusual. Aún cuando el plagiador traduzca a mano ciertos pasajes, es probable que la estructura de las frases sea atípica. Sin embargo, todo el texto podría ser traducido, con lo cual no habría cambios en el estilo.

Como es de esperar, existen varios sistemas de detección de plagio en distintos idiomas que incorporan una fase de traducción y luego utilizan métodos de detección de plagio con referencia. Estos métodos no son utilizados directamente, si no que se deben generalizar antes. ¿Porqué? Cada palabra puede ser traducida de varias maneras, con lo cual tenemos un conjunto de términos por cada término del documento original. Por ejemplo, en el caso de usarse un modelo que representa documentos como vectores de términos, tendríamos que comparar un vector (el documento que no fue traducido) contra una matriz (el documento traducido) [14]. Estos métodos son llamados enfoques basados en diccionarios (dictionary-based approaches), ya que necesitan un diccionario bilingüe para hacer las traducciones.

Hay otros sistemas que en vez de traducir el texto y transformarlo en un problema de detección de plagio monolingüístico usan directamente modelos que contemplan que los documentos están en distintos idiomas. En inglés estos son conocidos como Cross-language retrieval models. A continuación, hablaremos de algunos de estos modelos.

CL-ASA A diferencia del método anterior, éste no hace traducciones. Usa un diccionario estadístico, utilizado para establecer posibles traducciones: con él se calcula

la probabilidad de que un cierto término sea la traducción de otro. Una vez que se tienen las traducciones más probables, se calcula la similitud entre los textos. El diccionario bilingüe estadístico se calcula en base a un corpus de documentos en los dos idiomas considerados.

Cross Language Explicit Semantic Analysis (CL-ESA) Está basado en ESA [15](un método de information retrieval monolingüístico, generalización del Vector Space Model) en el que se representa un documento en relación a otros documentos. Se lo representa con un vector v , donde v es el vector que representa el documento d , v_i^* es el vector que representa el documento i -ésimo en el corpus de documentos de referencia y φ es una medida de distancia, por ejemplo el coseno del ángulo entre los dos vectores. Luego, la distancia entre dos documentos, d y d' se calcula como $\varphi(d, d')$.

$$d = (\varphi(v, v_1^*), \dots, \varphi(v, v_n^*))^T \quad (2.1)$$

En CL-ESA se compararan dos documentos en distintos idiomas, con lo cual los documentos de referencia no son exactamente los mismos; los autores decidieron usar un grupo de los artículos de Wikipedia que estuviera disponible en ambos idiomas (se supone que el corpus no debe ser específico a un dominio, debe ser amplio). Cada documento se compara con los documentos en su idioma, y luego se calcula la distancia nuevamente como $\varphi(d, d')$. La idea detrás de esto es que se compara el documento contra distintos conceptos y se obtiene una relación que es independiente del idioma. Así, no es necesario traducir el documento para compararlo contra otro en un idioma distinto.

Un modelo similar a este es el propuesto en Poliquien, Steinberger e Ignat[17]. En

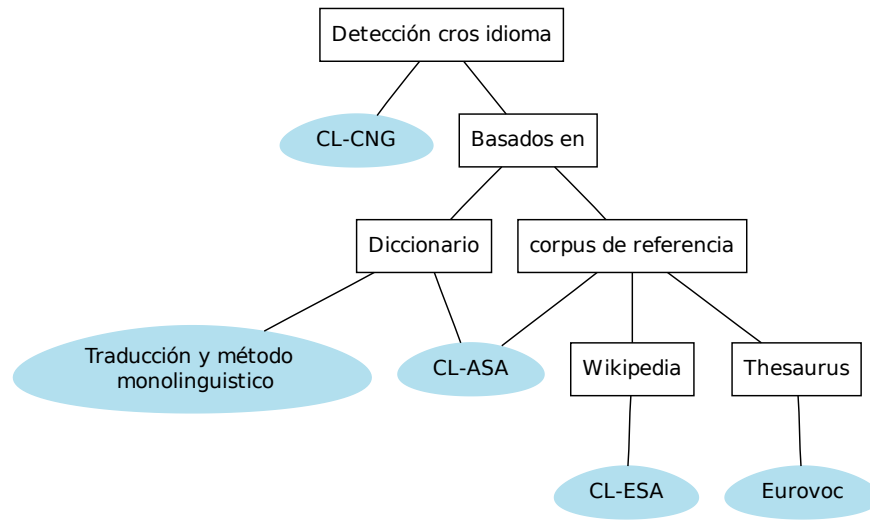


Figura 2.2: Distintos enfoques para la detección de plagio en distintos idiomas.

este modelo tampoco se traduce el texto, se usa el thesaurus multilingüe Eurovoc ¹. Los documentos son vinculados a descriptors y representados mediante un vector. Luego se calcula la similitud entre ambos vectores para determinar la similitud de los documentos.

Cross Language Character n-Gram Analysis (CL-CNG) Este es el más simple de los modelos. Simplemente consiste en dividir ambos textos en n-gramas, descartando todos los caracteres no alfanuméricos (espacios incluidos). Luego los n-gramas son pesados con tf-idf (term frequency–inverse document frequency). Y se calcula la distancia usando el coseno para cada representación. No se hace traducción de ningún tipo, por lo que este enfoque funciona cuando los idiomas tienen raíces en común (porque las palabras tienden a parecerse) [16].

¹<http://eurovoc.europa.eu/>

2.2. Metodologías para detección con referencia

Estructura general Los sistemas de plagio con referencia tienen en general la misma estructura [18][20]:

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

2.2.1. Taxonomía

A continuación se propone una taxonomía de los distintos métodos de detección de plagio extrínseca. Se podrían usar distintos criterios para armar esta clasificación:

- La unidad básica de comparación entre textos como párrafos, frases, palabras o todo el documento.
- Modelo de representación de documentos: métodos de espacio vectoriales, grafos, árboles de sufijos, etc.
- Análisis de similitud local y global [21]: la similitud global se refiere a características globales del documento (por ejemplo: frecuencias de términos del documento), mientras que la similitud local se refiere a características específicas de ciertas partes (por ejemplo: dos sub strings iguales constituyen una medida local).
- Distintos tipos de métodos: lingüísticos, estadísticos, etc.

La taxonomía será presentada discriminando los distintos tipos de métodos, sin dejar de mencionar los otros aspectos.

2.2.1.1. Métodos sintácticos

En esta sección hablaremos de métodos que hacen un análisis de los textos solo vistos como símbolos. No se les da ningún significado a estos símbolos: solo se analiza la frecuencia o presencia de ciertos símbolos (grupos de caracteres, palabras, etc).

Los métodos de pre procesamiento más habituales son: convertir todo a mayúsculas, eliminar acentos o signos diacríticos, eliminar signos de puntuación y todo símbolo no alfanumérico, reemplazo de números por un símbolo, etc.

Con estos métodos pueden convivir ciertas heurísticas lingüísticas como la eliminación de palabras comunes, lematización o uso de un tesoro² para generalizar palabras [22]. Se trata de heurísticas para mejorar el método, pero no son la esencia del método mismo.

²Diccionario de sinónimos y antónimos

A continuación describiremos distintos métodos sintácticos usados en el análisis preliminar que mencionamos antes.

Vector Space Model El modelo de espacio vectorial es una forma de representar textos (y objetos en general) que surge de la disciplina de recuperación de la información (Information Retrieval, en inglés). En ella, los documentos se representan a través de características esenciales que son cuantificadas (se les da un cierto peso) y almacenadas en un vector. En general se suele elegir como dimensiones términos del documento y sus frecuencias. Como peso se suele usar tradicionalmente lo que se conoce como tf-idf, esto es term frequency-inverse document frequency. La idea detrás de esto es que las palabras menos frecuentes en un texto (o un corpus de documentos) son las más relevantes del texto: por ejemplo la palabra “la” en castellano es muy frecuente y su presencia no es significativa; la palabra “vectorial” en cambio es más inusual y su presencia es mucho más significativa.

Luego, se usa una función de distancia entre dos vectores para medir la similitud entre los documentos. Esta medida suele ser el coseno del ángulo entre los vectores. Este modelo conlleva tres elecciones distintas: que característica utilizar para representar el texto, los pesos que se le asigna a cada dimensión y la medida de distancia entre vectores [23].

El hecho de representar de esta forma un documento hace que los términos sean vistos como independientes entre sí (lo cual no es necesariamente cierto, ya que las palabras tienen distintos significados según el contexto). Por este motivo decimos que es un método sintáctico: la representación es un vector con términos, pero éstos son vistos simplemente como cadenas de caracteres. Además, no se tiene en cuenta el orden de las palabras: esto

tiene un lado positivo y otro negativo. El hecho de usar un método que no es sensible al orden, nos habilita a encontrar plagios más complejos, en los que hay transposiciones de palabras (lo cual lo diferencia de varios métodos, como veremos más adelante). Por otro lado, el hecho de representar el documento así implica una pérdida de información: no podemos usar el orden de las palabras para detectar plagio.

A medida que los documentos crecen en longitud, tienen más términos, con lo cual sus respectivos vectores también crecen en longitud. Esto aumenta la complejidad de los cálculos (hay que tener en cuenta que son muchos los documentos a comparar), por lo que se suelen utilizar métodos para reducir las dimensiones y considerar las más importantes [23][24].

Como ya habíamos mencionado antes, una ventaja de este modelo es la flexibilidad. Podemos tener coincidencias parciales, pudiendo así detectar plagios más elaborados que un simple copy-paste. En recuperación de la información el valor de la distancia da un ranking entre los documentos: cuanto más cercano a uno sea el valor mas se parecen los documentos. En detección de plagio no se usa de esta manera: simplemente se hace una primera selección de los documentos que superen un cierto umbral; dos documentos se parecen o no se parecen, no es tan fuerte la idea de ranking, ya que luego se analizará en forma más detallada esos documentos. Es deseable que haya una gran separación entre los valores obtenidos por documentos plagiados y por los no plagiados. En otras palabras, que un documento tenga un valor de similitud de 0.9, no necesariamente implica que tenga más texto plagiado que otro con similitud 0.8, solo implica que ambos contienen plagio.

Shivakumar y García-Molina [25] critican este modelo. Sin embargo, sus objeciones no apuntan tanto al enfoque en sí mismo como a malas elecciones de funciones de peso

(como usar la frecuencia de palabras, sin tener en cuenta la frecuencias de las palabras en el documento/corpus, lo cual da más importancia a las palabras comunes) y malas elecciones de funciones de distancia (según ellos el coseno, del cual hablaremos más adelante, no es una buena elección para detectar plagio).

El modelo de espacio vectorial asume que los plagios ocurren en documentos de un mismo tópico [21]. Esto dejaría sin detectar algunos casos en los que tal vez un autor menciona superficialmente otra disciplina, copiando (algunas de) esas menciones.

¿Cómo se usa este modelo en detección de plagio? Así como lo describimos, sirve para calcular la similitud entre textos, y posibles plagios, pero no señala los pasajes sospechosos. Incluso, un par de frases copiadas textualmente (dentro de un documento largo) podrían no ser detectadas. Hay algunas formas de aplicar esto para obtener pasajes sospechosos. Zechner, Muhr, Kern y Granitzer[24] usan un vector por frase: las que superan un cierto umbral de similitud son consideradas plagio. Este método sufre los problemas habituales de analizar plagio por frases: al plagiar, se pueden reorganizar las oraciones de forma distinta, en modo que la parte plagiada quede separada. Además dividir un texto en frases suele tener problemas, ya que usar los puntos como separación no alcanza: abreviaciones, índices y ecuaciones suelen confundir a las herramientas [25].

Devi, Rao, Ram y Akilandeswari [26] utilizan el VSM para hacer una primera selección de documentos. Luego utilizan la información obtenida en el primer paso para marcar ciertas líneas como sospechosas. Calculan la probabilidad de que ese pasaje sea un plagio con otro algoritmo, función del coseno de esa parte (utilizan coseno como medida de distancia).

Otra alternativa es mencionada en Stein y Meyer zu Eissen [21]: si el valor de similitud entre dos textos supera un cierto umbral, se dividen ambos documentos en partes más

cortas y se vuelve a calcular la similitud, recursivamente.

Hashing Una técnica muy utilizada es la de hashing o fingerprinting. Consiste en seleccionar partes de un texto y calcular el hash de esa porción, utilizando una función de hash criptográfica (MD5 generalmente). El hash se almacena en un índice invertido. Se procesa el documento sospechoso de la misma manera que los de referencia y de haber varias coincidencias se considera que puede haber plagio.

Como se dijo en el primer capítulo, las funciones de hash criptográficas poseen el denominado efecto avalancha: si cambia un bit del input, cambia drásticamente el output. Entonces, si una de las partes que seleccionamos difiere muy poco, el hash de esa parte va a diferir completamente, evitando que esa similitud sea detectada. Esto hace que la longitud de las partes o chunks a hashear no pueda ser muy grande: si se tomaran frases por ejemplo, se podría eludir la detección modificando sólo una palabra. Al mismo tiempo, cuanto menor sea la longitud de los chunks, más costoso será el análisis (mayor tiempo de procesamiento, mayor espacio necesario para almacenar el índice).

Como chunks se suelen usar n-gramas de palabras o caracteres [19][28]: esto es n palabras (o caracteres) consecutivas. Se seleccionan algunos de estos chunks y se guarda su respectivo hash en el índice, ya que guardar los hashes de todos los chunks suele ser prohibitivo (especialmente si consideramos chunks con superposición). Para eso se utilizan distintas estrategias [28]: seleccionar cada i-ésimo hash, lo cual dificulta la detección de coincidencias en el caso de reordenamientos, agregados o borrados de frases o palabras; en otros casos se toman los hash que cumplen ser $0 \bmod p$ para algún p . El problema de esto último es que las coincidencias son detectadas solamente si los hash de los pasajes que coinciden son $0 \bmod p$. Además, no da ninguna garantía del espacio que puede haber

entre dos hashings seleccionados.

Para solucionar esto, Schleimer, Wilkerson y Aiken[28] proponen el algoritmo Winnowing para hashings de k-gramas. Este algoritmo ve el documento como una serie de ventanas superpuestas. Selecciona un hash de cada una de esas ventanas (el mínimo, y si hay más de uno el de más a la derecha). Tiene dos propiedades muy interesantes:

- Si hay un substring repetido de al menos t caracteres, el algoritmo garantiza que será detectado.
- No se detectan coincidencias menores a k (umbral de ruido).

El usuario define los valores de k y t , y sobre esa base queda definida la longitud de las ventanas. Se elige el mínimo porque es probable que el mínimo de varias ventanas superpuestas coincida, con lo cual disminuye la cantidad de hashings a almacenar.

Stein y Meyer zu Eissen [21] critican a las fingerprints por ser computacionalmente caras y señalan que la longitud de los chunks tiene que ser pequeña (ya que como se mencionó antes si cambia un sólo caracter del chunk, su hash va a ser completamente distinto). Al aumentar la cantidad de hashings necesarios esto impacta negativamente en el esfuerzo de cálculo, almacenamiento y comparación. Ellos proponen como solución usar fuzzy fingerprints (ver la correspondiente sección).

En ese sentido, el algoritmo Winnowing mejora el tema del almacenamiento y la comparación (usando el mínimo reduce la cantidad de hashings a almacenar), sin embargo no hace nada para mejorar el esfuerzo de cómputo: si bien selecciona solo algunos hashings para almacenar, primero se deben calcular todos porque es la única forma de saber cual es el mínimo para una determinada ventana. El costo de cálculo es elevado ya que los chunks son tomados en forma superpuesta.

Los autores usan un tipo de hash pensado para aliviar el cómputo: proponen usar un rolling hash, es decir, un hash que permite reutilizar el cálculo del hash anterior para calcular el siguiente (recordemos que se calculan en forma superpuesta con lo cual si el hash es de k caracteres, un hash y el siguiente tendrán $k - 1$ caracteres en común). El algoritmo Karp-Rabin es un ejemplo de rolling hash, pero tiene un defecto: el último carácter sólo afecta algunos bits del hash (se pierde el efecto avalancha); se propone una modificación que hace que el último carácter pueda potencialmente afectar todos los bits.

Este método se usa al igual que VSM para hacer una selección previa de documentos, pero aporta más información a la hora de buscar los pasajes específicos. Junto con cada hash se guarda información sobre la posición del chunk y el documento del cual proviene. Una forma de obtener los pasajes sería juntar esta información con algún algoritmo ad-hoc. Por ejemplo, Kasprzak, Brandejs y Kripac[19] usan un algoritmo para definir intervalos plagiados. Siguen reglas en las que por ejemplo: un intervalo válido tiene por menos veinte chunks en común; o que entre dos chunks plagiados en un intervalo haya como máximo cuarenta y nueve chunks no plagiados.

Una observación importante: en algún momento del análisis se debe controlar que si dos hashings coincidieron fue porque realmente las cadenas son iguales y no porque se trate de una colisión en la función de hash.

N-gramas En los últimos años, varias publicaciones se refirieron a sus métodos de detectar plagio como “basados en n-gramas” [27, 29, 30, 31, 32]. Los métodos tienen la misma estructura: se toman n-gramas del documento (n caracteres o palabras consecutivos), se calcula en algunos casos la frecuencia [32], en otros sólo el conjunto de n-gramas [29], y se computa la distancia con alguna función.

Este enfoque tiene similitudes con los dos discutidos anteriormente. Por un lado, se podría ver el conjunto de n-gramas de un texto como un hash, aunque no tenga una longitud fija (aunque esa podría considerarse una interpretación forzada). Por otro lado, puede verse como un VSM: el documento es representado como un vector, con los n-gramas como dimensiones, la frecuencias como peso y luego la función de distancia.

Cuando nos referimos a n-gramas podemos estar hablando de distintas cosas: n caracteres consecutivos, n palabras. Una variación de estos últimos es utilizar las longitudes en vez de las palabras [33]: por ejemplo, dado el bigrama “El gato”, la representación sería “2 4”.

¿Qué valores de n se suele usar? En general cuando se trata de n-gramas de palabras, el n suele estar entre dos y cinco: Barrón-Cedeño y Rosso[27] usan bigramas y trigramas, en Ferret[31] se usan trigramas, mientras que Clough[29] y Kasprzak[19] usan cinco-gramas. Grozea, Gehl y Popescu[30] usan dieciséis-gramas, porque usan n-gramas de caracteres (y un dieciséis-grama es equivale a dos o tres palabras). Mas allá de los números en sí mismos, lo que sucede es que al aumentar el n disminuye la cobertura (menos casos son detectados, hay más falsos negativos) pero aumenta la precisión (disminuyen los falsos positivos).

En general los n-gramas se toman en forma superpuesta, lo cual hace que la cantidad de n-gramas de un texto sea mucho mayor, pero que el método tenga mejor cobertura.

Éstos métodos no aportan tanta información para detectar pasajes copiados como los fingerprints. Es por eso que suelen ser combinados con algún método de análisis detallado.

Fuzzy fingerprints B. Stein y S. M zu Eissen proponen un método que combina fingerprints y vector space model[21]. Según ellos los enfoques que usan fingerprints tienen

como desventaja su alto costo y que la longitud de los chunks debe ser pequeña. Su método es más flexible, permitiendo la elección de chunks mucho más largos. Así cada documento será representado por menos hashings, disminuyendo el costo de cálculo, almacenamiento y comparación. En lugar de tomar los pasajes directamente desde el documento, usan una representación en forma de vector de los mismos.

Dividen los términos que aparecen en un documento en clases de equivalencia de forma tal que un término corresponde a una clase si comienza con cierto grupo de prefijos [35]. Por ejemplo: una clase de equivalencia formada por los términos que comienzan con X, Y o Z. La frecuencia de aparición de estas clases puede ser estimada usando algún corpus de documentos (en “Near Similarity Search and Plagiarism Analysis” [21] se usa el British National Corpus ³).

Para formar el vector, se calcula la frecuencia de cada clase de equivalencia para cada chunk, y se almacena la desviación de éstas con respecto a los valores calculados en base al corpus. Finalmente se aplica un método de dispersión al vector y se obtiene un hash de cada chunk.

Una vez construido el hash del documento, la comparación se hace de igual manera que en el caso de las fingerprints. Una colisión en el hash indica que el chunk correspondiente ha sido plagiado.

Este método posee varias ventajas respecto a fingerprinting y el modelo de espacio vectorial:

- Los chunks seleccionados son del orden de las cien palabras, contra los tres - diez utilizados en fingerprinting, resultando en un número mucho menor de hashings por documento.

³<http://www.natcorp.ox.ac.uk/>

- Posee una cobertura muy similar a la de los modelos de espacio vectorial (recordemos que detecta plagios más complejos, con un cierto nivel de reescritura).

Su desventaja es que se podrían producir colisiones pese a que los pasajes analizados no se parezcan. Otro inconveniente es que está ligado al idioma y a la existencia de un corpus grande en ese idioma para calcular las frecuencias de las clases de equivalencia.

Análisis detallado Los métodos vistos hasta ahora sirven para hacer un análisis preliminar de la similitud entre dos documentos. Son usados junto con algún método más detallado que permita encontrar los pasajes precisos en los que ocurrió el plagio. Algunos de los enfoques preliminares proveen más información que otros, lo cual podría ser eventualmente aprovechado en el análisis detallado.

Los métodos presentados en esta parte no lidian con grandes cantidades de documentos, sino que los comparan de a pares. Los textos a revisar son aquellos obtenidos en el análisis inicial. Esto cambia completamente los requerimientos sobre los algoritmos: deben ser muy precisos, y el precio a pagar es un aumento en el tiempo de cómputo.

Los enfoques de esta sección son también utilizados en bioinformática. En biología es habitual encontrarse con la necesidad de comparar distintas proteínas, secuencias de ADN, etc. Esto implica manejar grandes volúmenes de datos por lo que en la actualidad se hace por computadora. Los datos son representados con letras del alfabeto. Por ejemplo, el ADN se representa como cadenas de los caracteres A, C, G y



Figura 2.3: Una molécula de ARN vista como una cadena de caracteres.
Fuente: Wikipedia

T. Entonces, este tipo de problema biológico puede ser resuelto con la ayuda de distintos algoritmos de búsqueda de cadenas (string matching o string searching) que veremos a continuación.

Un matching es una relación entre dos subcadenas comunes a dos textos. Si ambas subcadenas son exactamente iguales lo llamaremos matching exacto; de lo contrario si las dos subcadenas son similares (según algún criterio) pero no idénticas se trata de un matching aproximado. Los matchings del segundo tipo tienen mayor cobertura⁴ porque detectan casos más complejos, ya que toleran reescritura.

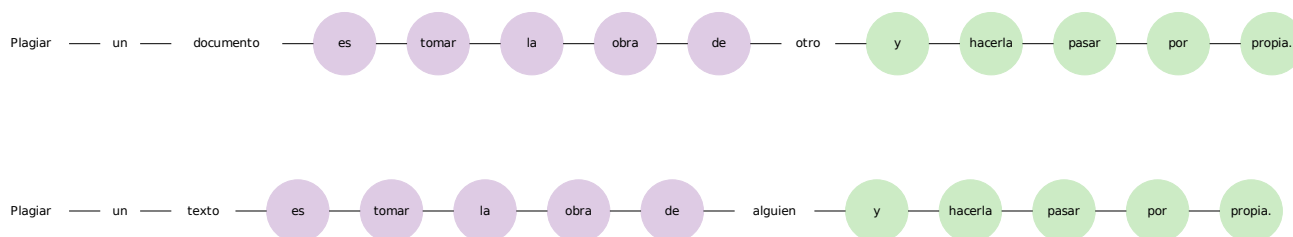


Figura 2.4: Un ejemplo de Máxima subcadena común en dos frases muy similares.

Máxima subcadena común Intuitivamente, la forma más simple de encontrar una copia, sería buscar una cadena muy larga de caracteres presente en ambos textos. Un problema relacionado es el de encontrar la máxima subcadena común (Longest Common Substring) entre dos textos. Zaslavsky et. al. [36] analizan su posible uso en la detección de plagio. Evalúan la construcción un árbol de sufijos, una estructura que contiene todos los sufijos de una determinada cadena, orientada a resolver operaciones con cadenas en

⁴Esta medida será discutida en detalle en la sección 2.4.

forma eficiente. Teóricamente se podría usar este árbol como índice y almacenar todos los documentos de una colección. La similitud se calcularía tomando la relación entre la LCS y la longitud del texto.

Construir uno de estos árboles es muy costoso en términos de espacio lo que lo hace impracticable para grandes corpus de documentos. Por eso es posible utilizarlo sólo como un análisis detallado. De todas formas se trata de un matching exacto, con lo cual solamente detecta casos de plagio textual.

Máxima subsecuencia común Otro problema similar es el de la máxima subsecuencia común. La diferencia es que una subcadena es un conjunto de caracteres consecutivos de una cadena, mientras que en una subsecuencia no necesariamente lo sean.

A diferencia del anterior, este enfoque es tolerante a cierto tipo de reescritura. En las figuras 2.4 y 2.5 se puede apreciar la diferencia entre ambos enfoques: mientras las subcadenas comunes máximas cubren cinco palabras, la máxima subsecuencia común cubre doce. La máxima subsecuencia común es tolerante a reescrituras tales como cambiar, agregar o borrar palabras.

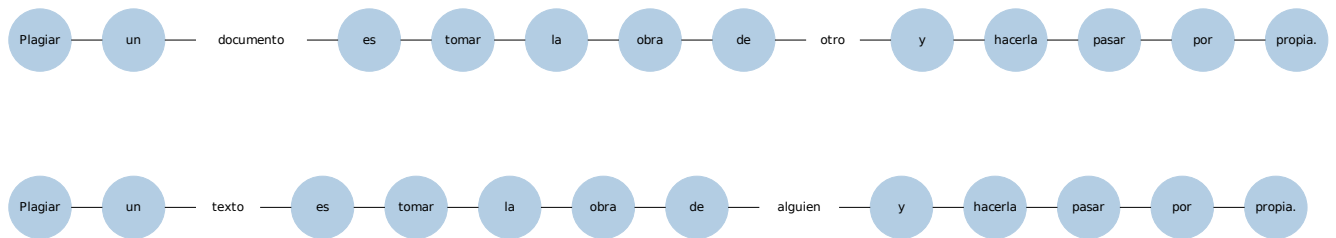


Figura 2.5: La máxima subsecuencia común de las mismas frases.

Tiene la gran desventaja de no ser tolerante a re ordenamiento de palabras (order preserving). En la figura 2.6 se puede ver como el fragmento “la obra de otro” aparece en ambas cadenas pero no en la subcadena común. Al buscar plagio, es importante detectar que ese fragmento también fue copiado. De todas formas se trata de un matching aproximado, ya que tolera algunos tipos de reescritura.

En el caso de la subcadena, puede ser simplemente mostrada al usuario si supera una cierta cantidad de caracteres, ya que fijando un valor no muy bajo es muy probable de que se trate de plagio. En cambio, que la longitud de la máxima secuencia común sea mayor a un cierto número no garantiza que se trate de plagio: puede tratarse de una gran cantidad de caracteres pero muy separados entre sí. Se deben analizar las subcadenas comunes dentro de la secuencia: el largo de cada una de ellas, la separación entre sí y la proporción de palabras (o caracteres) idénticos respecto al total de la secuencia. Este problema es común a los algoritmos de matching aproximado en general.

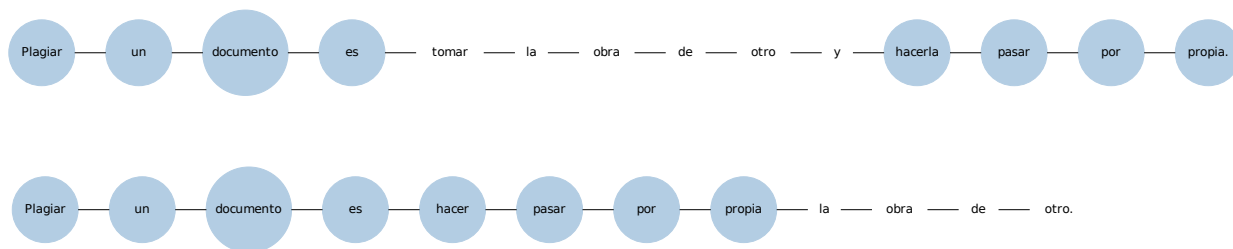


Figura 2.6: Un ejemplo de que la máxima subsecuencia común es sensible al orden de las palabras. En realidad el sufijo “la” de la palabra “hacerla” no pertenece a la subsecuencia, pero decidimos no reflejarlo para no complicar el dibujo.

Greedy String Tiling El problema de los métodos mencionados es que consideran una única subsecuencia o subcadena. Esto lleva a que los resultados de una simple transposición no sean detectados. Si en cambio se buscan todas las cadenas o secuencias comunes mayores a una cierta longitud, esto puede solucionarse.

La detección de pasajes plagiados puede plantearse como un conjunto de secuencias no superpuestas que cubran los textos maximizando la cantidad de caracteres cubiertos. No se conoce ningún algoritmo que pueda resolver este problema en tiempo polinomial. Sin embargo, como señala Wise[37], al buscar plagio se prefieren cadenas largas ya que a mayor longitud, mayor es la probabilidad de que el fragmento sea producto de una copia (y no una coincidencia casual). A partir de esto surge un algoritmo goloso conocido como Greedy String Tiling.

Wise usa el concepto de *matches*: un par de subcadenas comunes a ambos textos. Es una asociación temporal y no necesariamente única: una de las dos cadenas podría estar involucrada en más de un match. Por otro lado menciona los *tiles*, una asociación permanente y única de un par de subcadenas comunes. Los tiles son formados marcando matches. Una vez que un match fue marcado ya no podrá ser parte de otro tile. La longitud de los matches considerados debe ser igual o mayor a un valor, llamado longitud mínima del match.

Consta de dos etapas: en la primera se toman los matches maximales de una longitud mínima. En la segunda los matches se analizan en orden decreciente de longitud y aquellos pares en los que ninguna componente fue marcada forman un nuevo tile. De esta forma se obtienen matches que no se solapan y que son tan largos como es posible. Una vez analizados todos los matches maximales de esa longitud se disminuye la longitud considerada y se repite el proceso anterior. Esto termina cuando se alcanza la longitud

mínima del match.

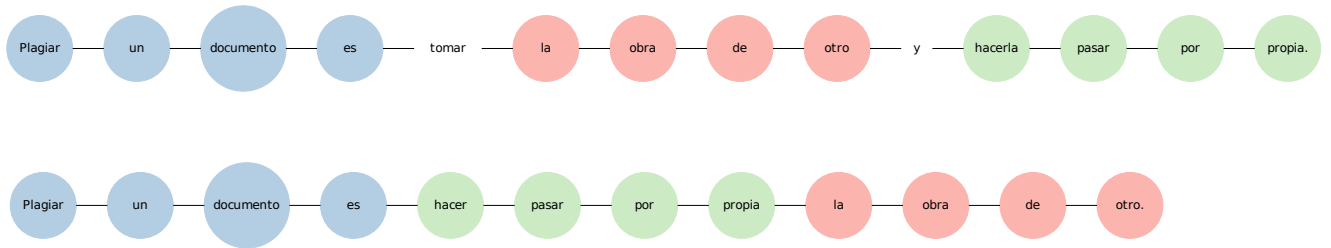


Figura 2.7: El ejemplo anterior analizado con GST. Los distintos colores representan los distintos tiles que retorna el algoritmo.

La primera etapa de GST se suele resolver utilizando Karp Rabin: un algoritmo de búsqueda de cadenas, basado en hashing. Para encontrar las cadenas comunes entre dos textos de por lo menos m caracteres, se calculan y almacenan los hashings de todas las cadenas de longitud m del primer documento⁵. Luego se calculan los hashings de las cadenas del otro texto, y se comparan entre sí. Cada vez que hay una coincidencia de hashings, se comparan ambas cadenas para descartar una colisión. La comparación se hace hasta encontrar un caracter que difiera, de forma tal de encontrar un match maximal.

Para que este enfoque resulte eficiente, la función de hash utilizada tiene que permitir que el cálculo de un hash a partir del anterior sea menos costoso que el primero. Recordemos que entre una cadena y la siguiente hay sólo un caracter de diferencia, lo que permite reutilizar algunos cálculos. Karp Rabin es simplemente una optimización para Greedy String Tiling. El enfoque de iterar sobre ambas cadenas para encontrar

⁵En esta parte siempre hablamos de cadenas sin marcar, es decir que no pertenecen a ningún tile.

aquellas comunes tiene complejidad $O(lm)$, donde l y m son las longitudes de los textos analizados. Karp Rabin tiene la misma complejidad en el peor caso (que haya colisiones en la función de hash que fueren una comparación carácter a carácter de las cadenas) pero en la práctica tiene complejidad $O(l+m)$. La complejidad de Greedy String Tiling es en peor caso $O((l+m)^3)$ pero en la práctica es $O(l+m)$.

El resultado de Greedy String Tiling es un conjunto maximal de pares de cadenas no solapadas comunes a los textos, como puede verse en la figura 2.7. Este conjunto es un matching aproximado: tolera agregado y borrado de caracteres pero también tolera reordenamiento.

El resultado del algoritmo suele ser procesado con algún algoritmo para combinar los tiles cercanos entre sí (merging algorithm)⁶ y considerarlos un solo pasaje, como hacen Clough et. al. [29]. En el ejemplo, el resultado estaría formado por todas las palabras pintadas de color. En el mismo algoritmo de merging se podría realizar el análisis necesario en los algoritmos aproximados que mencionamos antes (cuántas cadenas comunes hay o qué tan separadas están, por ejemplo).

Si bien el algoritmo es de matching aproximado, hasta ahora hablamos de tiles con matching exacto (vistos individualmente). Greedy String Tiling se puede generalizar para que cada tile sea un matching aproximado: Wise lo hace pero para resolver un problema de bioinformática [38]. Clough dice estar investigando formas de hacer los tiles más resistentes a reescritura (por ejemplo tolerando el cambio de una palabra por un sinónimo, el borrado o agregado de palabras) para que el enfoque sea aún más resistente a cambios en el texto [39]. Sin embargo, no hemos encontrado ninguna publicación al respecto.

⁶En general estos algoritmos establecen condiciones del tipo: si la diferencia entre dos tiles es menor a n , considerarlos parte del mismo pasaje plagiado.

Dotplot El Dotplot es una técnica para visualizar información. Se usa en biología para analizar secuencias genéticas, en computación para detectar código duplicado y para detectar similitud en documentos [40, 41]. Otra aplicación es detectar plagio en texto o código fuente, ya que permite comparar dos documentos entre sí.

Es un gráfico bidimensional en el que se representan las coincidencias entre las dos secuencias y en cada eje la posición del carácter, n-grama o palabra que coincide (por ejemplo, en el eje x la posición en el documento sospechoso, y en el eje y la posición en el original).

Una vez obtenido el gráfico, se deben analizar automáticamente los patrones obtenidos. Supongamos que hay una frase idéntica en ambos textos. Esto se verá en el dotplot como una línea diagonal. Por otro lado, si hay pequeños fragmentos copiados entre sí pero en distinto orden esto se verá reflejado como distintas diagonales. Frases con algunas palabras cambiadas, diagonales fragmentadas. Otra forma que indica plagio son los cuadrados: acumulación de coincidencias cortas en un mismo lugar, producto de plagio más ofuscado que en los ejemplos anteriores [32, 41]. Además de formas que indican plagio, en los dotplots suelen aparecer coincidencias pequeñas producto de frases usuales, nombres propios o casualidades. Estas coincidencias no nos interesan y hay que descartarlas; se las suele llamar “ruido”.

Es común leer en publicaciones e informes de hace algunos años (2003 o anteriores) [39, 42] que los dotplots tienen como desventaja no cuantificar la similitud entre dos textos. Esto es porque se refieren a detectar plagio o reuso de texto observando manualmente el resultado del gráfico. Actualmente se hace un análisis automático de los patrones del gráfico [32, 43, 30], mediante procesamiento de imágenes, permitiendo obtener las posiciones exactas de las frases plagiadas.

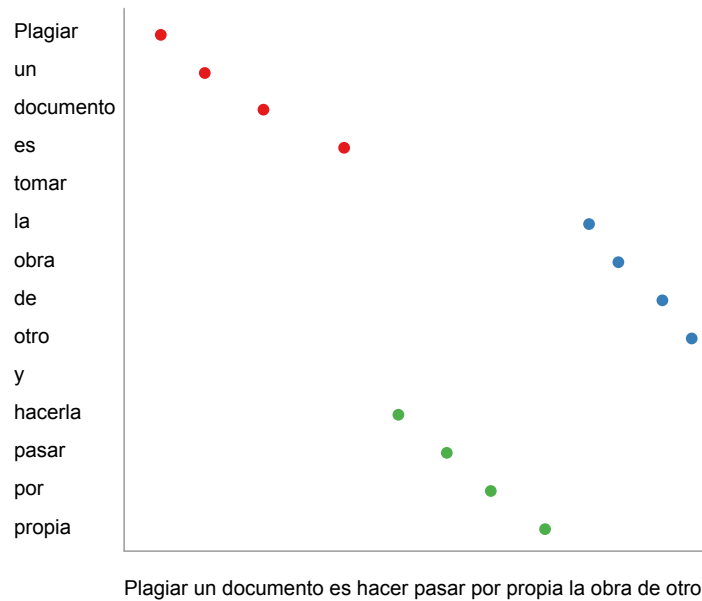


Figura 2.8: El mismo ejemplo de frases plagiadas alterando el orden, representado con un dotplot (se reemplazó las posiciones de las palabras por las mismas palabras).

Un problema que presenta este método es su complejidad: es cuadrático en función de las longitudes de los documentos analizados, tanto en tiempo como en espacio [30]. La complejidad proviene del hecho que en el gráfico se marcan todas las coincidencias: si en el primer texto aparece la expresión “Por ejemplo” tres veces y en el segundo cuatro, las coincidencias marcadas serán doce. Otra desventaja de este gráfico es la aparición de matches casuales (ruido) que ya mencionamos.

Grozea, Popescu y Gehl [30] consideran menor cantidad de coincidencias: la primera con la primera, la segunda con la segunda, etc. De esta forma, en el ejemplo anterior representarían tres coincidencias, frente a las doce de un dotplot tradicional.

Basile et al. [32] codifican los textos con números de forma similar al encoding T9 , utilizado en celulares: (a,b,c) \rightsquigarrow dos, (d,e,f) \rightsquigarrow tres, etc. Cero y uno son utilizados para

espacios y para otros símbolos. En general, dada una cadena de números codificada de esta manera, sólo se encontrará una cadena que tenga sentido en el idioma utilizado. La ventaja de esta forma de escribir las cadenas es que se puede almacenar de forma más eficiente, con un solo número, donde cada dígito corresponde a un carácter. Luego, por cada posición del documento original a comparar, almacenan la última posición en ese archivo de la misma cadena de largo siete y guardan en un vector de diez a la siete posiciones todas las cadenas posibles de longitud siete, con la última posición en la que aparecen. De esta forma, para cada posición del texto sospechoso se calcula el o los matches más largos en el original [44]. Esto hace que no se representen las coincidencias menores a siete, lo cual disminuye las coincidencias casuales (las cortas al menos).

2.2.1.2. Métodos semánticos o lingüísticos

Los métodos analizados en esta sección no ven las palabras como símbolos, sino que las relacionan con otros términos según su significado (como sinónimos, antónimos, hipónimos, etc). Son enfoques más complicados de implementar que los sintácticos, por lo que la cantidad de trabajos en este área es menor a la anterior. De todas formas, es un área en crecimiento y seguramente el futuro en la detección de plagio, ya que puede detectar casos que los enfoques tradicionales no pueden.

Todas las publicaciones que encontramos [45, 46, 34, 51] analizan los textos a nivel de frases y utilizan estos métodos para un análisis detallado - probablemente porque estos enfoques son más costosos computacionalmente.

Todos los enfoques que analizamos utilizan WordNet⁷, una base de datos lingüística que agrupa las palabras en conjuntos de sinónimos, *synsets*, desarrollada en la Universidad

⁷<http://wordnet.princeton.edu/>

de Princeton. Los synsets están vinculados entre sí mediante distintas relaciones sintácticas (ver tabla 2.1).

El preprocesamiento varía según la preferencia de los autores. En algunos se usa stemming, es decir la reducción de una palabra a su raíz, y remoción de palabras comunes. Muftah y Jabr en cambio hacen la remoción de palabras analizando la frase. En vez de eliminar términos basándose en una lista de vocablos comunes, remueven palabras según su clase: remueven artículos, conjunciones, preposiciones, etc.

PPChecker [34] utiliza la relación de sinonimia de WordNet para calcular la similitud de documentos. Computa este valor contando la longitud de las palabras comunes, considerando los sinónimos como la misma palabra.

Alzahrani y Salim miden la similitud de la frase, en base a la individual de cada término. Otorgan distintos puntajes a los pares de palabras: uno si trata de la misma palabra, medio si son sinónimos, cero en otro caso. Si el valor final obtenido supera un cierto umbral, se considera la frase plagio. Finalmente se combinan las oraciones consecutivas marcadas para conformar pasajes.

Tsatsaronis et al. buscan todos los conceptos relacionados con las palabras de las frases analizadas. Arman caminos entre pares de vocablos (uno perteneciente a cada oración) con un peso asociado, que depende del tipo de relación de los enlaces, de la especificidad de las palabras-nodo intermedias, y de la longitud del camino. Seleccionan el mayor peso de estos caminos, que representará la relación semántica entre esas dos palabras. Lo novedoso de este enfoque es utiliza Wikipedia para relacionar palabras que no están en WordNet. De esta forma son tenidos en cuenta nombre propios, como países o ciudades.

Por otro lado calculan la similitud léxica: estiman la especificidad de los términos de

las frases. El motivo que dan es que en los textos podría haber vocablos muy específicos que podrían no estar bien representados en WordNet o Wikipedia. Para el cómputo utilizan la media armónica de los pesos tf-idf de los términos.

Finalmente, se calcula el resultado final combinando los dos conceptos anteriores, eligiendo los pares de palabras de forma tal de maximizar la relación semántica.

Estos métodos resultan muy interesantes ya que son capaces de detectar plagio con altos niveles de ofuscación. Enfoques que consideren las palabras como algo más que secuencias de bits son necesarios tanto para detectar plagio con reelaboración como para detectar plagio en distintos idiomas.

Si bien WordNet está disponible solo en inglés, existen bases de datos con la misma interfaz en otros idiomas, como EuroWordNet ⁸, permitiendo que los métodos no estén ligados únicamente a un idioma.

⁸<http://www.illc.uva.nl/EuroWordNet/>

Relación semántica	Significado	Categoría sintáctica	Ejemplos
Sinonimia	mismo o similar significado	S V Adj Adv	rápido - veloz
Antonimia	significado opuesto	Adj Adv S V	rápido - lento
Hiponimia	término más específico	S	árbol - planta
Meronimia	parte del significado de la otra palabra	S	dedo - mano
Troponimia	hiponimia para verbos	V	caminar - desplazarse
Implicación (Entailment)	implicación - causa	V	roncar - dormir

Cuadro 2.1: S = Sustantivos, Adj = Adjetivos, V = Verbos, Adv = Adverbios

Relaciones semánticas en WordNet[47].

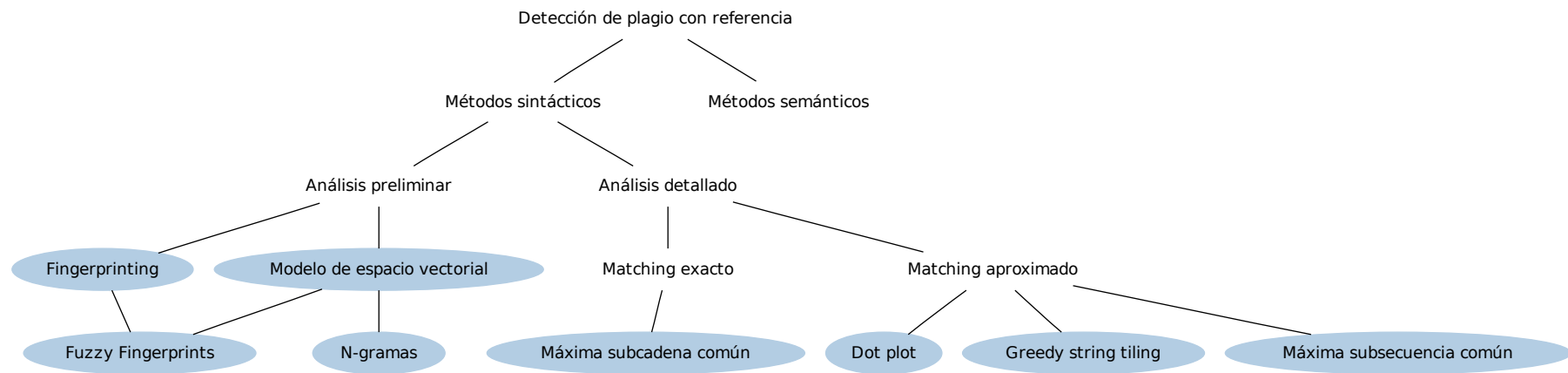


Figura 2.9: La taxonomía de métodos de detección de plagio con referencia.

2.2.2. Otros aspectos

2.2.2.1. Unidad de comparación

Un aspecto importante de los sistemas de detección de plagio, es la unidad de comparación a utilizar. Las unidades de comparación posibles son: todo el documento, párrafos, frases o (grupos de) palabras o caracteres.

Usar documentos como unidad de comparación es algo habitual en modelos como el de Vector Space Model para la comparación global. Esta unidad se usa también al hacer análisis detallado entre documentos: al tener que señalar los pasajes plagiados se deben revisar los dos textos completos.

Grupos de palabras o caracteres son la unidad de comparación utilizada en los métodos basados en n-gramas y fingerprints. Las frases son una unidad común, pero que pueden presentar problemas [25]: ecuaciones, figuras y abreviaciones suelen confundir al programa, y el texto no es correctamente dividido en frases.

Los párrafos son una unidad utilizada con menor frecuencia: tal vez eso suceda porque es una unidad muy grande para buscar textualmente (para guardar su fingerprint por ejemplo), pero chica para los enfoques que usan modelos como VSM.

No siempre se comparan unidades en forma simétrica (frases contra frases, por ejemplo), en algunos casos la comparación es asimétrica: Barrón-Cedeño y Rosso [27] dividen el texto sospechoso en frases y cada frase es comparada contra un documento completo.

2.2.2.2. Funciones de distancia

Mas allá del enfoque elegido, en los sistemas de detección de plagio suele aparecer el concepto de función de distancia, que puede estar definida sobre conjuntos o vectores.

Las medidas más utilizadas son coseno, el índice de Jaccard, el coeficiente de Dice y el de overlap. Toman valores entre cero y uno en general⁹. Se las conoce como medidas de similitud, ya que al aumentar las características o componentes similares de los documentos comparados, el valor de la función aumenta.

Coseno	$\frac{\sum_{i=1}^n x_i \cdot y_i}{\sqrt{\sum_{i=1}^n x_i^2 \times \sum_{i=1}^n y_i^2}}$
Jaccard	$\frac{\sqrt{\sum_{i=1}^n x_i \cdot y_i}}{\sqrt{\sum_{i=1}^n x_i^2} + \sqrt{\sum_{i=1}^n y_i^2} - \sqrt{\sum_{i=1}^n x_i \cdot y_i}}$
Dice	$\frac{2\sqrt{\sum_{i=1}^n x_i \cdot y_i}}{\sqrt{\sum_{i=1}^n x_i^2} + \sqrt{\sum_{i=1}^n y_i^2}}$

Cuadro 2.2: Funciones de distancia usadas para detección de plagio definidas sobre vectores[48].

Como mencionamos anteriormente, los documentos pueden ser vistos como vectores en los que cada término tiene un peso asignado: ya sea cero o uno en el caso de un modelo binario, como la frecuencia del término o el peso tf-idf. En el modelo de espacio vectorial, es muy común utilizar el coseno como función de distancia. Sin embargo, esta medida no es apropiada para detectar plagio.

Una gran cantidad de autores la critican. Kang et al. argumentan que el coseno es una

⁹El coseno va de menos uno a uno normalmente, pero al tratarse de vectores con componentes positivas, su rango va de cero a uno.

buena medida para calcular similitud pero no para detectar plagio [34]. Sostienen que lo indicado sería medir las coincidencias entre los textos en absoluto (en vez de medirlas en relación a la longitud de los textos), ya que según ellos cuantas más coincidencias hay, mayor es la evidencia de que hubo plagio. Como medida en su programa, PPChecker, usan una función del solapamiento entre palabras (con la particularidad de que consideran sinónimos como la misma palabra ¹⁰).

Shivakumar y Garcia-Molina tampoco consideran que sea una buena medida de plagio [25]. Afirman que sólo es apropiada cuando los valores de los vectores son del mismo orden de magnitud. Para solucionar esto definen el conjunto de proximidad de una palabra w_i como el conjunto de palabras que cumplen la condición:

$$\epsilon - \left(\frac{F_i(R)}{F_i(S)} + \frac{F_i(S)}{F_i(R)} \right) > 0 \quad (2.2)$$

donde $F_i(D)$ es la cantidad de ocurrencias de w_i en el documento D y ϵ es un parámetro de tolerancia en el rango $(2^+, \infty)$, configurable por el usuario. Si $F_i(S)$ o $F_i(R)$ son cero, la condición no se satisface.

Luego definen la medida subconjunto:

$$\text{subconjunto}(D_1, D_2) = \frac{\sum_{w_i \in c(D_1, D_2)} \alpha_i^2 * F_i(D_1)F_i(D_2)}{\sum_{i=1}^N \alpha_i^2 * F_i^2(D_1)} \quad (2.3)$$

donde α_i es el peso asociado a w_i . Esta medida es similar a coseno, pero es asimétrica. Al ser comparados con coseno, dos documentos pueden obtener un valor bajo, aún cuando

¹⁰Usan WordNet para encontrarlos.

uno es subconjunto de otro. Este problema se evita al normalizar el numerador sólo con respecto a un documento (por eso decimos que es asimétrica).

Finalmente la medida de similitud entre dos documentos es:

$$\text{sim}(D_1, D_2) = \max\{\text{subset}(S, R), \text{subset}(R, S)\} \quad (2.4)$$

En el caso en que $\text{sim}(D_1, D_2) \geq 1$, lo definimos como igual a uno, para que el rango esté entre cero y uno. Esta función está diseñada para obtener valores altos cuando se analiza un documento que es subconjunto o superconjunto de otro. Los autores afirman que esta medida es superior a la del coseno, sin embargo en la publicación no hay experimentos que las comparen.

Pese a las críticas, encontramos dos investigaciones en las que se comprueba que el coseno tienen un desempeño similar a Jaccard y Dice en detección de plagio. Barrón-Cedeño, Eiselt y Rosso [50] encontraron que Jaccard y el coseno tienen aproximadamente la misma cobertura, máximo match falso y separación (ver sección 2.4). Muftah y Jabr también experimentaron con estas medidas. En términos de cobertura, Dice y el coseno tienen valores similares, y el coeficiente de Jaccard tiene una performance inferior. Si bien es probable que medidas como la de PPChecker tengan mejor performance (ya que fue diseñada específicamente para plagio y tiene en cuenta sinónimos) el coseno resulta una buena opción en comparación con otras medidas utilizadas habitualmente.

Las medidas de distancia se pueden definir también sobre conjuntos. ¿Qué representan estos conjuntos? Pueden tratarse de n-gramas [49], fingerprints o cualquier otra forma de representar el documento mediante un conjunto. A continuación, haremos un análisis

Coeficiente de Dice	$\frac{2 A \cap B }{ A + B }$
Índice de Jaccard o Resemblance	$\frac{ A \cap B }{ A \cup B }$
Coeficiente de Overlap	$\frac{ A \cap B }{\min(A , B)}$
Containment	$\frac{ A \cap B }{ A }$

Cuadro 2.3: Funciones de distancia usadas para detección de plagio definidas sobre conjuntos [49, 29].

numérico de los índices de Jaccard, Dice y Overlap. Para estudiar los índices, armamos tres casos distintos, que se ven en las figuras de esta sección.

Los casos están formados por tres representaciones de los conjuntos con diagramas de Venn en las que se aumenta alguna variable (tamaño de un conjunto o tamaño de la intersección). Para cada sub-caso hay un gráfico comparativo de los valores de las distintas funciones de distancia. Por otro lado, hay para cada caso un scatter plot que grafica el valor de los coeficientes para distintos valores de la variable considerada.

- Caso 1: Aumentamos la intersección entre los dos conjuntos, manteniendo los tamaños de los conjuntos, para ver cómo incide en los coeficientes.
- Caso 2: Aumentamos el tamaño de uno de los conjuntos, manteniendo la intersección.
- Caso 3: Un conjunto está incluido en el otro. Aumentamos el conjunto más pequeño.

De estos casos sacamos las siguientes conclusiones:

- El número absoluto que toman estas funciones en realidad no nos importa, ya que se puede establecer un umbral a partir del cual el caso sea considerado plagio según sea necesario. Lo que realmente nos interesa es ver como se comportan las funciones al alterar ciertas variables.
- Al aumentar el tamaño de la intersección de los dos conjuntos, los valores aumentan. Esto es correcto, ya que al haber más elementos en común entre los conjuntos (sean n-gramas, palabras, etc) aumentan las probabilidades de que se trate de plagio.
- Si se aumenta el tamaño de uno de los conjuntos, pero manteniendo la misma intersección los coeficientes disminuyen, ya que la intersección pierde peso relativo al tamaño de los conjuntos.
- En el caso particular en el que un conjunto está incluido en otro, probamos aumentar el conjunto de menor tamaño y ver que pasaba. Tanto Jaccard como Dice aumentan, lo cual nos pareció razonable. Overlap en cambio se mantuvo estable: vale uno en todos los casos. Esta medida no indica si dos documentos son similares, sino si un documento está contenido en otro. Podría usarse para detectar copias que provienen de varias fuentes: en ese caso probablemente las otras medidas no tengan valor tan alto como esta.

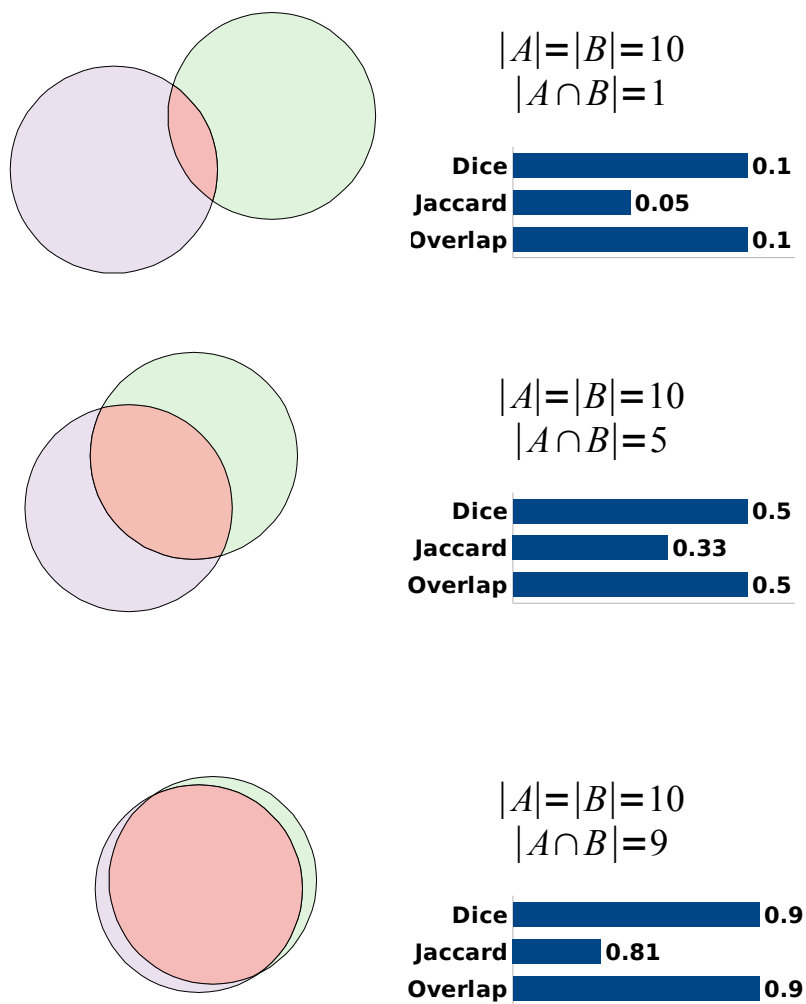


Figura 2.10: Caso 1: representación de los coeficientes de Dice, Jaccard y Overlap al aumentar la intersección de dos conjuntos.

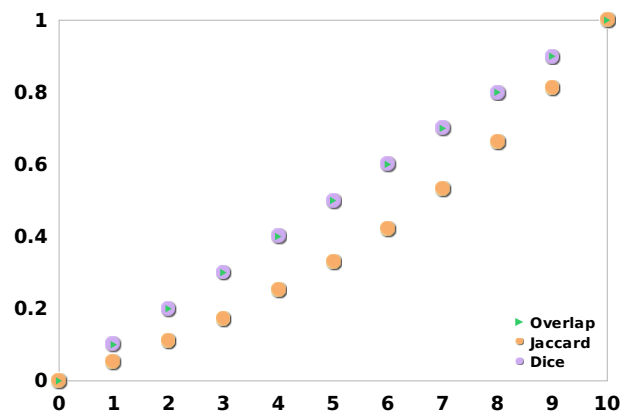


Figura 2.11: Caso 1: scatter plot de los coeficientes de Dice, Jaccard y Overlap al aumentar la intersección de dos conjuntos. En el eje x el valor de la intersección, y en el eje y el valor de los distintos índices.

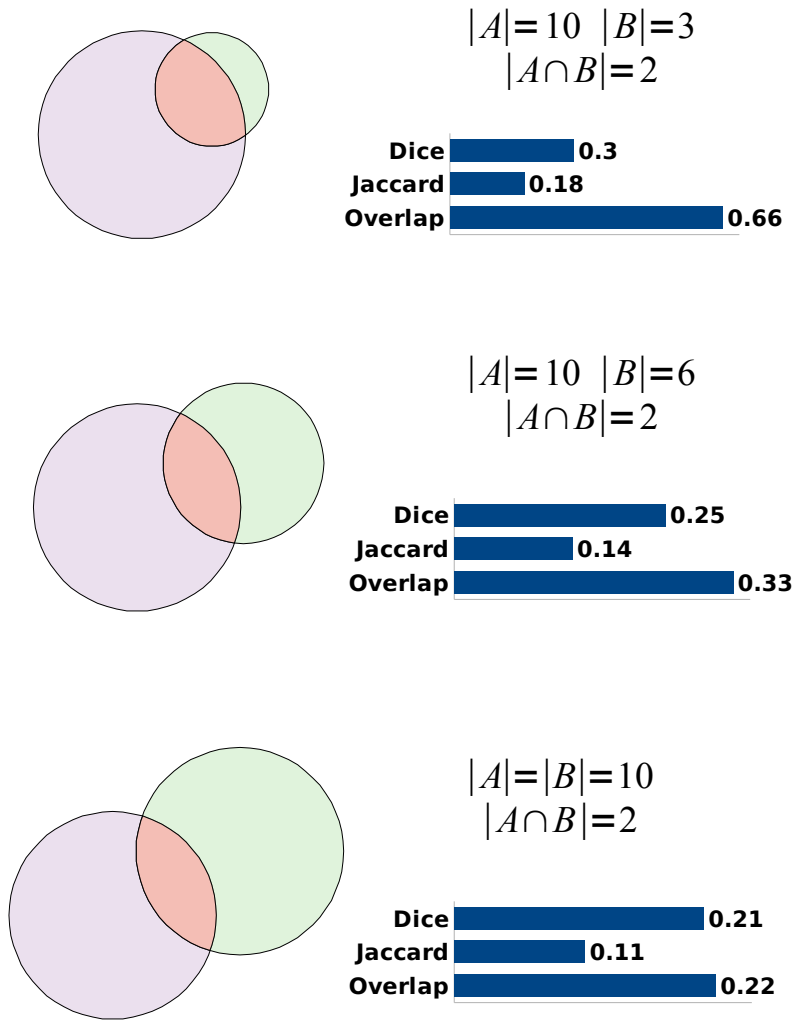


Figura 2.12: Caso 2: representación de los coeficientes de Dice, Jaccard y Overlap al aumentar el tamaño de uno de los dos conjuntos (manteniendo la intersección).

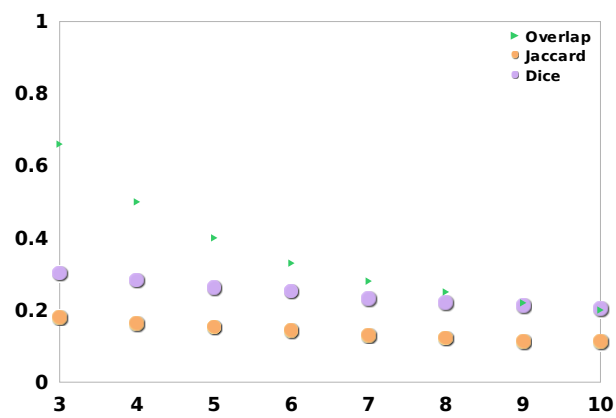


Figura 2.13: Caso 2: scatter plot de los coeficientes de Dice, Jaccard y Overlap al aumentar el tamaño de uno de los dos conjuntos (manteniendo la intersección). En el eje x está representado el tamaño del conjunto que aumenta y en el eje y los valores de las distintas funciones de distancia.

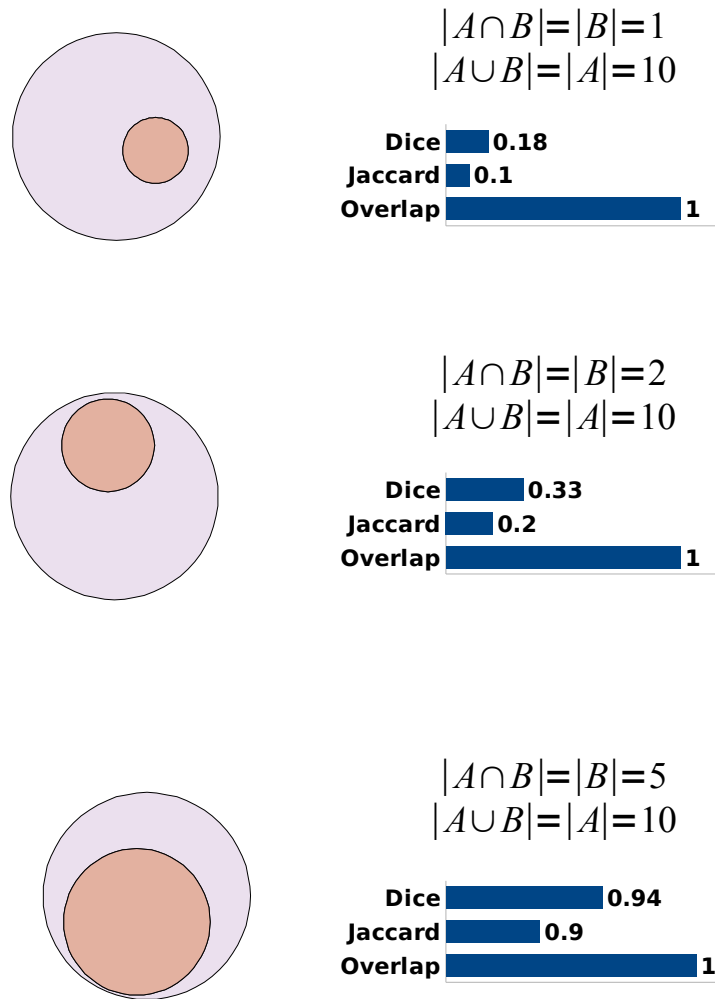


Figura 2.14: Caso 3: Un conjunto está incluido en el otro. Representación de Dice, Jaccard y Overlap al aumentar el tamaño del conjunto más pequeño.

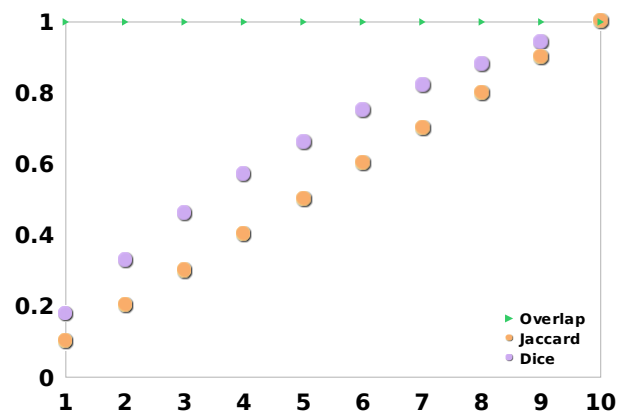


Figura 2.15: Caso 3: Un conjunto está incluido en el otro. Scatter plot de Dice, Jaccard y Overlap al aumentar el tamaño del conjunto más pequeño. En el eje x el tamaño del conjunto que aumenta, y en el eje y el valor de las funciones de distancia.

2.2.3. Más allá del corpus de referencia: búsqueda de documentos

Como ya mencionamos antes, la detección con un corpus de referencia se basa en la suposición de que el o los documentos fuentes del plagio se encuentran en la colección. Si esto no sucede, la detección intrínseca no es la única opción: se puede buscar el documento en Internet.

Al buscar el texto en la Web, estamos ampliando enormemente la cantidad de documentos disponibles. Además, nos permite buscar en documentos muy recientes, algo difícil de encontrar en una colección de referencia. De todas formas las búsquedas no son excluyentes: se puede hacer detección de plagio con un corpus de referencia y además buscar documentos en la Web. Si bien hay muchos artículos y publicaciones disponibles en Internet, algunos están protegidos por leyes de Copyright y solo pueden ser obtenidos mediante el pago de una suma, por lo que contar con una base de datos de publicaciones científicas resulta muy útil.

Este problema se conoce como recuperación de documentos web similares (Web document similarity retrieval problem) [52]. Para buscar los textos, se usa un enfoque conocido como meta-búsqueda, que consiste en utilizar la infraestructura existente de varios buscadores (como Google, Yahoo, Bing, etc) y combinar los resultados obtenidos de cada uno de ellos [53, 54].

El desafío está en generar múltiples consultas a partir de un documento de forma tal de poder enviarla a los distintos motores de búsqueda y obtener las fuentes del plagio (si existieran). También puede ser utilizado para buscar documentos similares.

Las consultas son generadas a partir de términos del documento sospechado. Un

enfoque sencillo, utilizado para encontrar documentos similares, es extraer términos del documento, eliminando las palabras comunes [53]. En detección de plagio este tipo de enfoque no es efectivo, ya que el documento podría contener más de una fuente y partes originales, con lo cual buscar términos al azar, o los términos mas frecuentes no es efectivo [55].

Pereira y Ziviani [55] usan secuencias de palabras para generar búsquedas. Primero seleccionan una palabra (*anchor*) según algún criterio, y luego toman n términos a la izquierda y n a la derecha, que conforman la lista de palabras a buscar. Para seleccionar los anchors los criterios son:

- Palabras al azar, distribuidas de forma uniforme.
- Términos que no pertenecen al diccionario. En general se trata de palabras con errores ortográficos o nombres propios.
- Palabras más frecuentes.
- Palabras menos frecuentes.

Stein et al[11] proponen utilizar técnicas de detección intrínseca, como punto de partida para buscar potenciales fuentes en la Web. Sin embargo, aún suponiendo que el plagio provenga de una misma fuente (no necesariamente es así) estos algoritmos sólo diferencian partes del documento escritos por distintos autores, sin señalar cuales son plagiados y cuales originales.

2.3. Corpus disponibles

Una parte fundamental de la investigación científica es la experimentación. En nuestro caso, nos referimos a probar que el programa desarrollado detecte casos de plagio y no señale como plagio documentos que no lo son. Para esto necesitamos un corpus de documentos sobre el cual hacer nuestras pruebas: lo ideal sería un conjunto de textos con algunos casos de plagio reales y documentos del mismo tema pero que no sean plagio.

Utilizar casos de plagio reales tiene varios problemas[58]:

- Los casos conocidos son encontrados accidentalmente y en general se trata de copias textuales o con muy poca reelaboración.
- Los textos no pueden ser publicados sin la autorización de los autores.
- Tampoco pueden ser publicados sin que se sepa quién fue el autor del plagio.
- Muchos casos ocurren en ámbitos educativos como escuelas o universidades. Ni los profesores ni los establecimientos educativos tienen interés en difundir estos casos.

2.3.1. DejaVu

Este es el único corpus que contiene plagios reales. Se trata de una base de datos de publicaciones médicas que contiene casos de plagio, o de auto-plagio (autores que repiten publicaciones sin citarse a sí mismos y/o sin agregar nueva información) [56]. La base de datos se encuentra disponible para ser consultada online <http://dejavu.vbi.vt.edu/dejavu/>, y también puede ser descargada. Existe también un motor de búsqueda, eTBLAST <http://etest.vbi.vt.edu/etblast3/> que entre otras cosas puede ser utilizado para detectar plagio.

La información está organizada como pares de publicaciones similares en las que se incluye:

- Autor
- Título
- Lugar y fecha de publicación
- Abstract
- Número de similitud
- Categoría, que se refiere a si se trata de un plagio a terceros, del mismo estudio que evolucionó, de un auto-plagio, etc.
- Idioma

El problema es que este corpus no está pensado como una herramienta para evaluar un sistema de detección de plagio, sino para que médicos o biólogos puedan acceder a la información, ver si alguien plagió sus publicaciones, etc. En la base de datos aparece el abstract de los documentos, pero no su contenido. Para que este corpus sea de utilidad, se debería tener acceso a la base de datos Medline[®], de donde fueron obtenidos los artículos.

De todas formas, incluso contando con los textos completos, la información de similitud es global, no se señalan pasajes concretos como plagiados. Además la revisión está encarada desde un punto de vista ético que no nos interesa: a efectos de evaluar nuestro sistema, una copia textual y una reentrega de un trabajo corrigiendo errores ortográficos (errata, como la llaman ellos) es lo mismo.

2.3.2. METER

El corpus METER (MEasuring TExt Reuse) es parte de un proyecto con el mismo nombre en el que se estudia la reutilización de texto en periodismo. Es una colaboración entre los departamentos de periodismo y computación de la Universidad de Sheffield, Reino Unido¹¹. Este corpus presenta casos reales e incluye los textos completos.

El corpus está formado por artículos escritos por una agencia de noticias, la Press Association, y otros sobre mismo tema, de distintos diarios de UK: The Sun, The Mirror, etc. Algunos de estos artículos se basan en los de la agencia y otros no. Para encontrar ejemplos interesantes, tomaron las siguientes medidas:

- La agencia elegida es la más importante en el Reino Unido, por lo que si el diario usó alguna fuente para la noticia muy probablemente haya sido la de esa agencia.
- Las noticias elegidas no son tan importantes, por lo que es más probable que el diario haya usado una historia prefabricada, sin enviar un periodista al lugar.
- Se trata de historias conocidas como “hard news”: basadas en hechos puntuales en vez de opiniones. Por un lado, este tipo de noticias son generalmente tomadas de agencias de noticias. Por otro lado, si se trata de un artículo escrito de forma independiente la estructura y el vocabulario serán muy similares, haciéndolo un caso interesante de prueba.
- Para cubrir distintos estilos de escritura, se tomaron noticias del mundo del espectáculo y de informes judiciales. Se seleccionaron diarios de diferente registro: periódicos más formales (quality press) y populares (tabloids).

¹¹<http://www.dcs.shef.ac.uk/nlp/meter/>

Trabajaron con un periodista que analizaba los textos y los clasificaba como:

- Totalmente derivado: la noticia de la PA es la única fuente.
- Parcialmente derivado: la noticia de la PA es una fuente, pero no la única.
- No derivado: El artículo fue escrito de forma independiente.

No se trata de casos de plagio sino de reuso de texto, previo pago de una suma a la agencia de noticias. Si bien probablemente el reuso de texto tenga similitudes con el plagio, no se trata del mismo fenómeno. El periodista que usa una noticia no está haciendo nada malo, simplemente la reescribe con el estilo del diario para el que trabaja, la alarga o acorta según sus necesidades. Esto difiere de la reescritura en un caso de plagio, que está orientada a evitar la detección. Los mismos autores lo propusieron en su momento como una alternativa para evaluar sistemas de detección de plagio, ya que no existían corpus diseñados para esto (hoy en día eso no es cierto, como veremos más adelante). De todas formas, sigue siendo una opción interesante para evaluar reescritura de textos, ya que se trata de casos reales. Para analizar la reelaboración en detalle el corpus provee anotaciones a nivel de frase o palabra, que también se dividen en tres categorías:

- Literal: el texto fue tomado palabra por palabra para expresar la misma información.
- Reescritura: el texto fue parafraseado para expresar la misma información.
- Nuevo: texto que no aparece en el artículo de la agencia.

Las anotaciones a nivel documento cubren todos los artículos, pero las detalladas están disponible solo para el veinticinco por ciento del corpus.

	METER	Deja Vu	PAN	Clough09
Más de un idioma		✓	✓	
Anotaciones detalladas	✓		✓	
Casos reales	✓	✓		
Casos simulados			✓	✓
Casos artificiales			✓	
Cantidad de documentos	1716	79383	68558	100
Longitud de los documentos	Pocas páginas	Un párrafo	1 - 1000 páginas	200 - 300 palabras
Más de un tema	✓		✓	
Plagio inter tema			✓	

Cuadro 2.4: Tabla comparativa de los distintos corpus disponibles.

2.3.3. PAN

PAN¹² es un taller internacional sobre análisis de plagio y atribución de autoría, entre otras cosas. Fue organizado desde el 2007, y a partir del 2009 cuenta con una competencia internacional de detección de plagio. Para la competencia se generó un corpus de casos de plagio ficticios, con el objetivo de evaluar detectores de plagio con referencia, intrínseco y detección de plagio en distintos idiomas.

La competencia resulta muy importante porque provee un marco para comparar distintos métodos de detección entre sí. Además, el corpus es público y las medidas usadas para comparar fueron publicadas, con lo cual cualquier científico del campo puede reproducir fácilmente las pruebas y comparar su enfoque con otros. Luego del certamen, cada equipo publica un informe sobre el método utilizado. De hecho, varias de las fuentes citadas son trabajos presentados en ese taller.

Los organizadores publican todos los años un documento de resumen de la competencia [60, 18] en el que señalan aspectos positivos y negativos de los enfoques presentados. De los trabajos presentados en 2010 rescatan la madurez con respecto a los del año anterior y que casi todos presentan tres fases: análisis preliminar, análisis detallado y postprocesamiento, en lugar de la comparación de a pares de documentos (algo no muy práctico en una colección de datos tan grande). La estructura facilita el uso de índices para hacer un análisis preliminar, ya que los textos están divididos entre sospechosos y originales.

La crítica que hacen es que algunos participantes toman medidas orientadas a ganar la competencia, pero no aplicables en un escenario real. Otra desventaja es que varios

¹²<http://pan.webis.de/>

de los métodos son solo aplicables a colecciones locales, pero no a detección de plagio extrayendo documentos de la Web.

El corpus fue creado utilizando dos métodos: plagio artificial y plagio simulado. Los casos de plagio artificiales son aquellos creados por un programa. Los organizadores del evento usaron varias estrategias para generar estos casos. Dado un pasaje, se genera otro plagiado de las siguientes formas[58]:

- Se reordenan, reemplazan, agregan o borran palabras del pasaje original al azar. Los agregados o reemplazos son tomados del documento donde se ubicará el pasaje plagiado.
- Se reemplazan palabras por sinónimos, antónimos, hipónimos e hiperónimos.
- Se analiza sintácticamente el texto original y se mezclan las palabras manteniendo la misma estructura.

Estas estrategias son combinadas y aplicadas en mayor o menor medida para obtener distintos niveles de ofuscación de plagio. El plagio es insertado inter e intra tópicos. Para obtener los casos en los que efectuar detección intrínseca, se generaron casos de la misma manera, sin incluir los originales en el corpus.

Los casos de plagio simulados fueron escritos por personas que los autores contrataron a través de un sitio de Internet¹³. Todos los contratados hablaban inglés fluidamente y todos los trabajos fueron controlados con posterioridad.

Además de estos casos, una parte del corpus está dedicada a evaluar plagio en distintos idiomas. Para construir este tipo de casos, tradujeron textos del alemán y castellano al inglés en forma automática.

¹³<https://www.mturk.com/mturk/welcome>

Por otro lado, una parte de los documentos no posee plagio alguno, de forma tal de poder medir la precisión de los detectores (más en la sección 2.4).

Una vez construido el corpus, los investigadores lo compararon con el corpus METER y el Clough09 (del que hablaremos a continuación), utilizando un modelo de n-gramas, con valores de n entre uno y diez. Su preocupación principal era ver si el plagio generado algorítmicamente era una alternativa viable al plagio construido manualmente. Llegaron a la conclusión de que los corpus se comportaban de manera similar.

Los resultados de la competencia nos resultaron muy útiles al estudiar la detección de plagio. Tenemos sin embargo una pequeña objeción: los resultados de la detección con referencia, la intrínseca y la detección en distintos idiomas se evalúan en forma conjunta. Para nosotros sería interesante tener resultados parciales de estas tres categorías por separado, ya que algunos grupos no intentan resolver el problema de plagio intrínseco o el de plagio en distintos idiomas. Así se pierde información valiosa: un equipo puede haber encontrado todos los casos de plagio extrínseco, y tener una puntuación global baja por no haber participado en las otras categorías.

2.3.4. Clough09

Este es un corpus pequeño de casos simulados. Está formado por cinco grupos distintos de documentos que son distintas respuestas a cinco preguntas de temas relacionados con Ciencias de la Computación. Los textos fueron escritos por estudiantes de grado o postgrado en Computación, de forma tal que tuvieran cierta familiaridad con los temas tratados. Para cada pregunta hay cinco respuestas cortas (entre doscientas y trescientas palabras):

- **Copia** Respuesta copiada del correspondiente artículo en Wikipedia. No se dieron instrucciones sobre qué partes del artículo copiar, esto era decisión del estudiante.
- **Revisión leve** Respuesta en la que el participante tomaba información de Wikipedia, pero podía alterarla levemente, sin alterar demasiado el orden de la información.
- **Revisión profunda** Respuesta en la que nuevamente el voluntario se basa en Wikipedia, pero se le pide que use las mismas ideas con distintas palabras y estructura. Esto puede incluir combinar o separar oraciones.
- **Respuesta sin plagio** Respuesta en la que el estudiante recibía material de lectura a partir del cual generar la respuesta. Bajo ninguna circunstancia podía leer el artículo de Wikipedia.

Éstas respuestas enumeradas son los textos sospechosos. Además, por cada grupo hay un original, que es el artículo de Wikipedia sobre el tema.

Los autores opinan que la ventaja de este corpus con respecto a otros, es que fue hecho manualmente y no algorítmicamente. Por otro lado tiene algunas desventajas:

- La colección es mucho más pequeña que las otras analizadas, tanto en la cantidad de documentos como en la longitud de los documentos.
- Todos los artículos pertenecen a la misma disciplina.
- No posee anotaciones detalladas.

2.4. Medidas para comparar enfoques

Una vez que un corpus es utilizado para probar un sistema de detección, se deben comparar los resultados obtenidos contra las anotaciones del corpus. Luego se debe cuantificar de alguna manera qué tan buenos son los resultados. Para eso existen algunas medidas de performance, que veremos a continuación.

Cobertura La cobertura es una medida ampliamente usada en Recuperación de la Información. Se calcula como el cociente entre las instancias correctas detectadas sobre las que realmente pertenecen al conjunto relevante. En el caso de la detección de plagio, es el cociente entre los casos de plagio correctamente detectados sobre el total de los casos de plagio. Expresa cuán completo es el algoritmo, cuántos casos logró detectar. Es inversamente proporcional a la cantidad de falsos negativos.

Veamos una definición más formal, tomada de la competencia PAN¹⁴. Sea s un pasaje plagiado, perteneciente a S , el conjunto de todos los pasajes plagiados, $|s|$ su longitud y $|S|$ su cardinal.

$$\text{Cobertura} = \frac{1}{|S|} \sum_{i=1}^{|S|} \left(\frac{\#\text{caracteres detectados de } s_i}{|s_i|} \right) \quad (2.5)$$

En la competencia se cuentan caracteres para poder evaluar distintos enfoques (que usen palabras, n-gramas, etc). Las detecciones de plagio con referencia incluyen tanto los caracteres del texto sospechoso como los del texto original. Una detección r tiene que coincidir en al menos un caracter tanto con el pasaje plagiado como con el pasaje original

¹⁴<http://www.uni-weimar.de/medien/webis/research/workshopseries/pan-10/task1-plagiarism-detection.html>

del correspondiente s . De no ser así, la detección tendrá precisión cero y no contribuirá a la cobertura de s .

Precisión Es otra medida habitualmente usada en recuperación de la información. Es el cociente entre los casos detectados correctamente sobre los casos totales detectados. Representa cuanta fidelidad tiene el algoritmo: dados los casos señalados como plagio, cuantos efectivamente lo son. Es inversamente proporcional a la cantidad de falsos positivos.

Nuevamente, ponemos la fórmula utilizada en PAN, donde r denota una detección del conjunto de detecciones R , $|r|$ su longitud y $|R|$ el cardinal del conjunto.

$$\text{Precisión} = \frac{1}{|R|} \sum_{i=1}^{|R|} \left(\frac{\#\text{caracteres plagiados de } r_i}{|r_i|} \right) \quad (2.6)$$

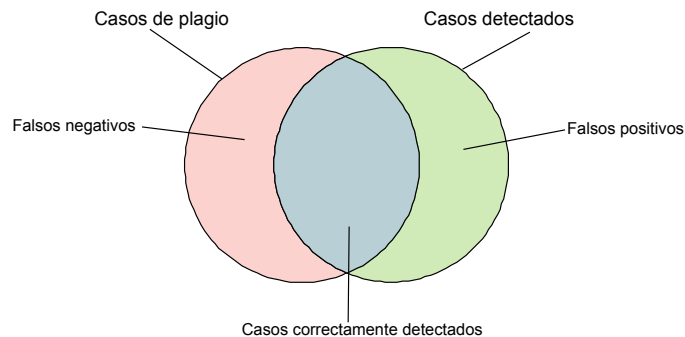


Figura 2.16: Casos detectados y casos de plagio.

Granularidad Esta medida no es una de las medidas utilizadas en la competencia y nos pareció interesante. Se usa para determinar si el algoritmo detecta pasajes plagiados

como un todo o como partes separadas, con o sin superposición.

Sean s un pasaje plagiado del conjunto $|S|$ de todos los pasajes plagiados, $|s|$ su longitud, R el conjunto de todas las detecciones, S_R un subconjunto de S para el cual existen detecciones en R y $|S_R|$ su cardinal.

$$\text{Granularidad} = \frac{1}{|S_R|} \sum_{i=1}^{|S_R|} \#\text{detecciones de } s_i \text{ en } R \quad (2.7)$$

Máximo match falso y separación Hoad y Zobel proponen otras medidas más orientadas a evaluar la fase de análisis preliminar [61]. Por un lado, definen el match falso más alto (Highest False Match), el mayor valor de similitud obtenido por un documento que no contiene plagio:

$$\text{Máximo match falso} = \frac{100 * \text{sim}(D_F, D_S)}{\text{sim}(D_S, D_S)} \quad (2.8)$$

Donde D_S es el documento sospechoso que está siendo analizado (comparándolo contra una colección de referencia) y D_F es el documento no plagiado con valor de similitud más alto. Esta medida está expresada como un porcentaje, por eso se multiplica por cien y se divide por el máximo valor posible, que es el de comparar el documento sospechoso contra si mismo¹⁵.

Esta medida se usa en conjunto con la separación, que representa la diferencia entre el máximo match falso y el mínimo match correcto (el documento plagiado con menor

¹⁵Así se tiene en cuenta medidas que pueden no estar normalizadas.

valor de similitud).

$$\text{Separación} = \text{Máximo match falso} - \text{Mínimo match correcto} \quad (2.9)$$

Por su definición, tiene sentido según sus autores sólo si el algoritmo encuentra todos los casos de plagio, es decir si la cobertura es igual a uno.

Las medidas anteriores comparaban el conjuntos de documentos detectados contra el conjunto de documentos con plagio, y pueden ser aplicadas a la comparación de varios documentos sospechosos contra un corpus de documentos. La separación y máximo match falso se refieren en cambio a la comparación entre *un* documento sospechoso contra una colección.

Capítulo 3

Implementación de software de detección de plagio

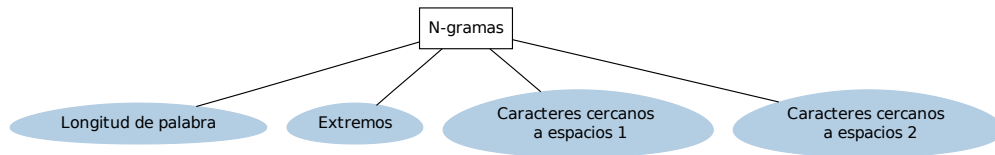
Luego de hacer un profundo análisis de la literatura en detección de plagio, implementamos un sistema de detección de plagio. Los objetivos incluían tanto investigar técnicas nuevas como obtener un software completo capaz de detectar plagio, que soporte distintos tipos de archivos y proporcione información al usuario sobre los pasajes plagiados, a nivel de los caracteres.

Como mencionamos en la primera parte de la tesis, los sistemas de detección de plagio actúan en dos etapas: un análisis preliminar y luego uno detallado. Buscamos nuevos algoritmos para la primera de estas etapas, y decidimos utilizar algún método conocido para la segunda.

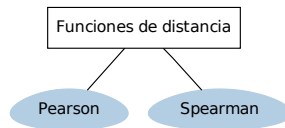
En un primer momento estudiamos la bibliografía existente sobre perceptual hashing, la mayoría de la cual se centra en imágenes audio y vídeo. Sin embargo, nuestro interés se inclinó por los de texto. Descubrimos que su principal aplicación es la detección de

plagio y por eso empezamos a investigar esta problemática.

Empezamos a pensar funciones de hash: se nos ocurrió en primer lugar representar las palabras a través de su longitud, y después a través de su primera y última letra. De acuerdo con nuestra investigación, decidimos tomar estas ideas e implementarlas dentro del modelo de n-gramas. El método de selección de los n-gramas, es decir el hash, fue el primer eje de la investigación. El segundo fue la función de distancia utilizada para comparar los vectores de n-gramas.



(a) N-gramas



(b) Funciones de distancia

Figura 3.1: Los temas investigados, con las distintas propuestas.

3.1. Análisis preliminar

Pre procesamiento Antes de analizar los textos, se los pasó a minúsculas y se eliminaron todos los signos diacríticos presentes y signos de puntuación, para evitar

que leves modificaciones al texto dificulten la detección de copias. También se eliminaron los números, ya que es común en casos de plagio de informes mantener el mismo texto alterando levemente las cifras.

Los espacios en blanco fueron mantenidos, pero tanto tabulaciones como saltos de línea fueron reemplazados por espacios. En el caso de varios espacios consecutivos, fueron reemplazados por uno solo.

3.1.1. N-gramas

Longitud de palabra Pensamos que una posible forma de representar las palabras era a través de su longitud¹. El n en este caso fijaría la cantidad de palabras (representadas con números) seleccionadas. Desafortunadamente, más adelante descubrimos que esta idea ya había sido recientemente desarrollada por Barrón-Cedeño et al.[33] (la publicación es del 2010).

Extremos Utilizamos la primera y la última letra de cada palabra, para representarla en un n -grama. Así un trigramma estaría formado por seis caracteres, los tres pares de extremos pertenecientes a cada palabra.

Caracteres cercanos a espacios Se trata de una variación de la idea anterior. En vez de utilizar el primer y último carácter de cada palabra, tomamos el primer carácter antes de un espacio y del siguiente. Se parece bastante al método anterior si n es mayor a uno: salvo el primero y el último, los otros caracteres del n -grama coinciden con extremos de

¹Permitimos una longitud máxima de veinte caracteres. Es decir, si una palabra es más larga, la representamos con el número veinte.

Algoritmo 1 Calcula el vector de n-gramas de extremos

Parámetros: n : longitud de los n-gramas, $texto$: texto a analizar

```
1: vectorResultado  $\leftarrow$  {}, listaPalabras  $\leftarrow$  {}
2: pos  $\leftarrow$  0
3: mientras #palabras después de pos  $\geq$  n hacer
4:   si pos == 0 entonces
5:     listaPalabras  $\leftarrow$  tomar las primeras n palabras de texto
6:   si no
7:     eliminar primer elemento de listaPalabras
8:     agregar la palabra que empieza en pos al final de listaPalabras
9:   fin si
10:  extremos  $\leftarrow$  ''
11:  para p  $\in$  listaPalabras hacer
12:    extremos  $\leftarrow$  extremos + '-' + p[0]+p[long(p)-1]
13:  fin para
14:  si extremos  $\in$  vectorResultado entonces
15:    vectorResultado[extremos]++
16:  si no
17:    vectorResultado[extremos]  $\leftarrow$  0
18:  fin si
19:  pos  $\leftarrow$  avanzar una palabra
20: fin mientras
21: para clave  $\in$  claves(vectorResultado) hacer
22:  vectorResultado[clave] = vectorResultado[clave] / #palabras en texto - (n-1)
23: fin para
```

palabras. Sin embargo, no era nuestra intención representar palabras, que son comunes entre distintos textos, sino relacionar palabras consecutivas, algo propio del texto. Por eso implementamos una variación de este método, que selecciona dos caracteres antes y dos después del espacio.

Algoritmo 2 Calcula el vector de n-gramas de c caracteres cercanos a espacios

Parámetros: n : longitud de los n-gramas, $texto$: texto a analizar, c : cantidad de caracteres a tomar

```
1: vectorResultado  $\leftarrow \{\}$ , listaEspacios  $\leftarrow \{\}$ 
2: pos  $\leftarrow c$ 
3: mientras (#espacios después de pos > n) o (#espacios después de pos == n y texto.longitud() -
   último espacio  $\geq c$ ) hacer
4:   si pos == 0 entonces
5:     listaEspacios  $\leftarrow$  tomar las posiciones de los n primeros espacios de texto
6:   si no
7:     eliminar primer elemento de listaEspacios
8:     agregar la posición del siguiente espacio después de pos
9:   fin si
10:  caracteres  $\leftarrow ''$ 
11:  para p  $\in$  listaEspacios hacer
12:    caracteres  $\leftarrow$  texto[p-c:p-1] + ' ' + texto[p+1:p+c]
13:  fin para
14:  si caracteres  $\in$  vectorResultado entonces
15:    vectorResultado[caracteres]++
16:  si no
17:    vectorResultado[caracteres]  $\leftarrow 0$ 
18:  fin si
19:  pos  $\leftarrow$  posición del último espacio + 1
20: fin mientras
21: para clave  $\in$  claves(vectorResultado) hacer
22:  vectorResultado[clave] = vectorResultado[clave] / #espacios en texto - (n-1)
23: fin para
```

N-gramas conocidos Decidimos además implementar n-gramas de caracteres y de palabras, como se usa en otras publicaciones, para poder comparar contra nuestras estrategias.

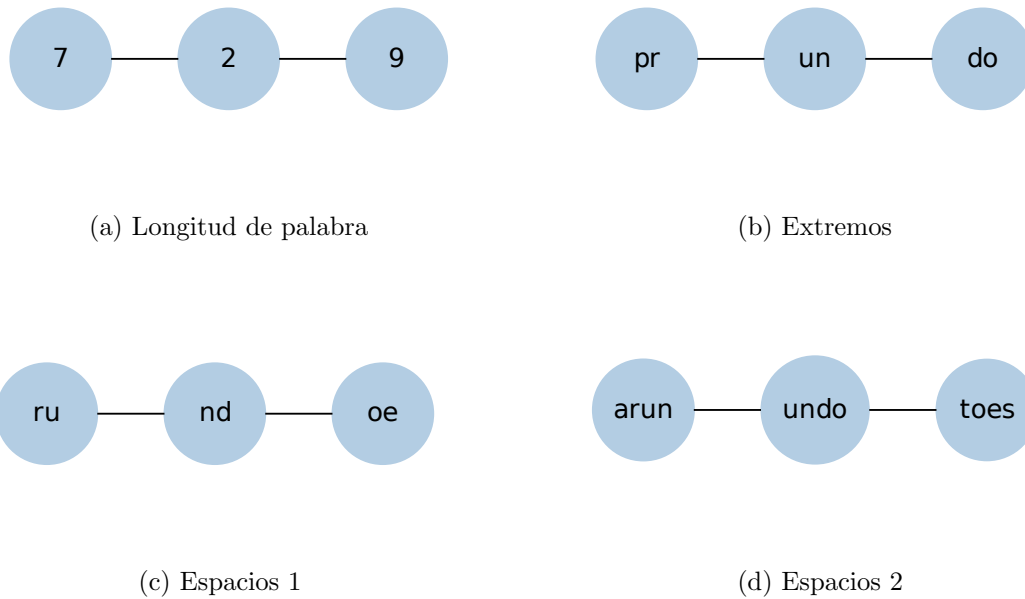


Figura 3.2: El primer trigramas de la frase “Plagiar un documento es” con las distintas estrategias de selección.

3.1.2. Funciones de distancia

Por otro lado, decidimos experimentar con otras funciones de distancia tomadas de la estadística, para ver si eran aplicables a detección de plagio. Utilizamos el coeficiente de correlación de Pearson, y el de Spearman. Éstos son utilizados para medir la correlación entre dos variables, en este caso la representación por medio de n-gramas

de los documentos a analizar.

El razonamiento detrás de esto es que si los vectores de los documentos se parecen, es probable que se trate del mismo documento, o tengan partes en común. Todos los documento escritos en un mismo idioma, tendrán ciertas similitudes. El desafío está en poder buscar características que sean comunes al texto y no al idioma, o al tema del documento, y por otro lado, en buscar una función de distancia que permita distinguirlos adecuadamente.

Pearson El coeficiente de correlación de Pearson está dado por la siguiente fórmula:

$$\frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}} \quad (3.1)$$

En este caso cada i representa un n -grama, y X_i la frecuencia de ese n -grama. Al tratarse de frecuencias, la media corresponde a $\frac{1}{n}$, con lo cual la fórmula quedaría:

$$\frac{\sum_{i=1}^n (X_i - \frac{1}{n})(Y_i - \frac{1}{n})}{\sqrt{\sum_{i=1}^n (X_i - \frac{1}{n})^2} \sqrt{\sum_{i=1}^n (Y_i - \frac{1}{n})^2}} \quad (3.2)$$

La fórmula es parecida a la de una medida utilizada ampliamente en detección de plagio: el coseno. La diferencia es que Pearson utiliza los vectores centrados con respecto a la media. Cuanto más grande sea n , más se parecerán ambas distancias.

El n al que se refiere la fórmula, es la cantidad de posibles n -gramas de la estrategia utilizada. Una posible forma de calcular este n , es contar la cantidad de caracteres del alfabeto y elevarlo a la n -ésima potencia. Esto en realidad sobrestima el número, ya que no todas las posibles combinaciones de caracteres son palabras de un lenguaje.

En el caso de tener n-gramas formados por caracteres, esta puede ser una forma razonable de calcularlos, ya que tienen una longitud fija. En el caso de tomar n-gramas de palabras, es más difícil computar el n de esta forma, ya que las palabras tienen longitud variable, con lo cual el cálculo sería:

$$\sum_{i=1}^l \#car^i \quad (3.3)$$

Donde $\#car$ es la cantidad de caracteres del alfabeto. Este número es mucho más grande que el anterior, y es nuevamente una sobrestimación. Para mejorarlo decidimos en el caso de las palabras utilizar una estimación de la cantidad de palabras del idioma inglés (el corpus que vamos a usar en los experimentos está en ese idioma). Una estimación de este tipo no se puede aplicar sobre los n-gramas de caracteres, ya que no hay datos al respecto: no hay cálculos disponibles sobre la cantidad de combinaciones posibles de n caracteres en una determinada lengua, ni tiene sentido hacerlos.

Por lo antes expuesto, para las estrategias de los extremos y los caracteres cercanos a espacios utilizamos todas las posibles combinaciones de caracteres (ya que estas estrategias tienen longitud fija). En el caso de las estrategias de longitud de palabra, estimamos el total contando todas las posibles combinaciones de longitudes (que limitamos a variar entre uno y veinte), aunque esto también es una sobrestimación (por ejemplo: encontrar tres palabras seguidas de longitud veinte en un texto es imposible).

Spearman El coeficiente de correlación de Spearman utiliza vectores de rangos. Para crear estos vectores, se ordenan las frecuencias y se les asignan números según este orden (al n-grama más frecuente se le asigna el número uno, al segundo el número dos, etc). En caso de haber componentes con la misma frecuencia, se utiliza el promedio de los

rangos de las mismas. Luego, en base a los vectores de rangos (que deben tener la misma longitud), se computa para cada n-grama la diferencia entre el rango en el primer vector y en el segundo. Es decir, el vector d se calcula con la siguiente fórmula:

$$d_i = r_{1i} - r_{2i} \quad (3.4)$$

Donde i representa los distintos n-gramas. Para que ambos vectores de rangos tengan la misma longitud, consideramos los n-gramas que aparecen en por lo menos uno de los dos vectores.

Finalmente, la fórmula de la función de distancia es la siguiente:

$$1 - \frac{6 \sum_{i=0}^{n-1} d_i^2}{n(n^2 - 1)} \quad (3.5)$$

Donde d_i es la posición i del vector de diferencias de rango, y n la cantidad de n-gramas considerados.

Funciones frecuentemente utilizadas Además, decidimos implementar otras funciones de distancia utilizadas en detección de plagio para poder comprarlas contra Spearman y Pearson. Las funciones elegidas fueron: Jaccard, Dice, Coseno y Overlap (ver sección 2.2.2.2)

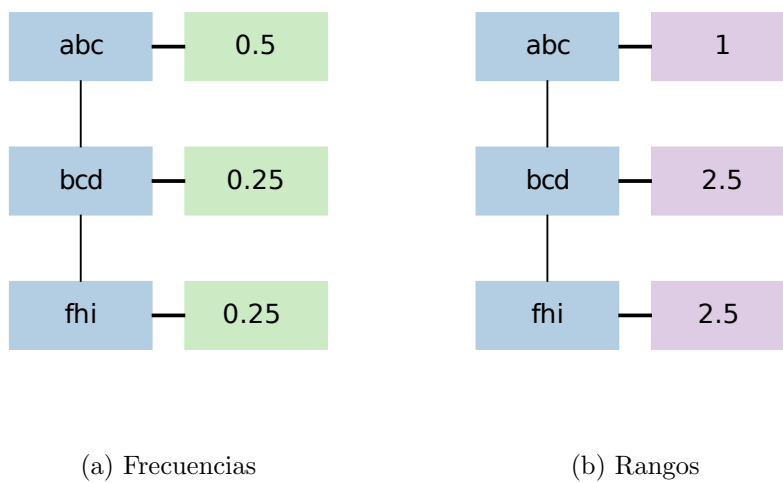


Figura 3.3: El vector con las frecuencias de los n-gramas, transformado a rangos.

3.2. Análisis detallado

Implementamos esta parte únicamente para que el software fuera completo, y pudiera responder al usuario exactamente qué pasajes fueron plagiados. En base a las investigaciones hechas en la primera parte, llegamos a la conclusión de que Dot Plot y Karp Rabin - Greedy String Tiling son los dos métodos que proporcionan mejores resultados, fundamentalmente porque soportan reordenamiento de las palabras. Implementamos este último por ser más simple de programar (Dot plot involucra algoritmos de procesamiento de imágenes digitales).

Para el desarrollo de esta parte nos basamos en el pseudocódigo que aparece en la publicación *Yap3: improved detection of similarities in computer programs and other texts*[37]. Como resultado de GST, obtenemos pasajes de texto que fueron plagiados. Luego, aplicamos una fase de post procesamiento bastante simple: si dos pasajes detectados estaban a menos de treinta caracteres en ambos textos, los unificamos y los

reportamos al usuario como uno solo.

El pre procesamiento aplicado en esta etapa tenía como condición no alterar los índices de los caracteres en los textos (es decir, no borrar ni agregar caracteres, solo reemplazar) para poder reportar los índices de los pasajes plagiados de forma sencilla. Por eso, lo único que hicimos fue pasar el texto a minúsculas y eliminar los símbolos diacríticos.

3.3. El programa

El programa funciona en tres etapas:

1. Creación de una base de datos (vacía) donde se almacenaran los vectores que representan los documentos. En este paso también se define el n que será utilizado (el tamaño de los n -gramas).
2. Agregado de un archivo a la base de datos. Se debe proporcionar la estrategia de selección de n -gramas a utilizar.
3. Comparación de un archivo contra los que están en la base de datos. Se debe proporcionar la estrategia de n -gramas y la función de distancia a utilizar.

El resultado final es un archivo en formato xml en el que se reportan la posición y longitud de los pasajes plagiados, junto con el nombre del archivo original de donde cada pasaje fue tomado. El formato es el mismo que fue utilizado en la competencia PAN del 2010 ².

El programa puede leer distintos formatos de archivo:

²<http://www.uni-weimar.de/medien/webis/research/events/pan-10/task1-plagiarism-detection.html>

- Texto plano
- Documentos de Microsoft Word
- Documentos PDF (Portable Document Format)

3.3.1. Tecnologías utilizadas

El lenguaje de programación utilizado fue C++, compilado con G++³, el compilador de GNU. Se utilizaron las librerías:

- GMP, GNU Multiple Precision Arithmetic Library ⁴ para las operaciones aritméticas. Esta librería soporta enteros grandes (el único límite al tamaño de los enteros es la memoria física disponible) y aritmética flotante con precisión arbitraria. Fue utilizada en los cálculos de las distancias entre vectores, ya que los tipos nativos del lenguaje no tenían la capacidad de manejar números tan grandes.
- Sqlite3⁵ como base de datos. La elegimos porque es fácil de utilizar, y es compatible con distintos Sistemas Operativos. De todas formas, el diseño del programa es extensible a otros tipos de base de datos, ya que si se decidiera utilizar el programa para una aplicación real con muchos documentos, probablemente habría que reemplazar la base por una más eficiente.
- Uni2ascii⁶, que convierte caracteres UTF-8 a caracteres Ascii. Se utilizó para eliminar los signos diacríticos (acentos, diéresis, etc) de los textos.

³<http://gcc.gnu.org/>

⁴<http://gmp1ib.org/>

⁵<http://www.sqlite.org/>

⁶<http://linux.die.net/man/1/uni2ascii>

Por otro lado probamos conversores a texto, para poder soportar distintos formatos. El software lee un archivo de configuración en donde se especifican los programas a utilizar y los comandos necesarios para invocarlos, dándole libertad al usuario para que elija los que él prefiera. De todas formas nosotros utilizamos Antiword⁷ para leer documentos de Word y Xpdf⁸ para PDF. Ambos pueden ser descargados gratuitamente de Internet y funcionan tanto en Windows como en Linux.

⁷<http://www.winfield.demon.nl/>

⁸<http://foolabs.com/xpdf/>

Capítulo 4

Experimentos

4.1. Análisis preliminar

4.1.1. Experimento 1

Una vez desarrolladas las estrategias de selección de n-gramas y las distintas funciones de distancia, nuestro objetivo era comparar las distintas combinaciones de estrategia/función de distancia/longitud del n-grama, y encontrar la mejor de ellas.

Corrimos el programa utilizando una selección de casos del corpus PAN (ver sección 2.3.3). No empleamos todo el corpus porque queríamos probar un gran número de combinaciones, lo cual hubiera llevado mucho tiempo. Por este motivo los valores de cobertura y precisión son seguramente mayores de lo que se obtendría utilizando un corpus más extenso. De todas formas el objetivo era comparar las distintas estrategias entre sí. La selección contenía cincuenta y cuatro documentos en la base de datos y trece textos sospechosos para ser comparados con la misma, por lo que el total de comparaciones para cada combinación es setecientos dos.

Para poder compararlas decidimos usar precisión y cobertura, que son medidas estándar (ver sección 2.2.2.1). Sin embargo, éstas cambian según el valor de corte a partir del cual se considera el documento plagio. Por eso decidimos graficar precisión y cobertura (a nivel documento) en función de este valor. Éstas son calculadas a partir de soluciones que provee el corpus PAN.

Primero hablaremos de las conclusiones generales que obtuvimos. Como sospechábamos, a medida que aumenta el umbral de corte, disminuye la cobertura y aumenta la precisión. De todas maneras, disminuye de formas muy distintas según la combinación elegida. La precisión no siempre decrece en forma monótona: a veces aumenta para luego disminuir y luego volver a aumentar (ver figura 4.1a). Nosotros suponemos que esto se debe a que hay algunos casos de plagio que obtienen un puntaje menor a otros que no lo son. Entonces, al incrementar el valor de corte, los casos que efectivamente eran plagio dejan de ser marcados como tales, disminuyendo la precisión. Luego, al subir nuevamente el valor, los casos que no eran plagio dejan de ser detectados, aumentando la precisión.

El objetivo es que tanto la precisión como la cobertura sean altas, ya que por un lado no detectar casos de plagio en esta etapa implica que no serán detectados en todo el proceso. Por otro lado, detectar casos de más es muy costoso en términos de tiempo, ya que el análisis detallado requiere mucho cómputo y las bases de datos para detectar plagio tienen un gran número de documentos; un pequeño porcentaje de error puede implicar muchas horas perdidas revisando documentos no relacionados y frustrar al usuario del sistema. Si la respuesta no es dada en un tiempo razonable, el software no es útil.

Lo que buscamos entonces es encontrar un valor de corte que maximice tanto la cobertura como la precisión, pero ésta es una situación de trade-off, ya que ganar precisión implica perder cobertura y viceversa. Un posible valor es el punto en el cual se cruzan

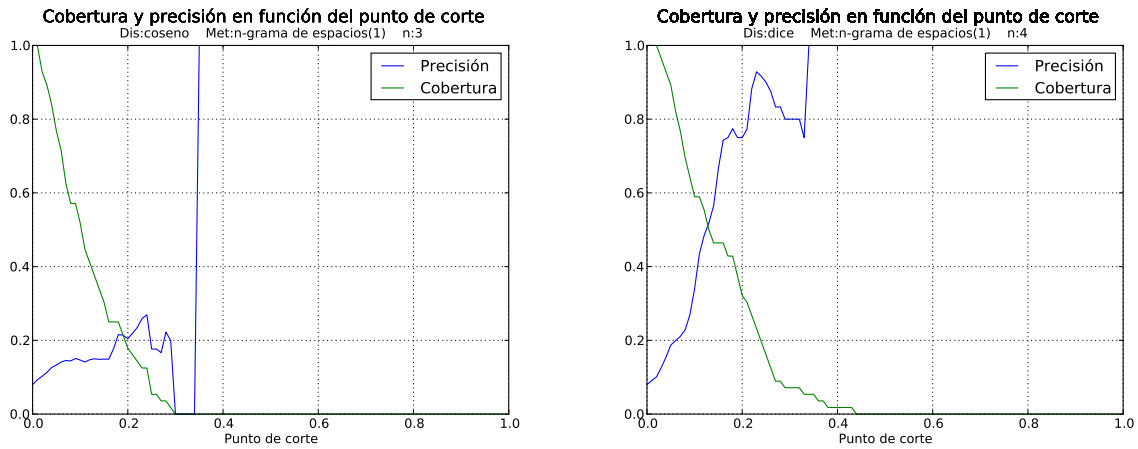


Figura 4.1: Dos ejemplos distintos de como varían precisión (en azul) y cobertura (en verde).

las dos funciones, ya que con ese valor cobertura y precisión son iguales. Otra posibilidad es combinar ambas medidas. Existe una medida en estadística que hace esto, conocida como F-measure:

$$F = 2 \cdot \frac{\text{precisión} \cdot \text{cobertura}}{\text{precisión} + \text{cobertura}} \quad (4.1)$$

Buscamos maximizar esta función para obtener la mejor combinación de las medidas.

Continuando con la consideraciones generales, observamos que la estrategia de selección junto con la longitud de n-grama apropiada para la misma tienen mucho más peso en el resultado que la función de distancia elegida. Por ejemplo: en los n-gramas de palabra, al utilizar coseno se obtienen mejores resultados con la longitud cinco. Si en cambio se utiliza Dice, el máximo se obtiene en siete. Por otro lado, si se utilizan n-gramas de longitud de palabra y coseno el mejor resultado se obtiene en nueve, pero con Dice

el máximo esa en trece. Esto se debe a que los n-gramas de longitud de palabra aportan mucha menos información que los otros: en vez de almacenar n palabras, se almacenan n números correspondientes a las longitudes de esas palabras; es probable encontrar n palabras que tengan las mismas longitudes que otras n palabras, pero no se trate de las mismas. Por eso es necesario considerar más palabras para que esto no suceda.

Consideraremos ahora las estrategias y funciones de distancia en particular. En nuestro análisis de la literatura comprobamos que la estrategia de selección de n-gramas más frecuentemente implementada es la de utilizar n-gramas de palabras, por lo que decidimos analizarla en detalle para luego comparar las otras estrategias con ésta. Analizamos distintas combinaciones de longitudes y distancias, los resultados pueden verse en la figura 4.2a.

La mejor combinación para esta estrategia es utilizar como función de distancia Dice, con longitud de n-grama siete. En general, las distancias que suelen ser más utilizadas son coseno y overlap, que tienen su máximo cuando la longitud es cinco. En varias publicaciones los autores sostienen que los mejores resultados se obtienen con longitudes entre dos y cinco. En su reporte de la competencia PAN[29], Clough utilizó la distancia overlap y obtuvo los mejores resultados utilizando cinco-gramas. El corpus utilizado para los documentos es el mismo que el nuestro, por lo que podría haber una relación entre los casos utilizados y con qué longitud se obtienen los mejores resultados.

La siguiente estrategia que analizamos fue la de longitud de palabras. Nuevamente obtuvimos los mejores resultados empleando la función de distancia Dice, pero esta vez con n igual a trece. En la publicación que encontramos al respecto[33], los autores usaron Jaccard y obtuvieron la mejor cobertura con doce-gramas. Considerando la misma distancia, nosotros también obtuvimos la mejor cobertura con la misma longitud y

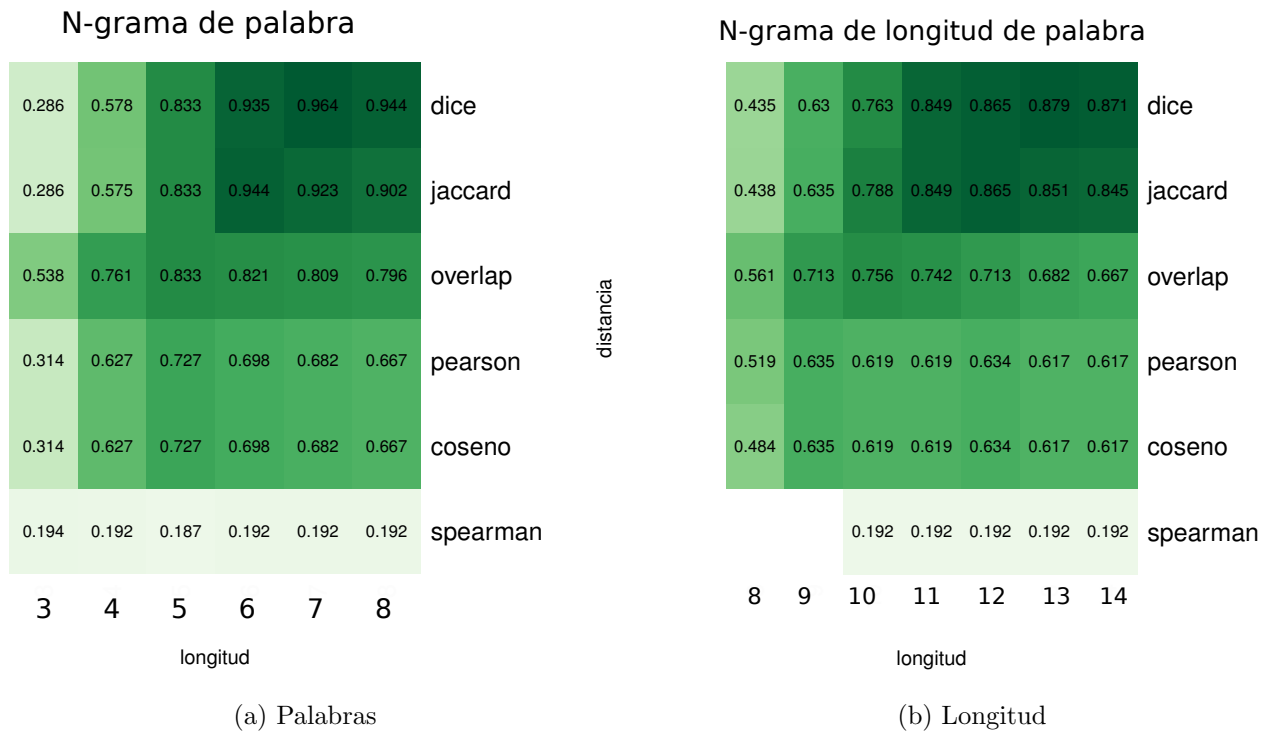
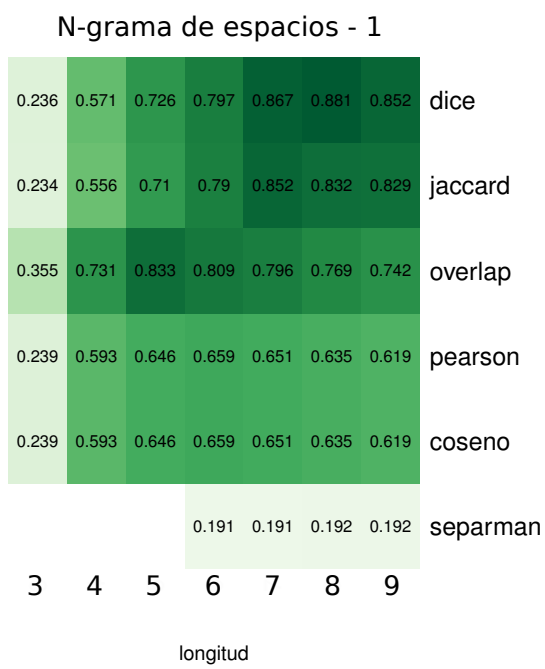


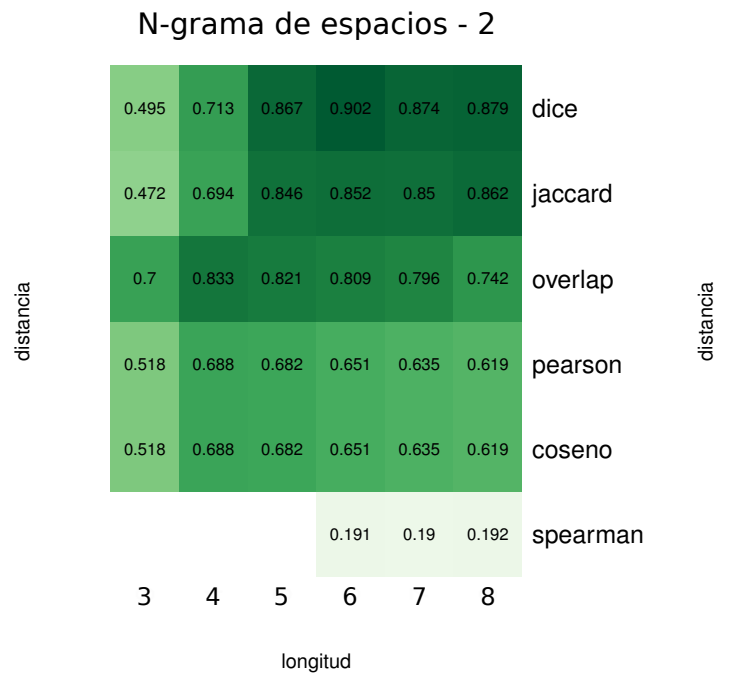
Figura 4.2: F-measure de los n-gramas de palabras con distintas longitudes y funciones de distancia.

además la mejor F-measure. Por último, notamos que los resultados obtenidos eran ligeramente inferiores a los de los n-gramas de palabras.

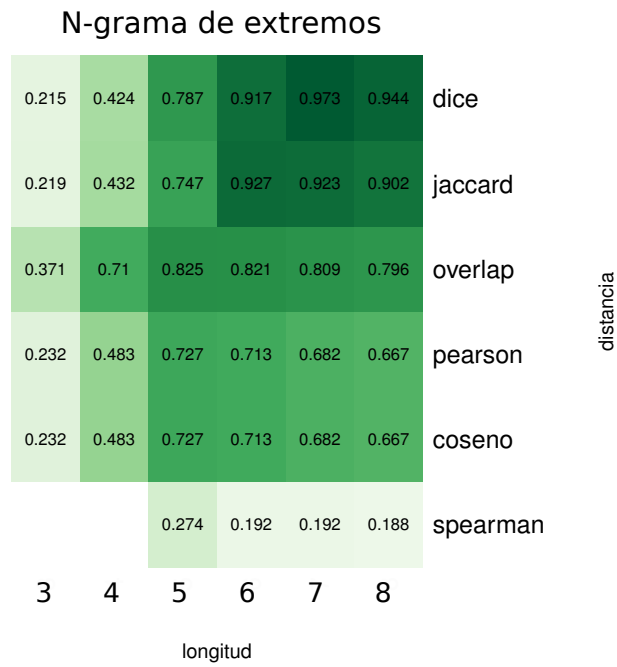
Luego analizamos las estrategias originales. Nuevamente, las funciones de distancia con las que obtuvimos mejores resultados -y bastante parecidos entre sí- son Dice y Jaccard. Con Overlap logramos resultados ligeramente inferiores, pero con la ventaja de que los mejores resultados se obtienen con longitudes menores (lo que implica menor esfuerzo en cómputo y almacenamiento).



(a) N-gramas de espacios 1



(b) N-gramas de espacios 2



(c) N-gramas de extremos

Figura 4.3: F-measure de las estrategias originales con distintas longitudes y funciones de distancia.

Por otro lado, con Coseno y Pearson obtuvimos resultados muy parecidos entre sí, pero inferiores a los alcanzados con las otras distancias. El coeficiente de correlación de Spearman no es una función de distancia apropiada para discriminar plagio, ni siquiera con valores altos de n .

Encontramos que los n -gramas de extremos también alcanzan el mejor valor de F-measure con longitud igual a siete, tal como pasaba con los n -gramas de palabras. Además, a partir de los cinco-gramas los valores de F-measure son muy similares (en varios casos idénticos), lo que indica que es muy poco probable que dada una secuencia s_1 de n palabras exista otra secuencia s_2 con los mismos extremos y haya al menos una palabra distinta s_1 y s_2 . Esto es muy importante, ya que indica que podemos reemplazar las palabras por sus extremos y obtener los mismos resultados, con menor costo de almacenamiento, un tema no menor en este tipo de sistemas, que deben contar con grandes bases de datos de documentos.

Los n -gramas de espacios no intentan representar palabras, a diferencia de los otros enfoques, sino la relación entre palabras consecutivas en un texto. De todas maneras, presentan valores de f-measure similares a los obtenidos con los n -gramas de palabra. En el caso de los n -gramas de espacio de un carácter, notamos que alcanzan su óptimo con longitudes mayores a las de los anteriores. En realidad esta versión de la estrategia se parece mucho a la de los extremos: salvo el primer y último carácter seleccionados, todo el resto son iguales.

La segunda versión de esta estrategia - que selecciona dos caracteres en vez de uno - alcanza mejores resultados con longitudes menores a las de las palabras. Creemos que esto se debe a que hay más información en cada n -grama por dos motivos: por un lado representa $n+1$ palabras, por otro al tomar dos caracteres de cada palabra, es más difícil

que al reemplazar una palabra por otra sin que esto sea detectado.

Luego de este experimento llegamos a la conclusión de que las medidas de distancia que propusimos no son tan efectivas como las utilizadas en otras publicaciones. Sin embargo, las estrategias de selección de n-gramas alcanzaron buenos resultados.

4.1.2. Experimento 2

Una vez que verificamos cuáles son las mejores combinaciones de estrategia/función de distancia/longitud del n-grama, decidimos seguir analizándolas, esta vez con una selección mayor de casos de test y midiendo tiempo y el tamaño ocupado por la base de datos de documentos. De las ciento ochenta y siete combinaciones del experimento 1, seleccionamos noventa y cuatro para esta segunda prueba.

La nueva selección de casos de test (también tomada del corpus PAN) contiene ciento veinticinco documentos originales y sesenta y tres sospechosos, es decir siete mil ochocientos setenta y cinco comparaciones (diez veces más que la selección anterior).

Las pruebas fueron ejecutadas en una PC de escritorio con un procesador AMD Opteron 165 de 2100MHz y 2GB de memoria RAM. Los resultados de las 15 mejores combinaciones pueden verse en la figura 4.4. Los tiempos incluyen el agregado de todos los archivos originales a la base de datos y las comparaciones de todos los textos sospechosos con la misma. De todas formas, la etapa de agregado a la base de datos no influye demasiado en los tiempos: se trata de aproximadamente uno o dos minutos, cuando las comparaciones pueden tardar hasta una hora y media, según la combinación utilizada.

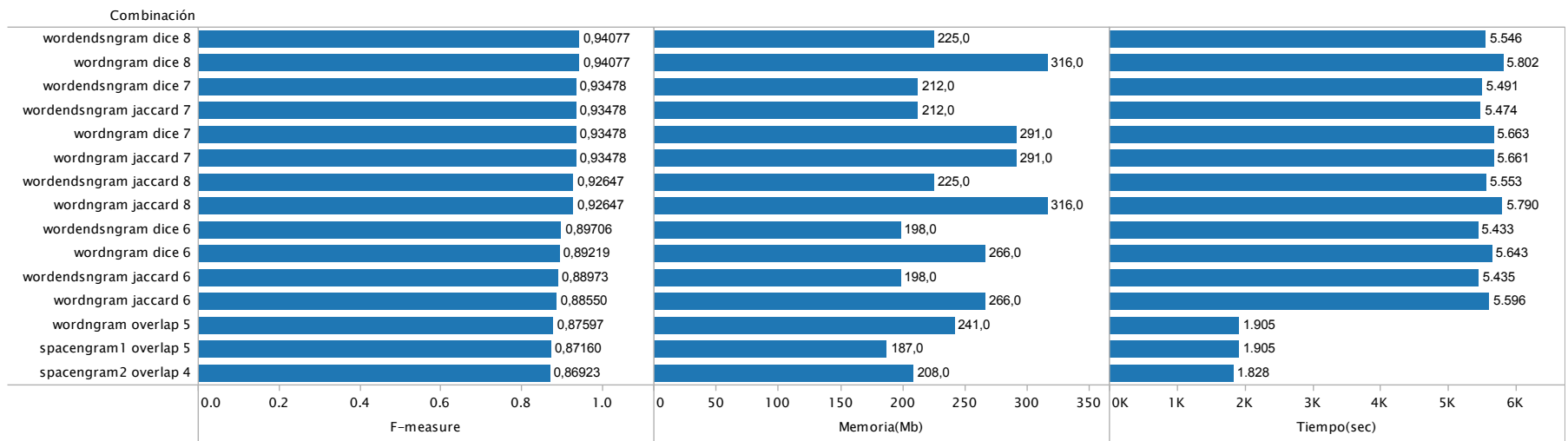


Figura 4.4: Las mejores quince combinaciones con sus respectivos valores de f-measure, memoria y tiempo de ejecución.

4.1.2.1. Funciones de distancia

Si bien hubo variaciones con respecto a los resultados obtenidos, éstas no fueron sustanciales. Notamos nuevamente que las funciones de distancia con las que se alcanza mejores valores son Dice y Jaccard (con resultados muy similares). La función de distancia overlap obtiene buenos resultados, pero es hasta un ocho por ciento menos efectiva. Las otras funciones ni siquiera aparecen dentro de las quince mejores combinaciones.

De los gráficos, podemos ver que utilizar overlap implica una drástica disminución de los tiempos de ejecución. Esto se debe a que esta medida solo tiene en cuenta la cantidad de n-gramas de cada vector, sin considerar las frecuencias (que implica recorrer todo el vector). Sin embargo, hay que tener en cuenta que esto puede traer como consecuencia una pérdida en la precisión, y que documentos que no contienen plagio sean analizados en detalle, algo muy costoso en términos de tiempo. El ahorro de tiempo es efectivo, solo si la precisión no disminuye demasiado.

En nuestro caso (ver tabla 4.1), las dos mejores combinaciones que utilizan overlap, tienen la mayor precisión, con lo que el ahorro de tiempo será considerable. Sin embargo, la cobertura es quince por ciento menor.

No hacemos consideraciones en cuanto a la memoria, ya que la función de distancia elegida no tiene impacto en ello. El tamaño ocupado por la base de datos está determinado por el tamaño del vector de n-gramas, producto a su vez de la estrategia y longitud de los n-gramas elegidos.

4.1.2.2. Estrategias

Las mejores estrategias de selección son las de n-gramas de palabras y n-gramas de extremos. Dado que se trata de longitudes altas (seis o más), los resultados obtenidos con ambas (tomando el mismo n y la misma función de distancia) son idénticos.

El tiempo de ejecución es prácticamente el mismo para las dos, con una muy leve ventaja de parte de los n-gramas de extremos. En cuanto a la memoria, la estrategia de extremos presenta un ahorro de memoria hasta treinta por ciento con respecto a la de palabras (la diferencia depende de la longitud de n-grama utilizado).

Las estrategias de espacios, que nosotros veíamos como una mejora de las de extremos, resultaron ser peores. Las de longitud de palabra, tienen una performance aún menor. La mejor de las combinaciones que involucra n-gramas de longitud es doce por ciento menos efectiva que la mejor combinación de n-gramas de palabras.

A continuación, un cuadro con las mejores combinaciones en términos de F-measure, con sus respectivos valores de precisión y cobertura.

Función de distancia	Estrategia	N	F-measure	Cobertura	Precisión
Dice	N-grama de extremos	8	0.940	0.944	0.937
Dice	N-grama de palabra	8	0.940	0.944	0.937
Dice	N-grama de extremos	7	0.934	0.902	0.969
Dice	N-grama de palabra	7	0.934	0.902	0.969
Jaccard	N-grama de extremos	7	0.934	0.902	0.969
Jaccard	N-grama de palabra	7	0.934	0.902	0.969
Jaccard	N-grama de extremos	8	0.926	0.881	0.976
Jaccard	N-grama de palabra	8	0.926	0.881	0.976
Dice	N-grama de extremos	6	0.897	0.853	0.945
Dice	N-grama de palabra	6	0.892	0.839	0.952
Jaccard	N-grama de extremos	6	0.889	0.818	0.975
Jaccard	N-grama de palabra	6	0.885	0.811	0.974
Overlap	N-grama de palabra	5	0.875	0.790	0.982
Overlap	N-grama de espacios-1	5	0.871	0.783	0.982
Overlap	N-grama de espacios-2	4	0.869	0.790	0.965
Overlap	N-grama de espacios-2	7	0.844	0.741	0.981
Overlap	N-grama de extremos	5	0.844	0.797	0.897
Overlap	N-grama de espacios-1	8	0.835	0.727	0.981

Dice	N-grama de longitud	14	0.829	0.867	0.794
Jaccard	N-grama de longitud	14	0.826	0.797	0.857
Overlap	N-grama de longitud	10	0.825	0.713	0.980
Jaccard	N-grama de longitud	13	0.808	0.839	0.779
Dice	N-grama de longitud	13	0.804	0.832	0.777

Cuadro 4.1: Tabla con las combinaciones con f.measure de 0.8 o más

4.2. Análisis detallado y postprocesamiento

También experimentamos con la fase de análisis detallado y de postprocesamiento, en la que utilizamos el algoritmo Greedy String Tiling. El objetivo era por un lado implementar el algoritmo para entenderlo más profundamente y por otro comprobar su eficiencia en la práctica.

Creíamos que teniendo el pseudocódigo (como mencionamos, lo tomamos del paper *Yap3: improved detection of similarities in computer programs and other texts*[37]) la programación sería trivial. Sin embargo, nos dimos cuenta de que había varias decisiones que tomar:

- Longitud mínima de un match: a partir de cuántos caracteres se considera que un fragmento está plagiado. Cuánto más grande sea este valor, más certeza se tendrá de que se trata de plagio pero si se toma un número demasiado alto, es probable que no se detecten ciertos fragmentos. Hay que tener en cuenta que muchas veces las personas que cometen plagio, copian frases textualmente, pero alterando algunas palabras en el medio, para dificultar la detección. Esto hace que los fragmentos idénticos sean bastante más cortos que todo el pasaje plagiado. Por eso decidimos fijar este valor en veinte caracteres.
- Longitud de búsqueda inicial: la longitud de los chunks a hashear en la primera iteración. Se buscarán coincidencias en los textos iguales o mayores a esta longitud. Este parámetro no afecta el resultado del algoritmo ya que en GST el valor de búsqueda crece o decrece según sea necesario, pero una mala elección puede aumentar considerablemente el tiempo de ejecución del algoritmo. Si es muy elevado, se harán varias búsquedas sin resultados, y se irá disminuyendo

paulatinamente. Si es muy pequeño, se lo deberá incrementar, terminando la búsqueda actual para comenzar una nueva. Decidimos utilizar ciento sesenta caracteres como longitud inicial.

- El primo a utilizar en Karp-Rabin: este valor se utiliza para la operación módulo. Si se toma un valor pequeño, la cantidad de hashings será pequeña y las colisiones habituales, degradando la performance del algoritmo. Por este motivo, elegimos utilizar números primos de veintisiete o más bits. De todas formas, elegir un número exageradamente grande no es conveniente ya que es más costoso en términos de cómputo y de uso de memoria.
- Igualdad de cadenas: nos preguntamos si debíamos tener en cuenta los espacios en blanco o las mayúsculas. Decidimos pasar todo el texto a minúsculas para evitar esta distinción y consideramos todos los caracteres blancos (espacios, salto de línea, tabulación) como equivalentes. Mantuvimos los signos de puntuación.

Por otro lado, en nuestro algoritmo de postprocesamiento, también tuvimos que tomar otra decisión: cuál es el número máximo de caracteres no idénticos que puede haber entre dos fragmentos plagiados para que éstos sean considerados uno solo. Decidimos fijar este número en treinta.

Verificamos estas decisiones comparando uno de los archivos sospechosos contra la base de datos, de seis fragmentos plagiados de distintos documentos y longitudes variables. No continuamos las pruebas con mayor cantidad de documentos por que este análisis es muy costoso y consume mucho tiempo (estas seis comparaciones insumieron casi dieciocho horas). Los resultados obtenidos fueron los siguientes (calculados con un script provisto por los organizadores de la competencia PAN):

Precisión	0.556952969607
Cobertura	0.428437447681
Granularidad	24.6

Cuadro 4.2: Resultados del análisis detallado del primer documento sospechoso

Para este documento en particular, los valores de cobertura y precisión son iguales a uno, así que no influyen en las medidas. Nosotros creemos que el bajo valor de precisión obtenido se debe a que el valor de longitud mínima era demasiado bajo. Por otro lado, el valor alto de granularidad (ver sección 2.4) se debe a que el valor elegido en la fase de postprocesamiento también es bajo. Aumentar este valor probablemente produzca un aumento en la cobertura.

Decidimos alterar estos dos valores y probar nuevamente. Mantuvimos sin alteraciones el valor de búsqueda inicial, ya que observamos que debía ser aumentado pocas veces (siempre disminuye hasta alcanzar la longitud mínima de tile). Fijamos la longitud mínima del match en cincuenta caracteres y la cantidad de caracteres distintos del algoritmo de postprocesamiento en doscientos. Los resultados obtenidos son los de la tabla 4.3.

La precisión aumentó considerablemente, como consecuencia de haber aumentado el valor de la longitud mínima del match. Sin embargo, esto también produjo una disminución notable de la cobertura, algo que nos sorprende, ya que sabemos que éstas compiten entre sí: es usual que al aumentar una, disminuya la otra. La granularidad bajó notablemente, como consecuencia del aumento de la longitud mínima de match y

Precisión	0.956406031602
Cobertura	0.284205389563
Granularidad	16.5

Cuadro 4.3: Resultados del análisis detallado del primer documento sospechoso con distintos parámetros

del aumento de los caracteres de tolerancia entre dos fragmentos. De todas formas sigue siendo alta.

Encontrar el punto de equilibrio de estos parámetros está de todas formas fuera del alcance de la Tesis. Lo que concluimos de este experimento es que éstos parámetros son muy importantes, ya que al variarlos obtenemos cambios muy grandes en los resultados.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Detección de plagio

En este trabajo hicimos un estudio del estado del arte en detección de plagio, estudiando las distintas líneas de investigación abiertas, los recursos disponibles y la forma de comparar distintas técnicas entre sí. Encontramos otras publicaciones, informes y tesis que también estudian el campo pero en general quedaron rápidamente desactualizadas (la detección de plagio tuvo un gran crecimiento en los últimos años).

Las líneas de investigación más recientes son las de detección en distintos idiomas, y estilometría, que permite detectar plagio mediante cambios de estilo en el documento, sin la necesidad de contar con el documento original. En los últimos años aparecieron varias publicaciones de estos temas, y el corpus de la competencia internacional PAN cuenta con ejemplos pensados para ser detectados mediante estas técnicas.

Además de búsqueda en una colección de referencia y detección intrínseca al documento, existe la alternativa de buscar el origen del plagio en Internet. Un software

ideal debería combinar estas tres técnicas, junto con la detección de plagio en distintos idiomas.

De todas formas, es difícil hacer una acusación de plagio mediante técnicas de análisis intrínseco, ya que no contamos con el documento original. Con este método sólo se prueba estrictamente que el texto fue escrito por dos personas distintas[62].

5.2. Detección mediante n-gramas

Los enfoques de n-gramas son uno de los enfoques más comunes en las publicaciones de plagio actualmente. Se utilizan distintas longitudes y distintas funciones de distancia. Siempre se menciona el trabajo de otros, pero muy pocas veces se comparan los métodos existentes con los propios.

La competencia internacional de detección de plagio PAN, es justamente un esfuerzo para dar un marco de comparación a los distintos enfoques. Nosotros quisimos seguir en esta línea, por lo que comparamos el método que desarrollamos con los utilizados en la mayoría de las publicaciones. Este experimento fue otro aporte: hicimos una comparación organizada de distintas estrategias de selección de n-gramas, distintas funciones de distancia y distintas longitudes, algo que no encontramos en otras publicaciones. Lamentablemente, al hacer tantas combinaciones distintas, tuvimos que restringir la selección de documentos utilizados para las pruebas. Nos hubiera gustado hacer estas comparaciones con todo el corpus o con una selección más grande, pero los tiempos de ejecución eran prohibitivos.

Nuestro aporte novedoso es una nueva estrategia de selección de n-gramas, que tiene la misma cobertura y precisión que la utilizada actualmente, pero con un ahorro de espacio

en disco de entre veinte y treinta por ciento (según la longitud de n-grama utilizada). Esto es importante ya que la fortaleza de los sistemas de detección con una colección de referencia residen en el tamaño (y pertinencia) de la base de datos de documentos a comparar. De todas maneras, este tipo de enfoque debería ser complementado con alguno de búsqueda de documentos en Internet y algún método de estilometría, para lograr detectar casos de plagio aunque la fuente original no se encuentre en la base de datos.

5.3. Perceptual Hashing

Si bien nuestro objetivo al iniciar la tesis era aplicar la idea de los perceptual hashings, el método que obtuvimos es un híbrido entre un perceptual hashing y un método de selección de n-gramas. Nosotros creemos que un posible perceptual hashing sería utilizar el concepto de sinónimos. Una frase en la que se cambió una palabra por un sinónimo es un ejemplo de dos representaciones distintas de lo mismo.

Sin embargo, incorporar la búsqueda de sinónimos al análisis preliminar es demasiado costoso. De hecho, todos los enfoques semánticos son utilizados para el análisis detallado.

Bibliografía

- [1] N. Shivakumar y H. Garcia-Molina. Finding near-replicas of documents on the web. En WEBDB: International Workshop on the World Wide Web and Databases, WebDB. LNCS, 1999.
- [2] G. Vargas Aignasse. Modificación al artículo 172 (estafas y otras defraudaciones) del código penal; sobre plagio.
- [3] <http://www.phash.org/>
- [4] <http://isis.poly.edu/projects/percehash>
- [5] B. Coskun y N. Memon. Confusion/Diffusion Capabilities of Some Robust Hash Functions. En: Information Sciences and Systems, 2006 40th Annual Conference.
- [6] V. Monga, A. Banerjee, B. Evans. A clustering based approach to perceptual image hashing. En: Information Forensics and Security, IEEE Transactions, Marzo 2006 Vol: 1 Issue:1, pp 68 -79.
- [7] C. E. Shannon. Communication theory of secrecy systems, Bell System Technical Journal, Vol. 28 (1949), pp. 656-715.

- [8] E. Merlo. Detection of Plagiarism in University Projects Using Metrics-based Spectral Similarity. En Proceedings of Dagstuhl Seminar 06301: Duplication, Redundancy, and Similarity in Software, 10pp.
- [9] B. Baker. On finding duplication and near-duplication in large software systems. En WCRE '95 Proceedings of the Second Working Conference on Reverse Engineering.
- [10] S. Eissen y B. Stein. Intrinsic Plagiarism Detection. En Proceedings of the 28th European Conference on IR Research, ECIR 2006 Londres, pp. 565-569, Springer 2006.
- [11] S. Eissen, B. Stein y M. Kulig. Plagiarism Detection without Reference Collections. En Selected Papers from the 30th Annual Conference of the German Classification Society (GfKl) Berlin, pp. 359-366, Springer 2007.
- [12] R. Coyotl-Morales, L. Villaseñor-Pineda, M. Montes-y-Gómez y P. Rosso. Authorship Attribution using Word Sequences. En Lecture Notes in Computer Science, 2006, Volume 4225/2006, pp. 844-853, Springer.
- [13] C. Grozea y M. Popescu. Who's the Thief? Automatic Detection of the Direction of Plagiarism, Lecture Notes in Computer Science, Vol. Volume 6008/2010, book "Computational Linguistics and Intelligent Text Processing", pp. 700-710, DOI 10.1007/978-3-642-12116-6, 2010.
- [14] M. Mohammadi, M. Analouei. Using Word Distance Based Measurement for Cross-lingual Plagiarism Detection. (IJCNS) International Journal of Computer and Network Security, 125 Vol. 2, No. 6, Junio 2010

- [15] M. Potthast, B. Stein, and M. Anderka. A Wikipedia-Based Multilingual Retrieval Model. *Advances in Information Retrieval, Lecture Notes in Computer Science*, 2008, Volume 4956/2008, 522-530, DOI: 10.1007/978-3-540-78646-7_51, Springer.
- [16] A. Barrón-Cedeño, P. Rosso, E. Agirre y G. Labaka. Plagiarism Detection across Distant Language Pairs. *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*. Beijing, China, 2010. Association for Computational Linguistics.
- [17] B. Pouliquen, R. Steinberger y C. Ignat. Automatic Identification of Document Translations in Large Multilingual Document Collections. En *RANLP 2003 – Proceedings of the International Conference on ‘Recent Advances in Natural Language Processing*.
- [18] M. Potthast , A. Barrón-Cedeño, A. Eiselt, B. Stein y P. Rosso. Overview of the 2nd International Competition on Plagiarism Detection. En Martin Braschler and Donna Harman, editors, *Notebook Papers of CLEF 2010 LABs and Workshops*. Padova, Italia, 22-23 Septiembre 2010.
- [19] J. Kasprzak, M. Brandejs, y M. Kripac. Finding Plagiarism by Evaluating Document Similarities. En *Proceedings of the SEPLN’09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse*. Vyd. Vol. 502. San Sebastian, Španělsko : CEUR Workshop Proceedings, 2009. pp. 24-28. 10.9.2010, San Sebastian, Španělsko.
- [20] B. Stein, S. Meyer zu Eissen y M. Potthast. Strategies for retrieving plagiarized documents. En *SIGIR ’07 Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*.

- [21] B. Stein y S. Meyer zu Eissen. Near Similarity Search and Plagiarism Analysis. En Spiliopoulou et al. (Eds.): From Data and Information Analysis to Knowledge Engineering, Selected Papers from the 29th Annual Conference of the German Classification Society (GfKI), Magdeburg, ISBN 1431-8814, pp. 430-437, Springer 2006.
- [22] M. Chong, L. Specia y R. Mitkov. Using Natural Language Processing for Automatic Detection of Plagiarism. En: Proceedings of the 4th International Plagiarism Conference (IPC-2010), Newcastle-upon-Tyne, UK
- [23] R. Řehůřek. Plagiarism Detection through Vector Space Models Applied to a Digital Library. En RASLAN 2008. Vyd. 1, Brno : Masarykova Univerzita, 2008, pp. 75-83. 2008, Karlova Studánka.
- [24] M. Zechner, M. Muhr, R. Kern y M. Granitzer. External and Intrinsic Plagiarism Detection Using Vector Space Models. 3rd PAN Workshop/1st PAN Competition.
- [25] N. Shivakumar, H. Garcia-Molina. SCAM: A Copy Detection Mechanism for Digital Documents. En: 2nd International Conference in Theory and Practice of Digital Libraries (DL 1995), June 11-13, 1995, Austin, Texas.
- [26] S. L. Devi, P. Rao, V. Sundar Ram and A. Akilandeswari. External Plagiarism Detection. Lab Report for PAN at CLEF 2010.
- [27] A. Barrón-Cedeño y P. Rosso. On Automatic Plagiarism Detection Based on n-Grams Comparison. En: Boughanem et al. (Eds.) ECIR 2009, LNCS 5478, pp. 696-700, Springer-Verlag Berlin Heidelberg (2009)

- [28] S. Schleimer D. S. Wilkerson y A. Aiken. Winnowing: Local Algorithms for Document Fingerprinting. En SIGMOD '03 Proceedings of the 2003 ACM SIGMOD international conference on Management of data.
- [29] R. M. A. Nawab, M. Stevenson y P. Clough. University of Sheffield, Lab Report for PAN at CLEF 2010. <http://www.uni-weimar.de/medien/webis/research/workshopseries/pan-10/downloads/plagiarism-nawab-report.pdf>
- [30] Grozea, C., Gehl, C., Popescu, M.: ENCOPLLOT: Pairwise Sequence Matching in Linear Time Applied to Plagiarism Detection. En: 3rd PAN WORKSHOP. UNCOVERING PLAGIARISM, AUTHORSHIP AND SOCIAL SOFTWARE MISUSE. p. 10
- [31] C. Lyon, R. Barrett y J. Malcolm. A theoretical basis to the automated detection of copying between texts, and its practical implementation in the Ferret plagiarism and collusion detector, en: Plagiarism: Prevention, Practice and Policies Conference, Junio 2004.
- [32] C. Basile, G. Cristadoro, D. Benedetto, E. Caglioti, and M. Degli Esposti. A plagiarism detection procedure in three steps: selection, matches and squares. En 3rd PAN WORKSHOP. UNCOVERING PLAGIARISM, AUTHORSHIP AND SOCIAL SOFTWARE MISUSE, pag 19.
- [33] A. Barrón-Cedeño, C. Basile, M. Degli Esposti y P. Rosso. Word Length n-Grams for Text Re-use Detection. En: Gelbukh A. (ed.) CICLing 2010, LNCS (6008), pp. 687-699, Springer-Verlag, 2010.

- [34] Kang, N., Gelbukh, A.: PPChecker: Plagiarism Pattern Checker in Document Copy Detection. En: Sojka, P., Kopeček, I., Pala, K. (eds.) TSD 2006. LNCS, vol. 4188, pp. 661–667. Springer, Heidelberg (2006)
- [35] B. Stein. Fuzzy-Fingerprints for Text-Based Information Retrieval. En: Proceedings of the I-KNOW '05, Graz 5th International Conference on Knowledge Management Journal of Universal Computer Science, pp. 572-579.
- [36] A. Zaslavsky, A. Bia y K. Monostori. Using Copy-Detection and Text Comparison Algorithms for Cross-Referencing Multiple Editions of Literary Works. En: Research and Advanced Technology for Digital Libraries Lecture Notes in Computer Science, 2001, Volume 2163/2001, pp. 103-114.
- [37] M. Wise. Yap3: improved detection of similarities in computer programs and other texts. En SIGCSE'96, Philadelphia, U.S.A., Feb. 15–17, 1996, pp 130-134.
- [38] M. Wise. Running Karp–Rabin Matching and Greedy String Tiling. Basser Department of Computer Science Technical Report 463 (March 1993) Presentado en Software – Practice and Experience.
- [39] Clough, P. (2003), Old and new challenges in automatic plagiarism detection, National UK Plagiarism Advisory Service (Online).
- [40] J. Helfman. Dotplot patterns: a literal look at pattern languages. En: Journal Theory and Practice of Object Systems - Special issue on patterns archive Volume 2 Issue 1, 1996.

- [41] P. Clough. Dotplot: visually exploring patterns in text. En: <http://www.dcs.shef.ac.uk/nlp/meter/Workshop/Dotplot/Public/index.htm>
- [42] A. Granville. Detecting Plagiarism in Java Code. Undergraduate project Dissertation, Department of Computer Science, University of Sheffield, 2002.
- [43] T Grotton. External Plagiarism Detection Based on Standard IR Technology and Fast Recognition of Common Subsequences. Lab Report for PAN at CLEF 2010.
- [44] C. Basile. Entropy and semantics: textual information extraction through statistical methods. Tesi dell'Università di Bologna, Marzo 2010.
- [45] G. Tsatsaronis, I. Varlamis, A. Giannakopoulos, and N. Kanellopoulos. Identifying Free Text Plagiarism Based on Semantic Similarity. En: 4th International Conference on Plagiarism Detection, July 2010, Newcastle, UK.
- [46] S. Alzahrani, y N. Salim. Fuzzy Semantic-Based String Similarity for Extrinsic Plagiarism Detection. En: CLEF Notebook Papers/LABs/Workshops, (2010). www.clef2010.org/resources/proceedings/clef2010labs_submission_21.pdf
- [47] G. Miller. WordNet: A Lexical Database for English. En: Magazine Communications of the ACM CACM Homepage archive Volume 38 Issue 11, Nov. 1995.
- [48] J.Lewis, S. Ossowski, J. Hicks, M. Errami y H. Garner. Text similarity: an alternative way to search MEDLINE. En: Journal Bioinformatics. Volume 22 Issue 18, September 2006 Oxford University Press Oxford, UK.

- [49] A. Broder. On the resemblance and containment of documents. En: Proceeding SEQUENCES '97 Proceedings of the Compression and Complexity of Sequences 1997 IEEE Computer Society Washington, DC, USA.
- [50] A. Barrón-Cedeño, A. Eiselt y P. Rosso. Monolingual Text Similarity Measures: A Comparison of Models over Wikipedia Articles Revisions. En: Proceedings of ICON-2009: 7th International Conference on Natural Language Processing, pp. 29-38, Hyderabad, India, 2009. Macmillan Publishers.
- [51] A. Muftah y A. Jabr (2009) Document plagiarism detection algorithm using semantic networks. Masters thesis, Universiti Teknologi Malaysia, Faculty of Computer Science and Information Systems. <http://eprints.utm.my/11433/1/AhmedJabrAhmedMFSKSM2009.pdf>
- [52] F. Bravo-Marquez, G. L'Huillier, S.Ríos y J. Velásquez. Hypergeometric Language Model and Zipf-Like Scoring Function for Web Document Similarity Retrieval. En: Proceeding SPIRE'10 Proceedings of the 17th international conference on String processing and information retrieval Springer-Verlag Berlin, Heidelberg 2010. pp.303 308.
- [53] J. Singh y M. Kumar. A Meta Search Approach to Find Similarity between Web Pages Using Different Similarity. En: Communications in Computer and Information Science, 2011, Volume 125, Part 1, 150-160.
- [54] F. Bravo-Marquez, G. L'Huillier , S. Ríos, J. Velásquez y L. Guerrero. DOCODE-lite: a meta-search engine for document similarity retrieval. En: KES'10 Proceedings

of the 14th international conference on Knowledge-based and intelligent information and engineering systems: Part II Springer-Verlag Berlin, Heidelberg 2010.

- [55] A. Pereira y N. Ziviani. Retrieving similar documents from the web. *Journal of Web Engineering* vol 2 n. 4, pp. 247–261 (2004)
- [56] M. Errami, Z. Sun, T. Long, A. George y H. Garner. Deja vu: a database of highly similar citations in the scientific literature. En *Nucleic Acids Research*, Vol. 37, No. suppl 1. (1 January 2009), pp. D921-D924.
- [57] P. Clough, R. Gaizauskas y S. Piao. Building and annotating a corpus for the study of journalistic text reuse. En *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-02)*, volume V, pages 1678–1691, Las Palmas de Gran Canaria, Spain, 2002.
- [58] M. Potthast, B. Stein, P. Rosso y A. Barrón-Cedeño. An Evaluation Framework for Plagiarism Detection. En: *Proceedings of the 23rd International Conference on Computational Linguistics, COLING 2010 Beijing, China*.
- [59] P. Clough y M. Stevenson. Creating a Corpus of Plagiarised Academic Texts. En *Proceedings of Corpus Linguistics Conference, CL'09, 2009*.
- [60] M. Potthast, B. Stein, A. Eiselt, A. Barrón-Cedeño, y P. Rosso. Overview of the 1st International Competition on Plagiarism Detection. En B. Stein, P. Rosso, E. Stamatatos, M. Koppel, y E. Agirre, editors, *SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09)*, pp 1–9. CEUR-WS.org, September 2009. <http://ceur-ws.org/Vol-502>

- [61] T. Hoad y J. Zobel. 2003. Methods for Identifying Versioned and Plagiarized Documents. En: Journal of the American Society for Information Science and Technology, 54(3):203–215
- [62] B. Stein, S. zu Eissen. Intrinsic Plagiarism Analysis with Meta Learning. Stein, Koppel, and Stamatatos (editors) SIGIR Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection (PAN 07), pp. 45-50