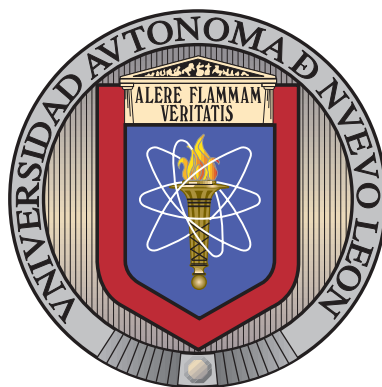


UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE LICENCIATURA



RECONOCIMIENTO DE ENTIDADES PARA
FILTRADO DE DUPLICADOS

POR

JORGE ALBERTO CORDERO CRUZ

EN OPCIÓN AL GRADO DE

INGENIERO EN MECATRÓNICA

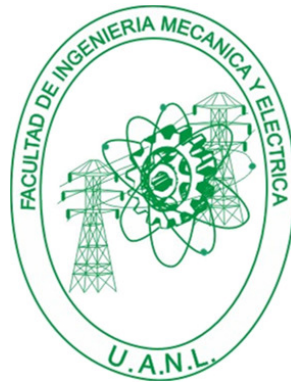
SAN NICOLÁS DE LOS GARZA, NUEVO LEÓN

MAYO 2014

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE LICENCIATURA



RECONOCIMIENTO DE ENTIDADES PARA
FILTRADO DE DUPLICADOS

POR

JORGE ALBERTO CORDERO CRUZ

EN OPCIÓN AL GRADO DE

INGENIERO EN MECATRÓNICA

SAN NICOLÁS DE LOS GARZA, NUEVO LEÓN

MAYO 2014

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica
División de Estudios de Licenciatura

Los miembros del Comité de Tesis recomendamos que la Tesis «Reconocimiento de Entidades para Filtrado de Duplicados», realizada por el alumno Jorge Alberto Cordero Cruz, con número de matrícula 1501046, sea aceptada para su defensa como opción al grado de Ingeniero en Mecatrónica.

El Comité de Tesis

Dra. Sara Elena Garza Villarreal
Asesor

Dra. Satu Elisa Schaeffer
Coasesor

M.C. Carlos Bernardo Garza Treviño
Revisor

Vo. Bo.

M.C. Arnulfo Treviño Cubero
División de Estudios de Licenciatura

San Nicolás de los Garza, Nuevo León, Mayo 2014

AGRADECIMIENTOS

Gracias a la Universidad Autónoma de Nuevo León y a la Facultad de Ingeniería Mecánica y Eléctrica por el apoyo prestado durante el transcurso de la carrera.

A mis padres Catalino Cordero Nolasco y Floreida Cruz Morales que nunca perdieron la fe en mí y me apoyaron desde el inicio hasta el final de la carrera.

A la Dra. Satu Elisa Schaeffer por dejarme ser parte de sus becarios, prestarme sus libros, dejarme entrar a sus clases como oyente y ser la patrocinadora oficial de los recursos para las competencias de robótica a las que asistí durante la carrera.

A la Dra. Sara Elena Garza Villarreal por apoyarme, darme consejos y tenerme paciencia durante el desarrollo de esta tesis.

Al profesor Carlos Bernardo Garza Treviño por las observaciones hechas al contenido de la tesis.

A la Dra. Sara Verónica Rodríguez Sánchez y al Dr. Hugo Jair Escalante Balderas por permitirme participar en un proyecto de investigación.

A mis hermanos, amigos y compañeros que fueron muy pacientes conmigo estos últimos cinco años.

A PROMEP por la beca recibida para el proyecto 103.5/12/7884.

RESUMEN

Jorge Alberto Cordero Cruz.

Candidato para el grado de Ingeniero en Mecatrónica.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio:

RECONOCIMIENTO DE ENTIDADES PARA FILTRADO DE DUPLICADOS

Número de páginas: 87.

OBJETIVOS Y MÉTODO DE ESTUDIO: El objetivo de este trabajo de tesis es crear un sistema capaz de detectar de manera automática documentos duplicados de un repositorio. Para lo cual se aplican técnicas de reconocimiento de entidades de texto al contenido de los documentos con la finalidad de extraer la información referente a tiempos, lugares, nombres de personas, nombres de organizaciones y sucesos descritos. A partir de la información extraída para cada documento, se crean representaciones estructuradas (*metamodelos*) de los mismos. Finalmente se aplican algoritmos de aprendizaje máquina a las representaciones estructuradas para realizar la detección de los documentos duplicados.

Se trabajó con reportes ciudadanos del *Centro de Integración Ciudadana* (CIC). Se definió un sistema de detección de documentos duplicados general y se creó la

implementación correspondiente para los reportes ciudadanos. En el sistema se implementaron tres algoritmos de cómputo inteligente: un algoritmo para realizar el reconocimiento de entidades de texto, un algoritmo de agrupamiento de documentos y un algoritmo de clasificación de documentos duplicados. Para las pruebas del sistema se crearon conjuntos artificiales de documentos con los que se probó el desempeño durante la detección de duplicados.

CONTRIBUCIONES Y CONCLUSIONES: La principal contribución de esta tesis es el diseño e implementación de un sistema de detección de documentos duplicados. Del sistema de detección de duplicados cabe destacar:

- El uso del metamodelo que permite estructurar el contenido de un documento para poder aplicar técnicas de cómputo inteligente a la información del documento.
- El establecimiento de criterios de detección de duplicados entre dos documentos. Estos criterios toman en cuenta la similitud que existe en los contenidos de descripción de sucesos, los contenidos de lugar y los contenidos de tiempo de un par de documentos.
- El aprendizaje automático de los criterios de detección de duplicados, debido a que estos criterios varían de acuerdo a la información contenida en los documentos.

El enfoque utilizado para la detección de duplicados aprovechó las características de la información contenida en los reportes ciudadanos. Se obtuvieron resultados satisfactorios durante las pruebas a la implementación del sistema de detección de duplicados.

Firma del asesor: _____

Dra. Sara Elena Garza Villarreal

ÍNDICE GENERAL

Resumen	v
1. Introducción	1
1.1. Definición del problema	2
1.2. Motivación	3
1.3. Hipótesis	4
1.4. Objetivos	5
1.5. Caso de estudio	5
1.6. Estructura de la tesis	7
2. Marco Teórico	9
2.1. Limpieza de texto	9
2.2. Procesamiento de texto	11
2.3. Reconocimiento de entidades en un texto	12
2.3.1. N-gramas	13
2.3.2. Etiquetado mediante el algoritmo Viterbi	13
2.3.3. Cálculo de los parámetros del algoritmo Viterbi	16

2.4. Representación estructurada de texto	17
2.5. Representación de documentos como vectores	18
2.6. Funciones de distancia	21
2.7. Agrupamiento de documentos de texto	22
2.8. Clasificación supervisada	24
2.9. Desempeño de un clasificador supervisado	26
3. Trabajos Relacionados	29
4. Metodología	36
4.1. Preprocesamiento de documentos	37
4.1.1. Extracción de la información	37
4.1.2. Limpieza de la información	38
4.1.3. Etiquetado de la información	39
4.1.4. Creación del metamodelo	40
4.2. Agrupamiento de metamodelos	42
4.3. Entrenamiento de clasificadores	43
4.4. Detección de metamodelos duplicados	44
4.5. Conclusión	44
5. Caso de Estudio	46
5.1. Reportes ciudadanos del Centro de Integración Ciudadana	46
5.2. Preprocesamiento de reportes ciudadanos	49

5.2.1. Extracción de la información	49
5.2.2. Limpieza de la información	49
5.2.3. Etiquetado de la información	52
5.2.4. Creación del metamodelo	55
5.3. Agrupamiento de metamodelos	57
5.4. Entrenamiento de clasificadores	62
5.5. Detección de duplicados	64
5.6. Conclusión	65
6. Experimentos y Resultados	67
6.1. Experimentos con etiquetador	67
6.1.1. Configuración experimental	67
6.1.2. Resultados	69
6.1.3. Discusión	69
6.2. Detección de duplicados	70
6.2.1. Configuración experimental	71
6.2.2. Resultados	75
6.2.3. Discusión	77
6.3. Conclusión	78
7. Conclusiones	79
7.1. Comentarios finales	80
7.2. Contribuciones	80

7.3. Trabajo a futuro	81
---------------------------------	----

ÍNDICE DE FIGURAS

2.1. Algunas de las palabras definidas en el proyecto Snowball para el idioma español.	10
2.2. Resultado de aplicar el proceso de limpieza a un texto sucio.	10
2.3. Cadena de texto.	12
2.4. Cadena de texto etiquetada.	13
2.5. Parte de los parámetros del algoritmo Viterbi obtenidos a partir de un conjunto de datos de entrenamiento.	17
2.6. Metamodelos generados en formato XML y JSON.	18
2.7. Agrupamiento de documentos de texto en tres categorías distintas. . .	23
2.8. Código de entrenamiento de una SVM.	26
2.9. Método de creación de conjuntos de entrenamiento y prueba, basado en el método de validación cruzada K iteraciones.	27
4.1. Sistema propuesto de detección de documentos duplicados.	36
4.2. Contenido de documentos en dos diferentes formatos.	38
4.3. Texto etiquetado utilizando las etiquetas SUCESO , LUGAR , HORA y 0 . . .	40
4.4. Estructura de metamodelos en formato XML y JSON.	40

4.5. Metamodelos generados a partir un texto etiquetado con las etiquetas SUCESO, LUGAR, HORA y O.	41
4.6. Generación de un metamodelo a partir de documentos en diferentes formatos.	42
4.7. Diagrama de un clasificador utilizado para detectar pares de docu- mentos duplicados.	44
5.1. Parte de un reporte ciudadano en formato JSON.	47
5.2. Lista de las abreviaturas que aparecen con mayor frecuencia en los reportes ciudadanos.	51
5.3. Lista de los símbolos vacíos que son eliminados de las palabras en un texto.	52
5.4. Texto limpiado antes de ser etiquetado.	53
5.5. Texto etiquetado después de haber sido limpiado.	54
5.6. Metamodelo generado a partir de un texto etiquetado.	55
5.7. Lista de palabras vacías utilizada para limpiar el contenido de los metamodelos.	57
6.1. Palabras obtenidas de los reportes del CIC descargados para el entre- namiento del etiquetador.	68
6.2. Parte del contenido del archivo de parámetros de etiquetado.	69
6.3. Metamodelo original.	72
6.4. Metamodelo similar al original.	73
6.5. Metamodelo distinto al original	74

-
- 6.6. Porcentajes de precisión obtenidos durante las pruebas de desempeño de los sistemas de detección de duplicados híbrido y supervisado. . . . 76
- 6.7. Porcentajes de exhaustividad obtenidos durante las pruebas de desempeño de los sistemas de detección de duplicados híbrido y supervisado. 76
- 6.8. Porcentajes de valor-F obtenidos durante las pruebas de desempeño de los sistemas de detección de duplicados híbrido y supervisado. . . . 77

ÍNDICE DE CUADROS

2.1. N -gramas obtenidos a partir del texto “semaforo descompuesto en la avenida”	13
2.2. Vectores de frecuencias de términos obtenidos al contar el número de veces que un término del vocabulario de términos aparece en los documentos: $d_1 =$ “UANL festeja 80 aniversario”, $d_2 =$ “la UANL premia la excelencia” y $d_3 =$ “ITESM celebra 70 aniversario”.	20
2.3. Vectores ponderados obtenidos después de aplicar el método <i>tfidf</i> a los vectores de frecuencias de los documentos $d_1 =$ “UANL festeja 80 aniversario”, $d_2 =$ “la UANL premia la excelencia” y $d_3 =$ “ITESM celebra 70 aniversario”.	21
2.4. Descripción de los conjuntos verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.	28
5.1. Diferentes grupos y las categorías de cada grupo a las que puede pertenecer un reporte ciudadano.	48
5.2. Ejemplo de una matriz de confusión generada durante el proceso de agrupamiento de los metamodelos utilizados para la creación del vocabulario de términos; el algoritmo de agrupamiento utilizado es k -medias. Los porcentajes de las categorías dominantes para cada grupo se muestran en negritas.	59

6.1. Matriz de confusión generada durante el proceso de agrupamiento y obtención de centroides del método híbrido.	76
---	----

CAPÍTULO 1

INTRODUCCIÓN

En la actualidad, es posible crear y compartir contenido de texto con gran facilidad, lo cual genera grandes repositorios donde la información puede ser consultada y analizada. Esta información permite a los usuarios conocer acontecimientos, tales como noticias en tiempo real, reportes de daños ocasionados por un desastre natural a una comunidad o descubrimientos científicos (entre muchos otros).

Es común que el contenido de estos repositorios cambie constantemente, ya sea por la actualización, el borrado o la agregación de nuevos documentos. Debido a esto, se vuelve más difícil el proceso de monitorear si se están generando documentos duplicados— es decir, documentos con contenido redundante.

La presencia de documentos duplicados en un repositorio presenta los siguientes problemas:

- Los usuarios que consultan un repositorio buscan obtener documentos distintos, por lo tanto requieren invertir tiempo para descartar duplicados.
- Los administradores de un repositorio buscan minimizar la cantidad de espacio necesaria para guardar los documentos; los duplicados hacen que este espacio se incremente lo cual produce un costo no deseable.
- Los duplicados introducen ruido cuando se quiere realizar análisis a los documentos y por lo tanto se pueden obtener resultados erróneos.
- El procesamiento de documentos se tarda más tiempo cuando se tienen dupli-

cados.

- Cuando los documentos corresponden a reportes o denuncias, el proceso de atención y respuesta se vuelve más tardado porque se tienen que descartar los duplicados.

En las situaciones anteriormente descritas se observa que la existencia de documentos duplicados en un repositorio representa una desventaja para los usuarios del mismo. Una manera automática para detectar documentos duplicados es de gran ayuda cuando se quiere mantener un repositorio libre de información redundante.

1.1 DEFINICIÓN DEL PROBLEMA

Los documentos duplicados en ciertos contextos generan redundancia o ruido y por lo mismo su posterior eliminación es deseable o necesaria. Existen ciertos tipos de documentos los cuales presentan campos de información referentes a: autores, fecha de elaboración, descripción de un suceso o un tema y lugares. Para este tipo de documentos se pueden establecer criterios de detección de duplicados de acuerdo al nivel de similitud que presenten los campos de los documentos comparados. Por ejemplo, se puede establecer un criterio que indique que dos documentos son duplicados si presentan las mismas fechas de elaboración, descripción de un tema y lugar en sus contenidos, independientemente de la información referente a los autores.

Los criterios de detección de duplicados pueden variar de acuerdo a la información contenida en los documentos. Por ejemplo, para detectar recetas de cocina duplicadas los campos de información referentes a lugar, fecha y autor son irrelevantes mientras que para detectar artículos científicos duplicados estos campos deben ser tomados en cuenta.

La detección de duplicados se puede realizar de manera manual, siendo una o varias personas quienes se encargan de buscar los duplicados existentes para un documento en un repositorio. A pesar de que una persona puede ser capaz de iden-

tificar documentos duplicados de manera sencilla, es propenso a errores; en cuanto la cantidad de documentos recibidos se vuelva considerable, una o varias personas no serán capaces de analizar tal cantidad de información. Un sistema automático de detección de duplicados puede ser capaz de analizar miles de documentos por día y se puede diseñar para obtener un buen desempeño al detectar duplicados.

1.2 MOTIVACIÓN

Este trabajo se realiza por la necesidad de tener un sistema de detección de documentos duplicados. Al tener un sistema de este tipo los documentos considerados como duplicados pueden ser mostrados a una persona que se encarga de corroborar que los documentos sean duplicados. Los documentos clasificados como duplicados sirven para retroalimentar al sistema con nueva información. De esta manera se tiene un sistema que se adapta a los cambios en la información de los documentos conforme pasa el tiempo y a una persona que sólo tiene que verificar que el sistema trabaje correctamente.

El problema de la detección de documentos duplicados ha sido abordado en otros trabajos con resultados satisfactorios; de los trabajos realizados podemos destacar:

- Un sistema de detección de duplicados para mensajes en Twitter realizado por Tao et al. [37].
- Un sistema de creación de noticias a partir del contenido de mensajes de Twitter realizado por Sankaranarayanan et al. [35].
- Un sistema de reconocimiento de noticias en mensajes de Twitter realizado por Agarwal et al. [11].
- Un sistema de detección de duplicados para páginas web realizado por Broder [14].

En este trabajo de tesis se propone un sistema de detección de duplicados que sea capaz de aprender los criterios de clasificación de duplicados de acuerdo al contenido de los documentos. Para esto se utilizarán técnicas de detección de entidades en el contenido de un texto, lo cual se realiza mediante la asignación de etiquetas a las palabras del texto para identificar el tipo de información presente. Mediante el etiquetado se identifican las palabras que describen sucesos, indican lugares, indican tiempos, indican nombres de personas e indican nombres de organizaciones.

Este presente trabajo busca crear una representación estructurada de la información relevante de un documento identificada mediante el etiquetado. Finalmente, aplicando técnicas de cómputo inteligente a las representaciones creadas se realizará la detección de documentos duplicados.

1.3 HIPÓTESIS

Este trabajo se realiza bajo la premisa de que la información contenida en documentos se puede estructurar de manera tal que se puedan aplicar técnicas de cómputo inteligente para realizar la detección de duplicados. Las técnicas de cómputo inteligente que se utilizan son las siguientes: minería de datos, aprendizaje máquina (ML por sus siglas en inglés “Machine Learning”) y procesamiento de lenguaje natural (NLP por sus siglas en inglés “Natural Language Processing”).

Las hipótesis formuladas en esta tesis son las siguientes:

- Utilizando reconocimiento de entidades de texto se puede obtener la información más relevante del contenido de un documento.
- Se puede representar el contenido de un documento mediante una representación estructurada que contiene la información más relevante obtenida del documento.
- Una representación estructurada permite utilizar técnicas de minería de datos y aprendizaje máquina al contenido de un documento.

- Las técnicas de minería de datos y aprendizaje máquina pueden ser utilizadas para detectar documentos duplicados en un repositorio.

1.4 OBJETIVOS

En esta tesis se plantea un objetivo general y tres objetivos específicos, los cuales son descritos a continuación.

OBJETIVO GENERAL Detectar de manera automática documentos duplicados en un repositorio de documentos.

OBJETIVOS ESPECÍFICOS

1. Extraer de un documento la información que hace referencia a tiempos, lugares, nombres de personas, nombres de organizaciones y la información que resume el contenido del documento.
2. Para cada documento crear una representación estructurada utilizando la información extraída.
3. Aplicar algoritmos de aprendizaje máquina a las representaciones estructuradas para detectar documentos duplicados.

1.5 CASO DE ESTUDIO

Las redes sociales facilitan el intercambio de información entre las personas; los usuarios de estas plataformas con mucha regularidad comparten sus gustos, intereses y actividades diarias con otras personas. Por ejemplo, algunas personas publican imágenes de los cines, teatros y estadios de fútbol a los que asisten y realizan comentarios acerca del evento presenciado.

Algunos usuarios de las redes sociales también realizan reportes y denuncias sobre situaciones tales como: vialidades en mal estado, servicio de recolección de basura sin pasar por las colonias, unidades de transporte público en mal estado, malos tratos recibidos en hospitales públicos. En ocasiones estos reportes ganan tanta notoriedad que terminan siendo atendidos por la autoridad correspondiente.

Una red social muy utilizada es **Twitter**¹, la cual genera una gran cantidad de información en tiempo real. Twitter es un servicio de microblogging que permite a sus usuarios publicar mensajes breves de hasta 140 caracteres, conocidos como *tweets*. Los usuarios pueden seguir a otros usuarios, lo cual implica poder ver los tweets que publican y también pueden elegir qué usuarios pueden ver sus tweets.

Como caso de estudio para la detección de documentos duplicados se seleccionaron los reportes ciudadanos recibidos por el Centro de Integración Ciudadana (CIC²); los cuales describen accidentes de vehículos, vialidades cerradas, semáforos descompuestos (entre otros). Los reportes ciudadanos recibidos son generados por medio de la herramienta web Proyecto Tehuan [2], aplicaciones disponibles para Android e iOS o enviando un tweet a la cuenta @Cicmty. Después de recibidos, a partir del contenido de cada reporte se crea manualmente un documento para estructurar la información contenida en el reporte.

Los reportes recibidos son canalizados con la autoridad correspondiente la cual toma las medidas necesarias para resolver la situación notificada. Debido a que varias personas pueden ser testigos del mismo suceso (como el choque de autos), se pueden generar varios reportes repetidos. Por lo tanto, la persona encargada de comunicar los reportes con las autoridades correspondientes debe estar atenta de evitar notificar duplicados.

La detección de reportes duplicados presenta la dificultad de no poder decidir si un par de reportes son duplicados en base únicamente al contenido de ambos. Por ejemplo, dos personas reportan el mismo accidente de autos, donde el primer repor-

¹<https://twitter.com>

²<http://www.cic.mx>

te describe el suceso como “choque de autos” y el segundo como “tsuru impacta chevy, no hay lesionados”. A pesar de que en ambos reportes se describe el mismo suceso, un análisis únicamente de texto indicará que en los reportes se describen sucesos distintos.

Además del contenido de texto que describe el suceso en un reporte, el lugar y la hora (tiempo) del suceso juegan un papel importante cuando se quiere detectar duplicados. Se observa que se trata de dos reportes diferentes cuando se notifican dos accidentes ocurridos en lugares distintos (colonias, avenidas, municipios), pero en caso de que se trate del mismo lugar se debe de tomar en cuenta la hora de ambos accidentes. Por ejemplo, se puede decidir que dos accidentes reportados son el mismo accidente si el lugar concuerda, el lapso de tiempo entre ambos reportes es de unos pocos minutos y la descripción de los sucesos es muy parecida.

Tomando en cuenta que las descripciones de los sucesos, lugares y tiempos para reportes duplicados pueden variar, se deben establecer criterios para decidir cuando dos reportes son duplicados en base su contenido. También se debe de tomar en cuenta que los criterios de decisión son dependientes del tipo de reporte. Por ejemplo, es posible que dos reportes que se refieren a un bache en el mismo lugar pero con un lapso de tiempo de 12 horas son duplicados, mientras que dos reportes que se refieren a un accidente en el mismo lugar y con el mismo lapso de tiempo probablemente sean dos reportes distintos.

1.6 ESTRUCTURA DE LA TESIS

Esta tesis se compone de los siguientes capítulos: introducción, marco teórico, trabajos relacionados, metodología, caso de estudio, experimentos y resultados y conclusiones.

En este capítulo se planteó la definición del problema y la motivación para trabajar con los reportes del CIC para implementar un sistema de detección de

duplicados.

El capítulo 2 presenta la notación y las definiciones de conceptos necesarios para familiarizar al lector con el contenido técnico presentado en la tesis.

En el capítulo 3 se presentan algunos trabajos que abordan el tema de la detección de documentos duplicados y se describe brevemente que métodos y que documentos se utilizan para cada trabajo.

El capítulo 4 presenta el sistema de detección de duplicados de forma general; se explican cuáles son los distintos procesos que forman al sistema independientemente del contexto en el que se apliquen.

En el capítulo 5 el sistema propuesto se aplica a los reportes ciudadanos del CIC. Aquí se describen las implementaciones de cada uno de los procesos que forman el sistema de detección de duplicados.

El capítulo 6 presenta los experimentos realizados para probar el funcionamiento del sistema de detección de duplicados, muestra los resultados de desempeño obtenidos y presenta conclusiones en base a esos resultados.

Finalmente en el capítulo 7 se presentan las conclusiones generales obtenidas a partir de los resultados del sistema de detección de duplicados durante las pruebas y se presentan las posibles mejoras que serán incorporadas como trabajos a futuro.

CAPÍTULO 2

MARCO TEÓRICO

En este capítulo se describen los métodos y la terminología más relevantes que se utilizan en esta tesis en capítulos posteriores. Por ejemplo, se muestra el algoritmo de etiquetado utilizado para extraer entidades de un texto y el método de validación cruzada utilizado en esta tesis.

2.1 LIMPIEZA DE TEXTO

La limpieza de un documento de texto se encarga de eliminar las partes del documento que resultan innecesarias y de preparar el documento para posteriormente ser procesado por técnicas de minería de texto, recuperación de la información (IR por sus siglas en inglés “Information Retrieval”), procesamiento de lenguaje natural (NLP por sus siglas en inglés “Natural Language Processing”) o aprendizaje máquina (ML por sus siglas en inglés “Machine Learning”). A continuación se definen algunos términos utilizados a lo largo de la tesis y se muestra un ejemplo de la limpieza de un texto.

Las *palabras vacías* son palabras que pueden eliminarse del texto sin afectar la información contenida en éste. Regularmente los artículos, preposiciones y pronombres se consideran palabras vacías. En la figura 2.1 se muestran algunas palabras vacías definidas en el proyecto Snowball para el idioma español [6]. Las palabras vacías se eliminan antes o después de que el texto sea procesado por técnicas de minería de texto, IR, NLP o ML.


```
de, la, que, el, en, y, a, los, del, se, las, por, un, para, con, no, una, su, al,
lo, como, más, pero, sus, le, ya, o, fue, este, ha, sí, porque, esta, son, entre,
está, cuando, muy, sin, sobre, ser, tiene, también, me, hasta, hay, donde, han,
quien, están, estado, desde, todo, nos, durante, todo, nos, durante, estados, todos,
uno, les, ni, contra, otros, fueron, ese, eso, había, ante, ellos, e, esto, mí,
antes, algunos, qué, unos, yo, otro, otras, otra, él, tanto, esa, estos, mucho,
quienes, nada, muchos, cual, sea, poco, ella, estar, haber, estas, estaba, estamos
```

Figura 2.1 – Algunas de las palabras definidas en el proyecto Snowball para el idioma español.

A lo largo de esta tesis se utilizará el término de *símbolos vacíos* para hacer referencia a los símbolos y caracteres que son eliminados del texto sin afectar la información contenida del mismo. Cuando un símbolo vacío forma parte de una palabra, éste se elimina si y sólo si se encuentra al inicio o final de la palabra. Por ejemplo, los caracteres del conjunto

$$\{*, -, ?, ", -, (,), :, @, ¡, ¿\}, \quad (2.1)$$

pueden considerarse símbolos vacíos.

A lo largo de esta tesis el término *vocal acentuada* se utilizará para hacer referencia a cualquiera de los caracteres contenidos en el siguiente conjunto:

$$\{á, é, í, ó, ú, à, è, ì, ò, ù, Á, É, Í, Ó, Ú, À, È, Ì, Ò, Ù\}. \quad (2.2)$$

En la figura 2.2a se muestra un texto el cual es limpiado eliminando palabras vacías, símbolos vacíos y quitando los acentos de las vocales acentuadas; en la figura 2.2b se muestra el texto resultante.

```
La *rabia* puede ¡transmitirse!
a cualquier mamífero.
```

(a) Texto antes de ser limpiado.

```
rabia puede transmitirse
cualquier mamifero.
```

(b) Texto después de ser limpiado.

Figura 2.2 – Resultado de aplicar el proceso de limpieza a un texto sucio.

2.2 PROCESAMIENTO DE TEXTO

En ocasiones en los textos se buscan palabras que siguen un cierto patrón, como por ejemplo: palabras que comienzan en mayúscula (*Alejandro, Monterrey, Santa*), palabras que constan de una letra mayúscula y un punto (*M., P., C.*), direcciones de correo electrónico (*jcordero@gmail.com, jc@live.com, jcc@yahoo.com*). Una manera de buscar en un texto palabras que tienen una letra mayúscula y un punto se describe a continuación.

- Crear un conjunto de todas las posibles palabras formadas por una letra mayúscula y un punto, $\{A., B., C., \dots, Z.\}$.
- Buscar cada una de las palabras del texto en el conjunto creado.
- Todas las palabras del texto encontradas en el conjunto son las palabras que siguen el patrón establecido.

El mismo procedimiento puede aplicarse en la búsqueda de palabras que sigan cualquier patrón. Una desventaja de este método es que tiene que listar todas las posibles palabras que siguen un patrón, lo cual se hace evidente cuando el patrón es todas las palabras que comiencen con la letra “a” $\{a, aa, aaa, \dots, a\dots a, ab, \dots\}$, en cuyo caso no se puede crear el conjunto de dichas palabras. Para evitar este problema al buscar palabras que siguen un patrón específico en un texto, se pueden utilizar expresiones regulares.

Una *expresión regular* es una secuencia de caracteres que describe a un conjunto de palabras que siguen un patrón en específico [21]. A continuación se muestran ejemplos de expresiones regulares y los patrones que describen.

- $a.*$: palabras que comienzan con la letra a.
- $[A-Z][a-z]*$: palabras que comienzan con letras mayúsculas seguidas de cero o más letras minúsculas.

- `[a-z0-9]+@gmail\.com`: direcciones de correo electrónico que comienzan con una o más letras minúsculas o números y terminan en `@gmail.com`.

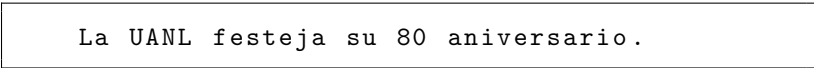
Las expresiones regulares se crean de acuerdo al patrón que se busca en un texto. Las expresiones regulares utilizadas en esta tesis siguen las reglas definidas en la librería estándar del lenguaje de programación Python [7]. Por ejemplo, en Python la expresión regular `a.*` se representa como `r'a.*'`.

2.3 RECONOCIMIENTO DE ENTIDADES EN UN TEXTO

En algunas ocasiones se requiere obtener la información más relevante del contenido de un documento de texto. Una manera de extraer la información relevante de un texto consiste en asignar *etiquetas* a las palabras, tales como `relevante` o `irrelevante`. Posteriormente, se eliminan las palabras etiquetadas como irrelevantes y se conservan las palabras etiquetadas como relevantes.

Además de utilizar las etiquetas `relevante` e `irrelevante`, también se puede utilizar otro tipo de etiquetas dependiendo del tipo de información que se quiere extraer de un texto. Las etiquetas `persona`, `lugar`, `puesto` y `empresa` pueden utilizarse para extraer de un texto el nombre de una persona, la empresa para la cual trabaja, su puesto, y la ubicación de la empresa o la casa de la persona. Las etiquetas definidas anteriormente de nada sirven en textos cuyo contenido son palabras que representan cadenas de ADN, lo cual indica que las etiquetas utilizadas dependen del tipo de textos que van a ser etiquetados.

Cada una de las etiquetas representa una entidad y al proceso de etiquetar un texto para obtener las entidades contenidas en el texto, se le conoce como *reconocimiento de nombres de entidades* (NER por sus siglas en inglés “Named Entity



La UANL festeja su 80 aniversario.

Figura 2.3 – Cadena de texto.

(La, IRRELEVANTE)	(UANL, ORGANISMO)	(festeja, SUCESO)
(su, IRRELEVANTE)	(80, TIEMPO)	(aniversario, TIEMPO)
(., IRRELEVANTE)		

Figura 2.4 – Cadena de texto etiquetada.

Cuadro 2.1 – N -gramas obtenidos a partir del texto “semaforo descompuesto en la avenida”.

N -grama	Secuencias obtenidas
Unigramas	(semaforo), (descompuesto), (en), (la), (avenida)
Bigramas	(semaforo, descompuesto), (descompuesto, en), (en, la), (la, avenida)
Trigramas	(semaforo, descompuesto, en), (descompuesto, en, la), (en, la, avenida)

Recognition”). En la figura 2.3 se observa una cadena de texto a la cual se le aplica NER y en la figura 2.4 se muestra cada una de las palabras con sus respectivas etiquetas, las cuales son *organismo*, *suceso*, *tiempo* e *irrelevante*.

2.3.1 N-GRAMAS

Un n -grama de una cadena de texto es una secuencia de n palabras contenidas en el texto. Cuando las secuencias son de una, dos y tres palabras se llaman *unigramas*, *bigramas* y *trigramas* respectivamente. En el cuadro 2.1 muestran los unigramas, bigramas y trigramas obtenidos a partir de la cadena “semaforo descompuesto en la avenida”.

2.3.2 ETIQUETADO MEDIANTE EL ALGORITMO VITERBI

Existen diferentes métodos para extraer entidades del contenido un documento de texto. En esta tesis se utiliza una implementación sencilla de un NER basada en un *Modelo Oculto de Markov* [19, 25, 30] (*HMM* por sus siglas en inglés “Hidden Markov Model”), a continuación se explica dicha implementación.

Para cada una de las palabras en una cadena de texto

$$\mathbf{x} = x_1x_2 \dots x_n, \quad (2.3)$$

se desea obtener la secuencia de etiquetas

$$\mathbf{y} = y_1y_2 \dots y_n \quad (2.4)$$

tal que; a cada palabra le pertenece una de las etiquetas contenidas en un conjunto

$$K = \{e_1, e_2, \dots, e_k\}. \quad (2.5)$$

La secuencia \mathbf{y} se obtiene al maximizar la *probabilidad conjunta* de una cadena de texto y su correspondiente secuencia de etiquetas

$$P(\mathbf{y}, \mathbf{x}) = P(\mathbf{x}|\mathbf{y})P(\mathbf{y}), \quad (2.6)$$

donde $P(\mathbf{x}|\mathbf{y})$ corresponde a la *probabilidad condicional* de generar la cadena de texto \mathbf{x} dada la secuencia de etiquetas \mathbf{y} y $P(\mathbf{y})$ corresponde a la distribución de *probabilidad a priori* sobre la secuencia de etiquetas \mathbf{y} [30, 34].

El cálculo de $P(\mathbf{y}, \mathbf{x})$ se simplifica al utilizar un Modelo Oculto de Markov de segundo orden, el cual aplica dos simplificaciones:

- Primero, supone que una etiqueta y_i depende unicamente de las dos etiquetas anteriores y_{i-1}, y_{i-2} (propiedad de Markov).
- Luego, se asume que cada palabra observada x_i depende unicamente de la etiqueta y_i .

Con estas dos simplificaciones la ecuación 2.6 se representa de la siguiente manera:

$$P(\mathbf{y}, \mathbf{x}) = \prod_{i=1}^{n+1} P(y_i|y_{i-1}, y_{i-2}) \prod_{i=1}^n P(x_i|y_i) \quad (2.7)$$

Para esta implementación del NER los parámetros del modelo HMM $P(y_i|y_{i-1}, y_{i-2})$ y $P(x_i|y_i)$ son fáciles de calcular debido a que están basados en

unigramas, bigramas y trigramas [16]. Para calcular estos parámetros se utiliza un conjunto de datos que consta de palabras etiquetadas.

Antes de presentar el cálculo de los parámetros del modelo HMM se definen los siguientes términos:

- El número de veces que el trigrama de etiquetas (u, v, w) aparece en los datos: $c(u, v, w)$. Por ejemplo, $c(\text{SUCESO}, \text{TIEMPO}, \text{TIEMPO})$.
- El número de veces que el bigrama de etiquetas (u, v) aparece en los datos: $c(u, v)$. Por ejemplo, $c(\text{SUCESO}, \text{TIEMPO})$.
- El número de veces que el unigrama de etiqueta (u) aparece en los datos: $c(u)$. Por ejemplo, $c(\text{SUCESO})$.
- El número de veces que el unigrama de etiquetas (s) aparece junto a la palabra x : $c(s \rightsquigarrow x)$. Por ejemplo, $c(\text{SUCESO} \rightsquigarrow \text{festeja})$.

Los parámetros del modelo HMM están dados por:

$$P(s|u, v) = \frac{c(u, v, s)}{c(u, v)}, \quad (2.8)$$

$$P(x|s) = \frac{c(s \rightsquigarrow x)}{c(s)}. \quad (2.9)$$

Después de calcular los parámetros del modelo HMM se necesita obtener la secuencia de etiquetas más probable para una cadena de texto. Lo cual se reduce al problema de encontrar

$$\arg \max_{y_1 \dots y_{n+1}} P(\mathbf{y}, \mathbf{x}), \quad (2.10)$$

donde el resultado de $\arg \max$ se obtiene entre todas las secuencias de etiquetas $y_1 \dots y_{n+1}$, de tal forma que $y_i \in K$ y $y_{n+1} = \text{STOP}$. La etiqueta **STOP** se utiliza para trabajar con cadenas de diferentes tamaños [15] y sus probabilidades se calculan de la misma manera que se calculan las probabilidades de las otras etiquetas.

Utilizando el *algoritmo Viterbi* [20, 25] se obtiene la secuencia de etiquetas para la ecuación 2.10. El algoritmo 1 muestra al algoritmo Viterbi utilizado en esta tesis.

Algoritmo 1 El Algoritmo Viterbi.

Entrada: una cadena de texto $x_1 \dots x_n$, parámetros $P(s|u, v)$ y $P(x|s)$.

Establecer $\pi(0, *, *) = 0$ para cada (u, v) tal que $u \neq *$ o $v \neq *$.

Para $k = 1 \dots n$ **Hacer**

Para $u \in K, v \in K$ **Hacer**

$$\pi(k, u, v) = \max_{w \in K} (\pi(k-1, w, u) \times P(u|w, u) \times P(x_k|v))$$

$$bp(k, u, v) = \arg \max_{w \in K} (\pi(k-1, w, u) \times P(u|w, u) \times P(x_k|v))$$

Fin Para

Fin Para

Establecer $(y_{n-1}, y_n) = \arg \max_{(u,v)} (\pi(n, u, v) \times P(\text{STOP}|u, v))$

Para $k = (n-2) \dots 1$ **Hacer**

$$y_k = bp(k+2, y_{k+1}, y_{k+2})$$

Fin Para

Retornar la secuencia de etiquetas $y_1 \dots y_n$

2.3.3 CÁLCULO DE LOS PARÁMETROS DEL ALGORITMO VITERBI

Para calcular los parámetros del algoritmo Viterbi se crea un conjunto de *datos de entrenamiento*; dicho conjunto consta de un grupo de palabras las cuales tienen una etiqueta asignada. En la figura 2.5a se muestra una parte de un archivo que contiene datos de entrenamiento, en los cuales se han utilizado las etiquetas REL (Información Relevante) e IRR (Información Irrelevante).

En la figura 2.5b se muestra parte del *archivo de parámetros de etiquetado*, el cual contiene los parámetros $c(u, v, s)$, $c(u, v)$, $c(u)$ y $c(s \rightsquigarrow x)$ calculados a partir de los datos en la figura 2.5a. El contenido de la figura 2.5b se lee de la siguiente manera:

- La primera columna indica las veces que uno de los parámetros $c(u, v, s)$,

rabia REL	1300	WORDTAG	REL	vacuna	
cuidado IRR	400	WORDTAG	REL	canino	
transmitirse REL	250	WORDTAG	IRR	calle	
perros REL	3400	1-GRAM	REL		
gatos REL	1660	1-GRAM	IRR		
siempre IRR	2500	2-GRAM	REL	REL	
mamiferos REL	4600	2-GRAM	REL	IRR	
clinica REL	5400	2-GRAM	IRR	REL	
mes IRR	2300	3-GRAM	REL	IRR	IRR
colonia IRR	1200	3-GRAM	REL	REL	IRR
	3400	3-GRAM	IRR	IRR	REL
	430	3-GRAM	REL	REL	REL

(a) Conjunto de datos de entrenamiento. (b) Parámetros del algoritmo Viterbi.

Figura 2.5 – Parte de los parámetros del algoritmo Viterbi obtenidos a partir de un conjunto de datos de entrenamiento.

$c(u, v)$, $c(u)$ y $c(s \rightsquigarrow x)$ fue obtenido a partir de los datos de entrenamiento.

- La segunda columna indica el tipo de parámetro calculado, WORDTAG se refiere a $c(s \rightsquigarrow x)$, 1-GRAM a $c(u)$, 2-GRAM a $c(u, v)$ y 3-GRAM a $c(u, v, s)$.
- Las columnas restantes muestran como se componen los parámetros WORDTAG, 1-GRAM, 2-GRAM y 3-GRAM.

Por ejemplo, el primer renglón de la figura 2.5b indica que (REL \rightsquigarrow vacuna) apareció 1,300 veces en los datos del conjunto de entrenamiento, y el último renglón indica que el trigramma (REL, REL, REL) apareció 430 veces en los datos del conjunto de entrenamiento.

2.4 REPRESENTACIÓN ESTRUCTURADA DE TEXTO

La mayoría de los métodos de minería de datos sólo son capaces de trabajar con información estructurada. Por lo tanto, cuando se requiere aplicar métodos de

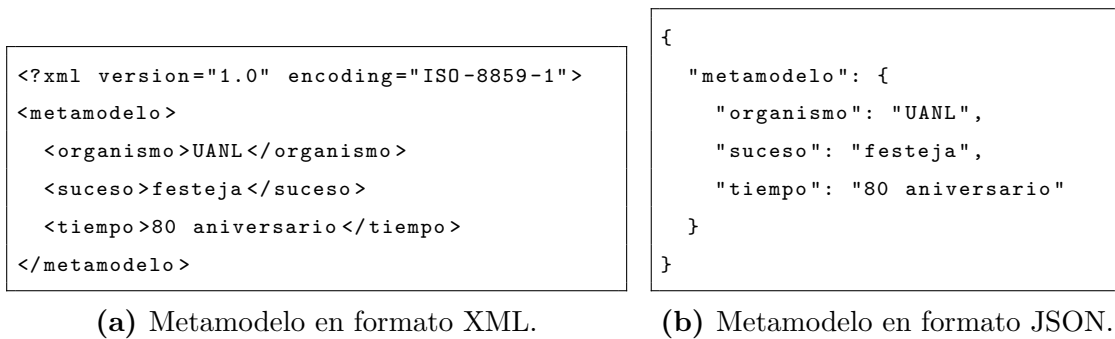


Figura 2.6 – Metamodelos generados en formato XML y JSON.

minería de datos a un documento de texto, el contenido del documento primero debe ser transformado y después se aplican los métodos correspondientes.

Existen varios formatos para crear una representación estructurada de un documento, dos de los más utilizados son: el lenguaje de marcas extensible (XML por sus siglas en inglés “Extensible Markup Language”) [10] y la notación de objetos de Javascript (JSON por sus siglas en inglés “JavaScript Object Notation”) [3]. La figura 2.6a muestra una representación estructurada en formato XML del contenido etiquetado mostrado en la figura 2.4 y en la figura 2.6b se muestra la representación estructurada en formato JSON; en ambas representaciones las palabras marcadas como irrelevantes no son agregadas.

A lo largo de la tesis se utiliza el término *metamodelo* para hacer referencia a la representación estructurada de un documento de texto etiquetado, para cualquier tipo de formato que se quiera usar (XML, JSON).

2.5 REPRESENTACIÓN DE DOCUMENTOS COMO VECTORES

En el campo de recuperación de la información es muy común representar el contenido de un documento de texto como un vector. A continuación se describe una manera de representar documentos como vectores.

Un documento de texto d en su forma más simple, puede ser visto como una secuencia de símbolos y palabras separadas por espacios en blanco. Por ejemplo, “Precaución, carretera en mal estado.” se considera un documento de texto, donde {“Precaución”, “carretera”, “en”, “mal”, “estado”} son las palabras y {“,”, “.”} son los símbolos. A partir de aquí se considerará que todos los documentos que van a ser transformados a vectores contienen sólo palabras (los símbolos se eliminan mediante un proceso de limpieza).

A partir de una colección de documentos de texto $D = \{d_1, d_2, \dots, d_n\}$, se crea un *vocabulario de términos* V [38], el cual está formado por las diferentes palabras (términos) contenidas en los documentos. Por ejemplo, para los documentos $d_1 = \text{“UANL festeja 80 aniversario”}$, $d_2 = \text{“la UANL premia la excelencia”}$ y $d_3 = \text{“ITESM celebra 70 aniversario”}$ se obtiene el siguiente vocabulario

{“70”, “80”, “aniversario”, “celebra”, “excelencia”, “festeja”, “ITESM”, “la”, “premia”, “UANL”}.

Una vez obtenido el vocabulario de términos, un documento de texto d se puede representar como un *vector de frecuencia de términos* \vec{d} . El vector \vec{d} tiene $|V|$ posiciones, una por cada término t del vocabulario de términos. Cada posición contiene el número de veces que t aparece en d . En el cuadro 2.2 se muestran los documentos con los que se formó el vocabulario de términos representados como vectores de frecuencia de términos; cada columna corresponde a un vector. Por ejemplo, $\vec{d}_2 = [0, 0, 0, 1, 1, 2, 1, 0, 0, 0]^T$.

Aplicando el método de ponderación frecuencia de término - frecuencia inversa de documento (*tfidf* por sus siglas en inglés “Term frequency - Inverse document frequency”) [15] a los vectores de frecuencia de términos, se obtienen *vectores ponderados*. La ponderación *tfidf* a un término se calcula mediante la formula

$$tfidf(j) = tf(j) \times \log \left(\frac{N}{df(j)} \right), \quad (2.11)$$

donde $tf(j)$ corresponde a la frecuencia del j -ésimo término en un documento, $df(j)$ corresponde al número de documentos de la colección D en los que aparece el j -ésimo

Cuadro 2.2 – Vectores de frecuencias de términos obtenidos al contar el número de veces que un término del vocabulario de términos aparece en los documentos: $d_1 =$ “UANL festeja 80 aniversario”, $d_2 =$ “la UANL premia la excelencia” y $d_3 =$ “ITESM celebra 70 aniversario”.

Términos	Vectores de frecuencia		
	\vec{d}_1	\vec{d}_2	\vec{d}_3
80	1	0	0
aniversario	1	0	1
festeja	1	0	0
UANL	1	1	0
excelencia	0	1	0
la	0	2	0
premia	0	1	0
70	0	0	1
celebra	0	0	1
ITESM	0	0	1

término y N al número de documentos de la colección D .

El cuadro 2.3 muestra los vectores ponderados obtenidos para los documentos d_1 , d_2 y d_3 después de haber sido ponderados con el método *tfidf*. Por ejemplo, para el documento d_2 se obtiene $\vec{d}_2 = [0, 0, 0, 0.15, 0.40, 0.80, 0.40, 0, 0, 0]^\top$.

Para un nuevo documento no visto en la colección de documentos del vocabulario de términos, también se puede crear su representación vectorial ponderada. Por ejemplo, para el documento $d_j =$ “la FIME festeja su 63 aniversario” obtiene el vector ponderado $\vec{d}_j = [0, 0.25, 0.68, 0, 0, 0.68, 0, 0, 0, 0]^\top$.

Cuadro 2.3 – Vectores ponderados obtenidos después de aplicar el método *tfidf* a los vectores de frecuencias de los documentos $d_1 =$ “UANL festeja 80 aniversario”, $d_2 =$ “la UANL premia la excelencia” y $d_3 =$ “ITESM celebra 70 aniversario”.

Términos	Vectores ponderados		
	\vec{d}_1	\vec{d}_2	\vec{d}_3
80	0.633	0	0
aniversario	0.244	0	0.208
festeja	0.633	0	0
UANL	0.244	0.149	0
excelencia	0	0.403	0
la	0	0.807	0
premia	0	0.403	0
70	0	0	0.564
celebra	0	0	0.564
ITESM	0	0	0.564

2.6 FUNCIONES DE DISTANCIA

Para comparar el nivel de similitud entre dos vectores \vec{p} y \vec{q} se utiliza una función de distancia $d(\vec{p}, \vec{q})$, la cual produce un resultado numérico. A continuación se describen dos funciones de distancia para vectores, las cuales se mencionan en capítulos posteriores.

La *distancia Euclidiana* entre dos vectores n -dimensionales \vec{p} y \vec{q} está dada por

$$\|\vec{p} - \vec{q}\|^2 = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}; \quad (2.12)$$

por ejemplo, para los vectores $\vec{p} = [1, 2, 3]^T$ y $\vec{q} = [5, 2, 7]^T$, $d(\vec{p}, \vec{q}) = 5$.

La *similitud cosenoidal* [38] entre dos vectores n -dimensionales \vec{p} y \vec{q} está dada por

$$SC(\vec{p}, \vec{q}) = \frac{\vec{p} \cdot \vec{q}}{|\vec{p}| |\vec{q}|}, \quad (2.13)$$

donde $\vec{p} \cdot \vec{q}$ corresponde al producto punto entre \vec{p} y \vec{q} y está dado por

$$\vec{p} \cdot \vec{q} = p_1 \times q_1 + p_2 \times q_2 + \dots + p_n \times q_n. \quad (2.14)$$

El valor de $|\vec{p}||\vec{q}|$ corresponde a la multiplicación de las magnitudes de los vectores \vec{p} y \vec{q} y está dado por

$$|\vec{p}||\vec{q}| = \sqrt{p_1^2 + p_2^2 + \dots + p_n^2} \times \sqrt{q_1^2 + q_2^2 + \dots + q_n^2}. \quad (2.15)$$

Al aplicar similitud cosenoidal a un par de vectores se obtienen valores en un rango de $[-1, 1]$. Para el caso de vectores de documentos ponderados el rango de valores es $[0, 1]$; entre más cercano a uno sea el resultado obtenido al aplicar similitud cosenoidal más parecidos son los vectores y entre más cercano a cero más diferentes son los vectores.

2.7 AGRUPAMIENTO DE DOCUMENTOS DE TEXTO

Dado un conjunto de documentos de texto, es muy común agrupar esos documentos de acuerdo a la información contenida en los mismos. En la figura 2.7 se muestra un conjunto de cinco documentos de texto los cuales son asignados a uno de los tres grupos (grupo 1, grupo 2 y grupo 3).

Existen dos tipos de agrupamiento de documentos muy utilizados los cuales se describen a continuación:

Agrupamiento rígido un documento es asignado solamente a un grupo.

Agrupamiento difuso un documento puede ser asignado a uno o más grupos.

Existen varios algoritmos de agrupamiento de texto; uno de ellos es el algoritmo *k-medias* [38], el cual realiza un agrupamiento rígido a los documentos.

A partir de un conjunto de documentos de texto $\{d_1, d_2, \dots, d_n\}$ se obtienen los vectores $\{\vec{d}_1, \vec{d}_2, \dots, \vec{d}_n\}$; el objetivo del algoritmo *k-medias* es agrupar los vectores

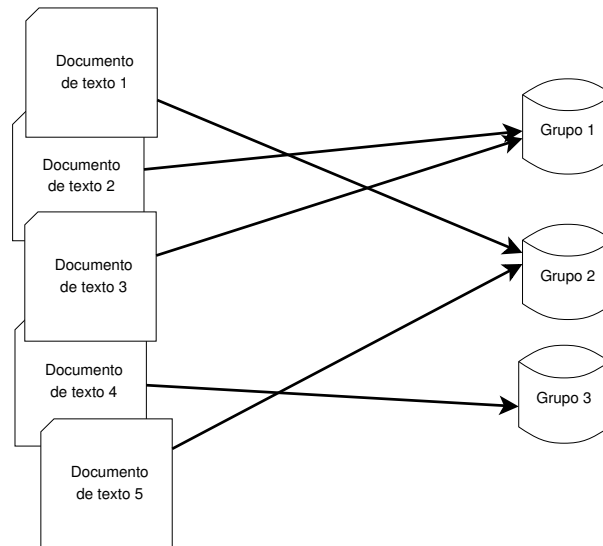


Figura 2.7 – Agrupamiento de documentos de texto en tres categorías distintas.

en k contenedores (grupos) de vectores similares. Al agrupar los vectores representativos también se agrupan los documentos a partir de los cuales fueron generados los vectores. El algoritmo k -medias funciona de la siguiente manera:

1. Distribuir los vectores $\{\vec{d}_1, \vec{d}_2, \dots, \vec{d}_n\}$ entre los k contenedores.
2. Calcular el *centroide* (vector promedio) $\vec{\mu}$ para los vectores de cada contenedor.
3. Comparar cada vector contra los k centroides mediante

$$d(\vec{d}_i, \vec{\mu}_c) = \|\vec{d}_i - \vec{\mu}_c\|^2 \quad (2.16)$$

y anotar el centroide más similar.

4. Mover los vectores a sus contenedores más similares.
5. En caso de que ningún vector haya sido movido a un contenedor diferente, terminar; de lo contrario volver al paso 2.

2.8 CLASIFICACIÓN SUPERVISADA

La clasificación de documentos de texto consiste en asignar a cada uno de los documentos, una etiqueta que indica la categoría a la cual pertenece un documento. En el área de aprendizaje máquina existe un método de clasificación que se conoce como *método de clasificación supervisada*, el cual se explica a continuación.

Suponiendo que se tiene un conjunto de documentos los cuales se quieren clasificar en dos categorías, categoría de **deportes** y categoría de **salud**, para cada documento se crea un vector de documento $\vec{x} = [x_1, x_2, \dots, x_n]^\top$. A cada vector \vec{x} se le aplica una función $h(\vec{x})$, la categoría correspondiente al documento se asigna en base al resultado de $h(\vec{x})$. Una función h muy simple se muestra a continuación:

$$h(\vec{x}) = \begin{cases} 1, & \text{si } \vec{w}^\top \cdot \vec{x} \geq 0, \\ -1, & \text{si } \vec{w}^\top \cdot \vec{x} < 0, \end{cases} \quad (2.17)$$

donde $\vec{w} = [w_1, w_2, \dots, w_n]^\top$ es un vector de pesos; más adelante se explica cómo obtener este vector. Para el ejemplo anterior se puede clasificar en la categoría de **deportes** a un documento cuando el resultado de $h(\vec{x}) = 1$ y en la categoría de **salud** cuando el resultado de $h(\vec{x}) = -1$. La selección del valor que representa a una categoría, se realiza de forma arbitraria, de igual manera se puede elegir 1 para la categoría de **salud** y -1 para la categoría de **deportes**.

Para obtener los valores del vector de pesos \vec{w} se necesita un conjunto de documentos los cuales han sido previamente asignados a una de las dos categorías mencionadas; a este conjunto se le conoce como *conjunto de datos de entrenamiento* o *conjunto de entrenamiento*. Para dicho conjunto de entrenamiento se crea una matriz \mathbf{X} , que contiene a los vectores de documentos

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)} \\ x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)} \\ \vdots \\ x_1^{(m)}, x_2^{(m)}, \dots, x_n^{(m)} \end{bmatrix}, \quad (2.18)$$

donde $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}]^\top$ representa al i -ésimo vector de documento creado a partir de los documentos del conjunto de entrenamiento, m es el número de documentos del conjunto de entrenamiento y n es el número de características de los vectores o el número de palabras del diccionario creado con los documentos del conjunto de entrenamiento (ver sección 2.5).

Cada uno de los vectores $\mathbf{x}^{(i)}$ tienen asignada una etiqueta la cual indica la categoría a la que pertenecen, para este caso las etiquetas corresponden a los valores 1 y -1 . Utilizando las etiquetas se crea un vector de etiquetas como el siguiente: $\mathbf{y} = [1, -1, \dots, -1]^\top$.

Finalmente aplicando un algoritmo de optimización a la matriz \mathbf{X} y el vector \vec{y} , se obtiene el vector de pesos \vec{w} [19]. Una vez obtenido \vec{w} , se pueden clasificar documentos nuevos que no pertenecen al conjunto de entrenamiento. A lo largo de la tesis la palabra *clasificador* hara referencia a la función h y la palabra *parámetros del clasificador* al vector \vec{w} .

Algunos de los clasificadores supervisados más conocidos son: máquinas de soporte vectorial (SVM por sus siglas en inglés “Support Vector Machine”) [26], redes neuronales (por sus siglas en inglés “Neural Networks”) [19], regresión logística [13, 23, 24] y clasificador bayesiano ingenuo [13, 23].

En la figura 2.8 se muestra un ejemplo del entrenamiento de un clasificador de tipo máquina de soporte vectorial utilizando la librería scikit-learn de Python [8]. En las líneas 2, 3 y 4 del código se crean la matriz \mathbf{X} y el vector de etiquetas \vec{y} a partir del conjunto de entrenamiento, en la línea 5 se crea un clasificador de tipo SVM y en la línea 6 se realiza el entrenamiento del clasificador.


```
1 from sklearn import svm
2 X = [[0.23, 0.13, 0.17],
3      [0.89, 0.92, 0.95]]
4 Y = [0, 1]
5 clasificador = svm.SVC()
6 clasificador.fit(X,Y)
```

Figura 2.8 – Código de entrenamiento de una SVM.

2.9 DESEMPEÑO DE UN CLASIFICADOR SUPERVISADO

La validación cruzada [32] utiliza dos conjuntos de datos (*conjunto de entrenamiento* y *conjunto de prueba*) para determinar el desempeño de los algoritmos de clasificación supervisada. El conjunto entrenamiento se utiliza para crear los clasificadores supervisados y el conjunto de prueba para determinar el porcentaje de precisión o error de cada clasificador.

K iteraciones es un método de validación cruzada en el cual un conjunto de datos D se divide aleatoriamente en K grupos del mismo tamaño. Con $K - 1$ grupos se entrena un clasificador y con el grupo restante se prueba el clasificador, este procedimiento se realiza K veces. Cada uno de los grupos se utiliza una vez para probar el clasificador.

En esta tesis se utiliza una variación del método K iteraciones para crear conjuntos de entrenamiento y prueba. A continuación se explica el funcionamiento de este método mostrado en la figura 2.9, aplicado a documentos de texto.

Para un conjunto de documentos de texto se crean $K = m + n$ grupos de documentos, de los cuales m grupos se utilizan para formar el conjunto de entrenamiento y n grupos para el conjunto de pruebas. Durante la i -ésima iteración se selecciona el i -ésimo grupo y los siguientes $m - 1$ grupos para formar el conjunto de entrenamiento, el conjunto de pruebas se crea con los n grupos restantes. Cuando se llega al último grupo y no se han completado los m grupos de entrenamiento, los grupos restantes se seleccionan a partir del primer grupo.

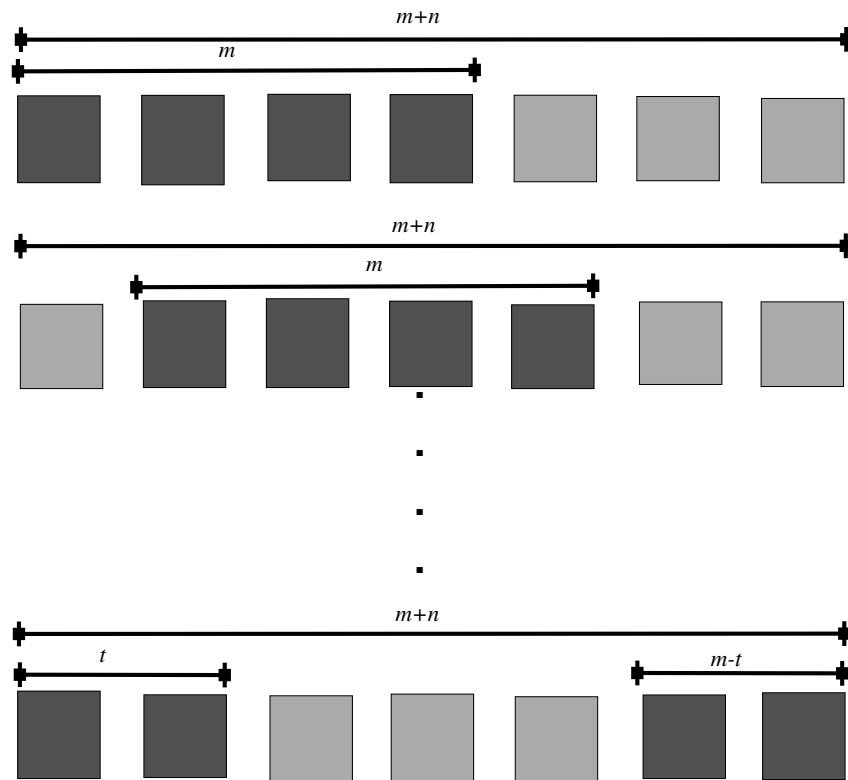


Figura 2.9 – Método de creación de conjuntos de entrenamiento y prueba, basado en el método de validación cruzada K iteraciones.

Para un grupo de documentos que pueden ser clasificados como positivos o negativos, se puede utilizar un clasificador binario para asignar la categoría correspondiente a cada documento. Los documentos clasificados pueden caer en uno de los cuatro conjuntos mostrados en el cuadro 2.4, los cuales son:

Verdaderos positivos (TP) son los documentos positivos clasificados como positivos.

Falsos positivos (FP) son los documentos negativos clasificados como positivos.

Verdaderos negativos (TN) son los documentos negativos clasificados como negativos.

Falsos negativos (FN) son los documentos positivos clasificados como negativos.

Cuadro 2.4 – Descripción de los conjuntos verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.

	Positivos	Negativos
Clasificados positivos	TP	FP
Clasificados negativos	FN	TN

Utilizando los valores de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos se obtienen las siguientes medidas:

- La *precisión* [17] corresponde al porcentaje de documentos clasificados como positivos y que realmente son positivos y está dada por

$$PR = \frac{TP}{TP + FP}.$$

- La *exhaustividad* [17] corresponde al porcentaje de documentos positivos que son clasificados como positivos y está dada por

$$RE = \frac{TP}{TP + FN}.$$

- El *valor-F* [17] corresponde a una media ponderada de la precisión y exhaustividad y está dado por

$$F_1 = 2 \cdot \frac{PR \cdot RE}{PR + RE}.$$

Las formulas de precisión, exhaustividad y valor-F definidas se utilizan para calificar el desempeño de un algoritmo de clasificación supervisada. Para este trabajo se busca obtener valores de precisión altos al mismo tiempo que se trata de evitar que los valores de exhaustividad obtenidos sean muy bajos.

CAPÍTULO 3

TRABAJOS RELACIONADOS

En este capítulo se presentan cuatro trabajos relacionados con el sistema de detección de reportes duplicados desarrollado en esta tesis. Se realiza una breve descripción de cada uno de los trabajos y se explican las similitudes y diferencias entre esos trabajos y este trabajo de tesis.

DETECCIÓN DE DOCUMENTOS CASI DUPLICADOS EN TWITTER

En el trabajo realizado por Tao et al. [37], los autores presentan un método para la detección de casi duplicados en Twitter. El proceso de detección de duplicados se realiza en dos pasos: (1) decidir si un par de tweets son duplicados y (2) determinar el nivel de similitud (duplicación). Existen cinco niveles de duplicación: (1) copias exactas, (2) copias casi exactas, (3) casi duplicados fuertes, (4) casi duplicados débiles y (5) poco traslape.

A continuación se muestran algunas de las características obtenidas a partir de un par de mensajes de Twitter (t_a, t_b) . Se obtienen características sintácticas como la distancia Levenshtein [19, 22, 28], porcentaje de palabras compartidas, porcentaje de hashtags compartidos, porcentaje de localizadores uniformes de recursos (URL por sus siglas en inglés “Uniform Resource Locator”) compartidos. Utilizando sistemas de reconocimiento de entidades, se obtienen características semánticas como el porcentaje de entidades compartidas y el porcentaje de los tipos de entidades compartidas. También se obtiene información correspondiente a la diferencia de tiempo entre los mensajes y la similitud de los usuarios que crean los mensajes.

Tao et al. [37] utilizan seis diferentes estrategias para realizar la detección de duplicados; para tener una estrategia base contra la cual comparar el desempeño de las estrategias definidas se utiliza la distancia Levenhstein. Una de las estrategias usa únicamente características semánticas realizando detección de duplicados en base al contenido de los mensajes. Otra estrategia utiliza características semánticas y sintácticas, también utilizan una estrategia que hace uso de características semánticas, sintácticas y contextuales. Los autores mencionan que se pueden generar más estrategias mediante combinaciones de las diferentes características generadas para los pares de mensajes. Para realizar la detección de duplicados en los pasos (1) y (2) se utiliza regresión logística.

En el trabajo se presentan las siguientes preguntas: (1) *¿cuál es el nivel de precisión de detección de duplicados para las diferentes estrategias utilizadas?*, (2) *¿cuáles características son importantes en la detección de duplicados?* y (3) *¿cómo varía la precisión en los diferentes niveles de duplicados?*.

En ambos el trabajo realizado por Tao et al. [37] y el trabajo presentado en esta tesis se utilizan métodos de identificación de entidades para obtener características semánticas, también se obtiene información referente al tiempo en ambos casos. La diferencia está en que en el trabajo de Tao et al. se generan hasta 22 características mientras que en la tesis se obtienen un máximo de cinco características (ver capítulo 5). Para decidir si dos textos son duplicados en el trabajo de Tao et al. se utiliza regresión logística y en la tesis se utilizan máquinas de soporte vectorial (ver capítulo 5).

EXTRACCIÓN DE NOTICIAS EN TWITTER

En el trabajo de Sankaranarayanan et al. [35] se describe la implementación de un sistema de extracción de noticias utilizando mensajes de Twitter. La idea principal consiste en obtener los tweets correspondientes a las noticias más recientes y agruparlos de acuerdo al tema y lugar de las noticias reportadas. El método utilizado se puede dividir en tres partes: (1) separar los tweets que contienen alguna noticia de

aquellos que no (llamados “ruido”), (2) agrupar los tweets que contienen información sobre la misma noticia y (3) detectar el tipo de noticia y el lugar de la noticia para cada grupo de tweets creado.

Los tweets que ingresan al sistema son separados en noticias y ruido, la selección de noticias se lleva a cabo utilizando un clasificador bayesiano ingenuo, los parámetros del clasificador se ajustan usando un conjunto de tweets de entrenamiento los cuales están etiquetados como noticias o ruido.

Los tweets clasificados como noticias son agrupados con otros tweets de acuerdo al tema de la noticia. Cada grupo de tweets tiene asociado un vector de características obtenido al aplicar *tfidf* a las palabras de los tweets en el grupo y un centroide de tiempo obtenido al calcular el promedio del tiempo de publicación de cada tweet. Durante la agrupación de los tweets existe una lista de grupos activos a los cuales se permite agregar nuevos tweets, los grupos se marcan como inactivos cuando sus centroides de tiempo están atrasados tres días o más con respecto a la fecha actual. Para asignar un tweet a un grupo primero se representa el tweet por un vector de características por medio de la aplicación de *tfidf* al contenido del tweet, luego se compara el contenido del tweet con el contenido de cada grupo activo de tweets utilizando una variación de la distancia cosenoidal y finalmente se asigna el tweet al grupo con el que tiene mayor similitud. Finalmente para cada grupo de tweets se obtiene la noticia a la que refieren los tweets y se obtiene la localización geográfica que pertenece al grupo utilizando técnicas de reconocimiento de entidades y la información contenida en los metadatos de los tweets.

Del trabajo realizado por Sankaranarayanan et al. [35] se destaca el uso de *tfidf* para crear representaciones de los tweets, la comparación de similaridad mediante una variante de la similitud cosenoidal, la extracción de la posición geográfica referida en el contenido de los tweets y la agrupación de tweets de acuerdo al contenido. En esta tesis también se utiliza *tfidf*, la similitud cosenoidal y se extrae la información de lugares contenida en los reportes. En el trabajo de Sankaranarayanan et al. el número de grupos que contienen tweets puede aumentar conforme aparecen nuevas

noticias, mientras que en el trabajo de tesis el número de grupos de documentos se mantiene constante, sin importar la cantidad de nuevos documentos que sean introducidos al sistema de detección de duplicados.

DETECCIÓN DE NOTICIAS LOCALES EN TWITTER

Agarwal et al. [11] presentan un trabajo enfocado en la detección de noticias locales reportadas en Twitter, la información relacionada a dichos eventos es encontrada en el contenido de varios tweets. Una de las principales dificultades para detectar las noticias locales es el escaso número de tweets correspondientes a una noticia local, además todos los tweets relacionados con un evento no aparecen en un instante corto de tiempo, más bien aparecen poco a poco conforme pasa el tiempo después de haber sucedido el evento reportado. Los autores se enfocan en los eventos relacionados a “incendios en fábricas” y “huelgas de trabajo”. El proceso de detección se realiza en cuatro pasos, a continuación se describe brevemente cada uno.

Como primer paso se detectan los tweets que reportan la ocurrencia de un evento. La detección se realiza rechazando los tweets que no corresponden a los eventos de “incendios de fábricas” y “huelgas de trabajo”, los tweets que no son rechazados son clasificados como eventos. Se utilizan expresiones regulares para rechazar los mensajes que contienen un patrón específico. Para los mensajes que las expresiones regulares no pueden rechazar se usan clasificadores supervisados. Algunas de las características utilizadas por los clasificadores son las entidades correspondientes a nombres de organizaciones, lugares y gente contenidas en los tweets, dichas entidades se obtienen utilizando el NER de Stanford [9].

Como segundo paso, los nuevos tweets clasificados como eventos se comparan con los tweets de las últimas 24 horas; los tweets que coinciden en el evento reportado son agrupados. Los autores llaman a este proceso *correlación entre mensajes*. En caso de que el evento reportado por un tweet no corresponda a los eventos reportados por los tweets de las últimas 24 horas, este tweet forma su propio grupo

que corresponde a un nuevo evento. Los grupos creados definen eventos parciales, los cuales posteriormente son correlacionados en base a información semántica.

En el tercer paso se extraen características para cada uno de los grupos de tweets formados. Las características obtenidas son las siguientes:

1. *Aspectos temporales* que capturan el tiempo de ocurrencia y la duración de los eventos.
2. *Aspectos generativos* que corresponden a los metadatos de los tweets. Por ejemplo, `Tweeter-ID` y hora de creación del tweet.
3. *Aspectos espaciales* que definen la ubicación de los eventos.
4. *Aspectos descriptivos* que corresponden a la información acerca de los eventos. Por ejemplo, el tipo de evento.

Para extraer la ubicación contenida en un tweet se utiliza NER y extracción basada en un vocabulario de conceptos.

El cuarto paso consiste en agrupar los eventos parciales tomando en cuenta que si dos eventos parciales pertenecen al mismo evento su contexto, ubicación, tiempo de aparición y propiedades descriptivas deben coincidir.

En el trabajo presentado por Agarwal et al. [11] se utiliza NER para obtener entidades correspondientes a tiempo, lugar y descripción de los eventos reportados; estas entidades posteriormente son utilizadas para agrupar eventos parciales. Un proceso similar se utiliza en esta tesis (ver capítulo 5), en la cual se obtienen características correspondientes a tiempo, lugar y descripción de los reportes; utilizando estas características, se realiza la detección de duplicados entre dos reportes. El método de obtención de entidades de lugar presentado en la tesis es más sencillo que el método presentado por Agarwal et al. [11]. En su método la agrupación de tweets reportando el mismo evento se realiza imponiendo una restricción que indica que los tweets que reportan el mismo evento deben aparecer dentro de un rango de 24 horas,

mientras que en la tesis dos reportes duplicados pueden aparecer dentro de un rango de tiempo de unos pocos minutos, un par de horas, varios días y en ocasiones muy raras incluso uno o dos meses.

DETECCIÓN DE DOCUMENTOS CASI DUPLICADOS

En el trabajo realizado por Broder [14] se presenta un método de detección de documentos casi duplicados (documentos muy similares); este método compara el contenido de pares de documentos y de acuerdo al porcentaje del contenido compartido por ambos documentos, éstos se clasifican como duplicados o no duplicados. A continuación se describe este método.

En primera instancia, el contenido de los documentos es reducido a una forma canónica, eliminando signos de puntuación, substituyendo letras mayúsculas por letras minúsculas, eliminando el formato del texto, etc. Una vez obtenida la forma canónica, del documento se extraen subcadenas de texto de un largo definido (n -gramas) y el documento es asociado con el conjunto de n -gramas obtenidos. Posteriormente para, cada uno de los n -gramas se calcula un identificador numérico (*firma*) mediante un proceso similar a la aplicación de una función *hash* a los n -gramas. A partir de las firmas obtenidas se crea una representación reducida del documento conocida como *bosquejo*.

Después de reducir cada documento a su bosquejo correspondiente, se agrupan los bosquejos mediante el uso de los algoritmos de *ordenamiento por mezcla* [36] y *unión-buscar* [36]; los grupos resultantes están formados por bosquejos con cierto grado de similitud. En los grupos creados se realiza una comparación entre pares de bosquejos de cada grupo y se clasifican como duplicados a aquellos bosquejos que superan un valor de umbral al ser comparados.

Una similitud encontrada entre el método presentado por Broder [14] y el propuesto en esta tesis es la creación de una representación de los documentos originales para facilitar el proceso de detección de documentos muy similares. Pero mientras

que Broder utiliza únicamente el texto contenido en los documentos para generar las representaciones, nuestro método incorpora información correspondiente a lugares y tiempos contenida en los reportes como se explica en el capítulo 5.

CONCLUSIÓN

Los trabajos relacionados mostrados presentan una o varias de las características principales del sistema de detección de duplicados propuesto. Para los trabajos que utilizan reportes de Twitter se aplican técnicas de NER, agrupamiento y clasificación supervisada. También se aplican medidas de similitud y el método de ponderación *tfidf*. En el trabajo restante se presenta una representación estructurada del documento original, a la que se le puede aplicar los algoritmos de detección de duplicados de forma más eficiente que al documento original.

CAPÍTULO 4

METODOLOGÍA

En este capítulo se explica el propósito de cada uno de los procesos que forman el sistema de detección de documentos duplicados mostrado en la figura 4.1, comenzando con el proceso de *extracción de la información* y terminando con el proceso de *detección de documentos duplicados*. También se explica la diferencia entre los *procesos dependientes del tipo de documentos procesados* y los *procesos independientes del tipo de documentos procesados*.

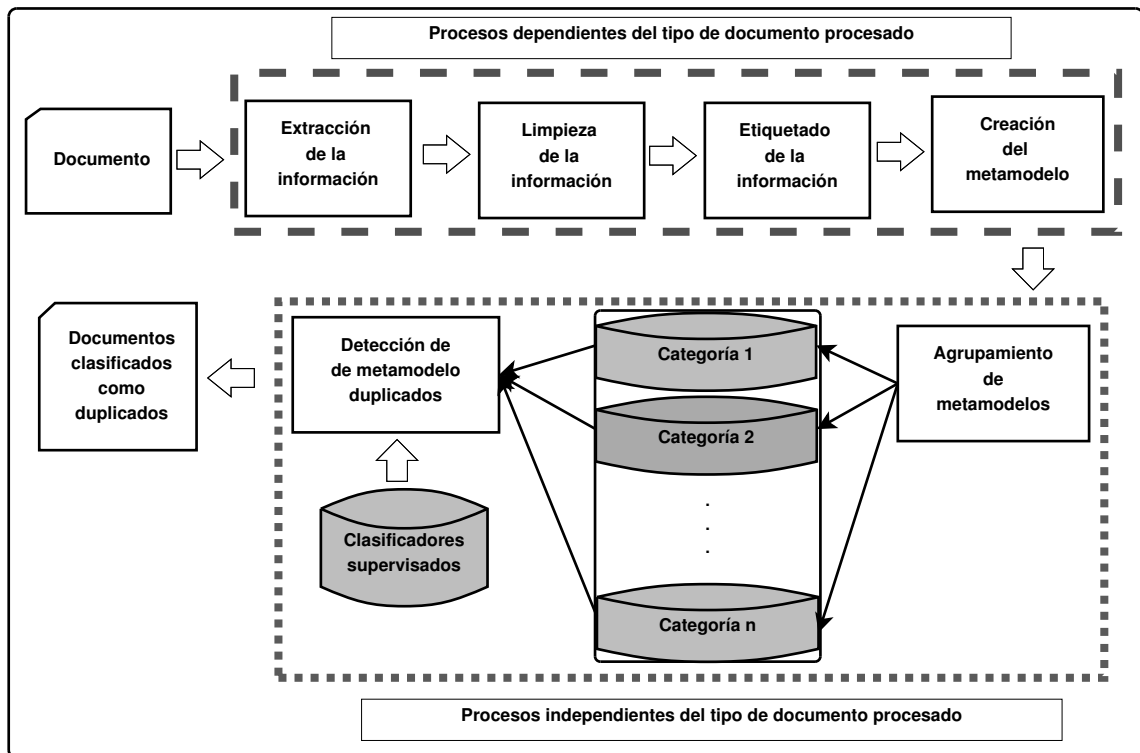


Figura 4.1 – Sistema propuesto de detección de documentos duplicados.

4.1 PREPROCESAMIENTO DE DOCUMENTOS

El preprocesamiento se encarga de obtener la información relevante de un documento y estructurarla de tal manera que luego pueda ser analizada. Durante el preprocesamiento de documentos se realizan los siguientes procesos: extracción de la información, limpieza de la información, etiquetado de la información y creación de un metamodelo.

4.1.1 EXTRACCIÓN DE LA INFORMACIÓN

Este proceso se encarga de extraer el texto (información) contenido en un documento. La manera de extraer el texto del documento depende del tipo de documento que se quiere procesar. A continuación se presentan algunos ejemplos de extracción de información en documentos de distintos formatos:

- Para extraer el contenido de un documento en formato PDF (figura 4.2a) se convierte el documento PDF a un documento de texto plano (.txt) y se extrae el contenido del documento de texto plano.
- Para extraer el contenido de un documento en formato CSV (figura 4.2b), basta con abrir el archivo .csv y obtener la información de cada una de las columnas separadas por coma (“,”).
- Para documentos en formato XML se puede utilizar un procesador que use la interfaz DOM (por sus siglas en inglés “Document Object Model”) para acceder al contenido del documento.

De un documento se debe extraer únicamente la información necesaria. Por ejemplo, para el caso del documento mostrado en la figura 4.2b, la información de interés puede estar contenida en las dos primeras columnas y por lo tanto sólo se extrae el contenido de esas dos columnas.

we show an efficient algorithm for computing this curve. Finally, we also note differences in the two types of curves are significant for algorithm design. For example, in PR space it is incorrect to linearly interpolate between points. Furthermore, algorithms that optimize the area under the ROC curve are not

Veracruz ,71699 ,7270413
Oaxaca ,93952 ,3521715
Campeche ,57924 ,791322
Durango ,123181 ,1547597
Coahuila ,151571 ,3055395

(a) Documento en formato PDF.

(b) Documento en formato CSV.

Figura 4.2 – Contenido de documentos en dos diferentes formatos.

4.1.2 LIMPIEZA DE LA INFORMACIÓN

Este proceso se encarga de limpiar el texto obtenido de un documento. Algunas de las operaciones que se pueden aplicar a un texto durante este proceso son:

- Eliminación de palabras vacías del texto.
- Eliminación de símbolos vacíos del texto.
- Sustitución de abreviaturas comunes en el texto por las palabras originales.
- Sustitución de palabras comunes en el texto por sinónimos.
- Eliminación de acentos en las palabras del texto.

Durante este proceso no es necesario aplicar todas las operaciones antes mencionadas. De acuerdo al tipo de documentos ingresados al sistema, se seleccionan cuales de las operaciones mencionadas tienen que aplicarse. También pueden aplicarse operaciones de limpieza diferentes a las mencionadas.

La lista de palabras vacías utilizadas puede variar de acuerdo al tipo de documentos ingresados al sistema. Por ejemplo, para textos en inglés y para textos en español se usan dos diferentes listas de palabras vacías. La lista de símbolos vacíos también varía de acuerdo al tipo de documentos ingresados al sistema.

4.1.3 ETIQUETADO DE LA INFORMACIÓN

En este proceso se aplica NER sobre un texto limpio, asignando a cada una de las palabras una etiqueta (ver sección 2.3). Las etiquetas se definen de tal manera que capturen la información necesaria para que el sistema sea capaz de realizar la detección de documentos duplicados. El número y el nombre de las etiquetas son definidas de acuerdo al tipo de información contenida en los documentos que ingresan al sistema. Por ejemplo, para documentos que contienen información médica se pueden definir las etiquetas `paciente`, `medico`, `sintoma`, `medicamento` e `irrelevante`, mientras que para documentos que contienen información de quejas de usuarios por productos fallidos se pueden definir las etiquetas `usuario`, `producto`, `empresa`, `fecha`, `queja` e `irrelevante`.

La etiqueta `irrelevante` se utiliza para etiquetar las palabras que no pertenecen a ninguna de las entidades importantes en el texto; como los signos de puntuación, los artículos, adjetivos, etc. Es muy común utilizar la letra `O` para representar la etiqueta irrelevante; en los trabajos presentados en [18, 31] se muestra el uso de la etiqueta `O`.

Existen implementaciones de NER disponibles en diferentes lenguajes de programación. Por ejemplo, para el lenguaje de programación Java existe la librería Stanford NER [9] y para Python la librería NLTK [5]. Independientemente de la implementación de NER que sea utilizada, se necesita un conjunto de palabras etiquetadas para entrenar los parámetros del etiquetador. Una manera de formar dicho conjunto es creando un archivo que contiene una palabra y su correspondiente etiqueta por cada fila (ver sección 2.3.3), siguiendo la misma convención que se utiliza para crear el conjunto de datos de entrenamiento de un NER para la sexta versión de CoNLL (por sus siglas en inglés “Conference on Computational Natural Language Learning”) en español [1].

```
(semaforo, SUCESO), (no, descompuesto), (funciona, SUCESO)
(lleva, 0) (desde, 0), (las, 0), (12pm, HORA), (en, 0)
(la, 0) (ave., LUGAR), (Juarez, LUGAR) (y, 0), (padre, LUGAR)
(mier, LUGAR), (favor, 0) (de, 0) (arreglar, 0)
```

Figura 4.3 – Texto etiquetado utilizando las etiquetas SUCESO, LUGAR, HORA y 0.

4.1.4 CREACIÓN DEL METAMODELO

La creación de un metamodelo consiste en obtener una representación estructurada del contenido etiquetado de un documento. La manera en que se crean los metamodelos se explica a continuación.

A partir del contenido etiquetado de un texto, para cada una de las etiquetas (sin incluir la etiqueta irrelevante) se crea una cadena de texto conocida como el *contenido de la etiqueta*. Esta cadena se forma concatenando las palabras que tienen la misma etiqueta. Por ejemplo, para el texto etiquetado mostrado en la figura 4.3 se obtienen las siguientes cadenas de texto: **semaforo no funciona**, **12pm** y **ave. Juarez padre mier**.

Después, utilizando la estructura mostrada en la figura 4.4a o la estructura mostrada en la figura 4.4b se crea el metamodelo con los contenidos de etiqueta obtenidos. Por ejemplo, para el texto de la figura 4.3 se obtiene el metamodelo en

```
<?xml version="1.0" encoding="ISO-8859-1">
<metamodelo>
  <etiqueta1>contenido</etiqueta1>
  <etiqueta2>contenido</etiqueta2>
  .
  .
  .
  <etiquetan>contenido</etiquetan>
</metamodelo>
```

(a) Estructura de un metamodelo en formato XML.

```
{
  "metamodelo":{
    "etiqueta1":"contenido",
    "etiqueta2":"contenido",
    .
    .
    .
    "etiquetan":"contenido"
  }
}
```

(b) Estructura de un metamodelo en formato JSON.

Figura 4.4 – Estructura de metamodelos en formato XML y JSON.

formato XML mostrado en la figura 4.5a y el metamodelo en formato JSON mostrado en la figura 4.5b.

La creación del metamodelo depende del tipo de documento que se procesa en el sistema. En la figura 4.6 se muestran documentos con formatos PDF, HTML, XML Y JSON, para los cuales se crean sus respectivos metamodelos. Para cada tipo de documento se realizan los procesos de extracción de la información, limpieza de la información, etiquetado de la información y creación del metamodelo, tomando en consideración el formato y contenido de los documentos.

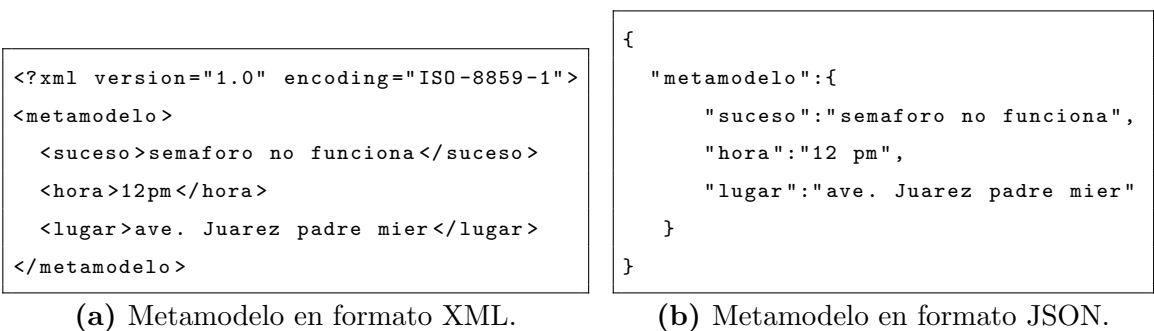


Figura 4.5 – Metamodelos generados a partir un texto etiquetado con las etiquetas SUCESO, LUGAR, HORA y O.

Los metamodelos permiten aplicar los mismos algoritmos de detección de duplicados a documentos con formatos diferentes; ésta es la principal motivación para el uso de los metamodelos. Cuando se quiere trabajar con un nuevo tipo de documento en el sistema de detección de duplicados, sólo se tiene que crear su respectivo metamodelo utilizando los métodos de procesamiento mostrados en la figura 4.6.

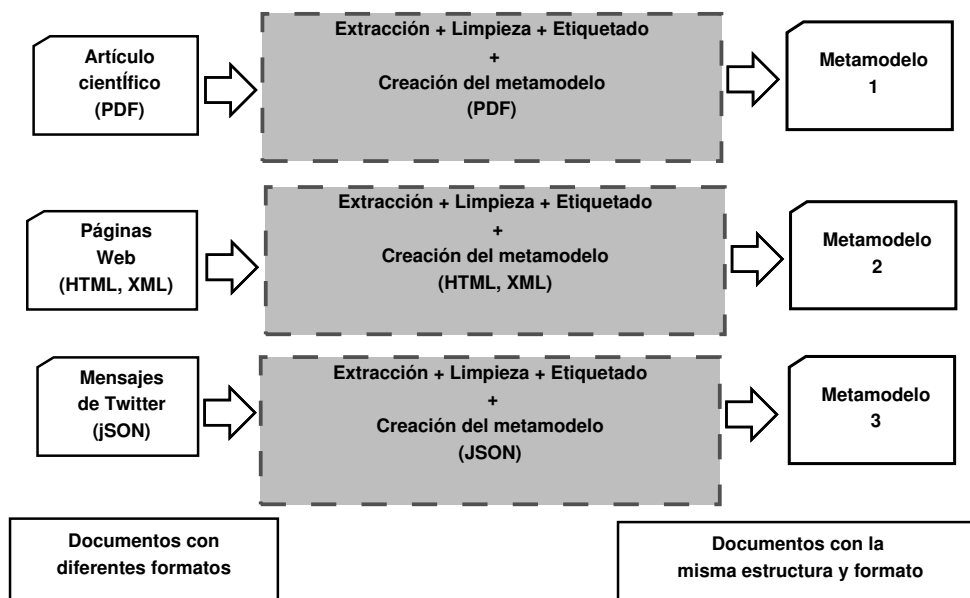


Figura 4.6 – Generación de un metamodelo a partir de documentos en diferentes formatos.

4.2 AGRUPAMIENTO DE METAMODELOS

El agrupamiento de metamodelos se encarga de juntar un nuevo metamodelo que ingresa al sistema de detección de duplicados con los metamodelos que ingresaron antes al sistema. Al trabajar con metamodelos este proceso es independiente del tipo de documentos ingresados al sistema de detección de duplicados. Lo mismo aplica para los demás procesos que trabajan con metamodelos.

El agrupamiento crea conjuntos de metamodelos similares, con la finalidad de reducir el número de metamodelos contra los cuales se tiene que comparar un nuevo metamodelo al realizar la detección de documentos duplicados. Existen diferentes maneras de agrupar los metamodelos; a continuación se describen dos métodos de agrupamiento que pueden ser utilizados.

El primer método consiste en medir el grado de similitud que existe entre el nuevo metamodelo y cada uno de los centroides de los grupos de metamodelos. Para después asignar el metamodelo al grupo que pertenece el centroide con el que se

obtuvo un mayor grado de similitud. Para este método existen dos variantes:

- Tener un número fijo de grupos a los cuales pueden ser asignados los metamodelos.
- Tener un número inicial de grupos y crear un nuevo grupo por cada nuevo metamodelo que no pertenece a ningún grupo, en el trabajo realizado por Sankaranarayanan et al. [35] utilizan este enfoque.

El segundo método consiste en utilizar un clasificador supervisado para asignar el metamodelo a una categoría que contiene metamodelos similares.

4.3 ENTRENAMIENTO DE CLASIFICADORES

En este proceso se lleva a cabo el entrenamiento de los clasificadores utilizados para realizar la detección de duplicados en los grupos de metamodelos descritos en la sección anterior. Para cada uno de los grupos se asigna un clasificador el cual es entrenado para detectar pares de documentos similares, clasificando los respectivos metamodelos como duplicados o no duplicados.

Para detectar metamodelos duplicados en cada uno de los grupos se establecen criterios de decisión tomando en cuenta la información contenida en los metamodelos. Estos criterios no necesariamente son iguales para todos los grupos, por lo tanto para cada uno de los grupos se deben establecer los criterios para decidir si dos metamodelos son duplicados. Por ejemplo, los criterios utilizados para decidir si dos mensajes de Twitter son similares [35] son diferentes a los criterios utilizados para decidir si dos páginas web son similares [29].

Estos criterios pueden establecerse de forma manual, pero con la desventaja de que al cambiar el tipo de documentos que ingresan al sistema, se tienen que volver a establecer. Por esta razón se utilizan algoritmos de clasificación (clasificadores) para realizar la detección de duplicados. Estos clasificadores se entrenan para aprender

automáticamente los criterios de decisión correspondientes a cada uno de los grupos de metamodelos. De tal forma que si cambian los tipos de documentos que ingresan al sistema de detección de duplicados, sólo hay que volver a entrenar los clasificadores del sistema.

4.4 DETECCIÓN DE METAMODELOS DUPLICADOS

En la detección de metamodelos duplicados se comparan pares de metamodelos utilizando un clasificador para buscar por duplicados. Un par de metamodelos son ingresados al clasificador correspondiente, el cual produce un resultado que indica si los metamodelos son duplicados; cuando dos metamodelos resultan ser duplicados los documentos a partir de los cuales se crearon los metamodelos también se consideran duplicados. En la figura 4.7 se ilustra la detección de duplicados utilizando un clasificador.

Cuando un nuevo metamodelo ingresa a un grupo de metamodelos se compara con los demás metamodelos del grupo. Los documentos que resultan duplicados después de comparar los metamodelos se muestran en una lista, como se observa en la figura 4.1.

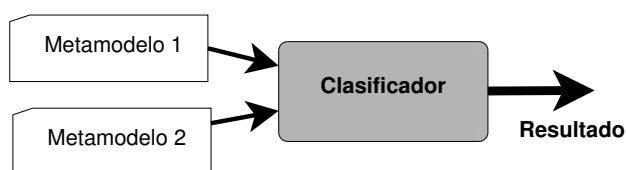


Figura 4.7 – Diagrama de un clasificador utilizado para detectar pares de documentos duplicados.

4.5 CONCLUSIÓN

En este capítulo se presentó de manera general el sistema de detección de duplicados propuesto en la tesis. Se mostró que el sistema está compuesto por dos tipos

de procesos: los que dependen de los tipos de documentos procesados y los que son independientes de los tipos de documentos procesados. Se introdujo el uso del meta-modelo para tener una representación estructurada del contenido de documentos en distintos formatos (páginas web, documentos de texto, etc.), se explicó cómo hacer para agregar documentos en nuevos formatos al sistema de detección de duplicados. Finalmente se presentaron los procesos que aplican métodos de agrupamiento y clasificación supervisada para realizar la detección de duplicados.

CAPÍTULO 5

CASO DE ESTUDIO

En este capítulo se describe la implementación del sistema de detección de duplicados para los reportes ciudadanos del CIC. Se realiza una descripción de los reportes ciudadanos y de las implementaciones de cada uno de los procesos descritos en el capítulo 4.

5.1 REPORTE CIUDADANOS DEL CENTRO DE INTEGRACIÓN CIUDADANA

El CIC proporciona un sistema en línea que permite que los ciudadanos realicen reportes sobre situaciones tales como: baches en calles y avenidas, problemas en la vialidad, situaciones de riesgo, emergencias, etc. Los reportes pueden ser generados por medio de la herramienta web Proyecto Tehuan, aplicaciones disponibles para Android e iOS o enviando un tweet a la cuenta @Cicmty.

Los reportes recibidos se muestran en la plataforma para desarrolladores CICMty-API¹ en formato JSON. En la figura 5.1 se muestra una parte de un reporte ciudadano. A continuación se explican los campos del reporte.

- **ticket:** contiene el identificador asignado a cada reporte ciudadano.
- **content:** contiene la información que describe el suceso que está siendo repor-

¹<http://www.developers.cic.mx/api/>

tado.

- **created_at**: contiene la fecha en la que fue creado el reporte ciudadano en formato JSON.
- **address_detail**: contiene los datos que forman la dirección del lugar donde ocurre el suceso que está siendo reportado, a continuación se describe el contenido de cada uno de estos datos.

```
1 {
2   "ticket": "#7YPC",
3   "content": "*ACCIDENTE* En gonzalitos altura de vuelta izquierda
4     a Insurgentes MIY #mtyfollo 17:37",
5   "created_at": "2013-08-04T17:49:56-05:00",
6   "address_detail": {
7     "formatted_address": "Gonzalitos 655, Sin Nombre de Colonia 31,
8       Monterrey, NL, México",
9     "zipcode": "64000",
10    "county": {
11      "long_name": "Monterrey",
12      "short_name": "Monterrey",
13    },
14    "state": {
15      "long_name": "Nuevo León",
16      "short_name": "Nuevo León"
17    },
18    "neighborhood": {
19      "long_name": "Centro",
20      "short_name": "Centro"
21    }
22  },
23   "group": "Vialidad y Transito (SS)",
24   "categories": ["ACCIDENTE"]
25 }
```

Figura 5.1 – Parte de un reporte ciudadano en formato JSON.

- **formatted_address**: contiene la calle, el número, la colonia, el municipio, el estado y el país.
 - **zipcode**: contiene el código postal.
 - **county**: contiene el nombre del municipio.
 - **state**: contiene el nombre del estado.
 - **neighborhood**: contiene el nombre de la colonia, lugar o zona de la ciudad donde ocurre el suceso.
- **group**: contiene el grupo al que pertenece el reporte ciudadano, un grupo está formado por una o varias categorías.
 - **categories**: contiene las categorías a las que pertenece el reporte ciudadano.

Cuadro 5.1 – Diferentes grupos y las categorías de cada grupo a las que puede pertenecer un reporte ciudadano.

Grupos	Categorías
Comunidad	Avisos, Evento público, Observador ciudadano
Emergencia	Emergencias
Propuestas ciudadanas	Propuesta comunidad, Propuesta seguridad, Propuesta servicios públicos, Propuesta vialidad
Seguridad	Incendio, Robo, Situación de riesgo
Servicios públicos	Alcantarillas, Alumbrado público, Falta electricidad, Fuga, Parques descuidados, Recolección de basura, Otros
Vialidad y tránsito	Accidente, Bache o vía dañada, Obras y/o vía cerrada, Semáforo descompuesto, Vialidad

En el cuadro 5.1 se muestran los grupos y las categorías a las que puede pertenecer un reporte ciudadano. Para este trabajo de tesis se seleccionaron reportes pertenecientes a las siguientes categorías: Accidente (acc), Alumbrado público (alu), Semáforo descompuesto (sem), Bache o vía dañada (bac), Evento público (eve), Alcantarillas (alc), Obras y/o vías cerradas (obr) y Situación de riesgo (sit).

5.2 PREPROCESAMIENTO DE REPORTES CIUDADANOS

En esta sección se realiza la implementación de cada uno de los procesos pertenecientes al preprocesamiento descrito en la sección 4.1, siendo aplicados a los reportes ciudadanos del CIC.

5.2.1 EXTRACCIÓN DE LA INFORMACIÓN

Como los reportes ciudadanos son documentos en formato JSON, se utilizó la librería `json` [4] del lenguaje de programación Python para extraer la información necesaria de los reportes. Los campos utilizados para extraer la información de los documentos son los siguientes:

- `'ticket'` (número de ticket).
- `'content'` (contenido).
- `'created_at'` (fecha de creación).
- `'address_detail': 'formatted_address'` (dirección con formato).
- `'categories'` (categorías).

En caso de que el campo de dirección con formato se encuentre vacío, se reemplaza este campo con la información contenida en `'address_detail': 'county'`, `'address_detail': 'state'` y `'address_detail': 'neighborhood'`.

5.2.2 LIMPIEZA DE LA INFORMACIÓN

La limpieza de la información extraída de los reportes ciudadanos se realiza mediante la aplicación de filtros de texto.

Para el texto presente en los campos de contenido y dirección con formato se aplican de manera secuencial los siguientes filtros:

- Se eliminan los acentos en las palabras porque algunos usuarios utilizan los acentos al escribir y otros los ignoran. Por ejemplo, las palabras **público** y **pùblico** pasan a ser **publico**.
- Si el texto contiene alguna de las abreviaturas mostradas en la figura 5.2, se sustituye el punto (.) en la abreviatura por la palabra `_dot_`. Este procedimiento se realiza porque el punto indica el final de una oración en el contenido de los reportes y se puede confundir una abreviatura con el final de una oración. Por ejemplo, la palabra `ave.` se convierte a `ave_dot_`.
- Se eliminan los enlaces que aparecen en el texto, eliminando las palabras que comienzan con `http`.
- Se usan expresiones regulares para encontrar abreviaturas de nombres y sustituir el punto por la palabra `_dot_`.
 - `r' [A-Z]\. [A-Z] [a-z]+'` para nombres como: M. Matamoros, G. Sada y P. Livas.
 - `r' [A-Z]\. ?[A-Z]\. (?:(del |de)?[A-Z] [a-z]+'` para nombres como: M. M. del Llano, M.M. de Llano y F. I. Madero.
- Se usan expresiones regulares para encontrar y sustituir los caracteres @ y - con espacios en blanco en palabras que refieren a tiempos y/o lugares.
 - `r' [?:[A-Z]+-]? [0-9] [0-9]? \. [0-9] [0-9]? @ [0-9] [0-9]? : [0-9] [0-9] ?'` para palabras como: MTY-06.15@11:46 y 06.15@11:46.
 - `r' [A-Za-z] [a-z]+ - [A-Za-z] [a-z]+'` para palabras como: Matias-Romero, Pablo-Livas y Venustiano-Carranza.
- Se eliminan de las palabras del texto los símbolos vacíos mostrados en la figura 5.3. Por ejemplo, la palabra `*ACCIDENTE*` queda como `ACCIDENTE`.
- Se eliminan las palabras encontradas por las siguientes expresiones regulares.

```

a., a.c., ant., aux., av., ave., ant., apo., b., blvd., c.,
cp., cfe., c.f.e., clin., carr., cam., col., const., c.v.,
calz., dr., don., e., esq., f., fr., frac., fracc., gza., g.,
hda., heb., h.e.b., h., ind., ing, i., izq., j., jua., l.,
lic., libr., ma., mty., m., mtro., mpal., n.l., nl., nte.,
ote., pte., profr., prof., prol., priv., p., prot., r., rep.,
s.a., s.v.t., sta., s., sto., sr., sra., sept., sec., svt.,
sc., t., trasp., u., vo., univ.

```

Figura 5.2 – Lista de las abreviaturas que aparecen con mayor frecuencia en los reportes ciudadanos.

- `r#[a-zA-Z_0-9]+` encuentra palabras como: `#1Accidente`, `#Ruta203`, `#calle_sucia`, etc.
 - `r'via @[a-zA-Z_0-9]+` encuentra palabras como: `via @CicMty`, `via svtMTY`, `via tito12`, etc.
 - `r'cc @[a-zA-Z_0-9]+` encuentra palabras como: `cc @CicMty`, `cc @colli3`, `cc @Zavala12`, etc.
 - `r'@[a-zA-Z_0-9]+` encuentra palabras como: `@DoRo4`, `@circoMty`, `@svtGPE`, etc.
- Se reemplazan espacios en blanco múltiples por espacios en blanco unitarios.
 - Se utiliza el punto final para separar las oraciones del texto. Las oraciones tienen un número máximo de palabras permitido, el cual sirve para evitar que el etiquetador tenga un mal desempeño al etiquetar oraciones muy largas. En este trabajo el número máximo de palabras permitido es 60.

Para el campo fecha de creación, se extrae el día y la hora de una representación de la fecha en formato de tiempo universal coordinado (UTC por sus siglas en inglés “Coordinated Universal Time”). Por ejemplo, de la fecha `2013-06-15T15:59:51-05:00` se extrae `2013-06-15` como el día y `15:59:51-05:00` como la hora.

Durante el proceso de limpieza de la información no se eliminan las palabras

vacías. Estas palabras proporcionan información importante para que el etiquetador detecte las palabras en el texto que refieren a lugares. Por ejemplo, es muy común encontrar las siguientes frases en los reportes; **entre Avenida P. Livas y Las Americas, en Paseo de los Leones, rumbo a Lazaro Cardenas.**

, *, :, /, (,), %, °, [,], ?, ¿, !, ¡, °, ;, -, \, ", ^

Figura 5.3 – Lista de los símbolos vacíos que son eliminados de las palabras en un texto.

5.2.3 ETIQUETADO DE LA INFORMACIÓN

El etiquetado de la información en las palabras contenidas en los reportes ciudadanos del CIC permite identificar entidades correspondientes a *lugares*, *tiempo* e *información relevante* presentes en el contenido del reporte.

Identificar correctamente los lugares mencionados en un reporte permite saber que dos reportes sobre accidentes a la misma hora son distintos porque los lugares (avenidas, calles, colonias, municipios, etc.) son diferentes. De manera similar, identificar correctamente la hora/tiempo del suceso mencionado en un reporte, permite decidir si dos reportes de un accidente en el mismo lugar refieren al mismo suceso cuando la diferencia de tiempo es de unos pocos minutos o si son dos sucesos diferentes cuando la diferencia de tiempo es de varias horas o días. Por último, identificar correctamente la información que describe el suceso reportado permite diferenciar entre accidentes de autos y reparaciones de calles en mal estado.

Para el proceso de etiquetado de información se definieron seis etiquetas, cinco para la información relevante (LOC, TIME, NAME, ORG, TTERM) y una para la información irrelevante (0) igual que en el trabajo realizado en [31]. Las etiquetas se asignan de la siguiente manera:

- La etiqueta LOC se asigna a las palabras que indican un lugar o a las palabras cuyo contexto indica un lugar. Por ejemplo, en el texto “**En la calle Juarez**

ACCIDENTE en Ave. Garza Sada sin lesionados 6:30 pm MTY NL gracias
--

Figura 5.4 – Texto limpiado antes de ser etiquetado.

hay un auto descompuesto”, la etiqueta LOC se asigna a las palabras {En, la, calle, Juarez}.

- La etiqueta TIME se asigna a las palabras que indican tiempo o a las palabras cuyo contexto indica tiempo. Por ejemplo, en el texto “Hoy a las 8:00 pm se volcó un trailer”, la etiqueta TIME se asigna a las palabras {Hoy, a, las, 8:00, pm}.
- La etiqueta NAME se asigna a las palabras que indican nombres de personas o a las palabras cuyo contexto indica el nombres de una persona. Por ejemplo, en el texto “El Sr. Juan Antonio necesita donadores de sangre”, la etiqueta NAME se asigna a las palabras {El, Sr., Juan, Antonio}.
- La etiqueta ORG se asigna a las palabras que indican nombres de organizaciones y/o empresas o a las palabras cuyo contexto indica el nombre de una organización y/o empresa. Por ejemplo, en el texto “El gobierno del estado felicita a los profesores de Nuevo León”, la etiqueta ORG se asigna a las palabras {El, gobierno, del, estado}.
- La etiqueta TTERM se asigna a las palabras que describen el suceso reportado y que además no tienen otra etiqueta asignada. Por ejemplo, en el texto “Un auto chocó en la avenida F. U. Gomez”, la etiqueta TTERM se asigna a las palabras {Un, auto, chocó}.
- La etiqueta 0 se asigna a las palabras que no tengan asignada ninguna de las otras etiquetas.

Para asignar las etiquetas definidas al texto se utilizó el algoritmo Viterbi descrito en la sección 2.3.2; los parámetros del algoritmo Viterbi son calculados como se describe en la sección 2.3.3. En la figura 5.4 se muestra un texto limpio extraído

```
[ACCIDENTE, TTERM], [en, LOC], [Ave., LOC], [Garza, LOC],  
[Sada, LOC], [sin, TTERM], [lesionados, TTERM], [6:30, TIME],  
[pm, TIME], [NL, LOC], [gracias, 0]
```

Figura 5.5 – Texto etiquetado después de haber sido limpiado.

de un reporte ciudadano y en la figura 5.5 se muestra el mismo texto después de haber sido etiquetado con las seis etiquetas definidas.

Las palabras que van a ser etiquetadas y aparecen muy pocas veces (menos de cinco veces) o que no aparecen en el archivo de parámetros de etiquetado causan que el proceso de etiquetado tenga un mal funcionamiento (en ocasiones todas las palabras se etiquetan con 0). Para evitar este problema se utilizan *pseudo-palabras*, para sustituir las palabras que aparecen con poca frecuencia o que no aparecen en el archivo de parámetros de etiquetado, este enfoque se propone en el trabajo realizado por Bikel et al. [12]. Las pseudo-palabras utilizadas en esta tesis son las siguientes:

- `_TWO_POINT_HOUR_` representa horas como 10:13:13 y 10:13.
- `_ONE_POINT_HOUR_` representa horas como 9:13:03 y 9:13.
- `_DAY_` representa fechas como 2001-09-30.
- `_NUMERIC_` representa números como 9932.
- `_ALL_CAPS_` representa palabras escritas en letras mayúsculas como MONTERREY.
- `_INIT_CAP_POINT_` representa palabras que inician con mayúscula y terminan en punto como Cd.
- `_INIT_CAP_NOPOINT_` representa palabras que inician con mayúscula como Alba.
- `_LAST_CAP_` representa palabras que terminan en mayúscula como mX.
- `_RARE_` representa palabras muy poco frecuentes que no son mapeadas a ninguna de las pseudo-palabras anteriores.

```
<?xml version="1.0" encoding="ISO-8859-1">
<metamodelo>
  <tterm>ACCIDENTE sin lesionados</tterm>
  <loc>en Ave. Garza Sada</loc>
  <time>6:30 pm</time>
</metamodelo>
```

Figura 5.6 – Metamodelo generado a partir de un texto etiquetado.

5.2.4 CREACIÓN DEL METAMODELO

La creación del metamodelo a partir de un texto etiquetado genera un documento en formato XML mediante el método descrito en la sección 4.1.4. La figura 5.6 muestra el metamodelo creado para los datos etiquetados de la figura 5.5.

Para acceder a la información contenida en los metamodelos se creó una interfaz que extrae el contenido del metamodelo y devuelve tres cadenas de texto, las cuales son `tinfo`, `tlugar` y `ttiempo`. El contenido de cada una de estas cadenas de texto se describe a continuación.

- La cadena de texto `tinfo` se forma con el texto que resulta después de limpiar el contenido de las etiquetas `TTERM`, `NAME` y `ORG`. Esta cadena corresponde al texto que describe el suceso que está siendo reportado.
- La cadena de texto `tlugar` se forma con el texto que resulta después de limpiar el contenido de la etiqueta `LOC`. Esta cadena corresponde al texto que indica el lugar del suceso.
- La cadena de texto `ttiempo` se forma con el texto que resulta después de limpiar el contenido de la etiqueta `TIME`. Esta cadena corresponde al texto que indica la fecha del suceso.

El texto guardado en cada una de las etiquetas del metamodelo contiene información innecesaria para el proceso de detección de duplicados. Por lo tanto, se

aplican filtros para limpiar el contenido de las etiquetas del metamodelo durante su extracción. Para el contenido de las etiquetas `TTERM`, `LOC`, `NAME` y `ORG` se aplican de manera secuencial los siguientes filtros:

- Las letras mayúsculas de las palabras son cambiadas por letras minúsculas para evitar tener diferentes representaciones de la misma palabra. Por ejemplo, las palabras `ACCIDENTE` y `Accidente` son convertidas a la palabra `accidente`.
- Se eliminan las palabras vacías porque no aportan ninguna información relevante al contenido del metamodelo; además durante la detección de duplicados provocan un mal desempeño de los clasificadores. La lista de palabras vacías utilizada en esta tesis se muestra en la figura 5.7.
- Cuando aparece alguna de las negaciones `sin` o `no` antes de una palabra, se crea una palabra compuesta que tiene la siguiente forma: `negación + _ + palabra`. Por ejemplo, a partir de `sin transito` se crea `sin.transito` y a partir de `no prende` se crea `no.prende`. Los mensajes contenidos en los textos `luminaria no prende arreglar` y `luminaria prende no arreglar` son diferentes, sin embargo los vectores ponderados (ver sección 2.5) creados para ambos textos son iguales. Al utilizar las palabras compuestas se obtienen los siguientes textos `luminaria no.prende arreglar` y `luminaria prende no.arreglar`; de los cuales se obtienen vectores ponderados diferentes.
- Cuando el final de un texto es `nuevo leon mexico`, se eliminan estas palabras porque no aportan información valiosa. Todos los reportes ciudadanos utilizados en este trabajo se realizaron en el estado de Nuevo León y el país de México. Por ejemplo, el texto `colonia mitras centro mty nuevo leon mexico` pasa a ser `colonia mitras centro mty`.
- Se reemplazan palabras en plural por su equivalente en singular para reducir el tamaño del diccionario de palabras utilizado para crear vectores ponderados para los documentos de texto. Por ejemplo, la palabra `accidentes` se reemplaza con `accidente`.

- Se reemplazan abreviaturas con las palabras que son abreviadas para evitar tener varias representaciones de una misma palabra. Por ejemplo, la abreviatura `aux.` se reemplaza con la palabra `auxilio` y `col.` con `colonia`.
- Se eliminan las palabras repetidas en un texto, debido a que es muy común que los nombres de avenidas, colonias y calles aparezcan varias veces. Por ejemplo, el texto `mitras centro colonia mitras centro 64000` pasa a ser `mitras centro colonia 64000`.

Para la fecha t contenida en el texto de la etiqueta `TIME` se crea una representación \hat{t} , la cual consiste en el número de minutos transcurridos desde el 1 de enero de 1970 hasta la fecha t . Por ejemplo, para la fecha correspondiente a las 13:00 horas del 21 de enero del 2013, se obtiene el número 23,132,220.

```
de, y, en, al, a, la, el, del, se, por, con, ya, nl, d, que, las
para, los, es, lo, o, q, le, h., hay, su, un, me, xk, p, m, han
hace, muy, mucho, casi, asi, como, mas, etc, aqui, #, yo, pero
todo, toda, mi, son, solo, una, ha, les, nos, +, mtyfollow, porque
si, yo, tan, sea, quiza, pq, poca, poco, pocos, ok, ni, nada, hubo
cada, les, ke, he, aun, da, varios, entre, dpersonal, intenso, leve
desde, esa, esta, ando, todos, ocasiones, habian, habia, pueda
espero, agradeceria, estamos, segun, seguido, diario, dicho, mucha
hacia, ve, deberian, ahi, parece, algo, unos, favor, tu, sobre
aprox, nunca, veo, darle, reporto, pronto, telefono, pasando
inmediato, avenida, calle, colonia
```

Figura 5.7 – Lista de palabras vacías utilizada para limpiar el contenido de los metamodelos.

5.3 AGRUPAMIENTO DE METAMODELOS

En este trabajo de tesis el agrupamiento de metamodelos se realizó de dos maneras distintas. Cada una de las maneras corresponde a un sistema de detección de

duplicados diferente. Al primer sistema se le conoce como sistema de detección de documentos duplicados híbrido y al segundo como sistema de detección de documentos duplicados supervisado.

Antes de explicar el funcionamiento de los métodos de agrupamiento se describe la creación del vocabulario de términos, el cual se utiliza al crear vectores ponderados a partir de metamodelos. Por comodidad a partir de aquí se conocerá a los métodos de agrupamiento de metamodelos como método de agrupamiento híbrido y método de agrupamiento supervisado.

Primero se crea una colección de metamodelos pertenecientes a los reportes de las ocho categorías seleccionadas en la sección 5.1

$$M = \{[m_1^{acc}, m_2^{acc}, \dots], [m_1^{alu}, m_2^{alu}, \dots], \dots, [m_1^{sit}, m_2^{sit}, \dots]\}, \quad (5.1)$$

donde m_i^{acc} es el i -ésimo metamodelo de la categoría **acc**, m_i^{alu} es el i -ésimo metamodelo de la categoría **alu** y m_i^{sit} es el i -ésimo metamodelo de la categoría **sit**. Los términos del vocabulario de términos se obtienen del contenido de **tinfo** y **tlugar** de cada uno de los metamodelos de la colección.

El método de híbrido funciona de la siguiente manera:

- Para cada metamodelo que ingresa al proceso de agrupamiento se extrae el contenido de **tinfo**, para agrupar los metamodelos de acuerdo a la información de los sucesos reportados.
- Luego se aplica el método *tfidf* a la cadena **tinfo** para crear un vector ponderado.
- Después se calcula la similitud cosenoidal entre el vector ponderado y cada uno de los centroides correspondientes a los grupos de metamodelos.
- El metamodelo se asigna al grupo que corresponde al centroide que presenta la mayor similitud con el vector ponderado.

Los centroides de los grupos de metamodelos se obtienen de la siguiente manera:

- Para cada uno de los metamodelos utilizados para crear el vocabulario de términos se extrae el contenido de `tinfo`.
- Luego se crea un vector ponderado aplicando el método *tfidf* a la cadena `tinfo`.
- Después de obtener los vectores ponderados para todos los metamodelos se aplica el algoritmo *k*-medias para crear ocho grupos de metamodelos.
- Para cada uno de los grupos creados se distingue la *categoría dominante* del grupo. Esta categoría dominante corresponde a la categoría que aporta el mayor porcentaje de metamodelos al grupo.
- Finalmente se obtienen los centroides de cada uno de los grupos creados. Un centroe se obtiene al calcular el promedio de los vectores ponderados de un grupo creado.

Cuadro 5.2 – Ejemplo de una matriz de confusión generada durante el proceso de agrupamiento de los metamodelos utilizados para la creación del vocabulario de términos; el algoritmo de agrupamiento utilizado es *k*-medias. Los porcentajes de las categorías dominantes para cada grupo se muestran en negritas.

Grupos obtenidos con el algoritmo *k*-medias

	g ₁	g ₂	g ₃	g ₄	g ₅	g ₆	g ₇	g ₈
acc	38.36	0.00	35.98	0.00	0.00	0.52	25.13	0.00
alu	12.66	0.00	0.00	0.00	86.69	0.65	0.00	0.00
sem	98.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00
bac	15.50	0.00	0.00	35.76	0.00	41.77	0.00	6.96
eve	96.00	0.00	4.00	0.00	0.00	0.00	0.00	0.00
alc	1.25	0.00	0.00	0.00	0.00	1.25	0.00	97.50
obr	42.40	0.63	0.00	0.00	0.00	55.06	0.00	1.90
sit	5.47	94.53	0.00	0.00	0.00	0.00	0.00	0.00

El cuadro 5.2 presenta una matriz de confusión que compara los metamodelos contenidos en las ocho categorías de reportes y los metamodelos contenidos en los ocho grupos creados con el algoritmo *k*-medias. En las filas de la matriz se muestra el porcentaje de metamodelos que aporta una categoría a cada uno de los ocho grupos

creados. En las columnas de la matriz se muestra el porcentaje de metamodelos que aporta cada una de las ocho categorías a un grupo creado. Por ejemplo, el grupo g_2 está formado por el 2% de los metamodelos de la categoría **sem**, 0.63% de los metamodelos de la categoría **obr** y 94.53% de los metamodelos de la categoría **sit**, por lo tanto la categoría dominante para este grupo es **sit**.

Durante el proceso de agrupamiento se utiliza un *valor de umbral* u , el cual se define como el valor mínimo del porcentaje de metamodelos que debe aportar una categoría dominante. También se realiza un conteo de los grupos para los cuales el valor de umbral es superado cu . La creación de grupos mediante el algoritmo k -medias se repite mientras el valor de cu obtenido, sea menor que el valor mínimo de cu establecido.

Con los centroides y las categorías dominantes se crea una lista de tuplas

$$l = [(cdg_1, centroideg_1), (cdg_2, centroideg_2), \dots, (cdg_8, centroideg_8)] \quad (5.2)$$

que contiene pares de categorías dominantes y centroides por cada grupo creado, donde cdg_1 corresponde a la categoría dominante del primer grupo y $centroideg_1$ es el centroide del mismo grupo.

El proceso de obtención de centroides y categorías dominantes se muestra en el algoritmo 2 y la selección del grupo correspondiente a un nuevo metamodelo que ingresa al sistema de agrupamiento híbrido se muestra en el algoritmo 3.

El método de agrupamiento supervisado es muy sencillo, para cada nuevo metamodelo que ingresa al proceso de agrupamiento se obtiene su categoría y se agrupa con otros metamodelos que tienen la misma categoría.

Algoritmo 2 Obtención de centroides y categorías dominantes utilizadas por el método de agrupamiento híbrido.

Entrada: M, u, cu

Salida: centroides lc , categorías dominantes lcd

```

1:  $lv = []$ 
2: Para  $m \in M$  Hacer
3:    $\vec{m} \leftarrow tfidf(tinfo(m))$ 
4:    $lv \leftarrow lv + \vec{m}$ 
5: Fin Para
6: bandera = Falso
7: Mientras bandera == Falso Hacer
8:    $(lc, lg) \leftarrow k-medias(lv)$ 
9:    $(n, lcd) \leftarrow categorias-dominantes(lg, M, u)$ 
10:  Si  $n > cu$  Entonces
11:    bandera  $\leftarrow$  Verdadero
12:  Fin Si
13: Fin Mientras
14: Retornar  $lc, lcd$ 

```

Algoritmo 3 Asignación del grupo correspondiente a un metamodelo mediante el método híbrido.

Entrada: metamodelo m , lista de centroides lc

Salida: id del grupo de metamodelos asignado

```

1:  $\vec{m} \leftarrow tfidf(tinfo(m)), id = 1, n = 1, sim = 0$ 
2: Para  $\vec{c} \in lc$  Hacer
3:    $nsim = SC(\vec{m}, \vec{c})$ 
4:   Si  $nsim > sim$  Entonces
5:      $sim \leftarrow nsim, id \leftarrow n$ 
6:   Fin Si
7:    $n \leftarrow n + 1$ 
8: Fin Para
9: Retornar  $id$ 

```

5.4 ENTRENAMIENTO DE CLASIFICADORES

Para realizar la detección de metamodelos duplicados se entrenan ocho clasificadores supervisados de tipo SVM, uno por categoría de reporte. Para llevar a cabo el entrenamiento se utiliza una colección de metamodelos distinta a la usada para la creación del vocabulario de términos.

Antes de explicar el proceso de entrenamiento se presenta la siguiente *medida de diferencia de tiempo*,

$$\Delta t(t_1, t_2) = 1 - \frac{1}{1 + \log_{10}(|t_1 - t_2|)}, \quad (5.3)$$

la cual mide la diferencia que existe entre el contenido de `t tiempo` de dos metamodelos donde t_1 corresponde al contenido de `t tiempo` obtenido del primer metamodelo y t_2 al contenido de `t tiempo` obtenido del segundo metamodelo.

Para cada una de las ocho categorías seleccionadas se entrena un clasificador; el proceso de entrenamiento de un clasificador se describe a continuación.

De la colección de metamodelos de entrenamiento se extraen los metamodelos que pertenecen a la categoría del clasificador a entrenar y se crea una lista de tripletas

$$le = [(m_1^1, m_2^1, dup^1), (m_1^2, m_2^2, dup^2), \dots, (m_1^{ne}, m_2^{ne}, dup^{ne})], \quad (5.4)$$

donde para la i -ésima tripleta, m_1^i y m_2^i son metamodelos diferentes y $dup^i \in [0, 1]$ es una etiqueta que toma el valor de 0 para indicar que los metamodelos son **no duplicados** y el valor de 1 para indicar que los metamodelos son **duplicados**.

Después para cada una de las tripletas contenidas en le se realiza lo siguiente:

- Se obtiene el contenido de `t info`, `t lugar` y `t tiempo` del par de metamodelos en la tripleta.
- Para el contenido de `t info` y `t lugar` de ambos metamodelos se crean vectores ponderados utilizando el método *tfidf*. Los vectores resultantes son \vec{ti}_1 y \vec{tl}_1 para el primer metamodelo y \vec{ti}_2 y \vec{tl}_2 para el segundo metamodelo.

- Se calcula la similitud cosenoidal de los vectores ponderados de **tinfo**,

$$si = SC(\vec{ti}_1, \vec{ti}_2). \quad (5.5)$$

- Se calcula la similitud cosenoidal de los vectores ponderados de **tlugar**,

$$sl = SC(\vec{tl}_1, \vec{tl}_2). \quad (5.6)$$

- Se calcula la diferencia de tiempo con el contenido **ttiempo** de ambos meta-modelos,

$$dt = \Delta t(tt_1, tt_2). \quad (5.7)$$

Con los valores de si , sl y dt calculados para cada tripleta se crea la matriz de características \mathbf{X} mostrada en la ecuación 5.8. Con los valores dup de cada tripleta se crea el vector \vec{y} mostrado en la ecuación 5.9.

$$\mathbf{X} = \begin{bmatrix} si^1, sl^1, dt^1 \\ si^2, sl^2, dt^2 \\ \vdots \\ si^{ne}, sl^{ne}, dt^{ne} \end{bmatrix}, \quad (5.8)$$

$$\vec{y} = \begin{bmatrix} dup^1 \\ dup^2 \\ \vdots \\ dup^{ne} \end{bmatrix}. \quad (5.9)$$

Utilizando la matriz \mathbf{X} y el vector \vec{y} se entrena el clasificador correspondiente a la categoría de los metamodelos utilizados. El entrenamiento de los clasificadores se lleva a cabo con la librería scikit-learn [8] del lenguaje de programación Python.

Finalmente, después de crear los clasificadores para cada una de las categorías se crea una lista de tuplas

$$lclf = [(acc, clf_{acc}), (alu, clf_{alu}), \dots, (sit, clf_{sit})], \quad (5.10)$$

donde cada tupla contiene un clasificador entrenado y la categoría correspondiente al clasificador y clf_{acc} es el clasificador correspondiente a la categoría acc .

5.5 DETECCIÓN DE DUPLICADOS

Una vez que un metamodelo es asignado a un grupo de metamodelos se inicia el proceso de detección de metamodelos duplicados, el cual se describe a continuación.

Primero se obtiene el contenido de **tinfo**, **tlugar** y **ttiempo** del nuevo metamodelo agrupado m y se crean los vectores ponderados $\overrightarrow{ti_m}$ y $\overrightarrow{tl_m}$ y la representación de tiempo tt_m . Luego se utiliza la categoría correspondiente al grupo que le fue asignado a m para seleccionar el clasificador clf de la lista $lclf$. En el método híbrido se utiliza la categoría dominante del grupo contenida en la lista l y en el método supervisado se utiliza la categoría del metamodelo. Después para cada uno de los metamodelos mg presentes en el grupo al que fue asignado m se realiza lo siguiente:

- Se obtiene el contenido de **tinfo**, **tlugar** y **ttiempo** y se crean los vectores ponderados $\overrightarrow{ti_{mg}}$ y $\overrightarrow{tl_{mg}}$ y la representación de tiempo tt_{mg} .

- Se calcula la similitud cosenoidal de los vectores ponderados de **tinfo**,

$$si = SC(\overrightarrow{ti_m}, \overrightarrow{ti_{mg}}). \quad (5.11)$$

- Se calcula la similitud cosenoidal de los vectores ponderados de **tlugar**,

$$sl = SC(\overrightarrow{tl_m}, \overrightarrow{tl_{mg}}). \quad (5.12)$$

- Se calcula la diferencia de tiempo con el contenido **ttiempo** de ambos metamodelos,

$$dt = \Delta t(tt_m, tt_{mg}). \quad (5.13)$$

- Utilizando los valores obtenidos para si , sl y dt el clasificador clf decide si los metamodelos m y mg son duplicados.

Finalmente, a partir de los metamodelos para los cuales m es clasificado como duplicado se crea una lista de reportes duplicados $ldup$. Esta lista contiene los

números de ticket de los reportes que son duplicados del nuevo reporte ingresado en el sistema. En el algoritmo 4 se muestra la obtención de los metamodelos duplicados para el nuevo metamodelo ingresado al sistema de detección de duplicados.

Algoritmo 4 Obtención de los metamodelos clasificados como duplicados, al comparar un nuevo metamodelo con cada uno de los metamodelos del grupo.

Entrada: nuevo metamodelo m , grupo de metamodelos G , categoría del grupo cat y lista de clasificadores para cada categoría $lclf$

Salida: lista de metamodelos duplicados $ldup$

- 1: $\overrightarrow{ti}_m \leftarrow tfidf(tinfo(m))$
 - 2: $\overrightarrow{tl}_m \leftarrow tfidf(tlugar(m))$
 - 3: $tt_m \leftarrow ttiempo(m)$
 - 4: $clf \leftarrow lclf[cat], ldup = []$
 - 5: **Para** $mg \in G$ **Hacer**
 - 6: $\overrightarrow{ti}_{mg} \leftarrow tfidf(tinfo(mg))$
 - 7: $\overrightarrow{tl}_{mg} \leftarrow tfidf(tlugar(mg))$
 - 8: $tt_{mg} \leftarrow ttiempo(mg)$
 - 9: $si \leftarrow SC(\overrightarrow{ti}_m, \overrightarrow{ti}_{mg})$
 - 10: $sl \leftarrow SC(\overrightarrow{tl}_m, \overrightarrow{tl}_{mg})$
 - 11: $dt \leftarrow \Delta t(tt_m, tt_{mg})$
 - 12: $dup \leftarrow clf(si, sl, dt)$
 - 13: **Si** $dup == Verdadero$ **Entonces**
 - 14: $ldup \leftarrow ldup + mg$
 - 15: **Fin Si**
 - 16: **Fin Para**
 - 17: **Retornar** $ldup$
-

5.6 CONCLUSIÓN

En este capítulo se describieron cada una de las partes que forman el sistema de detección de reportes duplicados presentado en el capítulo 4. Se mostraron las

técnicas de procesamiento de texto que utiliza el sistema de detección de duplicados, algunas de las cuales son:

- Búsqueda de palabras en una cadena de texto utilizando expresiones regulares.
- Uso de filtros para eliminar contenido no deseado de un texto (limpieza).
- Uso de NER para etiquetar palabras de acuerdo al tipo de entidad que representan.
- Representación estructura de un texto en formato XML (metamodelo).

También se mostró que dos maneras distintas de agrupar los metamodelos da lugar a dos sistemas de detección de duplicados distintos:

- Sistema de detección de duplicados híbrido, el cual realiza el agrupamiento utilizando el algoritmo k -medias.
- Sistema de detección de duplicados supervisado, el cual realiza el agrupamiento utilizando las categorías de los reportes utilizados para crear los metamodelos.

En el siguiente capítulo se muestran los resultados obtenidos de los experimentos realizados para ambos sistemas de detección de duplicados.

CAPÍTULO 6

EXPERIMENTOS Y RESULTADOS

En esta sección se explica el proceso de selección de los datos utilizados para el etiquetado, el método de detección de duplicados híbrido y el método de detección de duplicados supervisado. También se muestran y discuten los resultados obtenidos de los experimentos realizados para el etiquetado y ambos métodos de detección de duplicados y se presentan conclusiones sobre los resultados obtenidos.

6.1 EXPERIMENTOS CON ETIQUETADOR

A continuación se presentan la configuración del método de etiquetado automático utilizado en este trabajo, también se muestra el desempeño del etiquetador con un conjunto de datos de prueba y se presentan conclusiones sobre el desempeño del etiquetador.

6.1.1 CONFIGURACIÓN EXPERIMENTAL

Para poder usar el etiquetador descrito en la sección 5.2.3 se creó el archivo de parámetros de etiquetado; a continuación se explica la creación de dicho archivo.

Utilizando la plataforma CICMty-API se descargaron reportes ciudadanos a los cuales se aplicaron los métodos de extracción y limpieza de la información descritos en las secciones 5.2.1 y 5.2.2. Del contenido limpio de todos los reportes descargados se obtuvo un total de 123,583 palabras, las cuales se guardaron en un archivo de

texto (una palabra por renglón), la figura 6.1a muestra parte del archivo de texto.

A cada una de las palabras presentes en el archivo de texto se le agregó una de las seis etiquetas descritas en la sección 5.2.3 para obtener un archivo con palabras etiquetadas, la figura 6.1b muestra parte del archivo creado con palabras etiquetadas. La asignación de las etiquetas se realizó manualmente siguiendo los criterios establecidos en la sección 5.2.3.

ACCIDENTE	ACCIDENTE TTERM
en	en LOC
Bernardo	Bernardo LOC
Reyes	Reyes LOC
y	y 0
Madero	Madero LOC
,	, 0
se	se 0
envia	envia TTERM
personal	personal TTERM
SVT	SVT TTERM
16:36	16:36 TIME

(a) Palabras sin etiquetar.

(b) Palabras etiquetadas manualmente.

Figura 6.1 – Palabras obtenidas de los reportes del CIC descargados para el entrenamiento del etiquetador.

Utilizando un script de Python se calcularon los parámetros $c(u, v, s)$, $c(u, v)$, $c(u)$ y $c(s \rightsquigarrow x)$ para las palabras contenidas en el archivo de palabras etiquetadas manualmente y se creó el archivo de parámetros de etiquetado descrito en la sección 2.3.3. La figura 6.2 muestra una parte del contenido del archivo de parámetros de etiquetado.

```
339 WORDTAG LOC Constitucion
3 WORDTAG TIME 18:19
22 WORDTAG TTERM baches
6 WORDTAG 0 poco
1698 WORDTAG LOC _NUMERIC_
19672 1-GRAM TTERM
23691 1-GRAM 0
35 2-GRAM TIME TTERM
3507 3-GRAM TTERM LOC LOC
```

Figura 6.2 – Parte del contenido del archivo de parámetros de etiquetado.

Para probar la precisión del etiquetador se descargaron nuevos reportes que no habían sido procesados antes, el contenido de estos reportes fue extraído y limpiado para finalmente obtener un total de 5,099 palabras con las cuales se crearon dos archivos. El primer archivo contiene las 5,099 palabras sin etiquetar y el segundo archivo contiene las mismas 5,099 palabras etiquetadas manualmente.

6.1.2 RESULTADOS

Se probó el funcionamiento del etiquetador utilizando el archivo con las 5,099 palabras no etiquetadas, usando el etiquetador se generó un archivo con 5,099 palabras etiquetadas automáticamente. Utilizando un script de Python se compararon las etiquetas asignadas a las 5,099 palabras por el etiquetador contra las etiquetas que fueron asignadas manualmente y se obtuvo una precisión del 92%.

6.1.3 DISCUSIÓN

Como se muestra en el trabajo de Ritter et al. [33], las herramientas de reconocimiento de entidades existentes presentan un mal desempeño al aplicarse a mensajes de Twitter. Por ejemplo, en el trabajo de Klein et al. [27] se obtiene un valor de precisión 86.12% al aplicar la herramienta Stanford NER en documentos de noticias en inglés, mientras que en el trabajo de Ritter et al. se obtiene un 62% de

precisión utilizando la misma herramienta en mensajes de Twitter. Por lo tanto el nivel de precisión obtenido en las pruebas del etiquetador puede considerarse como un buen resultado.

Al revisar las etiquetas asignadas por el etiquetador a los textos etiquetados se observa que realiza un buen trabajo identificando lugares, tiempos y las descripciones de los sucesos, lo cual parte puede atribuirse a que el vocabulario utilizado en los reportes ciudadanos del CIC es muy limitado y por lo tanto el etiquetador es capaz de asignar etiquetas correctamente con mucha frecuencia. También se observa que muchos reportes parecen tener una estructura similar a la siguiente: **¿qué paso?**, **¿dónde paso?** y **¿cuándo paso?**, sin dar muchos detalles del suceso, facilitando así el trabajo del etiquetador al tener que asignar etiquetas a pocas palabras.

Revisando las equivocaciones del etiquetador se observa que se puede mejorar la precisión del etiquetado al incrementar el tamaño del conjunto de palabras etiquetadas manualmente disponibles para generar el archivo de parámetros de etiquetado. Varias etiquetas fueron mal asignadas a palabras que no se habían visto en los reportes descargados anteriormente.

Cabe mencionar que el archivo de palabras etiquetadas contra el que se comparó el resultado del etiquetador viene a ser lo que el autor de este trabajo considera los pares palabra-etiqueta correctos. Para una persona es muy fácil identificar a simple vista las partes del contenido de los reportes ciudadanos que indican lugares, tiempos, nombres de personas, nombres de organizaciones y descripciones del suceso. Por lo tanto se espera que si las palabras son etiquetadas por otras personas, las etiquetas sean asignadas de manera muy similar.

6.2 DETECCIÓN DE DUPLICADOS

Se realizaron pruebas a los sistemas de detección de duplicados híbrido y supervisado. A continuación se presentan las configuraciones realizadas para llevar a cabo

las pruebas; posteriormente se presentan los resultados obtenidos y se discuten los mismos. Finalmente se presentan conclusiones obtenidas a partir de los resultados de las pruebas realizadas.

6.2.1 CONFIGURACIÓN EXPERIMENTAL

Para ambos sistemas se utilizaron los mismos datos con la finalidad de realizar una comparación equitativa de los resultados obtenidos para cada uno de los sistemas.

CREACIÓN DEL VOCABULARIO DE TÉRMINOS

Para crear el vocabulario de términos descrito en la sección 5.3 se utilizaron 1,784 reportes diferentes a los usados en el entrenamiento y prueba del etiquetador. Estos reportes fueron descargados de la plataforma CICMty-API. Después de ser descargados el contenido de cada uno de los reportes fue extraído y limpiado utilizando los métodos descritos en las secciones 5.2.1 y 5.2.2. Posteriormente los contenidos preprocesados fueron etiquetados utilizando el etiquetador descrito en la sección 5.2.3 en la sección 5.4, se usó el archivo de parámetros del etiquetador que fue creado para probar la eficiencia del etiquetador. Finalmente, con los contenidos etiquetados de cada reporte se crearon los archivos de metamodelos correspondientes a cada reporte; a partir de estos metamodelos se creó el vocabulario de términos. Estos mismos metamodelos se usaron para crear los grupos utilizados por el método híbrido para realizar la detección de duplicados.

CREACIÓN DE METAMODELOS ARTIFICIALES

Para probar el desempeño de los dos sistemas de detección de duplicados se crearon metamodelos artificiales. Estos metamodelos fueron creados mediante la modificación del contenido de algunos metamodelos (originales) creados a partir de reportes ciudadanos. El proceso de creación de los metamodelos artificiales se describe

```
<metamodelo>
  <loc>
    En Olivo y Ruiz Cortines 1805 Moderna 64530 Monterrey NL Mexico
  </loc>
  <time>
    2013-06-20 12:03:07
  </time>
  <tterm>
    ACCIDENTE Transito de Monterrey acude al lugar
  </tterm>
</metamodelo>
```

Figura 6.3 – Metamodelo original.

a continuación.

Se descargaron 201 reportes de la plataforma CICMty-API diferentes a los usados por el etiquetador y el vocabulario de términos, de los cuales se obtuvieron 201 metamodelos usando los mismos procesos que se utilizaron con los reportes descargados para el vocabulario de términos. Para cada uno de los metamodelos creados se generaron dos metamodelos duplicados y dos metamodelos no duplicados obteniendo así un total de 1,005 metamodelos.

Los metamodelos artificiales fueron generados manualmente debido a que se buscó asegurar que los metamodelos generados como duplicados realmente fueran similares en la información contenida y los metamodelos generados como no duplicados realmente fueran distintos en la información contenida. Los cambios realizados a los metamodelos originales para crear los artificiales fueron modificaciones en la fecha (diferentes horas, días, meses), el lugar (diferentes avenidas, colonias, municipios) y el suceso reportado (diferentes sucesos) en los metamodelos originales. Los campos de nombres de personas y nombres de organizaciones no se modificaron, debido a que muy pocos de los 201 metamodelos originales tuvieron dicha información.

Para un metamodelo original y un metamodelo artificial duplicado las diferencias y similitudes de contenido se describen a continuación. En la figura 6.3 se muestra uno de los 201 metamodelos originales y en la figura 6.4 se muestra un metamodelo duplicado. Se observa que el contenido de lugar de ambos metamodelos

```
<metamodelo>
  <loc>
    En Olivo y Ruiz Cortines 1803 colonia Moderna Mty NL Mexico
  </loc>
  <time>
    2013-06-20 12:13:07
  </time>
  <tterm>
    ACCIDENTE unidad de SVT acude al lugar
  </tterm>
</metamodelo>
```

Figura 6.4 – Metamodelo similar al original.

refiere al mismo sitio, la diferencia de tiempo entre ambos metamodelos es de escasos diez minutos y en la descripción del suceso reportado en ambos metamodelos se hace referencia a que las unidades de tránsito han llegado a la escena del accidente, aunque en el metamodelo original se usa la palabra tránsito y en el duplicado la palabra SVT (SVT es una abreviación de Secretaría de Vialidad y Tránsito).

Para un metamodelo original y un metamodelo artificial no duplicado las diferencias y similitudes de contenido se describen a continuación. En la figura 6.5 se muestra el metamodelo no duplicado creado a partir del metamodelo de la figura 6.3. Se observa que en ambos metamodelos se hace referencia al mismo lugar, la descripción del suceso es diferente para ambos metamodelos y la diferencia en el contenido de tiempo de ambos metamodelos es de seis horas. A pesar de que los sucesos son descritos de manera diferente en ambos metamodelos la diferencia de tiempo es el principal factor que indica que el contenido de los metamodelos de las figuras 6.3 y 6.5 se refiere a dos accidentes distintos y por lo tanto ambos metamodelos no son duplicados. Se puede dar el caso de reportes que tienen el mismo lugar y la misma hora en su contenido y a pesar de que la descripción de suceso es diferente en realidad se trata del mismo suceso. Por ejemplo, dos personas reportando el mismo accidente donde una de ellas ve el accidente desde lejos y no reporta que hay heridos mientras que la otra persona está cerca del accidente y reporta que sí hay heridos.


```
<metamodelo>
  <loc>
    En avenida Olivo y Ruiz Cortines 1805 colonia Moderna Monterrey NL Mexico
  </loc>
  <time>
    2013-06-20 18:23:07
  </time>
  <tterm>
    ACCIDENTE persona atropellada
  </tterm>
</metamodelo>
```

Figura 6.5 – Metamodelo distinto al original

CREACIÓN DE CONJUNTOS DE ENTRENAMIENTO Y PRUEBA

Utilizando los 1,005 metamodelos obtenidos (metamodelos originales y artificiales) se crearon los conjuntos de entrenamiento y prueba utilizados por los clasificadores descritos en la sección 5.5. La creación de los conjuntos de metamodelos de entrenamiento y prueba se realizó de la siguiente manera:

- Los 201 metamodelos originales se agruparon por categoría.
- Utilizando la variación del método K iteraciones descrito en la sección 2.9 de cada categoría se seleccionaron la mitad de metamodelos para pertenecer al conjunto de entrenamiento y la otra mitad al conjunto de pruebas.
- Los metamodelos artificiales se agregaron al conjunto perteneciente al metamodelo original que les corresponde.

Una vez obtenidos los datos para el vocabulario de términos, el conjunto de entrenamiento y el conjunto de pruebas, se realizaron los experimentos para observar el desempeño del sistema de detección de duplicados híbrido y el sistema de detección de duplicados supervisado. A continuación se describen los experimentos realizados para cada sistema.

En el sistema híbrido se calcularon los centroides mediante el método descrito en la sección 5.3 estableciendo un valor de 53 para el umbral y 7 para los grupos que

deben superar el umbral. Luego se utilizó un ciclo para entrenar los clasificadores correspondientes a los centroides obtenidos y probar el desempeño de la detección de duplicados en múltiples ocasiones. Los metamodelos de entrenamiento y prueba usados en cada iteración del ciclo se obtuvieron mediante el método descrito en esta sección.

En el sistema supervisado se utilizó un ciclo para entrenar los clasificadores correspondientes a cada categoría y probar el desempeño de la detección de duplicados en múltiples ocasiones. Los metamodelos de entrenamiento y prueba usados en cada iteración del ciclo se obtuvieron mediante el método descrito en esta sección.

6.2.2 RESULTADOS

A continuación se muestran los resultados obtenidos en el proceso de detección de duplicados para el sistema supervisado y el sistema híbrido. Para el sistema supervisado se obtuvo una precisión promedio de 54.5 %, una exhaustividad promedio de 82.5 % y un valor-F promedio de 65.6 %. Para el sistema híbrido se obtuvo una precisión promedio de 46.9 %, una exhaustividad promedio de 84 % y un valor-F promedio de 59.6 %.

El cuadro 6.1 muestra la matriz de confusión generada durante el proceso de agrupamiento y obtención de centroides por parte del método híbrido. Las filas de la matriz corresponden a las categorías a las que pertenecen los metamodelos. Las columnas de la matriz corresponden a los grupos creados por el algoritmo de agrupamiento y de los cuales se obtuvieron los centroides.

Para ambos sistemas los resultados de precisión obtenidos se muestran en la figura 6.6, los resultados de exhaustividad obtenidos se muestran en la figura 6.7 y los resultados de valor-F obtenidos en la figura 6.8.

Cuadro 6.1 – Matriz de confusión generada durante el proceso de agrupamiento y obtención de centroides del método híbrido.

Grupos obtenidos con el algoritmo k -medias

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8
acc	26.98	0.00	0.00	0.00	73.01	0.00	0.00	0.00
alu	10.42	0.00	0.00	89.57	0.00	0.00	0.00	0.00
sem	3.25	0.00	96.75	0.00	0.00	0.00	0.00	0.00
bac	70.42	22.51	0.00	0.00	0.00	7.07	0.00	0.00
eve	5.88	0.00	0.00	0.00	0.00	0.00	0.00	94.12
alc	2.500	0.00	0.00	0.00	0.00	97.50	0.00	0.00
obr	97.45	0.00	0.00	0.00	0.00	1.91	0.64	0.00
sit	6.25	0.00	0.00	0.00	0.00	0.00	93.75	0.00

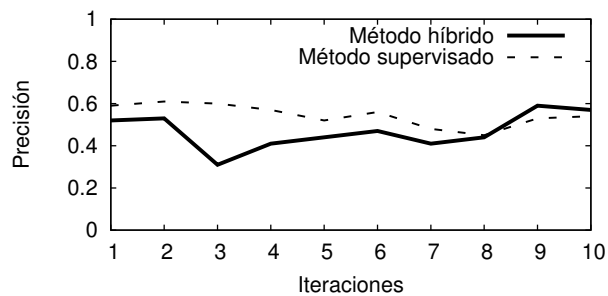


Figura 6.6 – Porcentajes de precisión obtenidos durante las pruebas de desempeño de los sistemas de detección de duplicados híbrido y supervisado.

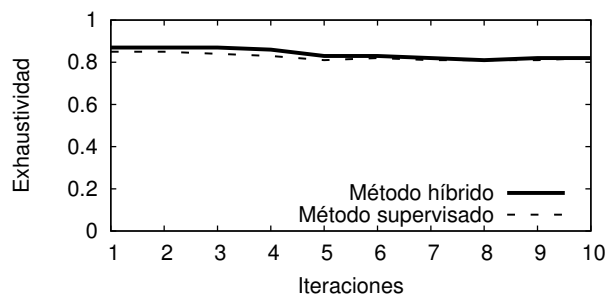


Figura 6.7 – Porcentajes de exhaustividad obtenidos durante las pruebas de desempeño de los sistemas de detección de duplicados híbrido y supervisado.

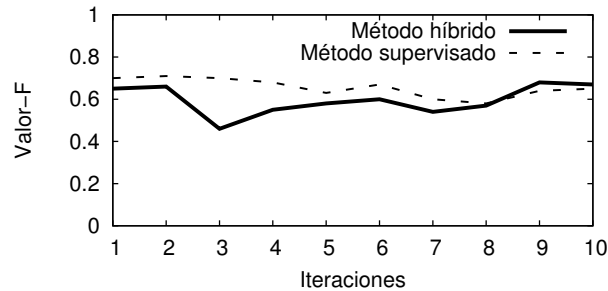


Figura 6.8 – Porcentajes de valor-F obtenidos durante las pruebas de desempeño de los sistemas de detección de duplicados híbrido y supervisado.

6.2.3 DISCUSIÓN

De los resultados obtenidos en las pruebas realizadas a los sistemas de detección de duplicados se observa que el sistema supervisado obtuvo un mejor desempeño que el sistema híbrido, los mejores valores de precisión obtenidos por ambos sistemas son cercanos al 60 % mientras que el peor valor de precisión obtenido por el sistema híbrido es de 31 % y por el sistema supervisado de 45 %.

Los resultados muestran que durante la agrupación de metamodelos y obtención de centroides por parte del sistema híbrido se crean algunos conjuntos que contienen metamodelos que pertenecen a diferentes categorías. Uno de esos conjuntos se muestra en la primera columna del cuadro 6.1; las categorías que más metamodelos aportan a este conjunto son: bache o vía dañada con 70.42 %, obras y/o vías cerradas con 97.45 %, accidentes con 26.98 % y alumbrado con 10 %. Este conjunto está formado principalmente por metamodelos correspondientes a reportes del grupo de vialidad y tránsito. Esta relación existente entre los reportes y mostrada por el sistema híbrido no se puede obtener mediante el sistema supervisado. Por lo tanto el sistema híbrido puede ser utilizado para realizar un análisis de la relación que existe entre los diferentes reportes que ingresan al sistema, mientras que el sistema supervisado se utiliza para realizar la detección de reportes duplicados.

6.3 CONCLUSIÓN

Los resultados obtenidos en las pruebas del etiquetador muestran que presenta un desempeño favorable al trabajar con los reportes del CIC. La información que describe los sucesos en estos reportes tiene una estructura parecida a los mensajes en Twitter; para este tipo de documentos las herramientas disponibles de reconocimiento de entidades presentan un mal desempeño. Además el etiquetador tiene una estructura muy sencilla, lo cual presenta una ventaja cuando se quiere trabajar con diferentes tipos de documentos.

El sistema de detección de duplicados supervisado obtuvo una precisión del 54.5%; lo cual es un resultado aceptable para el tipo de documentos con los que se trabaja. En la detección de duplicados para otro tipo de documentos los valores de precisión son mayores al 90%, sin embargo en la detección de duplicados para mensajes de Twitter los resultados son poco mayores al 65%.

El sistema de detección de duplicados híbrido presentó la capacidad de relacionar reportes que fueron agrupados en diferentes categorías pero cuyo suceso reportado es muy similar. Los reportes de las categorías de obras y/o vía cerrada y bache o vía dañada son considerados como reportes de la misma categoría por el sistema híbrido.

CAPÍTULO 7

CONCLUSIONES

Se presentó un enfoque para la detección de documentos duplicados el cual está basado en reconocimientos de entidades de un texto y aprendizaje computacional. Se trabajó con los reportes ciudadanos del Centro de Integración Ciudadana (CIC) como caso de estudio.

Del contenido de los reportes ciudadanos se extrajeron las entidades del texto correspondientes a *descripción del suceso, lugares, tiempos, nombres de personas y nombres de organizaciones*. A partir de la información relevante extraída de los reportes se crearon representaciones estructuradas en formato XML conocidas como metamodelos. Estos metamodelos se crearon con la finalidad de facilitar la aplicación de técnicas de minería de datos y aprendizaje máquina al contenido de los reportes ciudadanos.

Para comparar pares de metamodelos se utilizaron medidas de similitud para comparar las descripciones de los sucesos y los lugares; también se utilizó una medida de diferencia entre el tiempo presente en dos metamodelos.

Se utilizaron dos métodos de agrupamiento de los metamodelos para obtener conjuntos de metamodelos con información muy similar. Para detectar los metamodelos (documentos) duplicados en los grupos se utilizaron máquinas de soporte vectorial, las cuales fueron entrenadas con conjuntos de metamodelos generados artificialmente para contener duplicados.

7.1 COMENTARIOS FINALES

Una característica importante de este trabajo es el uso de entidades correspondientes a lugares, tiempos, sucesos, nombres de personas y nombres de organizaciones para realizar la detección de duplicados entre dos documentos. Para el tipo de documentos utilizados en este trabajo este enfoque resulta ser muy benéfico. Para otro tipo de documentos se podrá necesitar de otro tipo de entidades o enfoque para realizar detección de duplicados.

También cabe mencionar que al introducir el metamodelo como un elemento del sistema de detección de duplicados, se obtiene la flexibilidad de poder cambiar el formato de los documentos con los que se trabaja (conservando la misma información) sin afectar la parte de detección de duplicados del sistema.

7.2 CONTRIBUCIONES

La principal contribución de esta tesis es el diseño e implementación de un sistema de detección de documentos duplicados. El sistema propuesto contiene procesos de extracción de información, limpieza de información, etiquetado y reconocimiento de entidades, representación estructurada de la información, agrupamiento y detección de duplicados.

Del sistema de detección de duplicados cabe destacar:

- El uso del metamodelo que permite estructurar el contenido de un documento para poder aplicar técnicas de cómputo inteligente a la información del documento.
- El establecimiento de criterios de detección de duplicados entre dos documentos. Estos criterios toman en cuenta la similitud que existe en los contenidos de descripción de sucesos, los contenidos de lugar y los contenidos de tiempo de un par de documentos.

- El aprendizaje automático de los criterios de detección de duplicados, debido a que estos criterios varían de acuerdo a la información contenida en los documentos.

7.3 TRABAJO A FUTURO

Los resultados obtenidos en el sistema de detección de duplicados fueron satisfactorios, aun así los métodos y algoritmos utilizados pueden ser mejorados para obtener un mejor sistema de detección de duplicados. Como trabajo a futuro se propone:

- Llevar a cabo la resolución de las frases referidas a tiempos y lugares. Por ejemplo, detectar qué fecha es **el viernes pasado** y detectar si las palabras **en Juarez** se refieren a un municipio, una avenida, una colonia o una calle.
- Probar el uso del metamodelo para otro tipo de documentos. Por ejemplo, artículos científicos.
- Probar otras implementaciones de NER (Stanford NER, NER de la librería NLTK para Python) existentes para realizar el etiquetado.

BIBLIOGRAFÍA

- [1] Spanish Data of the CoNLL 2002 Shared Task with POS tags, 2002. <http://www.lsi.upc.edu/~nlp/tools/nerc/nerc.html>.
- [2] Proyecto-Tehuacan, 2013. <http://www.tehuacan.cic.mx/nuevo-reporte/>.
- [3] Introducing JSON, 2014. <http://www.json.org/>.
- [4] Librería para codificar y decodificar texto en formato JSON para el lenguaje de programación Python, 2014. <http://docs.python.org/2/library/json.html>.
- [5] NLTK Python, 2014. <http://nltk.org/index.html>.
- [6] Palabras vacías en español definidas en el proyecto Snowball. <http://snowball.tartarus.org/algorithms/spanish/stop.txt>, 2014.
- [7] Regular expressions operations, 2014. <http://docs.python.org/2/library/re.html>.
- [8] scikit-learn, Machine Learning in Python, 2014. <http://scikit-learn.org>.
- [9] Stanford NER, 2014. <http://nlp.stanford.edu/ner/>.
- [10] XML Tutorial, 2014. <http://www.w3schools.com/xml/>.
- [11] Puneet Agarwal, Rajgopal Vaithiyathan, Saurabh Sharma y Gautam Shroff. Catching the Long-Tail: Extracting Local News Events from Twitter. En *Proceedings of the 6th International AAAI Conference on Weblogs and Social*

- Media*, págs. 379–382, 2012. URL <https://www.aaai.org/ocs/index.php/ICWSM/ICWSM12/paper/view/4639>.
- [12] Daniel M. Bikel, Richard Schwartz y Ralph M. Weischedel. An Algorithm That Learns What’s in a Name. *Machine Learning*, 34(1-3):211–231, febrero 1999. ISSN 0885-6125. doi:10.1023/A:1007558221122. URL <http://dx.doi.org/10.1023/A:1007558221122>.
- [13] C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer-Verlag, Secaucus, NJ, EE.UU., 2006.
- [14] Andrei Z. Broder. Identifying and Filtering Near-Duplicate Documents. En *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, COM ’00, págs. 1–10, Londres, Reino Unido, 2000. Springer-Verlag. URL <http://dl.acm.org/citation.cfm?id=647819.736184>.
- [15] Christopher D. Manning, Prabhakar Raghavan y Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Nueva York, NY, EE.UU., 2008.
- [16] Kenneth Ward Church. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. En *Proceedings of the 2nd Conference on Applied Natural Language Processing*, ANLC ’88, págs. 136–143, Stroudsburg, PA, EE.UU., 1988. Association for Computational Linguistics. doi:10.3115/974235.974260. URL <http://dx.doi.org/10.3115/974235.974260>.
- [17] Jesse Davis y Mark Goadrich. The Relationship Between Precision-Recall and ROC Curves. En *Proceedings of the 23rd International Conference on Machine Learning*, ICML ’06, págs. 233–240, Nueva York, NY, EE.UU., 2006. ACM. doi:10.1145/1143844.1143874. URL <http://doi.acm.org/10.1145/1143844.1143874>.
- [18] Shipra Dingare, Malvina Nissim, Jenny Finkel, Christopher Manning y Claire Grover. A system for identifying named entities in biomedical text: How results

- from two evaluations reflect on both the system and the evaluations. *Comparative and Functional Genomics*, 6(1-2):77–85, febrero 2005. ISSN 1531-6912. doi:10.1002/cfg.v6:1/2. URL <http://dx.doi.org/10.1002/cfg.v6:1/2>.
- [19] Richard O. Duda, Peter E. Hart y David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, Nueva York, NY, EE.UU., 2000.
- [20] Roberto Esposito y Daniele P. Radicioni. Carpediem: Optimizing the viterbi algorithm and applications to supervised sequential learning. *Journal Machine Learning Research*, 10:1851–1880, diciembre 2009. ISSN 1532-4435. URL <http://dl.acm.org/citation.cfm?id=1577069.1755847>.
- [21] Jeffrey E. F. Friedl. *Mastering Regular Expressions (2nd Edition)*. O'Reilly & Associates, Inc., Sebastopol, CA, EE.UU., 2002.
- [22] Patrick A. V. Hall y Geoff R. Dowling. Approximate String Matching. *ACM Computing Surveys*, 12(4):381–402, diciembre 1980. ISSN 0360-0300. doi:10.1145/356827.356830. URL <http://doi.acm.org/10.1145/356827.356830>.
- [23] T. Hastie, R. Tibshirani y J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd Edition)*. Springer Series in Statistics. Springer, Nueva York, NY, EE.UU., 2009.
- [24] G. James, T. Hastie, D. Witten y R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Texts in Statistics. Springer, Nueva York, NY, EE.UU., 2013.
- [25] Daniel Jurafsky y James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, EE.UU., 2000.
- [26] Alexandros Karatzoglou, David Meyer y Kurt Hornik. Support Vector Machines in R. *Journal of Statistical Software*, 15(9):1–28, 4 2006. ISSN 1548-7660. URL <http://www.jstatsoft.org/v15/i09>.

- [27] Dan Klein, Joseph Smarr, Huy Nguyen y Christopher D. Manning. Named entity recognition with character-level models. En *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, págs. 180–183, Stroudsburg, PA, EE.UU., 2003. Association for Computational Linguistics. doi:10.3115/1119176.1119204. URL <http://dx.doi.org/10.3115/1119176.1119204>.
- [28] Vladimir Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.
- [29] Gurmeet Singh Manku, Arvind Jain y Anish Das Sarma. Detecting Near-duplicates for Web Crawling. En *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, págs. 141–150, Nueva York, NY, EE.UU., 2007. ACM. doi:10.1145/1242572.1242592. URL <http://doi.acm.org/10.1145/1242572.1242592>.
- [30] Natalia Ponomareva, Paolo Rosso, Ferrán Pla y Antonio Molina. Conditional Random Fields vs. Hidden Markov Models in a biomedical named entity recognition task. *Proceedings of International Conference Recent Advances in Natural Language Processing, RANLP*, págs. 479–483, 2007.
- [31] Lev Ratinov y Dan Roth. Design Challenges and Misconceptions in Named Entity Recognition. En *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, págs. 147–155, Stroudsburg, PA, EE.UU., 2009. Association for Computational Linguistics. URL <http://dl.acm.org/citation.cfm?id=1596374.1596399>.
- [32] Payam Refaeilzadeh, Lei Tang y Huan Liu. Cross-Validation. En Ling Liu y M.Tamer Özsu, editores, *Encyclopedia of Database Systems*, págs. 532–538. Springer US, 2009. doi:10.1007/978-0-387-39940-9_565. URL http://dx.doi.org/10.1007/978-0-387-39940-9_565.
- [33] Alan Ritter, Sam Clark, Mausam y Oren Etzioni. Named entity recognition in tweets: An experimental study. En *Proceedings of the Conference on Empi-*

- rical Methods in Natural Language Processing*, EMNLP '11, págs. 1524–1534, Stroudsburg, PA, EE.UU., 2011. Association for Computational Linguistics. URL <http://dl.acm.org/citation.cfm?id=2145432.2145595>.
- [34] S.M. Ross. *A first course in probability*. Pearson Prentice Hall, Upper Saddle River, NJ, EE.UU., 2010.
- [35] Jagan Sankaranarayanan, Hanan Samet, Benjamin E. Teitler, Michael D. Lieberman, y Jon Sperling. TwitterStand: News in Tweets. En *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, págs. 42–51, Nueva York, NY, EE.UU., 2009. ACM. doi:10.1145/1653771.1653781. URL <http://doi.acm.org/10.1145/1653771.1653781>.
- [36] Robert Sedgewick y Kevin Wayne. *Algorithms, 4th Edition*. Addison-Wesley, Boston, MA, EE.UU., 2011.
- [37] Ke Tao, Fabian Abel, Claudia Hauff, Geert-Jan Houben y Ujwal Gadiraju. Groundhog Day: Near-duplicate Detection on Twitter. En *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, págs. 1273–1284, Switzerland, 2013. International World Wide Web Conferences Steering Committee. URL <http://dl.acm.org/citation.cfm?id=2488388.2488499>.
- [38] S.M. Weiss, N. Indurkha, T. Zhang y F. Damerou. *Text Mining: Predictive Methods for Analyzing Unstructured Information*. Springer, Nueva York, NY, EE.UU., 2005.

FICHA AUTOBIOGRÁFICA

Jorge Alberto Cordero Cruz

Candidato para el grado de Ingeniero en Mecatrónica

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

RECONOCIMIENTO DE ENTIDADES PARA FILTRADO DE DUPLICADOS

Nací el 16 de febrero de 1990 en la ciudad de Coatzacoalcos, Veracruz, México. Soy hijo primogénito de Catalino Cordero Nolasco y Floreida Cruz Morales. A los 11 años me mudé a la ciudad de Monterrey, Nuevo León México. Realicé mis estudios de preparatoria técnica en el Conalep en la ciudad de Guadalupe Nuevo León en los años 2005-2008. Durante los años 2008-2013 realicé mis estudios como Ingeniero en Mecatrónica en la Facultad de Ingeniería Eléctrica de la Universidad Autónoma de Nuevo León. Durante el segundo y tercer año de la carrera fui becario de la Dra. Satu Elisa Schaeffer. En los dos últimos años de la carrera me desempeñé como practicante desarrollador de software en la empresa NIC México. En el cuarto año de la carrera participé en un proyecto de investigación con la Dra. Sara Verónica Rodríguez Sánchez y el Dr. Hugo Jair Escalante Balderas. En el último año de la carrera realicé mi tesis de licenciatura bajo la supervisión de la Dra. Sara Elena Garza Villarreal y la Dra. Satu Elisa Schaeffer.