Etiquetado Automático de Imágenes Digitales Mediante un Algoritmo de Ensamble Semi-supervisado

por

Heidy Marisol Marin Castro

Tesis sometida como requisito parcial para obtener el grado de

MAESTRO EN CIENCIAS EN LA ESPECIALIDAD DE CIENCIAS COMPUTACIONALES

Instituto Nacional de Astrofísica, Óptica y Electrónica

Febrero 2008

Tonantzintla, Puebla

Supervisada por:

Dr. Enrique Sucar Succar, INAOE
Dr. Eduardo Morales Manzanares, INAOE

Abstract

Content-based image retrieval is a technique that uses the visual content of images such as color, form or texture to search images from large image collections. One way to carry out content-based image retrieval consists of image annotation, that is, assigning a keyword to each object or region in the image and using these keywords to retrieve images.

Manual image annotation of a large collection of images is a complex and a subjective task, and it consumes a lot of time. An alternative approach is to automatic annotate images using machine learning techniques.

In this thesis, a new algorithm called WSA (Weighted Semi-Supervised AdaBoost) is proposed for automatic digital image annotation. WSA is based on an assemble of bayesian classifiers using a semi-supervised learning approach. The algorithm WSA uses Naive Bayes as its base classifier. A set of these is combined in a cascade based on the AdaBoost technique. However, when training the ensemble of Bayesian classifiers, it also considers the unlabeled images in each stage. These are annotated based on the classifier from the previous stage, and then used to train the next classifier. The unlabeled instances are weighted according to a confidence measure based on their predicted probability value; while the labeled instances are weighted according to the classifier error, as in standard AdaBoost.

The performance of WSA was evaluated using different databases and was compared against other classifiers like NaiveBayes, AdaBoost and the algorithm SA. In the experiments, WSA obtained better performance in the prediction of labels of the images.

Resumen

La *recuperación de imágenes por contenido* es una técnica que usa el contenido visual de las imágenes como color, forma y textura, entre otras, para guiar el proceso de búsqueda de imágenes dentro de grandes colecciones. Una de las formas para llevar a cabo la recuperación de imágenes por contenido consiste en etiquetar imágenes, asignando una palabra clave a cada uno de los objetos o regiones de la imagen y utilizar esas palabras clave para realizar búsqueda de las imágenes.

El etiquetado manual de una gran cantidad de imágenes es una tarea compleja, subjetiva y consume mucho tiempo. Un enfoque alternativo para realizar esta tarea es etiquetar automáticamente el conjunto de imágenes utilizando técnicas de aprendizaje computacional.

En este trabajo de tesis se propone un nuevo algoritmo para el etiquetado automático de imágenes digitales llamado WSA (Weighted Semi-Supervised AdaBoost). WSA está basado en un ensamble de clasificadores bayesianos bajo un enfoque de aprendizaje semi-supervisado, es decir, utiliza imágenes etiquetadas y no etiquetadas para predecir etiquetas de nuevas imágenes. En WSA un conjunto de clasificadores bayesianos son combinados en cascada en base a la técnica AdaBoost. Sin embargo, cuando el ensamble de clasificadores se entrena, éste considera las imágenes no etiquetadas en cada etapa. Estas imágenes son etiquetadas en base al clasificador de la etapa anterior y son usadas para entrenar al siguiente clasificador. Las instancias no etiquetadas son pesadas de acuerdo a una medida de confianza basada en su valor de probabilidad predictiva; mientras que las instancias etiquetadas son pesadas de acuerdo al error del clasificador, como en el algoritmo AdaBoost standard.

El desempeño del algoritmo WSA fue evaluado con diferentes bases de datos. Comparado contra otros clasificadores como el algoritmo probabilístico NaiveBayes, el algoritmo AdaBoost y el algoritmo SA, WSA obtuvo mejores resultados en la predición de etiquetas de nuevas regiones de imágenes.

Agradecimientos

Quiero agradecer al Consejo Nacional de Ciencia y Tecnología (CONACyT) por proporcionarme un apoyo financiero para llevar acabo mis estudios. Al Instituto Nacional de Astrofisica Óptica y Electrónica(INAOE) por la formación academica y los diferentes apoyos y servicios que me brindó.

Estoy muy agradecida con mis asesores, el Dr. Luis Enrique Sucar y el Dr. Eduardo Morales por su ayuda, guía, revisiones y aliciente para llevar acabo este proyecto de investigación. Me siento una persona afortunada de haber trabajado bajo su supervisión. Así también quiero agradecer al Dr. Jesús González, al Dr. José Francisco Martínez y al Dr. Manuel Montes quienes formaron mi cómite de evaluación, por sus valiosas sugerencias para mejorar mi trabajo de tesis.

A mi esposo Miguel por todo su entusiasmo y apoyo para realizar este proyecto de tesis. Ya que sin su apoyo no hubiera logrado la culminación de mi tesis.

A mi familia. Mis padres Miguel, Julia, y a mis hermanos Miguel Angel y Maribel por su amor y apoyo moral.

Gracias a dios por las infinitas bendiciones que me da día a día .....

A mi familia por su amor y apoyo.

Índice general

Resumen II

1. Introducción **1**

1.1. Recuperación de imágenes 1
1.2. Planteamiento del problema 4
1.3. Objetivos 5
1.3.1. Objetivo general 5
1.3.2. Objetivos específicos 5
1.4. Metodología de la solución propuesta 6
1.5. Estructura de la tesis 8

2. Aprendizaje computacional **10**

2.1. Introducción 10
2.2. Clasificación 12
2.2.1. Clasificación supervisada 13
2.2.2. Clasificador Bayesiano simple 14
2.2.3. Clasificación semi-supervisada 15
2.3. Clasificadores semi-supervisados 16
2.3.1. Self-Training 16
2.3.2. Algoritmo EM 16
2.4. Ensambles 17
2.4.1. Bagging 17
2.4.2. Boosting 18
2.4.3. AdaBoost 18
2.5. Ensambles semi-supervisados 19
2.5.1. Co-Training 20
2.5.2. ASSEMBLE 21
2.6. Evaluación de un clasificador 23

2.6.1. Validación simple 26
2.6.2. Validación cruzada 26
2.7. Sumario del capítulo 27

3. Etiquetado automático de imágenes **28**
3.1. Etiquetado automático de imágenes 28
3.2. Segmentación de imágenes 30
3.2.1. Técnicas de segmentación 31
3.2.2. Algoritmo Normalized Cuts 34
3.3. Extracción de características 37
3.4. Sumario del capítulo 39

4. Etiquetado automático mediante un enfoque semi-supervisado **41**
4.1. Estrategia de etiquetado 41
4.2. Pre-procesamiento 43
4.2.1. Segmentación de imágenes 43
4.2.2. Extracción de características 46
4.3. Aprendizaje semi-supervisado 47
4.3.1. Algoritmo AdaBoost Semi-supervisado (SA) 49
4.3.2. Algoritmo AdaBoost Semi-supervisado con Pesos - (WSA) 51
4.3.3. Sumario 53

5. Resultados **55**
5.1. Módulo de validación del nivel de confianza 56
5.2. Experimentos con bases de datos del repositorio UCI 56
5.3. Imágenes de Corel 60
5.4. Imágenes de las bases de datos de ImageCLEF 2007 65
5.5. Discusión de resultados 68

6. Conclusiones y trabajo futuro **70**
6.1. Sumario 70
6.2. Aportaciones 72
6.3. Conclusiones 72
6.4. Trabajo futuro 73

Referencias **78**

Índice de figuras

1.1. Esquema general de un sistema de recuperación de imágenes por contenido basado en consultas por ejemplo. 2
1.2. Esquema general de un sistema de recuperación de imágenes por contenido basado en consultas por texto. 3
1.3. Diagrama de bloques de la metodología utilizada. 7

3.1. Método basado en el histograma: (a) Imagen en escala de grises. (b) Histograma de intensidades de gris en la imagen. (c) Imagen segmentada 32
3.2. Segmentación basada en división de regiones (quad-tree): (a) Imagen original. (b) División de la imagen. (c) Segmentación quad-tree. 33
3.3. Método de detección de bordes: (a) Imagen original. (b) Detección de bordes por el método de Canny. (c) Detección de bordes por el método del filtro Laplaciano. 33
3.4. Imagen vista como un grafo completo: (a) Grafo completo no dirigido. (b) Arco entre cada par de pixeles de la imagen con un costo C_{ab} 35
3.5. (a) Imagen original. (b) Bordes de la imagen. (c) Imagen segmentada 35
3.6. Corte de un grafo: (a)Mejor corte agrupando el mayor número de pixeles. (b) Caso en donde el corte mínimo regresa una mala partición. 36
3.7. (a) Imagen 128 x 192. (b) Área de la región segmentada. (c) Área convexa de la región, (d) Perímetro de la región, (e) Eje mayor y menor de la región. 39

4.1. Diagrama de bloques de la estrategia de etiquetado. 42
4.2. Ejemplos de segmentación con Normalized Cuts. 44
4.3. Aplicación del algoritmo Normalized Cuts modificado. 45

4.4. Ejemplo: Obtención de los tres valores máximos en cada una de las bandas R, G y B. 47
4.5. Interfaz gráfica para segmentación y extracción de características. 48

5.1. Gráficas de comparación del desempeño de WSA, SA, AdaBoost y Naive Bayes para 6 bases de datos del repositorio UCI. 58
5.2. Gráficas de los intervalos de confianza al 95 % para seis bases de datos del repositorio UCI, comparando WSA, SA, AdaBoost y NaiveBayes con un sólo porcentaje de ejemplos etiquetados. . . . 60
5.3. Gráficas del desempeño del algoritmo WSA usando imágenes de la base de datos de Corel. 63
5.4. Ejemplos de imágenes difíciles de segmentar. 64
5.5. Gráficas de los intervalos de confianza al 95 % para cuatro bases de datos de Corel. 64
5.6. Imágenes de la base de datos de IAPR TC-12. 66
5.7. Imágenes de la base de datos de PASCAL 2007 VOC. 67

Capítulo 1

Introducción

1.1. Recuperación de imágenes

Con el crecimiento y proliferación de la Internet y la World Wide Web, la cantidad de imágenes digitales disponibles en las bases de datos ha crecido enormemente. Las bases de datos de imágenes digitales se estan volviendo cada vez más grandes por lo que surge la necesidad de crear sistemas de recuperación de imágenes efectivos y eficientes, usando técnicas basadas en recuperación mediante la información visual. La técnica de recuperación de imágenes por contenido consiste en usar propiedades o características del contenido visual de la imagen como forma, color, textura y relaciones espaciales, entre otras, para representar, indexar y buscar imágenes dentro de grandes colecciones [18].

La recuperación de imágenes por contenido puede llevarse a cabo de diferentes formas, una de ellas es la técnica denominada *"Consultas por ejemplo"*, la cual asume que las consultas se realizan a partir de imágenes ejemplo [28]. En este tipo de técnicas el usuario proporciona una imagen de consulta de la cual se extraen sus características visuales como color, textura o forma de algunos pixeles o regiones de la imagen y se almacenan en vectores multidimensionales llamados vectores característicos. Se calculan las similaridades y las diferencias que existen entre los vectores característicos de la imagen de consulta y los de las imágenes de la base de datos. Esta técnica no es totalmente efectiva para la recuperación de imágenes por contenido, ya que no es suficiente comparar el contenido visual a nivel global de las imágenes para obtener buenos resultados, además de que no

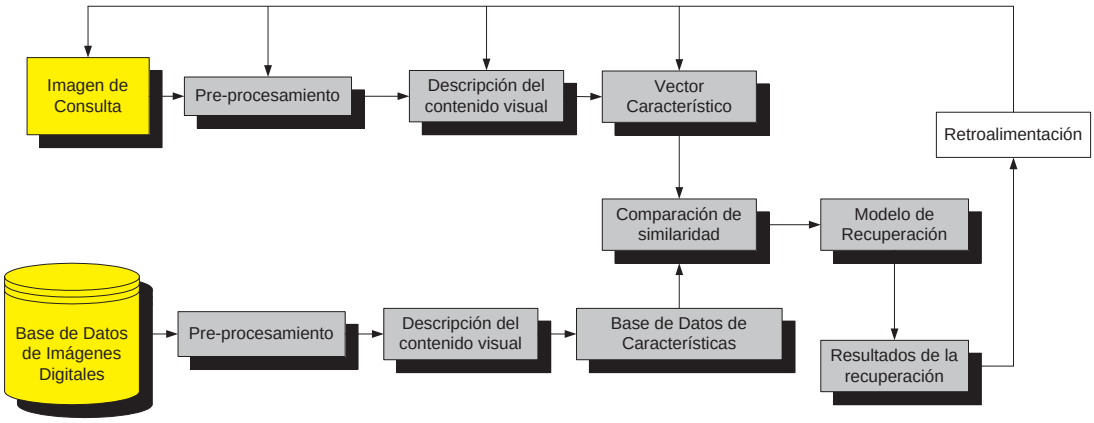

Figura 1.1: Esquema general de un sistema de recuperación de imágenes por contenido basado en consultas por ejemplo.

es la forma más natural de realizar consultas. Por ejemplo, si queremos recuperar imágenes que contengan edificios de iglesias, las características globales de las imágenes pueden ser variadas y por tanto los resultados de la consulta pueden regresar imágenes que contengan otros objetos predominantes en la imagen de consulta como árboles, gente, campo etc., en lugar de únicamente iglesias. En la figura 1.1 se muestra el diagrama de un sistema de recuperación de imágenes por contenido basado en consultas por ejemplo.

Existen también otras técnicas de recuperación de imágenes llamadas *"Consultas por texto"*, las cuales asumen que las consultas son listas de palabras clave [27]. Las búsquedas se realizan al tratar de encontrar una relación entre la palabra clave o palabras clave proporcionada(s) por el usuario y la descripción semántica o fragmentos de texto que tienen asociadas las imágenes de las bases de datos. Este tipo de técnica también puede ser poco efectiva ya que existen imprecisiones entre las descripciones semánticas que tienen las imágenes de las bases de datos y la palabra clave de consulta. Por ejemplo, supongamos que se desean encontrar imágenes que contengan un "banco" como mueble y como resultado de la búsqueda, regresa todas aquellas imágenes que contengan banco como edificio y mueble. En la figura 1.2 se muestra el diagrama general de un sistema de recuperación de imágenes por contenido basado consultas por texto.

Otra técnica para recuperar imágenes por contenido es mediante la *técnica de etiquetado de imágenes*, la cual consiste en asignar una palabra clave a cada región u objeto de la imagen. Las etiquetas son tomadas de un conjunto de clases

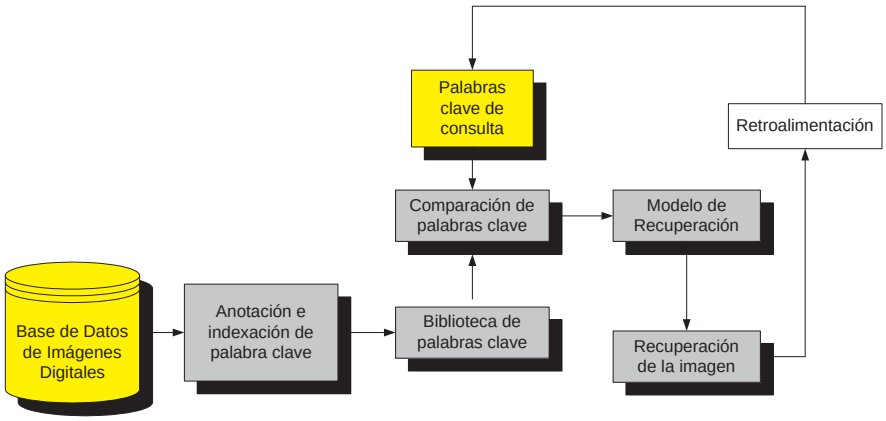

Figura 1.2: Esquema general de un sistema de recuperación de imágenes por contenido basado en consultas por texto.

predefinidas, consideradas como descriptores semánticos de alto nivel. La recuperación de imágenes se realiza a partir de las etiquetas asociadas a las imágenes. El etiquetado de imágenes puede realizarse manual o automáticamente. El etiquetado manual de imágenes es considerado como una buena técnica en términos de precisión, dado que las palabras clave son seleccionadas y asignadas de acuerdo al conocimiento de un humano. Sin embargo, el etiquetado manual es una tarea difícil, consume mucho tiempo y es subjetiva debido a que diferentes usuarios pueden considerar objetos con diferentes etiquetas en las imágenes, y de esta manera se pueden cometer errores.

El etiquetado automático de imágenes permite que las consultas puedan ser naturalmente especificadas por el usuario. Generalmente, requiere de técnicas de análisis de imágenes digitales como la segmentación de imágenes y técnicas de aprendizaje computacional, en particular, algoritmos de clasificación.

El etiquetado automático es mejor que el etiquetado manual en términos de eficiencia. Sin embargo, también puede carecer de eficacia ya que depende de otros procesos como la extracción de características y la segmentación de imágenes para poder llevarse a cabo, siendo este último un proceso no robusto e impreciso. A fin de realizar el etiquetado de manera automática, se han utilizando diferentes elementos para el reconocimiento computacional (automático) de objetos en imágenes, como búsqueda de primitivas o características visuales de bajo nivel (color, forma, textura, entre otras), así como también estrategias de aprendizaje computacional.

Se han desarrollado sistemas de etiquetado automático de imágenes, y algunos se encuentran disponibles en la web. Por ejemplo, ALIPR (Automatic Linguistic Indexing of Pictures - Real Time) es un sistema de etiquetado y búsqueda de imágenes desarrollado por investigadores de la universidad del estado de Pennsylvania. Este sistema funciona en tiempo real y maneja un vocabulario de hasta 332 palabras en inglés [25]. Otro ejemplo es Behold [42], un programa de búsqueda de imágenes a través del indexado de más de 1 millón de imágenes usando etiquetas generadas automáticamente.

1.2. Planteamiento del problema

La tarea de etiquetado de imágenes es un problema que comúnmente se presenta en el reconocimiento de objetos en las imágenes para llevar a cabo la recuperación de imágenes por contenido.

El etiquetado de imágenes es considerado como una tarea costosa debido a la gran brecha que existe entre el contenido visual de las imágenes y la asignación semántica de etiquetas. Por un lado, el etiquetado manual consume gran tiempo y está propenso a errores, además cuando se trabaja con grandes cantidades de imágenes éste resulta ser ineficiente. Por otro lado, el etiquetado automático requiere de la segmentación automática de imágenes, la cual es una tarea compleja e inexacta que aún no se ha resuelto completamente. Todo esto provoca que el proceso de etiquetado se vuelva más complicado.

Algunos enfoques para resolver el problema del etiquetado automático de imágenes han utilizado técnicas de aprendizaje computacional como la clasificación. Estas técnicas están basadas en un enfoque supervisado, en el cual se requiere que todas las imágenes que forman el conjunto de ejemplos con el cual se va a entrenar al clasificador, se encuentren previamente etiquetadas.

Otros trabajos [14, 17, 40] han utilizado técnicas de aprendizaje semi-supervisado para resolver la tarea de etiquetado automático. Dado que existe una gran cantidad de imágenes no etiquetadas disponibles, estas técnicas explotan este hecho al utilizar una gran proporción de imágenes no etiquetadas en combinación con una proporción pequeña de imágenes etiquetadas para realizar la predicción de etiquetas de nuevas imágenes.

El utilizar ejemplos no etiquetados no siempre es útil. Un problema que se puede presentar con el uso del aprendizaje semi-supervisado es que se genere un sesgo en los datos debido a que la distribución de los datos etiquetados y no etiquetados sea diferente. Otro problema que puede ocurrir es que la información que proveen los datos no etiquetados genere un modelo inadecuado.

Debido a la gran disponibilidad de imágenes y al alto costo de etiquetarlas manualmente, en este proyecto de tesis se plantea mejorar la tarea de etiquetado de imágenes de una forma automática, así como mejorar la predicción de etiquetas de nuevas imágenes mediante el uso de técnicas de aprendizaje semi-supervisado. Para reducir los problemas que pueden presentarse con el uso del aprendizaje semi-supervisado se proponen dos estrategias:

- Asignar pesos a los ejemplos no etiquetados en base a la confianza de su clase predicha, expresada por su probabilidad.
- Usar un ensamble de clasificadores, el cual es más robusto en comparación con un clasificador individual.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar un algoritmo para el etiquetado automático de imágenes utilizando un ensamble semi-supervisado, esto es, utilizar un conjunto de clasificadores conectados en cascada que exploten el uso de imágenes no etiquetadas en combinación con imágenes etiquetadas. A las imágenes no etiquetadas se les asigna un peso en base a una medida de confianza.

1.3.2. Objetivos específicos

- Segmentar imágenes y extraer características de bajo nivel como color, textura y forma.

- Diseñar e implementar un método de ensamble semi-supervisado para el etiquetado automático de imágenes usando imágenes etiquetadas y no etiquetadas.
- Definir una medida de confianza para asignar pesos a imágenes no etiquetadas de acuerdo a su valor de probabilidad predicha.
- Aplicar el algoritmo de etiquetado propuesto a un conjunto de imágenes de prueba para verificar su correcto funcionamiento y para evaluar su desempeño.

1.4. Metodología de la solución propuesta

En este trabajo se propone un algoritmo de aprendizaje semi-supervisado basado en un ensamble de clasificadores el cual explota el uso de imágenes no etiquetadas para el etiquetado automático de imágenes. Un ensamble de clasificadores es un conjunto de clasificadores cuyas decisiones individuales son combinadas para la construcción de una hipótesis de salida, la cual determina las clases o etiquetas de nuevas imágenes. La construcción del ensamble propuesto está basado en la estructura del algoritmo AdaBoost [19]. A las imágenes etiquetadas se les asigna un peso de acuerdo al error obtenido por el clasificador, como en el algoritmo AdaBoost original. A las imágenes no etiquetadas se les asigna un peso, de acuerdo a una medida de confianza basada en su valor de probabilidad predictiva.

El funcionamiento del ensamble fue evaluado mediante una variante de la técnica *k-fold-cross validation* [4] que se describe en la sección 5.1.

La metodología que se usó para el desarrollo del trabajo de tesis es la siguiente (ver la figura 1.3):

1. Realizar un estudio de algoritmos de segmentación automática existentes y seleccionar aquel algoritmo que demuestre tener un buen desempeño en términos de tiempo y precisión de segmentación.
2. Analizar y seleccionar algunos de los descriptores de características para la extracción de características de las imágenes.

3. Implementar el algoritmo de segmentación seleccionado, segmentar un conjunto de imágenes y realizar extracción de características a cada una de las regiones segmentadas.

4. Etiquetar manualmente una cantidad pequeña de imágenes para aplicar el enfoque semi-supervisado.

5. Diseñar e implementar un ensamble semi-supervisado de clasificadores bayesianos.

6. Usar los atributos de la extracción de características para entrenar el clasificador semi-supervisado y que éste sea capaz de predecir las clases de nuevas imágenes.

7. Determinar la eficacia del algoritmo propuesto utilizando un mecanismo para evaluar el nivel de confianza para clasificar nuevas imágenes no etiquetadas.

8. Realizar una comparación de los resultados arrojados por el ensamble semi-supervisado con otros algoritmos.

9. Analizar el comportamiento del ensamble usando distintos tipos de imágenes, lo que permitirá establecer criterios para mejorar la tarea de etiquetado automático de imágenes.

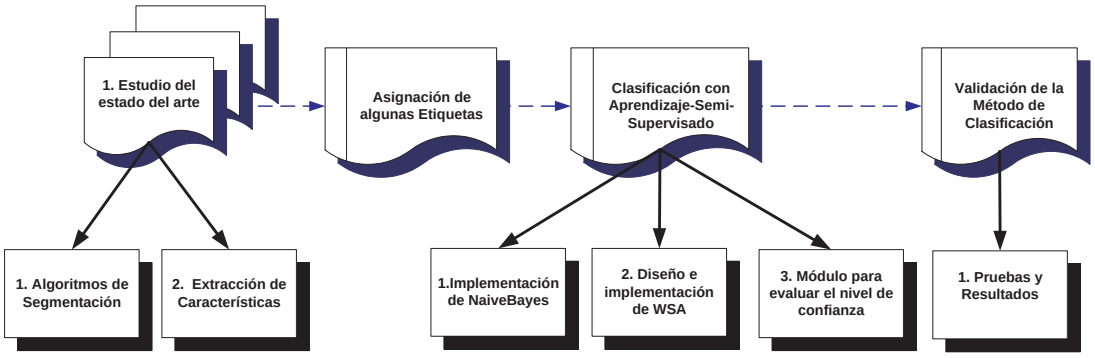

Figura 1.3: Diagrama de bloques de la metodología utilizada.

A partir de la metodología propuesta se diseñaron e implementaron dos algoritmos para resolver el problema de etiquetado automático de imágenes. El primero es un ensamble de clasificadores llamado SA (Semi-supervised AdaBoost),

el cual utiliza aprendizaje semi-supervisado manejando imágenes etiquetadas y no etiquetadas en porcentajes variables. SA maneja asignación de pesos en los datos etiquetados y no etiquetados, esto hace que sea un algoritmo eficiente en comparación con otros algoritmos como AdaBoost. Sin embargo, SA no utiliza ninguna medida de confianza que permita aumentar el peso de los ejemplos mal clasificados por el clasificador anterior del ensamble para que sean considerados por los nuevos clasificadores del ensamble. Ésto puede ocasionar que empeore la precisión de clasificación y que el modelo generado sea inadecuado. Un segundo algoritmo llamado WSA (Weighted Semi-Supervised AdaBoost) fue creado. En este algoritmo, a las imágenes no etiquetadas se le asigna un peso de acuerdo a una medida de confianza basada en la probabilidad de que la etiqueta predicha para esa imagen sea o no la etiqueta real. A las imágenes etiquetadas se les asigna un peso de acuerdo al error del clasificador como en el algoritmo original AdaBoost. Los resultados experimentales mostraron que el algoritmo WSA obtuvo mejores resultados en el nivel de predicción de nuevas etiquetas que SA y que otros algoritmos como AdaBoost y NaiveBayes.

1.5. Estructura de la tesis

En el siguiente capítulo se presentan conceptos del aprendizaje supervisado y semi-supervisado. Se expone la estructura y funcionalidad de algunos algoritmos semi-supervisados como *ASSEMBLE, AdaBoost y Re-weighting*, entre otros. También se presentan algunos de los mecanismos para evaluar el desempeño de un clasificador.

En el capítulo 3 se describe el proceso de etiquetado automático de imágenes, así como las dos tareas que se encuentran relacionadas con este proceso, la segmentación de imágenes y la extracción de características.

En el capítulo 4 se describe el diseño e implementación de la estrategia de etiquetado propuesta.

En el capítulo 5 se muestran los resultados obtenidos de las pruebas realizadas con la base de datos de imágenes Corel y con las bases de datos de imágenes de ImageCLEF2007.

Por último en el capítulo 6 se presentan las conclusiones de este trabajo de

tesis, un sumario de las aportaciones principales y las direcciones del trabajo futuro.

Capítulo 2

Aprendizaje computacional

En este capítulo se presentan algunas de las técnicas de aprendizaje computacional que se han utilizado recientemente para resolver la tarea de etiquetado automático de imágenes. Se describe el concepto de ensamble de clasificadores en el que se combina más de un clasificador simple para obtener mejores resultados en el proceso de predicción de etiquetas.

2.1. Introducción

Las técnicas de aprendizaje computacional se han utilizado frecuentemente para resolver problemas en donde se manejan grandes cantidades de datos y es necesario encontrar un patrón que permita determinar el comportamiento de éstos. Generalmente las técnicas de aprendizaje se utilizan para resolver problemas de clasificación. Recientemente se ha requerido del uso de técnicas de aprendizaje computacional para solucionar problemas de búsquedas de imágenes por contenido.

El aprendizaje computacional [4] es una rama de la inteligencia artificial cuyo objetivo es desarrollar modelos que sean capaces de aprender y generalizar comportamientos a partir de cierta información como ejemplos o experiencia previa. La funcionalidad de los modelos es extraer información implícita dentro de los datos para poder hacer predicciones y tomar decisiones sobre nuevos datos. Una definición de aprendizaje computacional dada por Tom Mitchell [11] es: "El aprendizaje automático es el estudio de algoritmos computacionales que van mejorando

automáticamente su desempeño a través de la experiencia". De manera más formal, esta definición sería [11]: "Un programa de computadora aprende a realizar una tarea T a partir de una experiencia E, si su rendimiento al realizar T, medido con P, mejora gracias a la experiencia E".

Existen varias tareas que se pueden realizar con sistemas de aprendizaje. Éstas pueden clasificarse como:

1. *Tareas de predicción*: se puede dividir en dos, clasificación y regresión o estimación.
   - La clasificación es una tarea básica en el análisis de datos y en el reconocimiento de patrones. En esta tarea, los datos son objetos caracterizados por atributos que pertenecen a diferentes clases discretas.
   - La regresión o estimación maneja clases continuas (reales). El objetivo de la regresión es inducir un modelo para poder predecir el valor de la clase dados los valores de los atributos. Por ejemplo, estimar la producción de leche que hay en un día en una determinada fábrica.
2. *Tareas Descriptivas*: son usadas para el análisis preliminar de los datos (características de los datos). Buscan derivar descripciones concisas de características de los datos, por ejemplo, medias, desviaciones estándares, entre otras.
3. *Tareas de segmentación*: tratan de buscar una separación de los datos en subgrupos o clases de acuerdo a un cierto criterio. Las clases pueden ser exhaustivas y mutuamente exclusivas o jerárquicas. En esta tarea se utilizan algoritmos de Clustering, el algoritmo EM, k-NN, centroides, SOM (Self-Organizing Maps) o Redes Kohonen, entre otras.
4. *Análisis de dependencias*: el valor de un elemento puede usarse para predecir el valor de otro. La dependencia puede ser probabilística, puede definir una red de dependencias o puede ser funcional.
5. *Detección de desviaciones, casos extremos o anomalías*: permiten detectar los cambios más significativos en los datos con respecto a valores pasados o normales y filtra grandes volúmenes de datos que son menos probables

de ser seleccionados. El problema se encuentra en determinar cuándo una desviación es significativa para ser de interés.

6. *Aprendizaje en base a experiencia*: se utiliza información y retro- alimentación de soluciones para mejorar el desempeño, como el caso del aprendizaje por refuerzo y aprendizaje de macro-operadores, entre otros.

7. *Tareas de búsqueda*: se utilizan principalmente para resolver algún problema de optimización. Involucran el uso de algoritmos genéticos, recocido simulado, ant-colony y técnicas de búsqueda local.

En esta tesis se lleva a cabo la tarea de clasificación para poder predecir las etiquetas de imágenes de forma automática. Esta tarea se describe a continuación con más detalle.

2.2. Clasificación

En los últimos años, las técnicas de clasificación han adquirido gran popularidad debido a su eficacia para resolver problemas de distinta índole como el descrubrimiento de medicamentos, en transacciones bancarias para predecir el comportamiento de cuentas de los clientes, diagnósticos médicos, predicción del clima, detección de fraudes, reconocimiento de caracteres, detección de anomalidades en los cromosomas, entre otros [4]. Esta técnica se ha utilizado también en procesos de recuperación de imágenes.

La clasificación ha sido estudiada extensamente en estadística, aprendizaje computacional, redes neuronales y sistemas expertos [29]. La clasificación consiste en asignar una de las diversas clases o categorias previamente establecidas a un objeto o fenómeno físico con la mayor presición posible. A tal objeto o fenómeno físico se le conoce como *instancia* o *ejemplo*. El objetivo de la clasificación es aprender una función $L : X \rightarrow Y$, denominada $clasificador$, donde X es el conjunto de aprendizaje y Y representa el conjunto de etiquetas o clases [35]. L establece una correspondencia entre vectores de entrada y un valor de salida, es decir, a cada instancia $x \in X$ le corresponde una única clase $y \in Y$. Además, y es nominal, es decir, puede tomar un conjunto finito de valores $y_1, y_2, ..., y_k$. Un algoritmo de aprendizaje tiene como objetivo extraer conocimiento de un conjunto

de datos y modelar dicho conocimiento para su posterior aplicación en la toma de decisiones.

La clasificación es una tarea básica en el desarrollo de este proyecto de tesis para la asignación automática de clases o etiquetas de nuevos ejemplos, que en este caso son imágenes. En las siguientes secciones se describen algunas de las características de la clasificación supervisada y semi-supervisada desde el punto de vista del aprendizaje computacional, para comprender el diseño del algoritmo de etiquetado automático propuesto.

2.2.1. Clasificación supervisada

La clasificación supervisada [29] es aquel tipo de clasificación donde se identifica el número de patrones que hay en un conjunto de datos, y los ejemplos que contienen estos patrones. Cada *instancia* está definida por un conjunto de valores llamados *atributos*. La *clase* es el atributo sobre el cual se realiza la predicción, por lo que es también denominada atributo de decisión para diferenciarla del resto de los atributos, denominados de condición. La *clase* indica la pertenencia del ejemplo o instancia de entrada a un determinado grupo de clase. Se denominan *etiquetas* de clase al conjunto o dominio de valores que la clase puede tomar. En el aprendizaje supervisado, la clase de cada instancia del conjunto de aprendizaje se conoce de antemano.

La clasificación supervisada parte de un conjunto de instancias descritas por un vector caraterístico y la clase a la que pertenece cada una de ellas [9] . A este conjunto de instancias se le conoce como *datos de entrenamiento o conjunto de aprendizaje*. Este conjunto es recolectado y agrupado en clases de acuerdo a las características que dichos ejemplos poseen. Así, utilizando este conjunto de datos de entrenamiento los clasificadores se entrenan y realizan la identificación de la clase correspondiente para nuevos ejemplos, aplicando el conocimiento adquirido y tratando de identificar sus clases con el menor error posible.

Existe una gran variedad de clasificadores supervisados [39] entre los que se encuentran los clasificadores bayesianos, redes bayesianas, redes neuronales, árboles y reglas de decisión, entre otros. En la siguiente sección se describe el funcionamiento del clasificador Bayesiano simple, el cual se utilizó como clasificador

base del ensamble desarrollado en este trabajo. El clasificador bayesiano simple fue utilizado debido a que es un clasificador sencillo de construir, permite realizar deducciones de manera muy rápida y tiene un alto nivel de precisión a pesar de su simplicidad.

2.2.2. Clasificador Bayesiano simple

Un clasificador Bayesiano simple (Naive Bayes) [4] es un algoritmo predictivo y descriptivo basado en la teoría de la probabilidad de Bayes [6]. Un clasificador bayesiano obtiene la probabilidad posterior de cada clase C_i, usando la regla de Bayes [6], como el producto de la probabilidad a *priori* de la clase por la probabilidad condicional de los atributos $E = \{E_1, E_2, ..., E_n\}$ dada una clase, dividido por la probabilidad de los atributos:

$$P(C_i|E) = \frac{P(C_i)P(E|C_i)}{P(E)} \tag{2.1}$$

El clasificador bayesiano simple asume que los atributos son independientes entre sí dada la clase, así que la probabilidad se puede obtener como el producto de las probabilidades condicionales individuales de cada atributo dado la clase:

$$P(C_i|E) = \frac{P(C_i)P(E_1|C_i)P(E_2|C_i)...P(E_n|C_i)}{P(E)} \tag{2.2}$$

Una vez calculadas las probabilidades individuales de cada atributo, el clasificador bayesiano simple regresa las probabilidades de las clases.

El clasificador bayesiano tiene la ventaja de ser sencillo de construir y entender, además de ser rápido para realizar inducciones. Sin embargo, el usar éste único clasificador en un enfoque supervisado no permite el uso de ejemplos o instancias no etiquetadas en el conjunto de entrenamiento.

En muchos dominios como el de imágenes, existe una mayor cantidad de imágenes no etiquetadas que contienen información interesante que puede ser utilizada para mejorar la clasificación. En la siguiente sección se presenta la descripción de la clasificación semi-supervisada en la que se explota este hecho al usar tanto datos etiquetados como no etiquetados para la clasificación de nuevos ejemplos.

2.2.3. Clasificación semi-supervisada

La clasificación semi-supervisada [44] es una forma especial de clasificación en donde se usa una gran cantidad de datos no-etiquetados conjuntamente con datos etiquetados para construir un mejor clasificador y de ésta manera poder maximixar el nivel de predicción de clases. Este enfoque surge a raíz de que las instancias etiquetadas son a menudo difíciles de obtener y consumen tiempo para etiquetarse. Caso contrario con los datos no etiquetados que pueden ser relativamente fáciles de obtener.

Un típico método de aprendizaje semi-supervisado consta de cuatro pasos [12]:

1. Entrenar un clasificador usando ejemplos de entrenamiento etiquetados E.
2. Etiquetar ejemplos de entrenamiento no-etiquetados E' usando el clasificador actual y obtener sus probabilidades de predicción.
3. Entrenar el clasificador con los ejemplos de entrenamiento E y E'.
4. Predecir las etiquetas de nuevos ejemplos del conjunto de prueba apartir del entrenamiento con los conjuntos E y E'.

El aprendizaje semi-supervisado ha llamado la atención recientemente en la literatura del aprendizaje computacional debido a su potencial para reducir la necesidad costosa de tener ejemplos etiquetados [14]. Aplicaciones como visión por computadora, clasificación de texto e investigación médica, son áreas donde los ejemplos no etiquetados pueden ser fáciles de obtener y resulta más fácil manejarlos junto con los ejemplos etiquetados.

En la siguiente sección se presentan algunos clasificadores representativos del aprendizaje semi-supervisado. Enfoques recientes han combinado más de un único clasificador para obtener mejores resultados de clasificación. Esta combinación se conoce como *ensamble de clasificadores*.

2.3. Clasificadores semi-supervisados

Existen diferentes algoritmos de clasificación semi-supervisada que explotan el uso de datos etiquetados en conjunto con datos no etiquetados. Algunos de los algoritmos más utilizados son los siguientes.

2.3.1. Self-Training

El funcionamiento de *Self-Training* es el siguiente [44]. Primero se entrena a un clasificador con una cantidad pequeña de ejemplos etiquetados. Después, el modelo generado se utiliza para clasificar ejemplos no etiquetados. De esta clasificación se consideran sólo aquellos ejemplos no etiquetados para los cuales su etiqueta predicha es más confiable, para formar el conjunto de entrenamiento. El clasificador se vuelve a entrenar y el proceso se repite. De ésta manera el clasificador utiliza sus mismas predicciones para aprender el mismo.

Self-Training ha sido utilizado en varias tareas de procesamiento natural del lenguaje. Yarowsky [41] usó Self-Training para desambiguar el sentido de las palabras, por ejemplo, diferenciar si la palabra "banco" significa un mueble para sentarse o un lugar en donde se realizan transacciones monetarias. En otro trabajo [32] se usó Self-Training para la identificación de sustantivos en archivos de texto.

2.3.2. Algoritmo EM

El algoritmo EM fue propuesto por Dempster, Laird y Ruin en 1997 [4]. El algoritmo EM se usa en estadística para encontrar una estimación del conjunto de parámetros del modelo, e intentar maximizar la probabilidad $P(O|\lambda)$ de generación de los datos de entrenamiento $O = \{O_0, ..., O_k\}$, a partir del modelo λ, de tal forma que la probabilidad asociada al nuevo modelo (λ^*) sea mayor o igual a la del modelo anterior.

$$P(O|\lambda^*) > P(O|\lambda)$$

El algoritmo EM puede aplicarse en muchas situaciones en las que se desea estimar un conjunto de parámetros θ que describen una distribución de proba-

bilidad subyacente, dada únicamente una parte observada de los datos completos producidos por la distribución. En general, se supone que en cada realización del experimento aleatorio se observa un parámetro z_i y que existe un parámetro oculto x_i. Sea $Z = \{z_1, z_2, ..., z_m\}$ el conjunto de datos observados en m realizaciones del experimento y $X = \{x_1, x_2, ..., x_m\}$ el conjunto de datos no observados. Sea $Y = Z \cup U$ el conjunto completo de datos. Los datos X pueden considerarse una variable aleatoria cuya distribición de probabilidad depende de los parámetros a estimar θ y de los datos observados Z. De la misma forma, Y es una variable aleatoria ya que está definida en términos de la variable aleatoria X. Se denomina h a la hipótesis actual de los valores de los parámetros θ, y h' a la hipótesis revisada que se estima en cada iteración del algoritmo EM.

En el algoritmo EM semi-supervisado se usa un clasificador NaiveBayes con los ejemplos etiquetados $L = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_m)\}$ para clasificar a los ejemplos no etiquetados $U = \{x_1, x_2, ...x_k\}$. Un siguiente modelo utiliza $L \cup U$ para crear un nuevo NaiveBayes, el cual clasifica al conjunto U. Este proceso se realiza iterativamente hasta que el resultado de clasificación comience a converger.

2.4. Ensambles

Un ensamble de clasificadores es un conjunto de clasificadores cuyas decisiones individuales son combinadas mediante algún esquema de toma de decisiones, normalmente por votación, para clasificar nuevos ejemplos [4]. Los ensambles obtienen mayor precisión de clasificación en comparación con un único clasificador [16]. A continuación se describen algunos ensambles.

2.4.1. Bagging

El ensamble de clasificadores Bagging *(bootstrap aggregating)* fue creado por Breiman en 1996 [4]. Este ensamble consiste en agregar, mediante votación simple, los resultados de varios clasificadores obtenidos de un mismo conjunto de entrenamiento mediante *bootstrap o muestreo con reemplazamiento*. Una muestra de ejemplos bootstrap se genera al muestrear uniformemente m instancias del conjunto de entrenamiento con reemplazo. Se generan T muestras, $B1, .., B_T$ y se

construye un clasificador C_i para cada muestra. Con estas muestras se construye un clasificador final C^* de todos los C_1 a C_T cuyo resultado es la salida mayoritaria de los clasificadores.

Para crear el primer clasificador C_i se construye una muestra de igual tamaño que la original, pero obtenida mediante extracción con reemplazo. Para una de las muestras, un ejemplo tiene la probabilidad de $1-(1-1/m)^m$ de ser seleccionado por lo menos una vez en las m veces que se selecciona una instancia. Para valores grandes de m esto se aproxima a $1-1/e$=63.2 %. Por lo que cada muestra tiene aproximadamente un 63 % de aparecer en los ejemplos de entrenamiento.

2.4.2. Boosting

El algoritmo Boosting fue propuesto por Freund-Shapire en 1996 y una de sus variantes más conocidas, es el algoritmo AdaBoost [19]. Al igual que Bagging, el algoritmo Boosting genera un conjunto de clasificadores y luego realiza una votación entre ellos. Sin embargo, existe una diferencia sustancial entre los dos algoritmos debido a que Bagging genera los clasificadores de manera independiente, mientras que Boosting no, ya que en éste los clasificadores se generan secuencialmente. En el algoritmo Boosting a todos los ejemplos se les asigna inicialmente un peso igual $(1/m)$, donde m es el número de instancias. Cada vez que se genera un clasificador, los pesos de los ejemplos de la muestra de entrenamiento se actualizan de acuerdo a los resultados hallados previamente por el clasificador, con el objetivo de minimizar el error esperado en las diferentes distribuciones de salida. Dado el número de muestras bootstrap T, se generan secuencialmente T muestras de entrenamiento ponderadas y se construyen T clasificadores $C_1, C_2, ..., C_T$. Finalmente se forma el clasificador C, usando el esquema de votación ponderada de cada clasificador. El peso de cada clasificador depende de su rendimiento en el conjunto de entrenamiento que se uso para construirlo.

2.4.3. AdaBoost

El método de etiquetado automático que se propone en este trabajo de tesis está basado en el ensamble AdaBoost multiclase [19]. AdaBoost es un método de

aprendizaje iterativo que combina una serie de clasificadores base usando una combinación lineal pesada para clasificar nuevos ejemplos. En cada iteración se genera un nuevo clasificador el cual trata de minimizar el error esperado asignándole más peso a los ejemplos que fueron mal clasificados, aumentando la probabilidad de que sean clasificados correctamente.

Formalmente, el algoritmo AdaBoost parte de un conjunto S de instancias etiquetadas donde a cada instancia x_i se le asigna un peso, $W(x_i)$. Se consideran N clases, donde la clase conocida de cada instancia x_i está dada por y_i. El clasificador base es C, y h_t es la hipótesis parcial de cada uno de los clasificadores base del ensamble en cada iteración t, $t \in \{1, 2, ..., T\}$. AdaBoost produce una combinación lineal de los t clasificadores base, $F(x) = \sum_{t=1}^{T} \alpha_t h_t$, donde α_t es el peso de cada clasificador. El peso es inversamente proporcional al error de cada clasificador sobre el conjunto de entrenamiento. Inicialmente los pesos son iguales para todas las instancias, y son usados para generar el primer clasificador h_1. Entonces el error e_1 de h_1 se obtiene mediante la sumatoria de los pesos de las instancias incorrectamente clasificadas. El peso $W(x_i)$ de cada instancia correctamente clasificada se disminuye mediante el factor $B_t = \frac{e_t}{1-e_t}$, y estos pesos $W(x_i)^{(t+1)} = W(x_i)^t \cdot B_t$ son usados para entrenar al siguiente clasificador. Este ciclo se repite hasta que $e_t > 0.5$ o cuando el número máximo predefinido de iteraciones se alcanza.

El ensamble AdaBoost multiclase que se muestra en el algoritmo 2.1 utiliza clasificadores en cascada, manipula los ejemplos de entrenamiento para generar múltiples hipótesis y predice resultados en base a factores probabilísticos. Algunos estudios empíricos [38] han mostrado que AdaBoost es un esquema eficiente para reducir las tazas de error de los clasificadores tales como árboles de decisión o redes neuronales.

2.5. Ensambles semi-supervisados

Los ensambles de clasificadores han demostrado tener un mejor desempeño que un único clasificador, además, en nuevos enfoques se ha utilizado el aprendizaje semi-supervisado en los ensambles con el objetivo de mejorar los niveles de precisión en la clasificación de nuevos ejemplos. En la siguiente sección se presentan algunos ensambles semi-supervisados.

Algoritmo 2.1 AdaBoost

Entrada: $S = (x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$: ejemplos etiquetados, T: iteraciones, C: clasificador base

Salida: Hipótesis Final : $H_f = argmax_{y \in Y} \sum_{t=1}^{T} (log \frac{1}{B_t}) \cdot h_t(x, y)$

1: Inicializar $W(x_i)^0 = \frac{1}{num\_Instancias(S)}$
2: **for** $t = 1$ hasta T **do**
3: Normalización $W(x_i)^t = \frac{W(x_i)}{\sum_{i=1}^{N} W_{(x_i)}}$
4: Llamar al algoritmo débil $h_t = C(S, W(x_i)^t)$
5: Cálculo del error $e_t = \sum_{i=1}^{N} W(x_i)^t$ if $h_t(x_i) \neq y_i$
6: **if** $e_t > 0.5$ **then**
7: Abortar ciclo
8: **end if**
9: $B_t = \frac{e_t}{(1-e_t)}$
10: Nuevo vector de pesos $W(x_i)^{(t+1)} = W(x_i)^t \cdot B_t$ if $h_t(x_i) = y_i$
11: **end for**

2.5.1. Co-Training

El algoritmo Co-Training es un ensamble propuesto por Blum y Mitchell en 1998 [9]. Este algoritmo asume que existen dos conjuntos de atributos independientes y compatibles para los ejemplos. Cada conjunto de atributos es suficientemente independiente para propósitos de aprendizaje y clasificación. Un clasificador que aprende de cada uno de estos conjuntos de atributos se puede usar para etiquetar datos para otro clasificador y así expandir el conjunto de entrenamiento de ambos clasificadores.

La idea principal del algoritmo Co-Training es que si dos algoritmos usan diferentes representaciones de sus hipótesis entonces pueden aprender dos modelos que pueden complementarse mutuamente, en algún momento uno de los algoritmos etiquetará algunos ejemplos no etiquetados y aumentará el conjunto de entrenamiento del otro algoritmo. Por ejemplo, si A y B son dos clasificadores, A etiqueta datos para B y B etiqueta datos para A hasta que ya no existan datos no etiquetados o hasta que ningún dato pueda ser etiquetado debido a la incertidumbre en la asignación de la etiqueta a esos datos. La decisión para etiquetar datos del otro clasificador se toma en base a técnicas estadísticas.

El conjunto de entrenamiento formado con los ejemplos etiquetados por ambos clasificadores permite al primer clasificador aprovechar la infomación acerca de la función objetivo con los ejemplos etiquetados por el segundo clasificador.

2.5.2. ASSEMBLE

El algoritmo ASSEMBLE (Adaptive Semi-Supervised Ensemble) fue desarrollado por Bennett *et al.*, en 2002 [7] para participar en la competencia *NIPS 2001 Unlabeled Data Competition*. En este algoritmo, Bennet introduce el concepto de *pseudo-clases*. La *pseudo-clase* de una instancia no etiquetada es definida como la clase predicha por el ensamble para esa instancia. La idea de ASSEMBLE es constuir un ensamble de clasificadores que trabaje en forma consistente tanto con ejemplos etiquetados como no etiquetados.

ASSEMBLE comienza con la asignación de *pseudo-clases* a las instancias en el conjunto no etiquetado para poder aplicar el algoritmo de aprendizaje semi-supervisado. Inicialmente las *pseudo-clases* son asignadas usando el algoritmo del vecino más cercano (k-NN).

Este método alterna entre la asignación de *pseudo-clases* a los ejemplos no etiquetados usando el ensamble existente y contruyendo el siguiente clasificador base usando los ejemplos etiquetados y las *pseudo-clases*. ASSEMBLE explota el uso de los ejemplos no etiquetados para reducir el número de clasificadores en el ensamble. ASSEMBLE ha utilizado árboles de decisión y redes neuronales como clasificadores base.

El algoritmo ASSEMBLE trabaja de la siguiente forma, ver el algoritmo 2.2. Parte de un conjunto de entrenamiento de datos etiquetados S y no etiquetados U y un clasificador base definido como $f_j(x)$. El parámetro $D_1(i)$ representa la distribución de costos para los datos etiquetados y no etiquetados, β es el peso asignado a los datos etiquetados y no etiquetados en la distribución D_0 y F_t representa el ensamble de clasificadores después de agregar el *t-ésimo clasificador*. T representa el número máximo de clasificadores para formar el ensamble. Dado que no se conoce la clase y_i de los datos no etiquetados U, ésta se obtiene usando el algoritmo del vecino más cercano (k-NN). Una vez asignadas las pseudo-clases con el algoritmo k-NN a los ejemplos no etiquetados, el clasificador base, $L(L +$

Algoritmo 2.2 ASSEMBLE

Entrada: S: ejemplos etiquetados , U: ejemplos no etiquetados, T: iteraciones
Salida: Hipotesis Final : $F_{(T+1)}$

1: $\ell = |S|$ y $u = |U|$
2: Inicializar:
$$D_1(i) = \begin{cases} \frac{\beta}{\ell} & \texttt{if } i \in L \\ \frac{1-\beta}{u} & \texttt{if } i \in U \end{cases}$$
3: $y_i = c$ donde c es la clase de k-NN para $i \in U$
4: $F_1 = L(L+U, Y, D_1)$
5: **for** $t = 1$ to T **do**
6: $\hat{y}_i = f_t(x_i), i = 1...\ell + u$
7: $\epsilon = \sum_i D_t \left[y_i \neq \hat{y}_i \right], i = 1...\ell + u$
8: **if** $\epsilon > 0.5$ **then**
9: Detener ciclo
10: **end if**
11: $w_t = 0.5 \cdot log(\frac{1-\epsilon}{\epsilon})$
12: $F_t = F_{t-1} + w_t f_t$
13: $y_i = F_t(x_i)$ if $i \in U$
14: $D_{t+1} = \frac{\alpha_i e^{-y_i F_{t+1}(x_i)}}{\sum_j \alpha_j e^{-y_j F_{t+1}(x_j)}} \forall i$
15: $S = Sample(L+U, \ell, D_{t+1})$
16: $f_{t+1} = L(S, Y, D_{t+1})$
17: **end for**

U, Y, D_1), se aplica a una distribución de ejemplos de S con etiquetas actuales Y, donde los datos tienen un peso de acuerdo a la distribución actual D_t. En seguida se calcula el error generado por la predicción de las etiquetas de los ejemplos etiquetados y no etiquetados, el cual está dado por $\epsilon = \sum_i D_t [y_i \neq \hat{y}_i]$ para un $\epsilon <= 0.5$. Se calcula la función de costo como $D_{t+1} = \frac{\alpha_i e^{-y_i F_{t+1}(x_i)}}{\sum_j \alpha_j e^{-y_j F_{t+1}(x_j)}} \forall i$, donde α_i toma un valor igual a 1; y se llama al siguiente clasificador base con la distribución de datos actualizada.

A diferencia del ensamble de clasificadores que se propone en este trabajo de tesis, el algoritmo ASSEMBLE utiliza en la primera iteración el algoritmo k-NN para asignar pseudoclases a las instancias no etiquetadas, además, no se ha utilizado con bases de datos de imágenes. ASSEMBLE es muy parecido al ensamble que se propone, por la asignación de pesos para distinguir los datos etiquetados de los no etiquetados y para eliminar el sesgo que se genere por el manejo de una distribución distinta de ambos conjuntos de datos.

ASSEMBLE asigna un peso inicial en forma arbitraria a los ejemplos no etiquetados y a partir de la segunda iteración todos los ejemplos, etiquetados y no-etiquetados son tratados de la misma forma. En este trabajo de tesis, los pesos asignados a los ejemplos no etiquetados dependen de su probabilidad predictiva y cambian dinámicamente en cada iteración del ensamble.

2.6. Evaluación de un clasificador

La evaluación del desempeño de un clasificador es un factor muy importante del aprendizaje computacional. La forma más común de medir la eficiencia de un clasificador es mediante la precisión predictiva del mismo. Cada vez que se introducen nuevos ejemplos a un clasificador, éste debe tomar la decisión correcta sobre la etiqueta que le asignará. La clasificación incorrecta de un ejemplo se considera como un error del clasificador. La tasa de error, o su complementaria, la tasa de acierto, se calcula fácilmente como [4]:

$$Taza\ de\ error = \frac{Número\ de\ errores}{Número\ total\ de\ casos} \tag{2.3}$$

Existen también otros dos conceptos importantes en la evaluación de un clasificador [4]:

1. *Taza de error verdadera.* Considerada como la probabilidad de que el modelo clasifique incorrectamente nuevos ejemplos que no se utilizaron en su construcción.

2. *Taza de error aparente.* Esta taza es demasiado optimista respecto a la realidad(o taza de error verdadera), ya que considera el error sobre las instancias utilizadas para inducir el modelo.

Además de estos dos conceptos también existen otras medidas de evaluación de la bondad o desempeño de un clasificador, las cuales se describen a continuación:

- Verdaderos Positivos (TP) = Es aquella instancia cuya hipótesis dice que debe ser positivo y en realidad es positivo.
- Verdaderos Negativos (TN) = Es aquella instancia cuya hipótesis dice que debe ser negativo y en realidad es negativo.
- Falso Positivo (FP) = Es aquella instancia cuya hipótesis dice que debe de ser positivo y en realidad es negativo.
- Falso Negativo (FN) = Es aquella instancia cuya hipótesis dice que debe de ser negativo y en realidad es positivo.

A partir de las medidas anteriores surgen otras medidas de evaluación de un clasificador las cuales se destacan a continuación [4]:

- *Precisión*: Porcentaje o proporción de predicciones positivas que son correctas, es decir, es la probabilidad de que una instancia x sea clasificada con la clase c, y esta instancia realmente pertenezca a esta clase.

$$Precisión = \frac{TP}{(TP + FP)} \tag{2.4}$$

- *Recall*: Porcentaje de verdaderos positivos predichos de entre todos los positivos, es decir, es la probabibilidad de que si una instancia x pertence a una clase c, el clasificador la clasifique correctamente.

$$Recall = \frac{TP}{(TP + FN)} \tag{2.5}$$

- *Accuracy*: Porcentaje de predicciones que son correctamente clasificadas.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{2.6}$$

- *Especificidad*: Porcentaje de instancias negativas que fueron predichas como negativas.

$$Especificidad = \frac{TN}{(TN + FP)} \tag{2.7}$$

Una buena medida de estimación del comportamiento del clasificador con nuevos datos, es la precisión. Sin embargo, si se calcula la precisión con el conjunto de ejemplos que se utilizó para entrenar el clasificador, se obtiene con frecuencia una precisión mayor a la real, es decir, serán estimaciones muy optimistas al utilizar los mismos ejemplos en la inducción del algoritmo y en su comprobación [35]. La idea básica es estimar el desempeño de un clasificador con una porción de los ejemplos y luego comprobar su válidez con el resto de los ejemplos. Esta separación es necesaria para garantizar la independencia de la medida de precisión resultante, de no ser así, la precisión del modelo será estimada. Para tener seguridad de que las predicciones sean robustas y precisas, se consideran dos etapas en el proceso de construcción de un clasificador, entrenamiento y prueba, partiendo los datos en dos conjuntos, uno de entrenamiento y otro de prueba.

Se disponen de varias estrategias de validación de un sistema de aprendizaje dependiendo de como se realice la partición del conjunto de datos. Básicamente se han utilizado dos técnicas para validar un clasificador, las cuales se describen a continuación.

2.6.1. Validación simple

El método de validación más sencillo es el de *validación simple* [35], el cual utiliza un conjunto de muestras para construir el modelo del clasificador. De entre la variedad de porcentajes utilizados, uno de los más frecuentes es tomar $2/3$ del conjunto total de ejemplos para el proceso de aprendizaje y $1/3$ para comprobar el error del clasificador. El inconveniente principal de esta técnica es el hecho de utilizar una parte de los ejemplos disponibles para llevar a cabo el aprendizaje, lo que causa que se pierda información útil en el proceso de inducción del clasificador.

2.6.2. Validación cruzada

La *validación cruzada* [24], es un método efectivo para probar la válidez de los algoritmos y evitar pérdida de información. En esta técnica el conjunto de datos D se divide aleatoriamente en k particiones mutuamente exclusivas, $D_1, D_2, ..., D_k$, conteniendo cada una el mismo número de ejemplos aproximadamente. La validación cruzada se ejecuta k-veces, por ello esta técnica es llamada en muchos casos como validación cruzada con k particiones o *k-fold-cross validation*. Un valor comunmente usado para k es 10. En cada evaluación se utiliza uno de los subconjuntos como conjunto de prueba, y se entrena el sistema con los $k-1$ conjuntos restantes. Así, la precisión estimada de clasificación es la media de las k tasas obtenidas. La técnica de validación cruzada se muestra en el algoritmo 2.3. La ventaja de usar *k-fold-cross validation* es que todos los ejemplos en el conjunto de datos son eventualmente usados para entrenamiento y prueba. La estimación de la precisión de *k-fold-cross validation* es el número completo de clasificaciones correctas sobre el número de instancias en el conjunto de datos. Una vez ejecutado este proceso k-veces, se obtiene el error E estimado como la taza de error promedio sobre los ejemplos de prueba.

$$E = \frac{1}{N} \sum_{i=1}^{N} E_i$$

Algoritmo 2.3 k-fold-cross validation

Entrada: $D_0 = \{(x_1, y_1), ..., (x_n, y_n)\}$: Conjunto de Datos
Salida: E: Error promedio

1: Dividir D_0 en k subconjuntos $m_1, m_2, ..., m_k$ de igual tamaño
2: $\{m_1 \cup m_2 \cup ... \cup m_k\} = D_0$
3: **for** $i = 1$ hasta k **do**
4: $m_i \leftarrow$ *conjunto de prueba*
5: $s_i \leftarrow$ *conjunto de entrenamiento*
6: $s_i \leftarrow D_0 - m_i$
7: Entrenar el modelo L usando s_i
8: $h_i \leftarrow L(s_i)$
9: Calcular el error E de h_i para m_i, $E_{m_i}(h_i)$
10: **end for**
11: Regresar el error promedio $E = \frac{1}{k} \sum_{i=1}^{k} E_{m_i}(h_i)$

2.7. Sumario del capítulo

En este capítulo se presentaron los fundamentos del aprendizaje computacional. Se describieron algunos algoritmos de clasificación bajo el enfoque supervisado y semi-supervisado. Un algoritmo supervisado requiere que todos los ejemplos se encuentren previamente etiquetados para entrenar al modelo de clasificación. En el enfoque semi-supervisado se incorpora el uso de ejemplos no etiquetados para mejorar la tarea de clasificación. En los ensambles se combinan más de un algoritmo de clasificación para predecir clases de nuevos ejemplos. Los ensambles han demostrado tener mejor desempeño que un único clasificador. La evaluación de un clasificador es una tarea muy importante en la predicción de clases de nuevos ejemplos ya que permite determinar la taza de error cometida por el clasificador.

ASSEMBLE está relacionado con el algoritmo que se propone en este trabajo de tesis. En ASSEMBLE, una vez que a los ejemplos no etiquetados se les asigna una clase, éstos son tratados de igual forma que el resto de los ejemplos, lo cual puede provocar sesgos indeseables y generar errores.

En el siguiente capítulo se describe el proceso de etiquetado automático de imágenes, así como las dos tareas que se encuentran relacionadas con este proceso, la segmentación de imágenes y la extracción de características.

Capítulo 3

Etiquetado automático de imágenes

En este capítulo se presenta el marco teórico para llevar a cabo la tarea de etiquetado automático, usando procesos como segmentación de imágenes, extracción de características y clasificación.

3.1. Etiquetado automático de imágenes

El etiquetado o anotación automática consiste en asignar una palabra clave o etiqueta (considerada como descriptor semántico de alto nivel) a cada una de las regiones u objetos de una imagen. El etiquetado permite a los usuarios accesar a una gran base de datos mediante consultas textuales, es decir, una vez anotadas las imágenes, el conjunto de palabras clave asociado a cada imagen o regiones de las imágenes se utiliza para realizar consultas.

Existen algunos trabajos sobre etiquetado automático de imágenes que han utilizado técnicas de aprendizaje automático para resolver esta tarea. Estas técnicas aprenden modelos estadísticos a partir de imágenes etiquetadas y se aplican para generar etiquetas de nuevas imágenes. Algunos de los modelos como máquinas de soporte vectorial (SVM), redes neuronales y modelos bayesianos, entre otros, usan clasificación supervisada, la cual requiere de un proceso de aprendizaje o entrenamiento por parte del clasificador que debe ser supervisado manualmente en mayor o menor medida [10]. Recientemente se han utilizado algunos modelos de clasificación automática supervisada para resolver la tarea de etiquetado. Estos

modelos parten de una serie de clases o categorías y de manera automática tratan de predecir las etiquetas de nuevas imágenes.

Sin embargo, estos modelos no han llegado a obtener buenos porcentajes de predicción debido a que utilizan tamaños de muestra pequeños, ya que generalmente no se tiene la información suficiente para formar un modelo confiable.

Algunos de los principales trabajos reportados para resolver la tarea de etiquetado automático se describen a continuación.

- Duygulu *et al.*, [17] proponen un modelo para reconocimiento de objetos asociando palabras clave a diferentes regiones mediante el algoritmo de aprendizaje EM. El proceso de etiquetado es visto como un proceso de traducción automática, donde una representación (regiones de imagen) se traduce a otra (palabras o etiquetas). En este trabajo, las imágenes son segmentadas en regiones usando el algoritmo Normalized Cuts [36]. Estas regiones son clasificadas de acuerdo a varios tipos de regiones usando una variedad de características visuales. A cada uno de los tipos de regiones se les asocian palabras clave, las cuales son sustantivos tomados de un vocabulario predefinido. Para llevar acabo esta asociación se utiliza el algoritmo de aprendizaje semi-supervisado EM, con el cual entrenan el modelo. Este modelo puede predecir una gran cantidad de palabras con una alta precisión. Se utilizaron 5,000 imágenes de Corel y un vocabulario de 371 palabras para la anotación.

- Barnard *et al.* [5], presentan un enfoque para predecir texto asociado tanto a imágenes completas (*auto-etiquetado*) así como a regiones particulares de la imagen (*nombramiento de región*). El auto-etiquetado facilita el acceso a grandes colecciones de imágenes mientras que el nombramiento de región es un modelo de reconocimiento de objetos visto como un proceso de traducción de regiones de imágenes a palabras de texto. Además, comparan varios métodos para predecir texto o palabras de imágenes. Los modelos son entrenados usando una gran colección de imágenes etiquetadas de escenas reales. Usan 16,000 fotografías de Corel con 155 palabras clave o etiquetas y 10,000 imágenes de prueba que son anotadas automáticamente.

- David Blei y Michael Jordan [8], consideran tres modelos jerárquicos probabilísticos para resolver el problema del etiquetado automático. Los autores se enfocan en casos en los cuales un tipo de dato puede ser visto como anotación o etiqueta de otro tipo de dato. Presentan una evaluación de los tres modelos con 7,000 imágenes y usan etiquetas de la base de datos de Corel. Blei y Jordan muestran que estos modelos pueden ser usados para asignar palabras a imágenes de gran tamaño.

- Yavlinsky *et al.*, [43] describen un sistema simple para el etiquetado automático de imágenes usando un enfoque "orientado a la escena". En este enfoque las imágenes pueden describirse con etiquetas de escenas básicas como calle, edificio o autopista. Usan modelos no paramétricos y estudian su comportamiento usando varias propiedades de la imagen como color y textura. Los autores muestran que bajo este simple sistema muchas propiedades de las imágenes pueden conducir a una precisión aceptable en la anotación.

La mayoría de estos trabajos requieren de procesos como la segmentación de imágenes y la extracción de características para resolver la tarea de etiquetado automático. En las siguientes secciones se describen estos dos procesos con más detalle.

3.2. Segmentación de imágenes

La segmentación de imágenes es una tarea compleja que aún no se ha resuelto completamente. Segmentar una imagen [31] consiste en dividir a la imagen en varios componentes u objetos con características similares mediante límites bien definidos. Estos componentes segmentados conocidos como *regiones* no deben de empalmarse. Cada región tiene un vector característico que lo diferencia de las otras regiones de la imagen. La segmentación es considerada como el primer paso en el análisis de imágenes [20]. El nivel de la subdivisión de la imagen depende del problema que se vaya ha resolver. En el etiquetado automático de imágenes se requiere un alto nivel de precisión en la subdivisión de la imagen de manera que facilite la identificación completa de los objetos de la imagen.

Clásicamente, la segmentación de imágenes se define como la partición de una imagen en regiones constituyentes no solapadas, las cuales son homogéneas con

respecto a alguna característica como intensidad o textura [34]. Si el dominio de la imagen está dado por I, entonces el problema de segmentación consiste en determinar los subconjuntos $S_k \subset I$ cuya unión es la imagen I completa. Por lo tanto, el conjunto que conforma la segmentación debe satisfacer:

$$I = \bigcup_{k=1}^{n} S_k \tag{3.1}$$

donde $S_k \cap S_j = \phi$ para $k \neq j$, y cada S_k está conectado. Idealmente, un método de segmentación encuentra aquellos conjuntos que corresponden a distintas estructuras o regiones de interés en la imagen.

Se han diseñado e implementado varias técnicas que tratan de resolver el problema de segmentación de imágenes. Estas técnicas tienen como objetivo principal agrupar píxeles que tengan gran similitud y separarlos de aquellos que no la tengan. A continuación se describen algunas técnicas de segmentación de imágenes.

3.2.1. Técnicas de segmentación

Existe un gran número de métodos para resolver la tarea de segmentación. Sin embargo, ninguno de ellos ha resuelto eficazmente esta tarea de manera general. Dentro de los métodos de segmentación clásicos se encuentran tres grandes grupos:

1. Métodos de umbralización.
2. Métodos basados en crecimiento de regiones.
3. Métodos basados en detección de bordes o fronteras.

Los métodos de umbralización son de los más eficientes y sencillos para segmentar [37]. Utilizan el histograma para indicar el número de puntos en los que la imagen posee un determinado nivel de gris. La umbralización trata de determinar un valor de intensidad, llamado umbral (threshold), que separa las clases deseadas, es decir, a partir de histogramas elige el punto de nivel de gris que separa los valores correspondientes al objeto y al fondo. En la figura 3.1 se muestra un ejemplo del histograma de una imagen, en donde aparentemente posee 5 posibles valores de umbral en donde cambia la intensidad de la imagen. Estos umbrales se

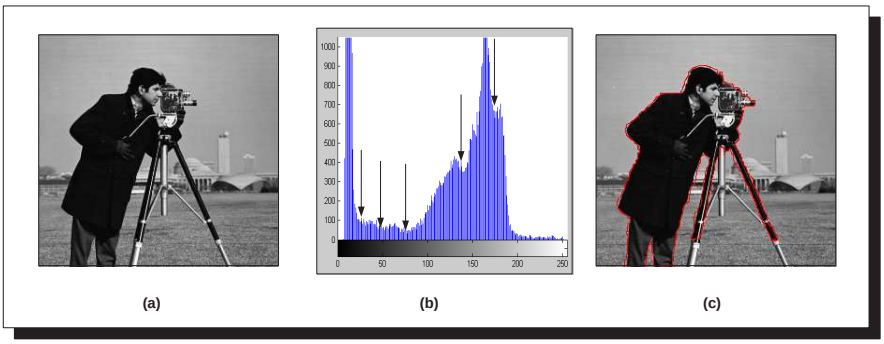

Figura 3.1: Método basado en el histograma: (a) Imagen en escala de grises. (b) Histograma de intensidades de gris en la imagen. (c) Imagen segmentada

observan en los valles del histograma de instensidades de gris. Generalmente, la umbralización (thresholding) es un método que busca segmentar imágenes creando una partición binaria de las intensidades de color o niveles de gris de la imagen. Su principal limitación es que usualmente no toma en cuenta las características espaciales de la imagen. Esto causa que sea sensible al ruido.

En los métodos basados en crecimiento de regiones [37], se comienza eligiendo aleatoriamente un conjunto de pixeles al que se les llama semillas. Apartir de las semillas, se van incorporando nuevos pixeles a las regiones utilizando un mecanismo de crecimiento. Este mecanismo detecta en cada etapa y en cada región a aquellos pixeles que aún no han sido clasificados para pertenecer a algún pixel del contorno de la región. Para cada pixel que se detecta se comprueba si cumple con la regla de homogeneidad. Cuando se elige más de una semilla por región es necesario unir o fusionar algunas de las regiones obtenidas.

En la figura 3.2 se muestra un ejemplo del resultado de aplicar un método basado en división de regiones, el cual ésta muy relacionado con los métodos de crecimiento de regiones que no requieren de una semilla para comenzar a generar la segmentación. Quad-tree es una estructura de árbol en la cual cada nodo interno apunta a un bloque de datos de la imagen. El método quad-tree que se muestra en la figura 3.2 realiza una descomposición de la imagen original en bloques más pequeños y crece iterativamente probando condiciones de bloques padre, basados en el contenido de textura de los bloques hijos hasta lograr encontrar regiones homogéneas en la imagen.

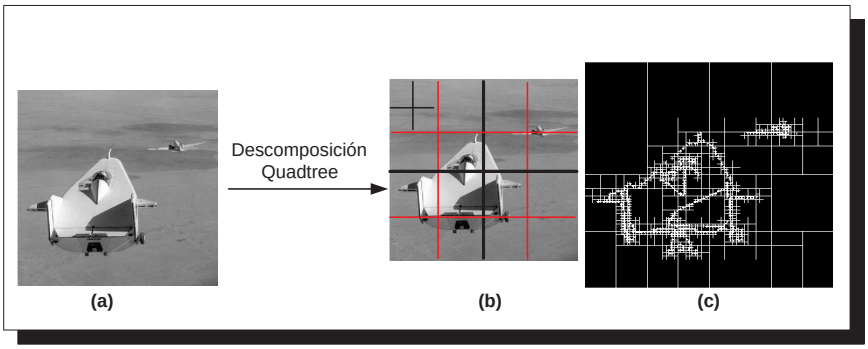

Figura 3.2: Segmentación basada en división de regiones (quad-tree): (a) Imagen original. (b) División de la imagen. (c) Segmentación quad-tree.

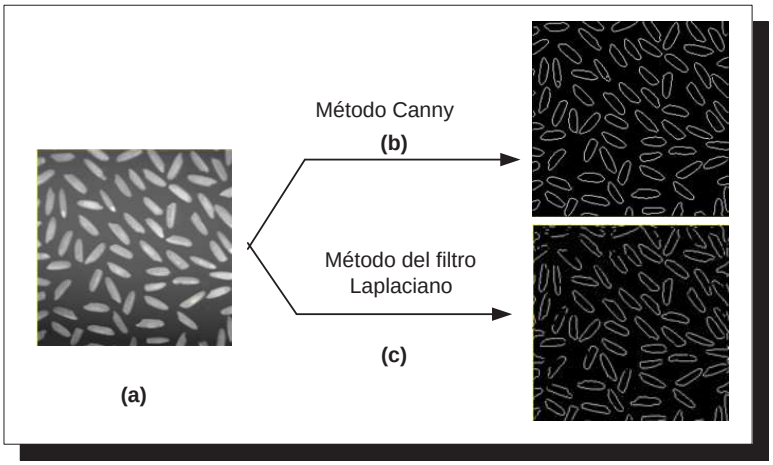

Figura 3.3: Método de detección de bordes: (a) Imagen original. (b) Detección de bordes por el método de Canny. (c) Detección de bordes por el método del filtro Laplaciano.

Los métodos basados en bordes consisten en detectar los puntos en una imagen en los cuales los cambios de luminosidad o intensidad son abruptos [45]. Estos cambios incluyen a) discontinuidad en el fondo de la imagen, b) discontinuidades en la superficie, c) cambios en propiedades de textura y d) variaciones en la iluminación. En la figura 3.3 se muestra un ejemplo del resultado obtenido de aplicar el método de segmentación Canny y el método de filtro laplaciano (ambos basados en la detección de bordes) a una imagen en escala de grises.

Para el desarrollo de este proyecto de tesis se utilizó el algoritmo de segmentación llamado Normalized Cuts, el cual esta basado en la partición de grafos y permite segmentar imágenes de forma automática. Además, este algoritmo ha

sido ampliamente utilizado en problemas de recuperación de imágenes, en donde se han logrado conseguir buenos resultados debido a su buen nivel de precisión para la segmentación [17, 38]. Además, Normalized Cuts es un algoritmo ampliamente conocido y no es muy complejo de implementar. En la siguiente sección se describe con más detalle su funcionamiento.

3.2.2. Algoritmo Normalized Cuts

El algoritmo *Normalized Cuts* [36] fue propuesto por Jianbo Shi y Jitendra Malik en 1997. El objetivo de *Normalized Cuts* es medir la bondad de la partición de una imagen apartir de un nuevo criterio téorico-gráfico llamado *Normalized Cuts*. Normalized Cuts mide tanto la diferencia total que existe entre los distintos grupos de la imagen como la similaridad total dentro de un mismo grupo. En esta técnica se consideran diferentes características de la imagen como intensidad, textura, continuidad de contorno y color.

En *Normalized Cuts* la imagen es vista como un grafo completo no dirigido, ver figura 3.4. Cada nodo del grafo representa un pixel de la imagen. Existe un arco entre cada par de pixeles a y b, el cual contiene un costo C_{ab} que mide la similaridad del par de pixeles. Este algoritmo trata de encontrar aquellos arcos que tengan un costo muy pequeño ya que resulta fácil eliminarlos mediante un corte del grafo. Además, cada uno de los arcos debe procurar mantener pixeles similares en los mismos segmentos y pixeles distintos en otros segmentos. La partición óptima del grafo es aquella que minimice los costos de los arcos que fueron removidos. El corte mínimo de segmentos se realiza mediante la ecuación 3.2, en donde $cut(A, B)$ es un corte que se realiza entre los segmentos A y B, $volume(A)$ es la suma de todos los costos de los bordes que toca A y $volume(B)$ es la suma de todos los costos de los bordes que toca B. En la figura 3.5 se muestra un ejemplo de segmentación de imagen con Normalized Cuts.

$$Ncut(A, B) = \frac{cut(A, B)}{volume(A)} + \frac{cut(A, B)}{volume(B)} \tag{3.2}$$

Sea $G = (V, E)$ un grafo completo no dirigido, donde V es un conjunto de nodos y E es un conjunto de arcos. G puede particionarse en dos conjuntos disjuntos A y B, donde $A \cup B = V$ y $A \cap B = \phi$, simplemente removiendo arcos que conectan a

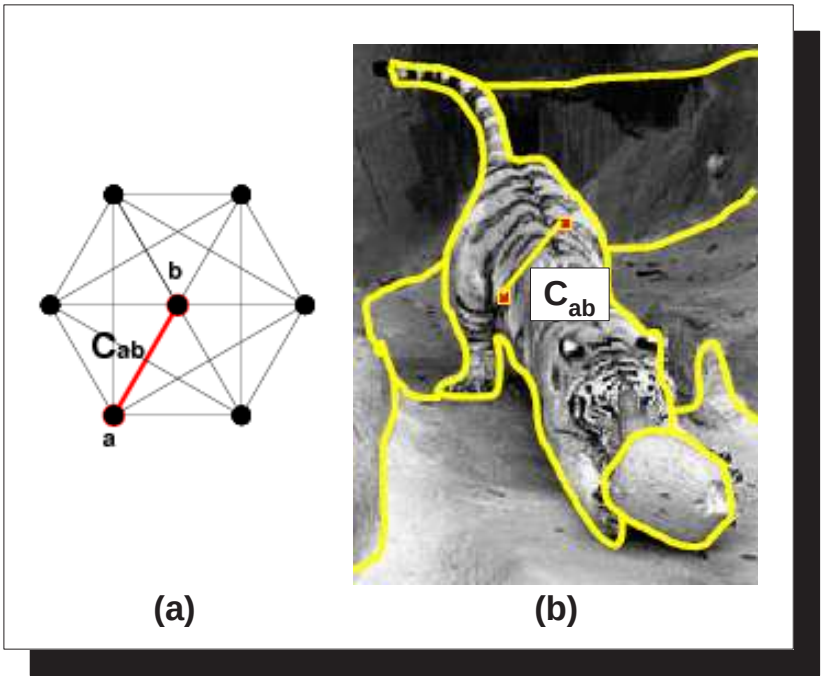

Figura 3.4: Imagen vista como un grafo completo: (a) Grafo completo no dirigido. (b) Arco entre cada par de pixeles de la imagen con un costo C_{ab}

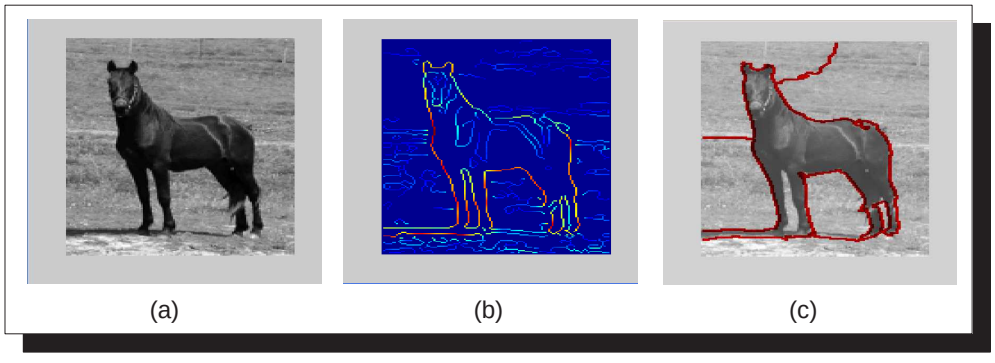

Figura 3.5: (a) Imagen original. (b) Bordes de la imagen. (c) Imagen segmentada

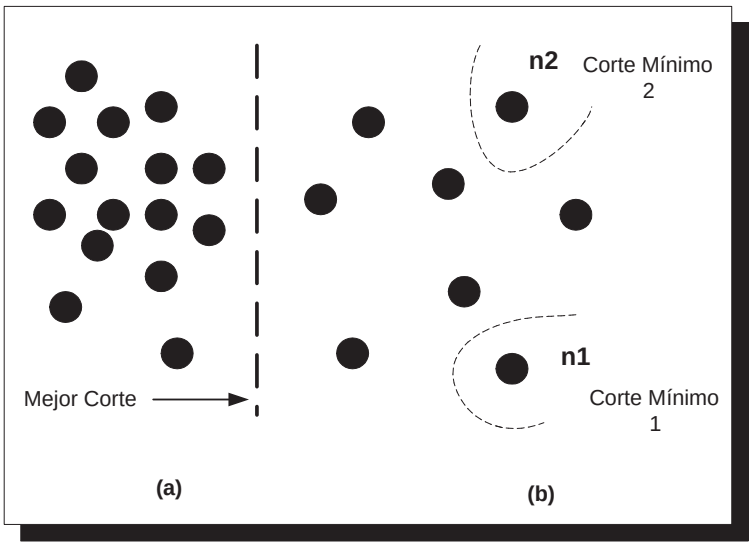

Figura 3.6: Corte de un grafo: (a)Mejor corte agrupando el mayor número de pixeles. (b) Caso en donde el corte mínimo regresa una mala partición.

ambos conjuntos. El grado de diferencia entre estos dos conjuntos se calcula como el costo total de los arcos que han sido removidos. A ésto se le llama *corte* y está dado por la ecuación 3.3:

$$cut(A, B) = \sum_{u \in A, v \in B} C_{uv} \tag{3.3}$$

La bipartición óptima de un grafo es aquella que minimiza el valor de corte. En la figura 3.6 se muestra un caso en donde el corte se incrementa con el número de arcos. El mejor corte permite agrupar al mayor número de pixeles que sean similares y descartar aquellos que se encuentren dispersos. En la figura 3.6 se observa que los *cortes mínimos* 1 y 2 no son los mejores, ya que sólo agrupan a un único pixel dada la distancia que existe de éstos hacia los demás pixeles más cercanos. Para este caso, el algoritmo regresa una mala segmentación. Sin embargo, en general el algoritmo *Normalized Cuts* realiza buenos cortes en el grafo.

3.3. Extracción de características

La extracción de características de imágenes es la base de la recuperación de imágenes por contenido. Las características pueden estar basadas en texto (palabras clave o anotaciones) o pueden ser características visuales (color, textura, forma, información espacial). Algunas de estas características visuales se describen a continuación.

Color. Esta característica es la más utilizada en recuperación de imágenes. Existen varias representaciones de la característica de color que han sido aplicadas en recuperación de imágenes incluyendo el histograma, momentos, conjuntos y distribuciones de color. El histograma de color es el más utilizado ya que denota la probabilidad conjunta de las intensidades de los tres canales de color. En este trabajo de tesis utilizamos el histograma de color para obtener los tres colores predominantes de cada región de una imagen. También se calculó el promedio global, considerado como el valor medio de luminosidad de todos los pixeles de la imagen y la desviación estándar de cada región, obtenida mediante las diferencias de todos los pixeles sobre el promedio global.

Textura. Otra de las características que juega un papel muy importante en los sistemas de recuperación por contenido es la percepción de las texturas. Esta característica se utilizó en este trabajo para obtener información acerca de las superficies de las imágenes. La textura es definida como una distribución estadística de dependencias espaciales de las propiedades del nivel de gris [3]. La textura puede ser descrita por la uniformidad, densidad, grosor, rugosidad, regularidad, intensidad y direccionalidad de medidas discretas del tono y de sus relaciones espaciales.

Existen varias representaciones de la textura que han sido investigadas en reconocimiento de patrones y visión por computadora. Básicamente los métodos de representación de la textura pueden clasificarse en dos categorias [18]: estructural y estadístico. Los métodos estructurales incluyen el uso de un operador morfológico o de un grafo de adyacencia. Los métodos estadísticos incluyen la transformada de Fourier, matrices de co-ocurrencia, componentes de análisis principales, campos aleatorios de Markov y técnicas de filtrado multiresolución como los filtros de Gabor y la transformada Wavelet.

Una de las herramientas más poderosas para el análisis de texturas son los filtros de Gabor [13], los cuales pueden verse como el producto de una componente pasobajo (gaussiana) y una ondulatoria (senoidal). Un filtro de Gabor de dos dimensiones está dado por la ecuación 3.4:

$$g(x,y) = \left(\frac{1}{2\pi\sigma_x\sigma_y}\right) exp\left[-\frac{1}{2}\left(\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2\right) + jw\,(xcos\theta)\right] \tag{3.4}$$

Donde θ representa la orientación de la señal senoidal que se encuentra dentro del intervalo de 0 a 360 grados. Las constantes σ_x, σ_y determinan la varianza de la gaussiana en los ejes x y y, respectivamente; y jw representa la frecuencia de la senoidal a lo largo del eje x. En esta tesis se utilizaron los filtros de Gabor variando el parámetro de orientación con los valores 0, 45, 90 y 135 grados.

Forma. La característica de forma [18] puede ser vista como la apariencia física o el contorno de un objeto. Las características de forma de los objetos o regiones se han usado en muchos sistemas de recuperación de imágenes por contenido. En comparación con las características de color y textura, las características de forma son usualmente descritas después de que las imágenes han sido segmentadas en regiones u objetos. En ocasiones se llega a descartar el uso de las características de forma debido a que es difícil lograr una gran precisión en la segmentación de imágenes.

Algunas características de forma son las siguientes:

1. *Área*, es un escalar que mide el número actual de píxeles en la región.
2. *Área convexa*, mide el número de píxeles en la región convexa de la imagen. El área convexa es el área del mínimo rectángulo que envuelve a la región de la imagen.
3. *Perímetro*, es la cantidad de pixeles que conforman el contorno de una región. Éste se calcula como $\sqrt{\frac{4*Area}{\pi}}$.
4. *Eje mayor y menor*, permiten medir la longitud en píxeles del eje mayor y menor de la elipse de la imagen.

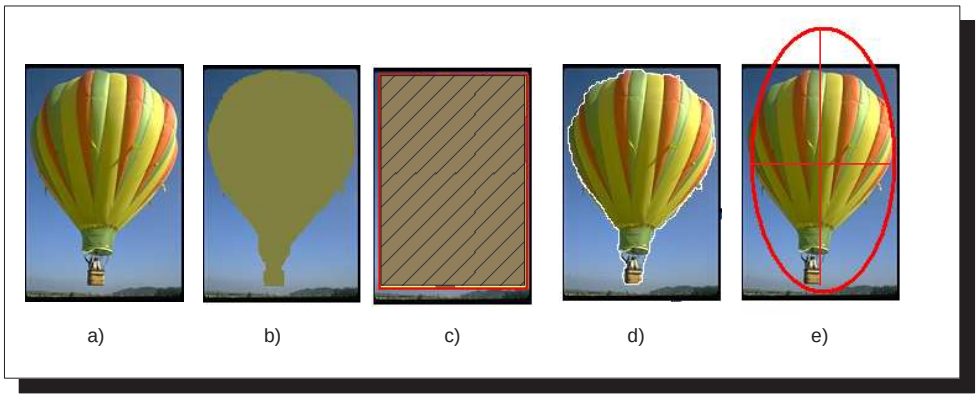

Figura 3.7: (a) Imagen 128 x 192. (b) Área de la región segmentada. (c) Área convexa de la región, (d) Perímetro de la región, (e) Eje mayor y menor de la región.

5. *Solidez*, es la proporción de pixeles en la convexidad completa de la región. Se calcula como $\frac{area}{area\ convexa}$.

6. *Excentricidad*, es la distancia entre los dos focos de la elipse de una imagen. La excentricidad medida sobre figuras normalizadas se encuentra en un rango entre 0 y 1. Por ejemplo para un cuadrado la excentricidad resulta 0, lo mismo que para un círculo ya que la elipse coincide con el círculo.

En la figura 3.7 se ilustran las características de forma mencionadas anteriormente. En este proyecto de tesis se consideraron estas características de forma para extraer información relevante de las imágenes.

3.4. Sumario del capítulo

En este capítulo se describió el etiquetado automático de imágenes, el cual permite asignar descriptores semánticos que ayuden a identificar objetos en las imágenes y realizar búsquedas de imágenes de manera más eficaz. Esta técnica necesita de procesos como la segmentación y extracción de características, siendo la segmentación una tarea compleja. El algoritmo de segmentación que se utilizó para el desarrollo de este proyecto de tesis es Normalized Cuts, el cual se ha utilizado en otros trabajos sobre recuperación de imágenes por contenido [17, 43].

Este algoritmo en general realiza una buena segmentación. La aplicación del proceso de extracción de características permite extraer información relevante de las imágenes para utilizarla en el etiquetado automático.

En el siguiente capítulo se describe la estrategia que se utilizó para resolver el problema del etiquetado automático de imágenes.

Capítulo 4

Etiquetado automático mediante un enfoque semi-supervisado

En este capítulo se describe la estrategia que se aplicó para resolver el problema de etiquetado automático de imágenes digitales basada en un enfoque de aprendizaje semi-supervisado. La estrategia propuesta está formada de 4 etapas, que parte del pre-procesamiento de imágenes hasta la validación del algoritmo de etiquetado.

4.1. Estrategia de etiquetado

La estrategia de etiquetado desarrollada se muestra en la figura 4.1. Esta estrategia parte de la segmentación de un conjunto de imágenes. Para esta etapa se utilizó el algoritmo de segmentación Normalized Cuts. De las regiones segmentadas se extrajeron las características de color, forma y textura, con el objetivo de obtener información relevante de cada una de las regiones. Para esta tarea se desarrolló un sistema interactivo que permite segmentar las imágenes, así como visualizar y extraer características de cada una de las regiones segmentadas. Cada una de las regiones segmentadas son representadas mediante vectores característicos que incluyen información visual de las regiones. Inicialmente a una parte del conjunto de vectores característicos se le asigna una etiqueta de forma manual (que es el equivalente a etiquetar las regiones de la imagen). Este conjunto de vectores formará parte de las instancias o ejemplos del conjunto para entrenar al

clasificador que se encargará de predecir etiquetas de nuevos vectores característicos (regiones).

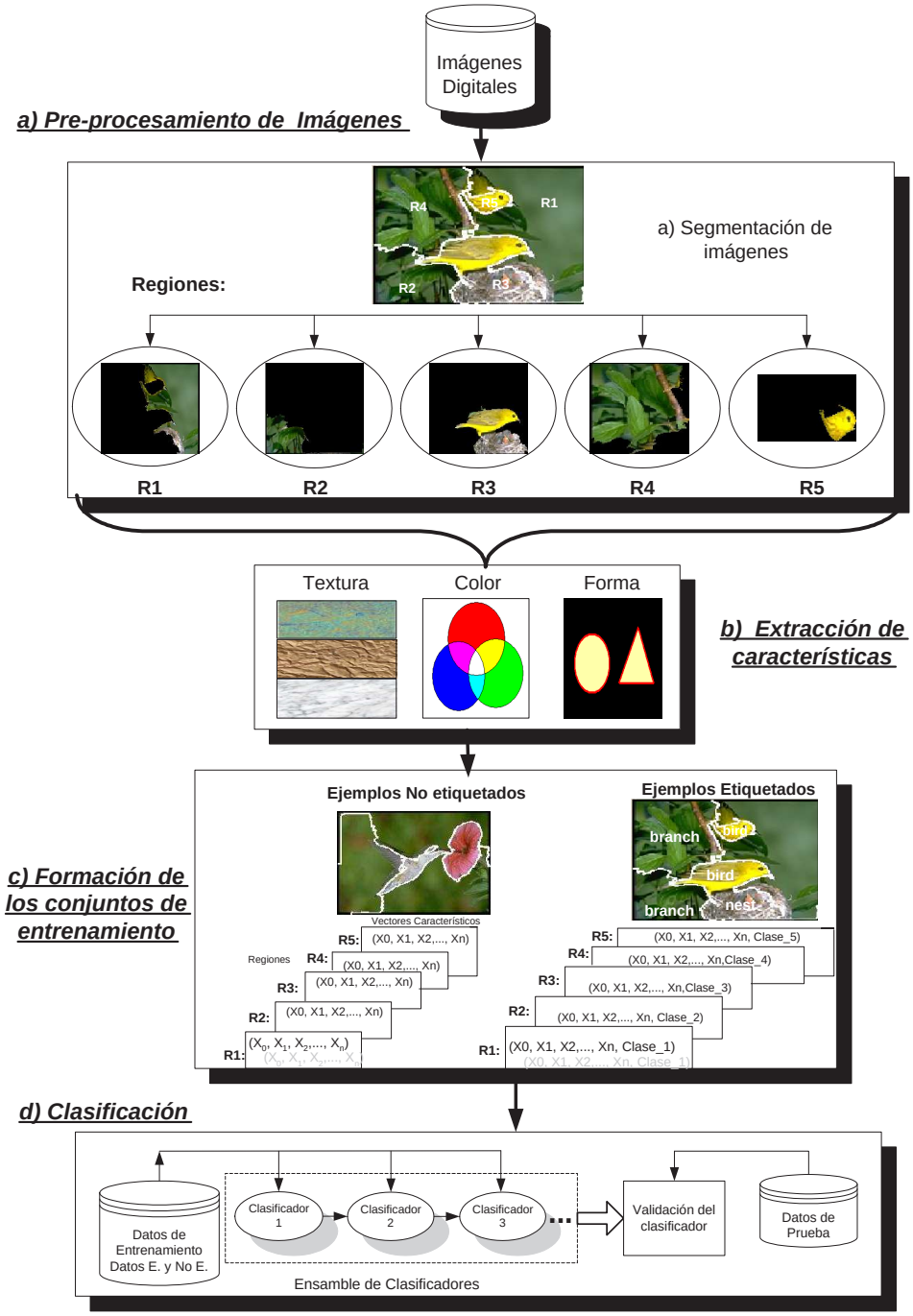

Figura 4.1: Diagrama de bloques de la estrategia de etiquetado.

El clasificador también utiliza ejemplos no etiquetados para el proceso de en-

trenamiento, es decir, vectores característicos que se obtienen del proceso de segmentación pero que no tienen asignada una etiqueta. El clasificador fue validado con base en su precisión sobre las etiquetas predichas de nuevas imágenes de prueba.

En las siguientes secciones se explica con más detalle cada una de las partes de la estrategia de etiquetado propuesta.

4.2. Pre-procesamiento

El pre-procesamiento de imágenes está formado de los procesos de segmentación de imágenes y extracción de características. En la primer tarea se realizó un estudio y análisis de las ventajas y desventajas de algunas técnicas de segmentación automática como segmentación Quad-Tree [35], segmentacion basada en cortes de grafos Graph Cut [33], segmentación basada en crecimiento de regiones [34], entre otras; para poder seleccionar de entre ellas la que tuviera una mayor precisión para dividir lo mejor posible cada una de las regiones de una imagen. En el proceso de extracción de características se trabajó con propiedades de bajo nivel como color, forma y textura que representaran mejor el contenido visual de las imágenes.

4.2.1. Segmentación de imágenes

Se aplicó el algoritmo de segmentación Normalized Cuts de la sección 3.2.2, para cada una de las imágenes. La idea principal de Normalized Cuts es realizar una segmentación buscando que la distancia entre pixeles (caracterización) de una misma región sea baja (parecida) y alta entre pixeles de diferentes regiones.

El algoritmo Normalized Cuts tiene las siguientes ventajas:

- Es un algoritmo de segmentación automática.
- El proceso de segmentación es más preciso en comparación con otros algoritmos de segmentación automática.
- Ha sido usado ampliamente en otros trabajos de recuperación de imágenes.

El algoritmo Normalized Cuts consume un tiempo promedio de 15 segundos en una PC Pentium Centrino a 1.7 GHz con 500MB de RAM para segmentar automáticamente imágenes que contienen pocos objetos. Para mejorar el nivel de segmentación se puede indicar el número de regiones en el que se dividirá la imagen. Por omisión el algoritmo establece una división de 5 regiones por imagen. En la figura 4.2 se presentan algunos ejemplos de los resultados obtenidos del proceso de segmentación con Normalized Cuts utilizando imágenes de la base de datos de Corel [1].

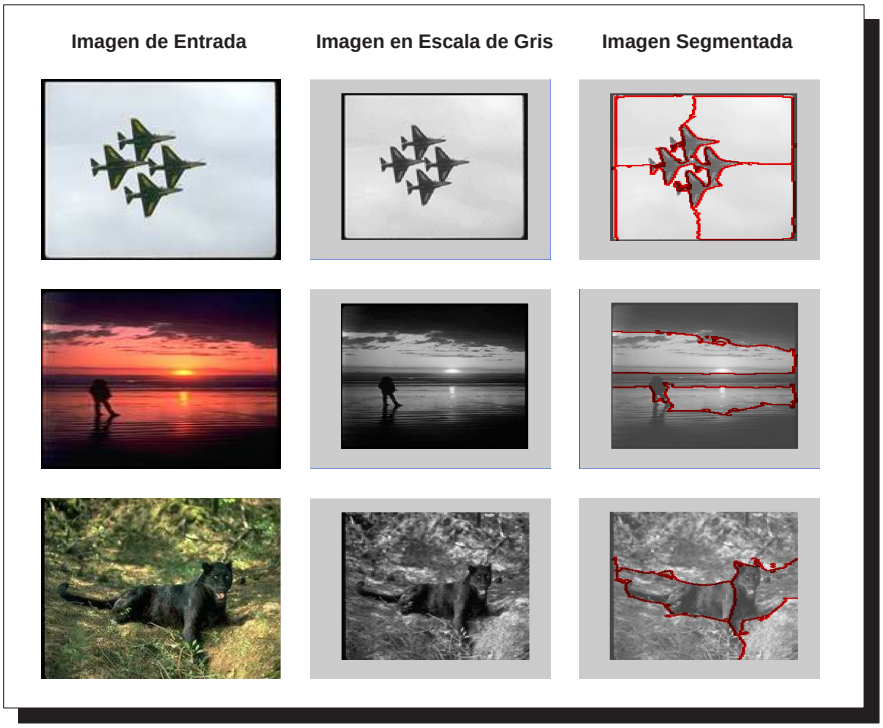

Figura 4.2: Ejemplos de segmentación con Normalized Cuts.

Como se puede observar en la figura 4.2, Normalized Cuts convierte la imagen de entrada en una imagen en escala de grises sobre la cual se realizan cortes normalizados. Sin embargo, para explotar mejor los resultados obtenidos del proceso de segmentación se realizaron modificaciones al algoritmo Normalized Cuts. Entre los cambios más significativos se encuentran los siguientes:

- Jianbo Shi autor de Normalized Cuts, utilizó en sus experimetos imágenes en escala de gris, y en caso de que las imágenes originales fueran muy grandes, éstas eran muestreadas. Uno de los cambios que se realizó al algoritmo fue

el trabajar con imágenes en formato RGB durante todo el proceso de segmentación, de manera que no se perdieran propiedades de color de la imagen. Para ello se aplicó una máscara o filtro entre la imagen original y la imagen segmentada en niveles de gris, para la ubicación de los pixeles de cada segmento de la imagen.

- Las regiones de las imágenes se almacenaron en pequeñas matrices para su posterior visualización y procesamiento de manera individual.
- Se añadió la posibilidad de poder establecer manualmente el número de segmentos en el que se dividirá la imagen de manera que se lograra un mejor resultado en el proceso de segmentación.

En la figura 4.3 se muestran los resultados obtenidos con Normalized Cuts con los mismos ejemplos después de haber realizado las modificaciones mencionadas anteriormente. En esta figura se muestra que la imagen de salida es una imagen a color en formato RGB y que cada una de las regiones de la imagen puede ser vista como una imagen independiente, de la cual se extraen sus características visuales.

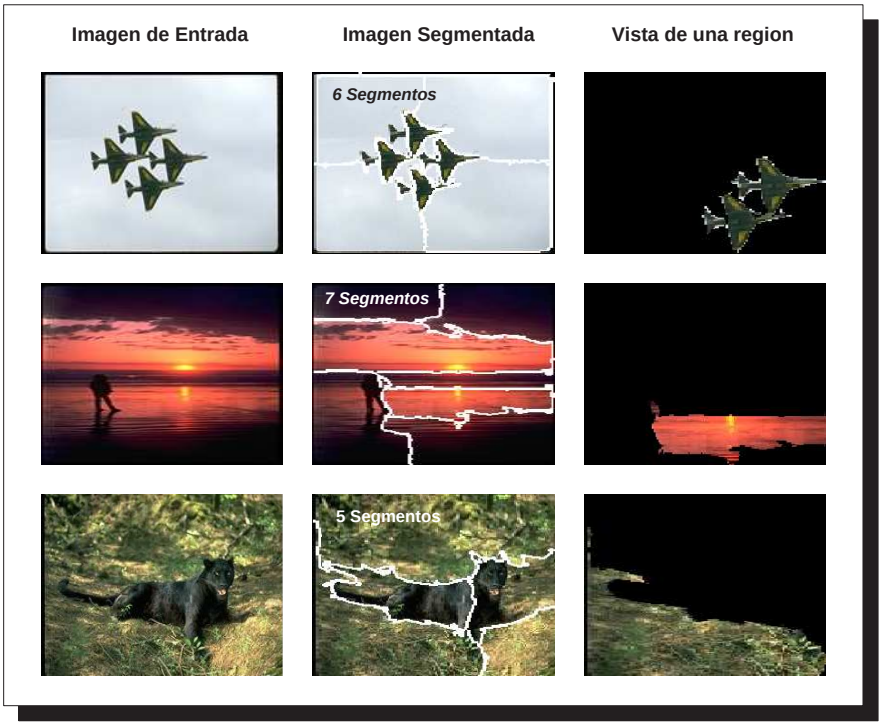

Figura 4.3: Aplicación del algoritmo Normalized Cuts modificado.

4.2.2. Extracción de características

Una vez segmentada la imagen, el siguiente paso que se realizó fue extraer información visual de cada región y representarla a través de un vector de atributos o vector característico $(x_1, x_2, x_3, ..., x_{30})$ de dimensionalidad 30, donde:

- x_1 - x_9: Representan los tres valores máximos en cada banda RGB, es decir los tres colores que son más frecuentes en cada una de las bandas, como se muestra en la figura 4.4.
- x_{10} - x_{12}: Representan el valor promedio de luminosidad de la región en las tres bandas R, G y B.
- x_{13} - x_{15}: Representan la desviación estándar de los colores de la región con respecto a su media, en las tres bandas.
- x_{16} - x_{17}: Indican respectivamente el área y perímetro de la región.
- x_{18} - x_{19}: Contienen la longitud en pixeles del eje mayor y menor de la elipse de la región de la imagen.
- x_{20}: Contiene el número de pixeles en la imagen convexa.
- x_{21}: Representa el número de pixeles en la convexidad completa de la región (solidez), calculada como *área/área convexa*.
- x_{22}: Representa la excentricidad, es decir, el cociente entre la distancia focal y la longitud del eje principal de la imagen [26].
- x_{23} - x_{30}: Corresponden a la media y la varianza de los filtros de Gabor en 4 orientaciones 0, 45, 90 y 135 grados.

Para las tareas de segmentación y extracción de características se desarrolló una interfaz de usuario, la cual se muestra en la figura 4.5. La interfaz contiene una sección para cargar la imagen y el número de segmentos en la que ésta será dividida, una sección para la visualización de los resultados del proceso de segmentación y de las distintas regiones obtenidas por este proceso, una sección para mostrar las características obtenidas en cada una de las regiones de la imagen y una sección para el etiquetado manual de algunas de las imágenes del conjunto de entrenamiento.

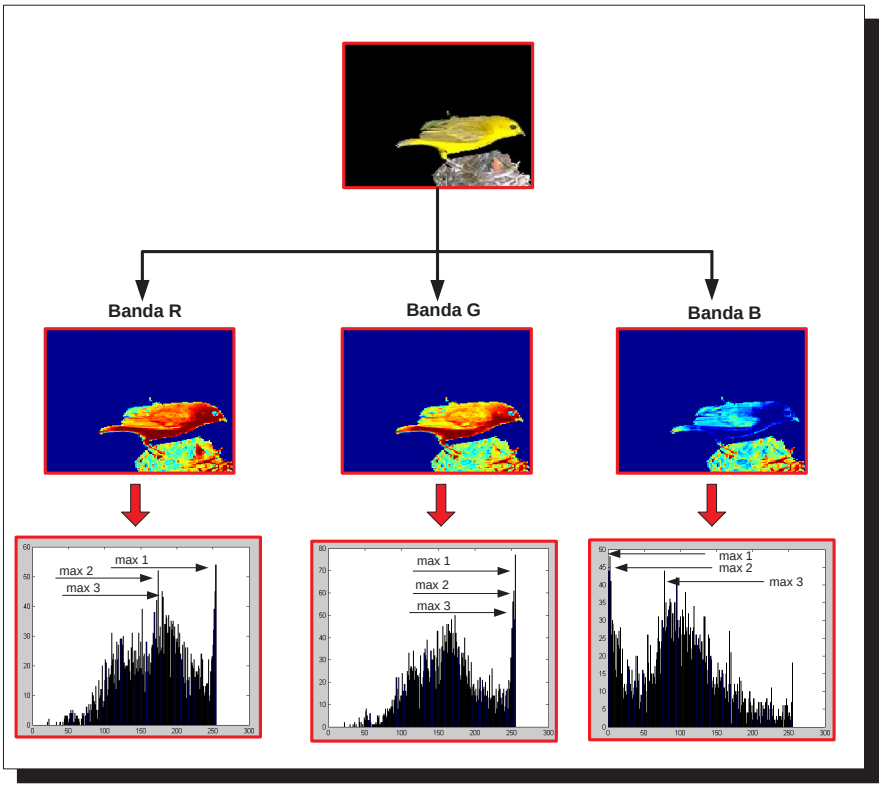

Figura 4.4: Ejemplo: Obtención de los tres valores máximos en cada una de las bandas R, G y B.

4.3. Aprendizaje semi-supervisado

En esta sección se describe la tercera etapa de la estrategía de etiquetado automático de imágenes usando un ensamble de clasificadores semi-supervisado. Esta etapa consiste en el diseño e implementación del algoritmo de aprendizaje semi-supervisado.

El paso clave de todo método semi-supervisado es saber hacer un buen uso de los ejemplos no etiquetados, sin hacer que éstos puedan empeorar la precisión del proceso de clasificación.

El enfoque propuesto en esta tesis asume que inicialmente se tienen dos conjuntos de imágenes segmentadas y que cada segmento de una imagen tiene asociado un vector de atributos. Uno de los conjuntos de imágenes segmentadas tiene una etiqueta asociada y el otro no. De esta manera, apartir de estos dos conjuntos, un conjunto más pequeño previamente etiquetado y otro conjunto más grande no

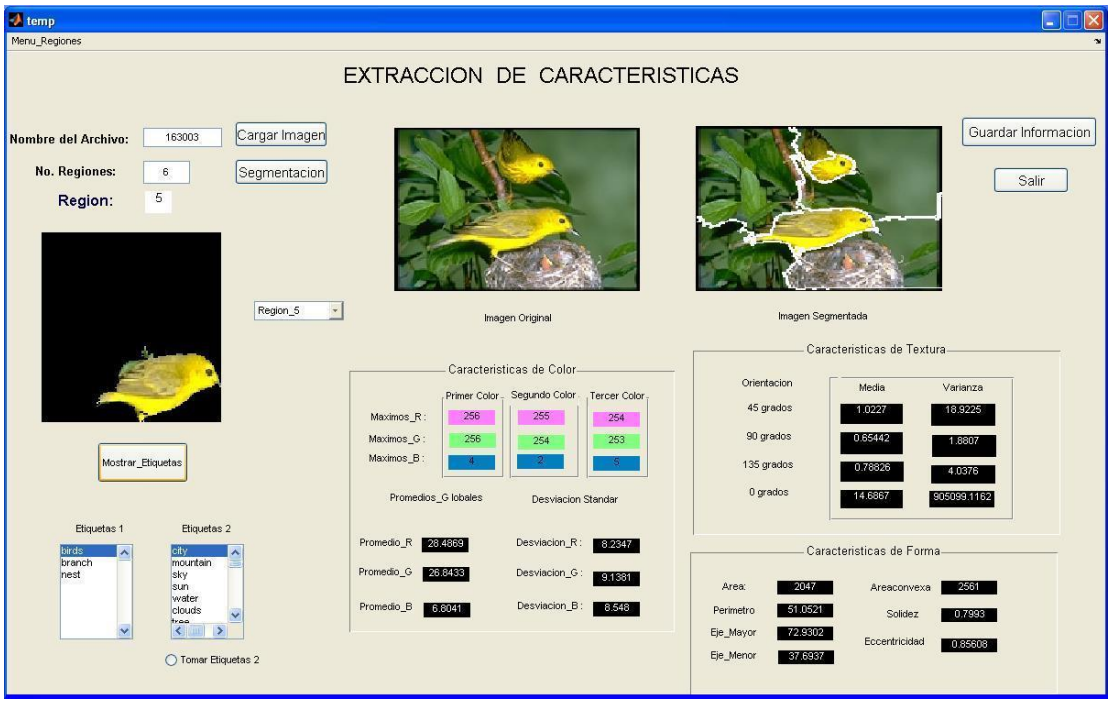

Figura 4.5: Interfaz gráfica para segmentación y extracción de características.

etiquetado, se aprenderá un modelo para predecir etiquetas para nuevas regiones de imágenes. Una vez entrenado, el modelo puede usarse para etiquetar imágenes automáticamente, realizando primero un proceso de segmentación y de extracción de características y posteriormente un proceso de clasificación de imágenes.

En las siguientes secciones se muestra la estructura de los algoritmos semi-supervisados que se desarrollaron en este trabajo para el etiquetado automático de imágenes. El primer algoritmo que se desarrolló fue llamado algoritmo SA (*Semi-supervised AdaBoost*). Este algoritmo está basado en la estructura del algoritmo AdaBoost [19] pero modificado a un enfoque semi-supervisado. SA realiza una combinación lineal de clasificadores en cascada para etiquetar nuevos ejemplos. La idea de SA es combinar datos etiquetados y no etiquetados otorgando ciertos pesos a los ejemplos mal etiquetados.

El segundo algoritmo que se desarrolló fue denominado WSA (*Weighted Semi-supervised AdaBoost*). Como en el algoritmo AdaBoost, la idea de generar un ensamble lineal de clasificadores en cascada, así como la asignación de pesos a los ejemplos no etiquetados, es adoptada tanto en el algoritmo SA como en el

algoritmo WSA. Sin embargo, la estructura de WSA y sus resultados son diferentes del los otros dos algoritmos. La diferencia clave es que en el algoritmo WSA se maneja una medida de confianza para la asignación de pesos en los datos no etiquetados, la cual permite aumentar el peso de los ejemplos mal clasificados por el clasificador anterior del ensamble para que sean reconsiderados por los nuevos clasificadores del ensamble. Esta medida de confianza hace de WSA un algoritmo mas confiable y robusto. La incorporación de pesos en los ejemplos permite eliminar el sesgo que se genera al manejar una distribución distinta de ejemplos etiquetados y no etiquetados. La principal ventaja del algoritmo WSA es el poder explotar el uso de una gran cantidad de datos no etiquetados para reducir la tarea de etiquetado de imágenes, y por consiguiente mejorar el aprendizaje en el ensamble.

Una descripción más detallada de los algoritmos SA y WSA se presenta a continuación.

4.3.1. Algoritmo AdaBoost Semi-supervisado (SA)

SA se muestra en el algoritmo 4.1. SA combina datos etiquetados y datos no etiquetados asignando el mismo peso a las instancias no etiquetadas. Inicialmente recibe como entrada un conjunto de instancias etiquetadas L y un conjunto de instancias no etiquetadas U. A cada instancia en L se le asigna un peso inicial $W_0(x_i) = 1/|L|$. Un clasificador inicial C_1 (NaiveBayes) se construye usando L y el peso $W_0(x_i)$ de las instancias en L. Las etiquetas en L permiten calcular el error en C_1, el cual se obtiene como $e_1 = \sum_{i=1}^{N} W_0(x_i) \quad \text{if } h_1(x_i) \neq y_i, \forall x_i \in L$. El factor de error B_1 se usa para actualizar los pesos de las instancias, disminuyendo el peso de las instancias correctamente clasificadas. SA asigna un peso $W_1(x_i) = 1/|U|$ a las instancias en U. El clasificador inicial C_1 se usa para predecir la clase de los ejemplos no etiquetados. Todos los pesos de $L \cup U$ son normalizados con el objetivo de que éstos no se vuelvan demasiado pequeños y de esta manera se eviten errores de precisión. El siguiente clasificador C_2 se construye utilizando los pesos y las clases predichas de $L \cup U$. El error del clasificador C_2 se obtiene apartir de las etiquetas de $L \cup U$. Este error es calculado como $e_t = \sum_{i=1}^{N} W_t(x_i) \quad \text{if } h_t(x_i) \neq y_i$. En las siguientes iteraciones los pesos de los ejemplos en L y U se van actualizando a partir del error cometido por C_i. Este proceso se realiza un

número predefinido de iteraciones T como en AdaBoost.

Algoritmo 4.1 Semi-Supervised AdaBoost (SA).

Entrada: L: Instancias Etiquetadas, U: Instancias No Etiquetadas, T: Iteraciones
Salida: Hipótesis final y probabilidades: $H_f = argmax_{y \in Y} \sum_{t=1}^{T} log\frac{1}{B_t} \cdot h_t(x, y)$,

1: $W_0(x_i) = \frac{1}{|L|}, \quad \forall x_i \in L$ {Pesos iniciales para L}
2: $h_1 = C_1(L, W_0(x_i))$ {Clasificador inicial}
3: Porcentaje de error sobre L: $e_1 = \sum_{i=1}^{N} W_0(x_i) \quad \text{if } h_1(x_i) \neq y_i, \forall x_i \in L$
4: $B_1 = \frac{e_1}{(1-e_1)}$
5: $W_1(x_i) = W_0(x_i) \cdot B_1 \quad \text{if } h_1(x_i) = y_i, \forall x_i \in L$
6: $W_1(x_i) = \frac{1}{|U|} \quad \forall x_i \in U$, Pesos para U
7: **for** t = 2 hasta T **do**
8: $\quad W_t(x_i) = \frac{W(x_i)}{\sum_{i=1}^{N} W(x_i)} \forall x_i \in L \cup U$ {Pesos Normalizados}
9: $\quad h_t = C_t(L \cup U, W_t(x_i))$
10: $\quad e_t = \sum_{i=1}^{N} W_t(x_i) \quad \text{if } h_t(x_i) \neq y_i$
11: $\quad$ **if** $e_t \geq 0.5$ **then**
12: $\quad\quad$ Salir del ciclo
13: $\quad$ **end if**
14: $\quad B_t = \frac{e_t}{(1-e_t)}$
15: $\quad W_{t+1}(x_i) = W_t(x_i)^t \cdot B_t \quad \text{if } h_t(x_i) = y_i \quad \forall x_i \in L$
16: $\quad W_{t+1}(x_i) = \frac{1}{|U|} \quad \forall x_i \in U$
17: **end for**

Las principales diferencias del algoritmo SA con el algoritmo ASSEMBLE son:

- En el algoritmo SA la clase inicial de los ejemplos de U se obtiene utilizando el algoritmo NaiveBayes, mientras que en el algoritmo ASSEMBLE la clase inicial o las pseudoclases de los ejemplos de U se obtienen utilizando el algoritmo k-NN.

- En ASSEMBLE la actualización de los pesos de todos los ejemplos de $L \cup U$ asume que las clases iniciales o pseudoclases de los ejemplos de U se encuentran correctamente predichas, por su parte en SA los pesos de L se actualizan si se conoce la clase de los ejemplos de U y luego se normalizan.

- ASSEMBLE usa sólo los ejemplos en $|L|$, tomando sólo una muestra en base a los pesos de $|L \cup U|$ de una distribución exponencial, mientras que SA usa todos los ejemplos de $L \cup U$.

4.3.2. Algoritmo AdaBoost Semi-supervisado con Pesos - (WSA)

El nuevo ensamble semi-supervisado de clasificadores que se propone llamado WSA, está basado en la técnica AdaBoost y usa el clasificador Naive Bayes simple como clasificador base. Un conjunto de estos clasificadores se combina en cascada basándose en la estructura de AdaBoost. WSA considera a las imágenes no etiquetadas en cada iteración del ensamble. Estas imágenes son etiquetadas por el clasificador del ensamble que se utilizó en la iteración anterior, y posteriormente son usadas para entrenar al siguiente clasificador. Las instancias no etiquetadas son pesadas de acuerdo a una medida de confianza basada en su valor de probabilidad predictiva, mientras que las instancias etiquetadas son pesadas de acuerdo al error del clasificador como en AdaBoost. El algoritmo WSA trabaja bien en la práctica, ya que los resultados muestran que el enfoque de este algoritmo es efectivo sobre un número de ejemplos de prueba, produciendo en algunos casos mayor precisión que los ensambles SA y AdaBoost usando los mismos bancos de entrenamiento y de prueba.

WSA permite determinar el número de clasificadores base que van a conformar el ensamble. Así, también permite que el conjunto de ejemplos de entrenamiento $L \cup U$ esté compuesto de $x_1, ..., x_m$ atributos m-dimensionales con clases conocidas $y_1, ..., y_m$. Para cada una de las pruebas que se realizaron se tomaron conjuntos de entrenamiento de distintos tamaños.

WSA se muestra en el algoritmo 4.2 y trabaja de la siguiente forma. Recibe como entrada un conjunto de datos etiquetados L y un conjunto de datos no etiquetados U. El ensamble de clasificadores F(X) esta formado de una combinación lineal de NB_j clasificadores base. Todos los ejemplos de L tienen un peso inicial $W_0(x_i) = 1/|L|$. Un clasificador inicial NB_1 se construye usando L. Las etiquetas en L se usan para evaluar el error de NB_1. Como en AdaBoost, el error se usa para asignar pesos a los ejemplos, decrementando el peso a los ejemplos bien clasificados, de manera que WSA multiplica el peso de todos los ejemplos correctamente etiquetados por un factor $B_t < 1$. El clasificador inicial, NB_1, se usa para predecir la clase de los ejemplos no etiquetados de U asignando cierta probabilidad a cada clase. La clase con la más alta probabilidad predictiva de cada instancia en U es seleccionada, y el peso de cada instancia se reduce multiplicando a éste

por la probabilidad de la clase. Los ejemplos no etiquetados con alta probabilidad sobre su clase predicha tendrán mayor influencia en la construcción del siguiente clasificador que los ejemplos con bajas probabilidades, reduciendo de esta manera una posible influencia presentada por etiquetas no confiables en el proceso de aprendizaje.

Todos los pesos de $L \cup U$ se normalizan, incrementando el peso de los ejemplos mal clasificados en L y reduciendo la influencia de los ejemplos en U con bajo valor de probabilidad sobre la clase. Los pesos de los ejemplos en U se asignan multiplicando la probabilidad predictiva de la clase $P(x_i, h_i)$ de los ejemplos no etiquetados por el factor de error B_t actualizado en cada clasificador del ensamble. El peso de los ejemplos en L se calcula de la siguiente forma. Si las clases predichas por el clasificador actual son iguales a las clases reales, se multiplica su peso actual $W_t(x_i)^t$ por el factor de error B_t actualizado, como en el algoritmo AdaBoost.

El siguiente clasificador NB_2 se construye usando los pesos y las clases predichas de $L \cup U$. NB_2 hace nuevas predicciones sobre U, y el error de NB_2 se usa para actualizar el peso de los ejemplos. Nuevamente el error se usa para obtener el factor B_{t+1}, el cual se usa para actualizar el error de clasificación de todos los ejemplos en L y todos los ejemplos en U. Este proceso continúa como en AdaBoost, para un número predefinido de iteraciones o cuando un clasificador tiene un error mayor o igual a 0.5, este umbral permite determinar que el clasificador tenga una precisión superior al 50 %. Como en AdaBoost, nuevas instancias se clasifican usando una suma pesada de la clase predicha de todos los clasificadores base construidos.

Las principales diferencias del algoritmo WSA con el algoritmo AdaBoost son:

- WSA usa datos etiquetados y no etiquetados.
- Los clasificadores base crean nuevas etiquetas de clase para las instancias no etiquetadas.
- Los pesos asignados a los datos no etiquetados originales dependen de la probabilidad de clase predicha, lo cual es también una diferencia con ASSEMBLE.

- WSA no guarda un historial de los pesos anteriores W_t de los ejemplos de U, ya que pueden cambiar de clase con el tiempo.

Algoritmo 4.2 Semi-supervised Weighted AdaBoost (WSA)

Entrada: L: Instancias Etiquetadas, U: Instancias No Etiquetadas, T: Iteraciones
Salida: Hipótesis final y probabilidades: $H_f = argmax_{y \in Y} \sum_{t=1}^{T} log\frac{1}{B_t} \cdot h_t(x, y)$,

1: $W_0(x_i) = \frac{1}{|L|}, \quad \forall x_i \in L$ {Pesos iniciales para L}
2: $h_1 = C(L, W_0(x_i))$ {Clasificador inicial}
3: $e_1 = \sum_{i=1}^{N} W_0(x_i) \quad \text{if } h_1(x_i) \neq y_i, \quad \forall x_i \in L$
4: $B_1 = \frac{e_1}{(1-e_1)}$
5: $W_1(x_i) = W_0(x_i) \cdot B_1 \quad \text{if } h_1(x_i) = y_i, \quad \forall x_i \in L$
6: $W_1(x_i) = P(x_i, h_1) \cdot B_1, \quad \forall x_i \in U$, donde $P(x_i, h_1)$ es el valor más alto de probabilidad de clase para la instancia i {Pesos iniciales para U}
7: **for** t = 2 hasta T **do**
8: $W_t(x_i) = \frac{W(x_i)}{\sum_{i=1}^{N} W(x_i)} \quad \forall x_i \in L \cup U$ {Pesos Normalizados}
9: $h_t = C(L \cup U, W_t(x_i))$
10: $e_t = \sum_{i=1}^{N} W_t(x_i) \quad \text{if } h_t(x_i) \neq y_i \quad \forall x_i \in L \cup U$
11: **if** $e_t \geq 0.5$ **then**
12: Salir del ciclo
13: **end if**
14: $B_t = \frac{e_t}{(1-e_t)}$
15: $W_{(t+1)}(x_i) = W_t(x_i)^t \cdot B_t \quad \text{if } h_t(x_i) = y_i, \quad \forall x_i \in L$
16: $W_{(t+1)}(x_i) = P(x_i, h_t) \cdot B_t, \quad \forall x_i \in U$
17: **end for**

4.3.3. Sumario

La estrategia de etiquetado que se propuso en este capítulo involucra dos procesos, la segmentación de imágenes y la extracción de características. En el proceso de segmentación de imágenes se utilizó un algoritmo automático llamado Normalized Cuts. En la extracción de características se consideraron atributos de bajo nivel de las imágenes como forma, color y textura. Se desarrolló una interfaz gráfica para llevar a cabo ambos procesos, y etiquetar de forma manual algunas imágenes, todo esto, para formar el conjunto de entrenamiento del clasificador.

El algoritmo de etiquetado automático propuesto, llamado WSA, es un nuevo ensamble semi-supervisado de clasificadores bayesianos que considera a las instancias no etiquetadas en cada una de sus etapas, a las cuales se les asigna un peso de

acuerdo a una medida de confianza basada en su valor de probabilidad predictiva. Ésta es la principal diferencia con el algoritmo ASSEMBLE que utiliza asignación de pseudo-clases mediante el algoritmo k-NN para las instancias no etiquetadas. En WSA las instancias etiquetadas son pesadas de acuerdo al error del clasificador como en el algoritmo AdaBoost.

El uso de pesos permite eliminar el error que se puede generar al trabajar con diferentes distribuciones de ejemplos etiquetados y no etiquetados. La información que proveen los datos no etiquetados puede mejorar la precisión de clasificación o puede empeorarla si no se maneja esa información correctamente, es decir, si se le da mayor influencia a los datos no etiquetados cuyo valor de clase se desconoce en lugar de los datos etiquetados en los cuales se conoce su valor de clase.

En el siguiente capítulo se presentan los resultados obtenidos al evaluar el ensamble semi-supervisado WSA con diferentes bases de datos. Además, el desempeño del algoritmo propuesto es comparado contra otros algoritmos de aprendizaje.

Capítulo 5

Resultados

En este capítulo se presentan los resultados de la evaluación experimental de los algoritmos SA y WSA para la tarea de clasificación, utilizando la estrategia de etiquetado automático que se describió en el capítulo 4. También se presenta una comparación de los resultados obtenidos por el algoritmo WSA contra otros algoritmos AdaBoost, SA y NaiveBayes.

El ensamble fue evaluado en 3 etapas. En la primer etapa de pruebas se utilizaron bases de datos del repositorio UCI de Machine Learning [30] para comprobar la eficiencia y el funcionamiento del ensamble. En la segunda etapa de pruebas se utilizó la base de datos de Corel [1], de la cual se selecionaron algunos conjuntos de datos de distintos tópicos. Las imágenes de Corel fueron consideradas para la segunda etapa de pruebas debido a su diversidad y claridad en el contenido de las imágenes. Además, estas imágenes han sido ampliamente utilizadas en otros trabajos sobre etiquetado y recuperación de imágenes. En la tercera etapa de pruebas se utilizaron imágenes a color de las bases de datos IAPR TC-12 [21] y PASCAL 2007 VOC [30] de ImageCLEF 2007. Estas bases de datos son de libre acceso y manejan una gran cantidad de imágenes de diferente tipo (aproximadamente 29,000 imágenes).

En cada una de las pruebas con imágenes se utilizó la interfaz gráfica de la sección 4.2.1 para las tareas de segmentación, extracción de características y etiquetado manual de las imágenes. Los ensambles de clasificadores bayesianos SA y WSA se implementaron en el lenguaje de programación C++.

5.1. Módulo de validación del nivel de confianza

Para validar el ensamble WSA se propuso el algoritmo 5.1. Este módulo de validación del nivel de confianza permite obtener la precisión del proceso de clasificación de WSA. El módulo de validación está basado en la estructura de la técnica *k-fold-cross validation* presentada en la sección 2.6. A diferencia de la técnica k-fold-cross validation, este módulo de validación utiliza los conjuntos de datos etiquetados L y no etiquetados U en distintos porcentajes P_U para formar el conjunto de entrenamiento, el cual se divide en k-particiones. Una de las k-ésimas particiones p_i de $L \cup U$ forma el conjunto de prueba. El clasificador L se entrena con las $k-1$ particiones restantes y se evalúa con la k-ésima partición de prueba. Este proceso se repite k-veces. En las pruebas realizadas se utilizó un valor de k=10.

El ensamble WSA fue validado en cada una de las pruebas realizadas con las distintas bases de datos de imágenes (UCI, Corel e ImageCLEF).

Algoritmo 5.1 Módulo de validacion

Entrada: L: Datos Etiquetados, U: Datos No-Etiquetados, P_U: Porcentaje de datos no etiquetados.

Salida: E_p : Error Promedio

1: $D = L \cup U$
2: $p_1 \cup p_2 \cup ... \cup p_k = L$ donde p_i representa el conjunto de prueba
3: **for** $i = 0$ hasta k **do**
4: $U' \leftarrow P_U$ Porcentaje de ejemplos del conjunto $\{D - p_i\}$
5: $L' = (D - p_i) - U'$
6: Entrenar el modelo $h_i = M(L' \cup U')$
7: Evaluar el modelo $E_{p_i}(h_i)$
8: **end for**
9: Regresar el error promedio $E_p = \frac{1}{k} \sum_{i=1}^{k} E_{p_i}(h_i)$

5.2. Experimentos con bases de datos del repositorio UCI

Se evaluó el desempeño del algoritmo semi-supervisado WSA utilizando conjuntos del repositorio UCI [30], para obtener el porcentaje de ejemplos correcta-

Tabla 5.1: Características de las bases de datos del repositorio UCI.

Nombre	Instancias	Atributos	Clases
Iris	150	4	3
Balance-Scale	625	4	3
Wine	178	14	3
Lymphography	148	18	4
Tic-tac-toe	958	9	2
Pima-indians-diabetes	768	8	2

mente clasificados y de esta manera medir la precisión de clasificación.

La mayoría de las bases de datos con las que se trabajó en esta primera prueba no contienen valores de atributos ausentes. Sin embargo, en caso de haber valores ausentes el algoritmo no los considera. Además, las instancias que forman parte de cada una de estas bases de datos que se utilizaron son instancias etiquetadas. Para formar el conjunto de entrenamiento se determinó el porcentaje de datos etiquetados con el cual se trabajaría, estos datos fueron seleccionados aleatoriamente del conjunto total de la base de datos y se consideraron como datos no etiquetados al porcentaje restante de todas las instancias de la base de datos.

En la primera etapa de pruebas se utilizaron seis bases de datos del repositorio UCI: *Iris, Balance-Scale, Wine, Lymphography, Tic-tac-toe y Pime-indians-diabetes*, cuyas características se muestran en la tabla 5.1.

Para realizar una justa comparación entre WSA, SA, AdaBoost y NaiveBayes utilizamos los mismos atributos y datos para todos los algoritmos. Los valores (reales) de los atributos fueron discretizados en 10 *bins* o particiones de igual tamaño usando la herramienta WEKA [39]. Los cuatro algoritmos fueron evaluados mediante su precisión predictiva usando el módulo de validación de las sección 5.1 y utilizando diferentes porcentajes de datos no etiquetados en los conjuntos de entrenamiento así como clases balanceadas.

La figura 5.1 muestra las gráficas del desempeño de WSA, SA, AdaBoost y el algoritmo NaiveBayes. Como se puede apreciar en las gráficas, WSA tiene una mejor precisión que AdaBoost al usar una gran cantidad de instancias no etiquetadas. WSA también tiene un mejor desempeño que SA, mostrando que el usar el valor de probabilidad de clase sobre las instancias no etiquetadas puede

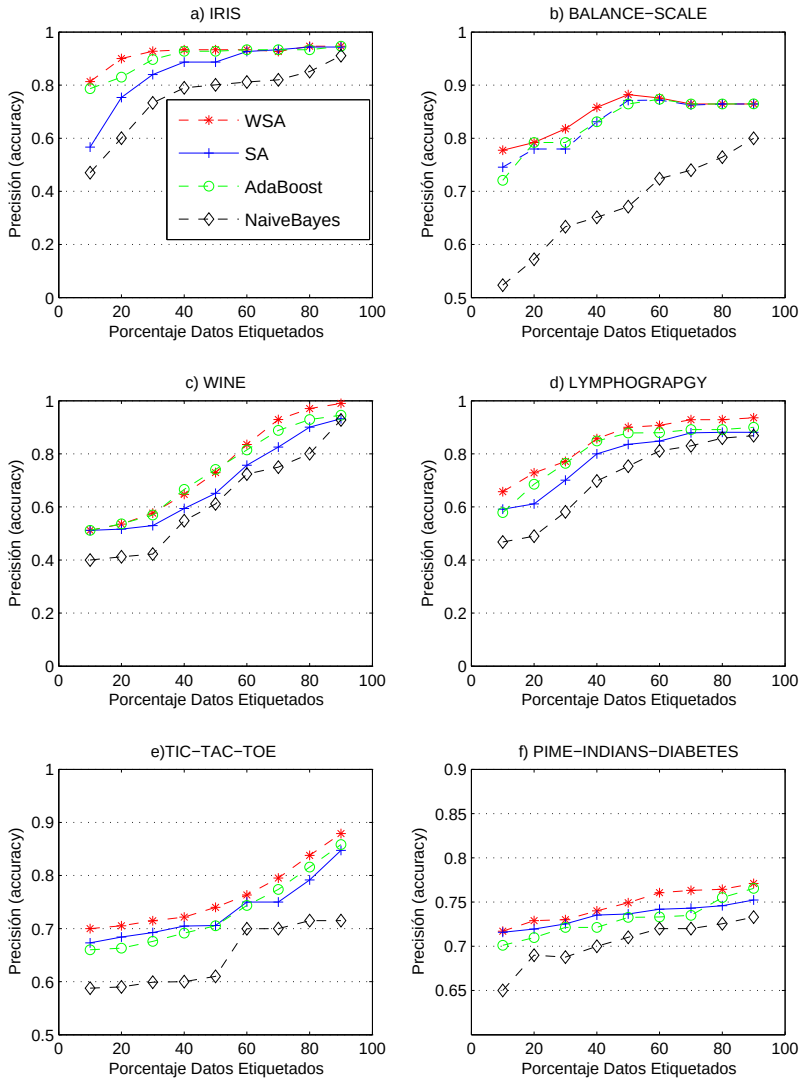

Figura 5.1: Gráficas de comparación del desempeño de WSA, SA, AdaBoost y Naive Bayes para 6 bases de datos del repositorio UCI.

tener un efecto positivo, tal como reducir el sesgo que los ejemplos no etiquetados pueden producir en el clasificador.

De la figura 5.1 se observa claramente que WSA es siempre mejor o al menos igual que los otros tres algoritmos. En algunos casos WSA obtiene mejores resultados cuando maneja una distribución pequeña de datos etiquetados, por ejemplo,

en las bases de datos de *Iris, Lymphography* y *Tic-tac-toe*, en donde cada una de estas bases de datos contienen una cantidad pequeña de ejemplos y manejan hasta tres clases. En la base de datos *Wine* se muestra claramente que el desempeño de WSA tiende a crecer con respecto al porcentaje de datos etiquetados que se maneja. Esta base de datos trabaja con un número de atributos más grande en comparación a las otras bases de datos. Con una misma distribución de datos etiquetados y no etiquetados el desempeño de WSA es similar al de los otros tres algoritmos. En conclusión, WSA es un ensamble semi-supervisado que muestra una buena precisión para clasificar nuevas instancias.

De los resultados obtenidos con las seis bases de datos del repositorio UCI, se realizó un análisis para medir el error del desempeño obtenido por WSA, SA, AdaBoost y NaiveBayes, utilizando la medida de estimación de error del algoritmo llamada *intervalo de confianza* [29], el cual fue construido mediante la siguiente distribución normal:

$$P(\text{-}1.96 < z < 1.96) = 0.95$$

donde z representa la probabilidad de error real, la cual suele estimarse empíricamente, clasificando una muestra de prueba independiente de tamaño n y haciendo un recuento del número de errores k: $z = \frac{k}{n}$.

En este caso se aplicó un intervalo de confianza del 95 %, ya que éste ha sido comunmente utilizado para estimar el error de un clasificador. Para este intervalo de confianza se consideró un 30 % de datos etiquetados. En la figura 5.2 se muestran los intervalos de confianza para las seis bases de datos. A partir de estos intervalos podemos afirmar el verdadero riesgo de error cometido por los algoritmos. De las gráficas de la figura 5.2 podemos observar que el error cometido por WSA es igual o menor que el error de los algoritmos AdaBoost y SA. Además, los ensambles WSA, SA y AdaBoost son significativamente mejores que NaiveBayes en todas las bases de datos. Por ejemplo, en la gráfica de la base de datos $Iris$, de la figura 5.2, se observa que el rango superior del intervalo de confianza del algoritmo WSA tiene una diferencia significativa de 0.09 comparada con el rango inferior del intervalo de confianza del algoritmo Naive Bayes. Esta diferencia permite establecer que el error cometido por WSA es menor en comparación con el algoritmo Naive Bayes.

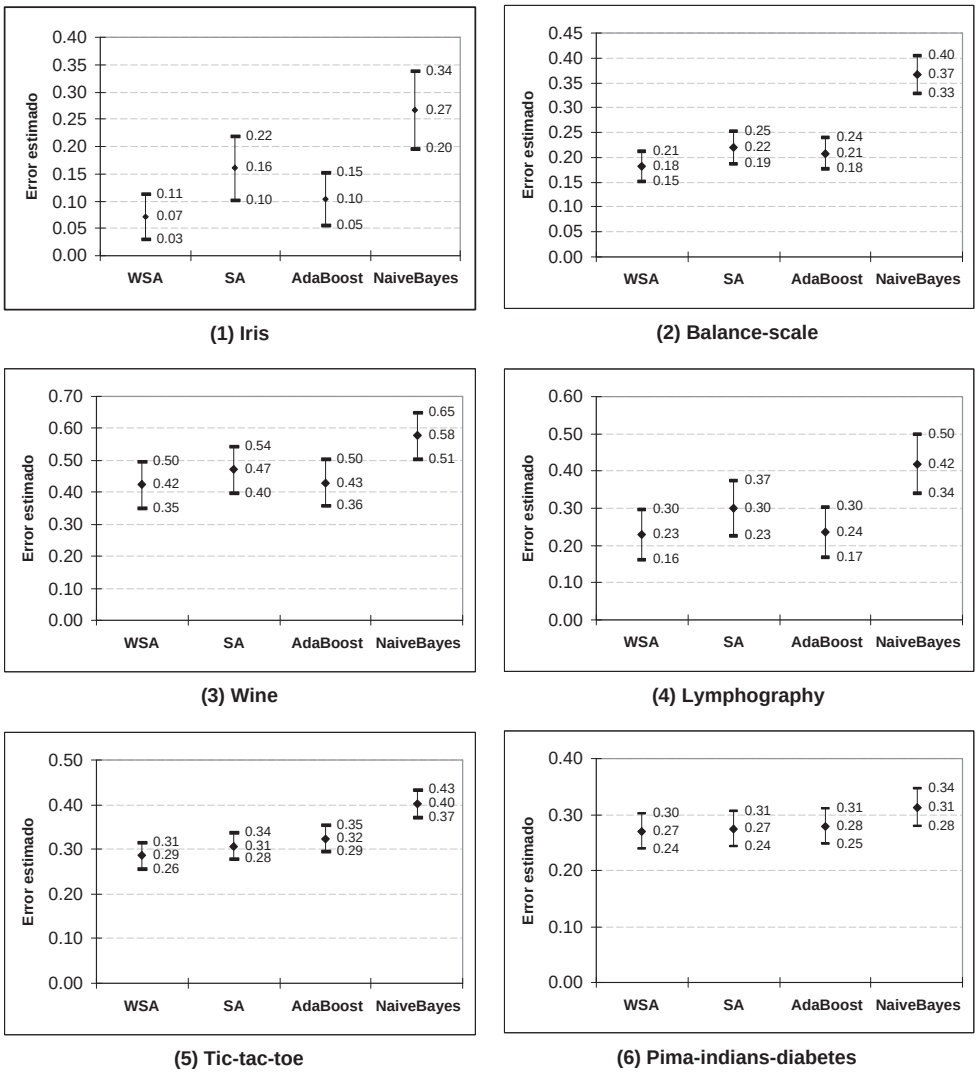

Figura 5.2: Gráficas de los intervalos de confianza al 95 % para seis bases de datos del repositorio UCI, comparando WSA, SA, AdaBoost y NaiveBayes con un sólo porcentaje de ejemplos etiquetados.

5.3. Imágenes de Corel

El algoritmo WSA fue probado sobre conjuntos de imágenes de la base de datos de Corel [1], las cuales están agrupadas de acuerdo a diferentes tópicos como paisajes, animales, construcciones, aeroplanos, autos y personas, entre otros. El tamaño de estas imágenes es de 192x128 pixeles. Las imágenes que se utilizaron fueron segmentadas con Normalized Cuts en 5 regiones y fueron extraídas 30 características visuales por cada región de la imagen, como se describió en la

Tabla 5.2: Características de los conjuntos de imágenes de Corel.

Nombre	Regiones	Clases	Distribución
Aeroplanos	127	sky	55
		jet	25
		clouds	31
		plane	12
		sunset	2
		helicopter	2
Aves	225	birds	59
		nest	32
		grass	47
		tree	14
		branch	45
		water	28
Paisajes	178	sun	57
		sea	31
		sky	19
		trees	19
		clouds	38
		building	14
Animales	148	fox	5
		sky	10
		reptile	9
		trees	8
		clouds	3
		bear	1
		grass	31
		water	11
		rock	34
		snow	36

sección 4.

De la base de datos de imágenes de Corel se utilizaron cuatro conjuntos de datos de los tópicos de: *Aeroplanos, Aves, Paisajes* y *Animales*. Para cada uno de estos cuatro conjuntos se utilizaron 100 imágenes, de las cuales se seleccionó para el entrenamiento un conjunto pequeño de regiones que mejor definieran al o a los objetos que se encontraban en las imágenes. Algunas de las características de estos cuatro conjuntos de imágenes se describen en la tabla 5.2.

En la tabla 5.3 se muestra el desempeño obtenido por WSA con los diferentes conjuntos de imágenes de la base de datos de Corel y distintos porcentajes de regiones etiquetadas. Para ello, se utilizó la interfaz gráfica de segmentación y extracción de características descrita en la sección 4.2.1. En la tabla 5.3 se

muestra que la precisión de clasificación de WSA es proporcional al procentaje de datos etiquetados que se utilizan. El porcentaje de precisión de clasificación se ve afectado por el tipo de imágenes que se utilice, por ejemplo, los mejores porcentajes de clasificación se obtienen con la base de datos *aeroplanos* debido a que este tipo de imágenes tienen pocos objetos y además, están bien definidos, lo cual hace que el nivel de clasificación sea mejor en comparación con los otros tipos de imágenes como las del conjunto de *animales*, el cual contiene imágenes difíciles de segmentar.

Tabla 5.3: Porcentaje de precisión de clasificación de WSA para los cuatro conjuntos de imágenes de Corel, utilizando porcentajes variables de imágenes etiquetadas.

Nombre	Instancias	Clases	**Porcentaje Datos Etiquetados**				
			10 %	30 %	50 %	70 %	90 %
Aeroplanos	127	6	40.43	55.00	76.66	90.08	99.16
Aves	225	6	32.08	51.16	74.16	86.25	90.10
Paisajes de Atardecer	178	6	35.00	40.62	43.75	64.37	80.75
Animales	148	10	22.50	40.00	48.75	58.75	60.00

En la figura 5.3 se hace una comparación de la precisión de los algoritmos WSA, SA, AdaBoost y NaiveBayes. En las gráficas se muestra que el ensamble WSA tiene mejor precisión de clasificación en comparación a SA y AdaBoost al incorporar la medida de confianza basada en la probabilidad predictiva para la asignación de pesos a los ejemplos no etiquetados. También se muestra que el ensamble es mejor que un clasificador como NaiveBayes. Uno de los aspectos que afecta la precisión de WSA es la segmentación automática de las imágenes, ya que algunas son difíciles de segmentar, como las que se muestran en la figura 5.4.

Nuevamente para comprobar el verdadero error de clasificación cometido por WSA se aplicó un intervalo de confianza del 95 %, considerando un 30 % de datos etiquetados ya que es uno de los porcentajes en donde existe una diferencia significativa de la precisión de WSA con respecto a los otros tres algoritmos. En la figura 5.5 se muestran los intervalos para cuatro conjuntos de imágenes de la base de datos de Corel.

A partir de los intervalos que se muestran en la figura 5.5, se puede afirmar que el error estimado por WSA, en la mayoría de los casos es menor que el error

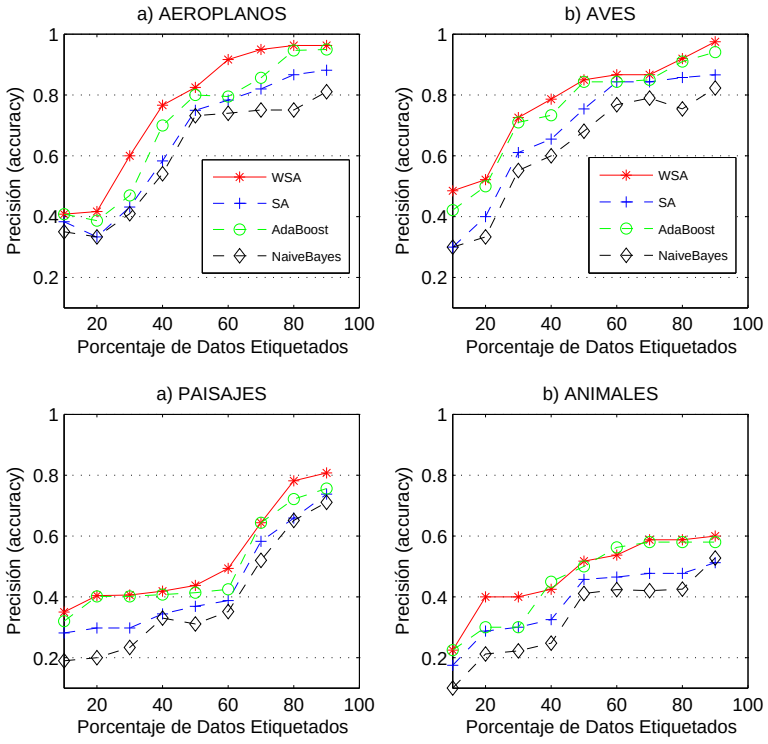

Figura 5.3: Gráficas del desempeño del algoritmo WSA usando imágenes de la base de datos de Corel.

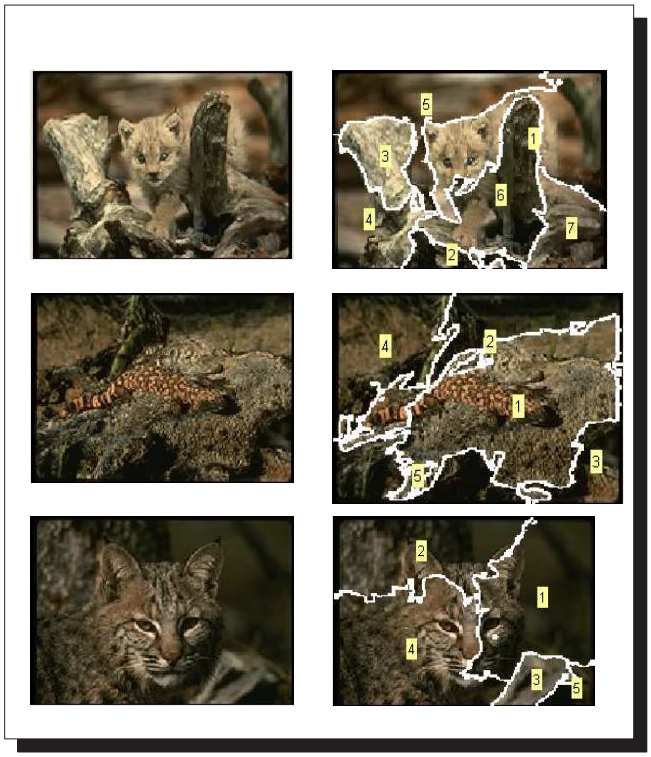

Figura 5.4: Ejemplos de imágenes difíciles de segmentar.

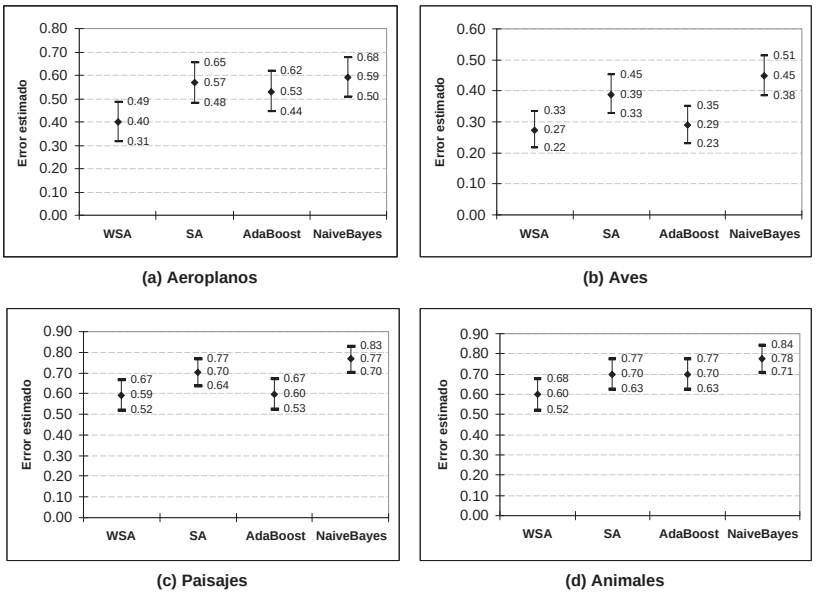

Figura 5.5: Gráficas de los intervalos de confianza al 95 % para cuatro bases de datos de Corel.

estimado en los otros algoritmos. Por ejemplo, en la gráfica de imágenes de aeroplanos se muestra una diferencia de error sigficativa entre WSA y los algoritmos SA y NaiveBayes. También se observa que el rango superior del intervalo de confianza de WSA toca parte del rango inferior del intervalo de confianza de SA. Sin embargo, hay una diferencia significativa entre la media muestral de ambos intervalos. Se observa que WSA tiene una media muestral menor a SA y por tanto el error estimado por WSA es menor.

5.4. Imágenes de las bases de datos de ImageCLEF 2007

Las bases de datos de imágenes IAPR TC-12 y Pascal 2007 VOC que se utilizaron en ImageCLEF 2007 [23] contienen una gran cantidad de imágenes en comparación al número de imágenes de Corel. La colección de imágenes Bechmark IAPR TC-12 [21] consta de 20,000 imágenes naturales sin movimiento. Esta colección contiene dibujos de diferentes deportes y acciones, fotografías de gente, animales, ciudades, paisajes y muchos otros aspectos de la vida contemporánea. En la figura 5.6 se muestran algunas de estas imágenes.

La colección de imágenes de la base de datos de Pascal 2007 VOC [2] tiene un total de 9,963 imágenes y aproximadamente 24,640 objetos anotados. Los objetos anotados de cada imagen tienen un archivo de anotación dado por una etiqueta por cada objeto presente en la imagen. Además de que múltiples clases pueden estar presente en la misma imagen. En la figura 5.7 se muestran algunas de estas imágenes.

Para evaluar el desempeño del algoritmo semi-supervisado WSA se realizó una prueba utilizando la base de datos Pascal 2007 VOC. En esta prueba se etiquetaron manualmente alrededor de 2412 regiones de la base de datos. De este conjunto se utilizaron diferentes porcentajes de ejemplos etiquetados que formaron parte del conjunto de entrenamiento del ensamble WSA. En la tabla 5.4 se muestran los resultados del nivel de precisión obtenidos por el algoritmo WSA, variando el porcentaje de datos etiquetados.

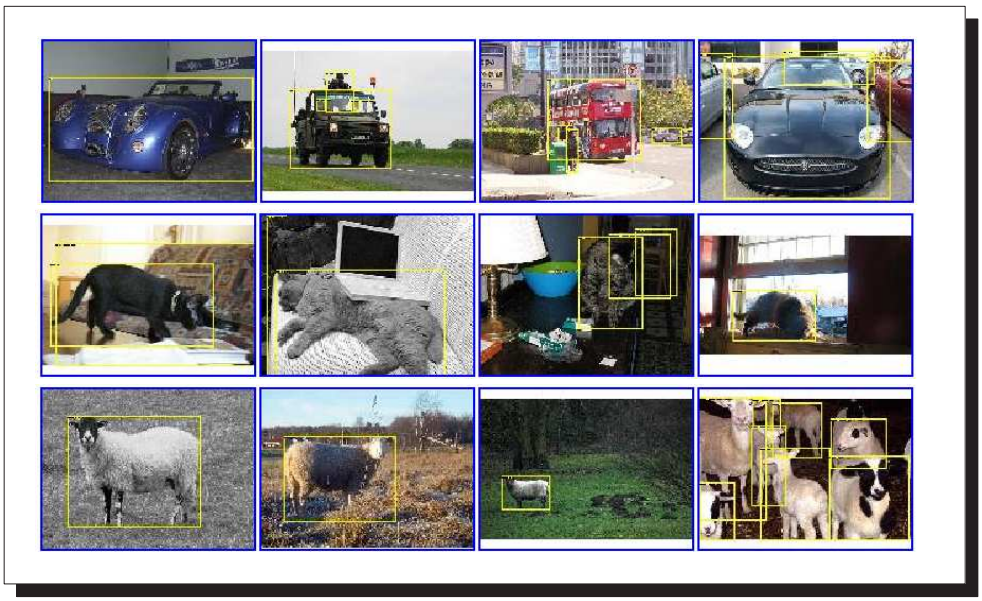

Figura 5.6: Imágenes de la base de datos de IAPR TC-12.

En esta prueba se obtuvo una precisión de clasificación menor, comparada con la precisión obtenida en las pruebas realizadas con la base de datos de imágenes de Corel, debido a distintos factores. Algunos de estos factores fueron:

- El contenido de las imágenes es variable, es decir, no existe alguna organización de tópicos sobre el tipo de imagen, todas las imágenes se encuentran en el mismo conjunto.
- El conjunto de etiquetas fue reducido, lo que ocasionó tener una diversidad

Tabla 5.4: Precisión de WSA utilizando las imágenes de la base de datos de Pascal 2007 VOC.

Nombre	Instancias	**Porcentaje Datos Etiquetados**			
		60 %	70 %	80 %	90 %
Prueba-1	229	38.636	45.000	63.181	82.272
Prueba-2	485	36.315	44.736	47.894	60.000
Prueba-3	1058	37.000	37.619	38.380	39.046
Prueba-4	1938	31.956	32.000	34.803	35.845
Prueba-5	2412	31.333	33.120	34.123	35.145

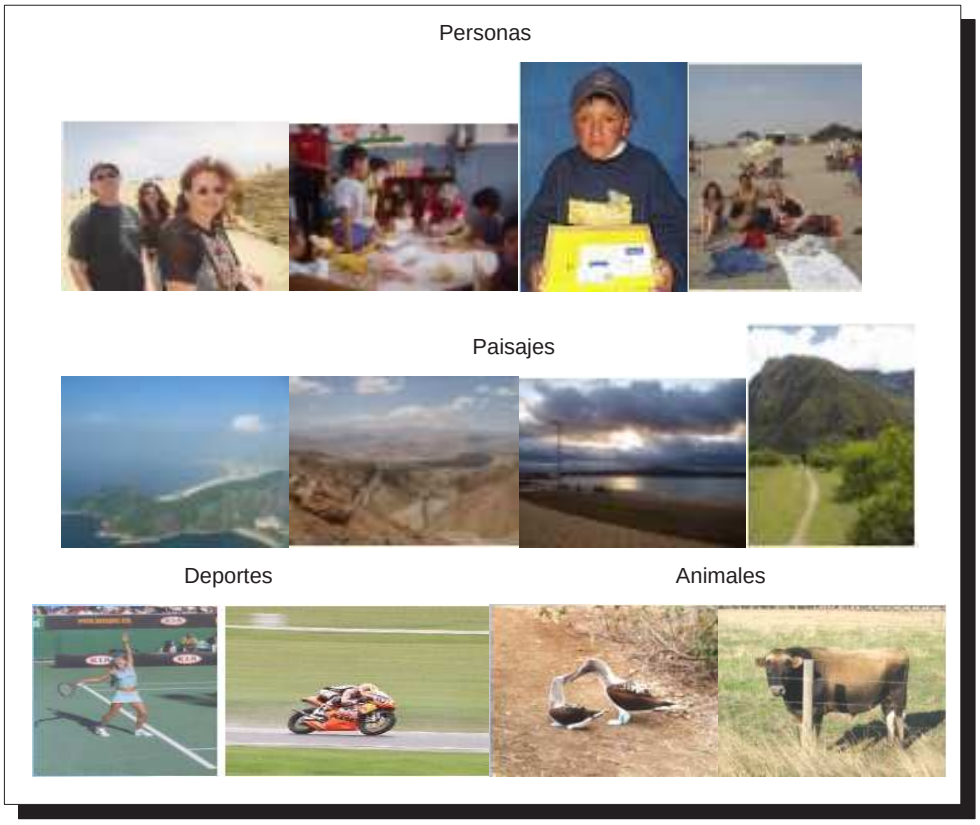

Figura 5.7: Imágenes de la base de datos de PASCAL 2007 VOC.

de imágenes con la misma etiqueta y que los criterios para la asignación de etiquetas no fuera consistente. Se consideraron 10 etiquetas: *rock, motorbike, bicycle, bus, person, sheep, sand, road, building y vehicle*.

- El conjunto de imágenes de Pascal 2007 VOC contenía imágenes difíciles de segmentar.

Es importante mencionar que los resultados obtenidos dependen del conjunto de entrenamiento que se obtenga del proceso de segmentación y extracción de características. En la mayoría de los casos, el algoritmo de segmentación no obtiene buenos resultados, lo cual ocasiona que el proceso de clasificación sea deficiente. En esta prueba se hizo evidente la necesidad de establecer y aplicar los mismos criterios para asignar las etiquetas al conjunto de entrenamiento, ya que una mala asignación de etiquetas lleva a tener resultados de clasificación deficientes.

El algoritmo WSA participó en la competencia de recuperación de objetos

ImageCLEF 2007, organizada por Allan Hanbury y Thomas Deselaers [23]. WSA fue usado para la tarea de etiquetado automático de imágenes digitales. Para el entrenamiento de los datos se utilizó la base de datos Pascal 2007 VOC y la base de datos IAPR TC-12 como el conjunto de prueba. Entre estas dos bases de datos se tienen aproximadamente unas 29,000 imágenes. Tomando en consideración que en la prueba realizada con la base de datos Pascal 2007 VOC se habian etiquetado alrededor de 2412 regiones, estas regiones fueron tomadas para formar el conjunto de entrenamiento. El nivel promedio de precisión alcanzado por WSA y otros de los algoritmos que participaron en esta competencia, se encuentra disponible en [22] y está por publicarse en [15]. WSA obtuvo un nivel promedio de precisión bajo a causa de algunos factores como el manejar un conjunto reducido de etiquetas, trabajar con imágenes difíciles de segmentar y utilizar imágenes de diferentes tópicos.

5.5. Discusión de resultados

Para evaluar el ensamble propuesto se utilizaron distintas bases de datos de imágenes y se realizaron comparaciones de precisión de clasificación con tres algoritmos, AdaBoost, SA y NaiveBayes. El ensamble WSA fue validado utilizando el módulo de validación que se propusó en la sección 5.1 en donde se consideraron como parte del conjunto de entrenamiento ejemplos etiquetados y no etiquetados.

En la primer prueba se utilizaron seis bases de datos del repositorio UCI. En esta prueba WSA obtuvó mejores porcentajes de precisión usando una cantidad pequeña de ejemplos etiquetados.

En la segunda prueba se utilizaron imágenes de la base de datos de Corel. Los resultados de precisión obtenidos por WSA dependieron del tipo de imágenes con el cual se trabajó. Por ejemplo, los mejores resultados se obtuvieron con imágenes que contienen objetos bien definidos y por tanto, se obtuvieron altos niveles de precisión de segmentación como en el caso de las imágenes de *aeroplanos*. Con imágenes difíciles de segmentar como las de *animales*, los resultados de clasificación fueron bajos.

En la tercera prueba se utilizaron imágenes de la base de datos Pascal 2007 VOC de ImageCLEF. A diferencia de las pruebas anteriores, en ésta se utilizó

una mayor cantidad de imágenes. Los resultados obtenidos por el ensamble WSA fueron bajos de aproximadamente un 30 % de precisión, a causa de diferentes factores como imágenes difíciles de segmentar, conjunto reducido de etiquetas, diversidad de imágenes de diferentes tópicos, entre otros. Sin embargo, cabe señalar que en la literatura se encuentran otros algoritmos de clasificación que han utilizado estas bases de datos y también han obtenido bajos niveles de precisión [23].

En general, los resultados de clasificación obtenidos con las bases de datos de imágenes se ven afectados debido a que hasta el momento no existe ningún algoritmo de segmentación automática que divida correctamente cualquier tipo de imágenes.

A partir de los resultados obtenidos en estas pruebas, se demostró que el enfoque semi-supervisado permite mejorar los niveles de clasificación comparado con en enfoque supervisado, en donde no puede explotar el uso de ejemplos no etiquetados. Además la asignación de pesos basada en una medida de confianza a los ejemplos no etiquetados mejora la clasificación reduciendo el sesgo que se genere por el manejo de una distribución distinta de datos etiquetados y no etiquetados.

De acuerdo a los resultados mostrados y comparados con otro ensamble como AdaBoost, se demostró que el enfoque propuesto consigue una asignación de etiquetas más precisa.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Sumario

La solución que se plantea en esta tesis para el problema del etiquetado automático está dividida en 3 etapas: pre-procesamiento de las imágenes, proceso de clasificación y validación del nivel de predicción. En la primer etapa se utilizaron dos métodos para el pre-procesamiento de las imágenes:

1. Segmentación de imágenes.
2. Extracción de características.

Se aplicó un algoritmo de segmentación a cada una de las imágenes del conjunto de datos, con el objetivo de definir lo más posible las regiones u objetos que forman parte de la imagen. El algoritmo de segmentación seleccionado fue Normalized Cuts, el cual ha sido ampliamente utilizado en varios trabajos sobre recuperación de imágenes por contenido. Como cualquier otro algoritmo de segmentación, Normalized Cuts no funciona bien para cualquier imagen, es decir, sólo trabaja bien con aquellas imágenes que tienen una buena resolución y que contienen objetos bien definidos.

Por otra parte, las características que fueron extraídas de cada una de las regiones de las imágenes permitieron considerar una mayor información de los objetos presentes en la imagen.

En la segunda etapa de la solución se diseño e implementó el ensamble semi-supervisado de clasificadores bayesianos WSA para el proceso de clasificación de las imágenes. En esta etapa se consideraron tres enfoques para la clasificación:

1. Ensamble supervisado AdaBoost construido de una manera tradicional, en donde todas las predicciones se obtienen mediante un voto mayoritario.
2. Ensamble semi-supervisado SA que trabaja con ambos tipos de datos sin otorgarle una medida de confianza de probabilidad para las instancias no-etiquetadas.
3. Ensamble semi-supervisado WSA que mejora a SA asignándole un peso a las instancias no etiquetadas.

En el primer enfoque, usando las bases de datos de Corel y del repositorio UCI se lograron obtener buenos resultados con precisiones mayores al 90 % de instancias correctamente clasificadas. Cuando se tiene una cantidad pequeña de instancias se mostró que es mejor utilizar el algoritmo WSA que el algorimo AdaBoost.

Con el segundo enfoque del ensamble semi-supervisado SA, se mostró que era indispensable establecer una medidad de confianza a las instancias no etiquetadas. Los resultados de SA estuvieron por debajo del ensamble AdaBoost supervisado.

En el tercer enfoque del ensamble WSA se obtuvieron mejores resultados comparado con SA y AdaBoost. Sin embargo, este enfoque podría mejorar si se generaran conjuntos de entrenamiento de imágenes que fueran confiables, es decir, que el conjunto de entrenamiento se formara con regiones bien divididas dentro de las imágenes y se aplicaran los mismos criterios para el etiquetado manual de las imágenes.

A partir de los resultados obtenidos en las pruebas, se demostró que el enfoque semi-supervisado en WSA, permite mejorar los niveles de clasificación que un enfoque supervisado, como AdaBoost. Además, al considerar WSA una medida de confianza para la asignación dinámica de pesos en los ejemplos no etiquetados,

se mejoró la clasificación a diferencia de SA que solamente realiza una asignación de pesos estática.

En las pruebas realizadas se aplicaron intervalos de confianza del 95 % para poder estimar el error del clasificador, en todos los casos WSA obtuvo el menor error de clasificación comparado contra los otros algoritmos. Además, mostró tener una diferencia significativa en comparación con el algoritmo NaiveBayes en imágenes bien segmentadas como las de *aves* y *aeroplanos* de la base de datos de Corel.

6.2. Aportaciones

Las principales contribuciones de este trabajo son:

- Un nuevo algoritmo de ensamble semi-supervisado de clasificadores bayesianos llamado WSA para el etiquetado automático de imágenes, el cual:
  - **a.** Asigna pesos a los ejemplos etiquetados y no etiquetados en base a su probabilidad de clase predicha para reducir la influencia de los ejemplos inicialmente no etiquetados cuyas probabilidades de clase sean bajas.
  - **b.** Incorpora una medida de confianza, la cual permite aumentar el peso de los ejemplos mal clasificados por el clasificador anterior del ensamble para que sean reconsiderados por los nuevos clasificadores del ensamble.
- Una herramienta para etiquetado manual de imágenes y extracción de características.

6.3. Conclusiones

En este proyecto de tesis se desarrolló un nuevo algoritmo llamado WSA para el etiquetado automático de imágenes, el cual está basado en un ensamble de clasificadores bajo un enfoque semi-supervisado. Como resultado de la investigación realizada se concluye que el aprendizaje semi-supervisado es igual o mejor que el enfoque supervisado aplicado al etiquetado automático de imágenes digitales. De acuerdo a los resultados obtenidos, el aprendizaje semi-supervisado permitió

extraer información importante de los ejemplos no etiquetados que sirvió para mejorar la precisión de clasificación.

La incorporación correcta de pesos mediante una medida de confianza en los ejemplos no etiquetados, permitió mejorar el nivel de clasificación. En general WSA tuvo buenos resultados usando porcentajes pequeños de datos etiquetados y grandes cantidades de ejemplos no etiquetados. La segmentación automática de imágenes es el punto clave en la estrategia de etiquetado que se propone. Debido a que este proceso aún es impreciso, afectó el desempeño del algoritmo de clasificación. De las pruebas realizadas se demostró que un ensamble de clasificadores como WSA es viable para utilizarse en sistemas de recuperación de imágenes ya que obtuvo mejores resultados que otros ensambles como AdaBoost o que un único algoritmo como NaiveBayes.

6.4. Trabajo futuro

Como trabajo futuro se propone:

1. Incorporar el algoritmo de etiquetado automático propuesto a un sistema de recuperación de imágenes por contenido.

2. Mejorar la segmentacion automática de imágenes. Uno de los problemas que se presenta en casi todas las técnicas de segmentación es que éstas se basan exclusivamente en características visuales de bajo nivel, y no consideran otro tipo de información que pueda ser relevante. Sin embargo, la segmentación de imágenes se puede afinar si las imágenes de entrada se etiquetan previamente y se combinan con las características visuales que se extraigan de las regiones segmentadas. Así, regiones con etiquetas similares se pueden unir. Con esto se esperaría que se tengan regiones mejor definidas que permitan entrenar mejor al clasificador.

3. Utilizar otras características visuales de la imagen como las relaciones espaciales entre los objetos.

Referencias

[1] Corel Images. University of California, Berkeley, 2003. `http://elib.cs.berkeley.edu/corel/`.

[2] The pascal visual object classes challenge 2007. EU-funded PASCAL Network of Excellence on Pattern Analysis, Statistical Modelling and Computational Learning, April 07 2007. `http://www.pascal-network.org/challenges/VOC/voc2007/`.

[3] S. Aksoy and R. Haralick. Textural features for image database retrieval. In *CBAIVL '98: Proceedings of the IEEE Workshop on Content - Based Access of Image and Video Libraries*, page 45, Washington, DC, USA, 1998. IEEE Computer Society.

[4] B. S. Aurajo. *Aprendizaje Automático: Conceptos básicos y avanzados*. Pearson-Prentice Hall, 2006.

[5] K. Barnard, P. Duygulu, N. de Freitas, D. A. Forsyth, D. Blei, and M. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.

[6] T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions Soc. London*, 53:370–418, 1763.

[7] K. Bennett, A. Demiriz, and R. Maclin. Exploiting unlabeled data in ensemble methods. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 289–296, New York, NY, USA, 2002. ACM Press.

[8] D. Blei and M. Jordan. Modeling annotated data. In *Proceedings of the 26th annual intetational ACM SIGIR conference*, 2003.

[9] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, pages 92–100, 1998.

[10] J. L. Carbajal and J. W. B. Bedoya. Comparison of models of automatic classification of textural patterns of minerals presentes in colombian coals. *Dyna, Revista de la Facultad de Minas-Universidad Nacional de Colombia*, 72:115–12, 2005.

[11] S. K. Chang, Q. Y. Shi, and C. W. Yan. Iconic indexing by 2-d strings. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(3):413–428, 1987.

[12] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, London, England, 2006.

[13] L. Chen, G. Lu, and D. Zhang. Content-based image retrieval using gabor texture features. In *In Proceedings of First IEEE Pacific-Rim Conference on Multimedia (PCM'00)*, pages 1139–1142, Sydney, Australia, 2000.

[14] I. Cohen, F. Cozman, N. Sebe, M. Cirelo, and T. Huang. Semisupervised learning of classifiers with application to human-computer interaction. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume Volume 26, 2004.

[15] T. Deselaers, A. Hanbury, and H. M. M. Castro. Overview of the imageclef 2007 object retrieval task. In *Por publicarse en Proceedings of the 2007 CLEF Workshop*, Budapest, Hungary, 2007. Springer LNCS.

[16] T. Dietterich. Machine learning research: Four current directions. T.G. Dietterich. Machine learning research: Four current directions. AI Magazine, 18(4):97–136, 1997., 1997.

[17] P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proceedings of the 7th European Conference on Computer Vision-Part IV*, page 97—112, London., 2002. Springer-Verlag.

[18] D. D. Feng, W.-C. Siu, and H.-J. Zhang. *Multimedia Information Retrieval and Management*. Springer, 2003.

[19] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[20] R. González and R. Woods. *Digital Image Processing*. Addison–Wesley, NY, 1992.

[21] M. Grubinger, P. Clough, and C. Leung. The IAPR TC-12 Image Benchmark. School of Computer Science and Mathematics, Victoria University Australia, 2006. `http://eureka.vu.edu.au/~grubinger/IAPR/TC12_Benchmark.html`.

[22] A. Hanbury and T. Deselaers. ImageCLEF 2007 - Object Retrieval Task: results. Oregon Health and Science University (OHSU) and Aachen University of Technology (RWTH), 2007. `http://www-i6.informatik.rwth-aachen.de/~deselaers/imageclef07/objret-results.html`.

[23] W. Hersh. ImageCLEF 2007 evaluation of image retrieval and annotation systems for photografic and medical images. Oregon Health and Science University (OHSU) and Aachen University of Technology (RWTH), 2006. `http://ir.shef.ac.uk/imageclef/`.

[24] Kohavi and Ron. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145, 1995.

[25] J. Li and J. Z. Wang. ALIPR – Automatic Linguistic Indexing of Pictures - Real Time. Penn State, Stanford University, November 2006. `http://www.alipr.com/`.

[26] J. Little and C. Moler. The mathworks accelerating the pace of engineering and science. Natick, Massachusetts, 1984. http://www.mathworks.com/.

[27] M. A. Melchor and J. M. V. González. Bases de datos para multimedia: Recuperación por contenido. CD:TEC–01–28, 2001.

[28] M. A. Melchor, J. M. V. González, and M. C. Rocamora. Recuperación por contenido en bases de datos de imágenes basada en wavelets. 2003.

[29] T. Mitchell. *Machine Learning*. McGraw Hill, Carnegie Mellon University, 1997.

[30] D.Ñewman, S. Hettich, C. Blake, and C. Merz. UCI Repository of machine learning databases. University of California, Irvine, 1998. `http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html`.

[31] G. Pajares and J. M. de la Cruz. *Visión por Computador. Imágenes digitales y aplicaciones.* Alfa-Omega, 2001.

[32] E. Riloff, J. Wiebe, and T. Wilson. Learning subjective nouns using extraction pattern bootstrapping, 2003.

[33] C. Rother, V. Kolmogorov, and A. Blake. Grabcut interactive foreground extraction using iterated graph cuts. volume 23, pages 309–314. ACM Press, 2004.

[34] P. K. Sahoo, S. Soltani, A. K. Wong, and Y. C. Chen. A survey of thresholding techniques. *Comput. Vision Graph. Image Process.*, 41(2):233–260, 1988.

[35] R. R. Sánchez. *Selección de atributos mediante proyecciones.* PhD thesis, Universidad de Sevilla, 2005.

[36] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[37] G. Stockman and L. G. Shapiro. *Computer Vision.* Prentice-Hall, Upper Saddle River, NJ, USA, 2001.

[38] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 511—518, Cambridge, 2001.

[39] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[40] J. Yao and Z. M. Zhang. Semi-supervised learning based object detecttion in aerial imagery. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume Volume 1, pages 99–106, Washington, DC, USA, 2005. IEEE Computer Society.

[41] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[42] A. Yavlinsky. Behold. Imperial College London, 2005. `http://photo.beholdsearch.com/search.jsp`.

[43] A. Yavlinsky, E. Schofield, and S. M. Rüger. Automated image annotation using global features and robust not parametric density estimation. In *CIVR*, pages 507–517, 2005.

[44] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005. http://www.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf.

[45] D. Ziou and S. Tabbone. Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, 8:537–559, 1998.