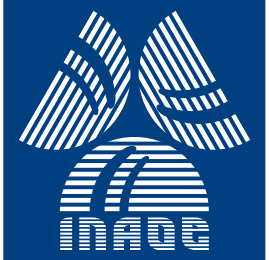

Método de clasificación multi-etapa para el direccionamiento de solicitudes en un centro de llamadas

por

Fernando Uceda Ponga

Tesis sometida como requisito parcial para obtener el grado de:

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

en el

Instituto Nacional de Astrofísica, Óptica y Electrónica

Enero 2009

Tonantzintla, Puebla.

Supervisada por:

Dr. Luis Villaseñor Pineda

Dr. Manuel Montes y Gómez

Agradecimientos

Al arquitecto del universo por haberme puesto en esta vida.

A mis asesores Dr. Luis Villaseñor Pineda y Dr. Manuel Montes Gómez por la guía y paciencia para realizar este trabajo.

A mis sinodales Dr. Aurelio López, Dr. Saúl E. Pomarez y Dr. Enrique Sucar por los consejos y comentarios para mejorar esta tesis.

A mi familia por su cariño y fuerza.

A Claus por su compañía, paciencia y amor, encontrarte es lo mejor que me ha ocurrido, te amo polle araña malvada.

A mis amigos y mis compañeros del INAOE, gracias por escucharme.

y a "$Hola\ Mundo$" sin el ésta tesis no existiria.

Polle, Manin, Mijita, Meme, Burguer y El Barbón
gracias por la paciencia y alegrias que me han dado.

Resumen

El reconocimiento del lenguaje hablado para la comunicación humana en los sistemas tecnológicos ha cobrado gran auge en los últimos años. Específicamente para su uso en centros de llamadas o *call centers.* Donde gracias a un sistema automático es posible enlazar al usuario con el servicio correspondiente. Típicamente, el direccionamiento automático de llamadas a través del habla requiere dos pasos. Durante el primero, se transcribe la elocución o intervención del usuario que llama utilizando un sistema de reconocimiento de habla. Un segundo paso, toma la transcripción automática y determina el servicio solicitado por el usuario. Este segundo paso es foco de atención de esta tesis y es abordado como un problema de clasificación automática de texto.

Sin embargo, aplicar los enfoques tradicionales de clasificación de textos a esta tarea no es posible. En primer lugar, las transcripciones provienen de elocuciones muy cortas, una frase de unas cuantas palabras con las que un usuario expresa su necesidad. Además, dado que aún no se cuenta con un reconocedor de habla perfecto, es posible encontrar transcripciones erróneas. Por otro lado, debido a la naturaleza propia de un centro de llamadas, siempre existe la posibilidad de buscar el apoyo de un operador humano. Gracias a esta posibilidad el sistema puede recurrir a un operador en caso de la incomprensión de una orden. De esta manera, aquellas intervenciones no reconocibles son rechazadas por el sistema y resueltas por un operador. Por supuesto, este rechazo implica un costo, mientras más alta sea la tasa de rechazo más alto será el costo.

El presente trabajo presenta un método de clasificación multi-etapa que permite encontrar un balance entre una baja tasa de rechazo y una alta precisión para un centro de llamadas. El esquema de clasificación propuesto se compone de dos etapas. Una primera etapa busca asegurar una alta precisión en la clasificación de las transcripciones incurriendo inevitablemente en un alto rechazo. Posteriormente, en una segunda etapa, las llamadas inicialmente rechazadas se revisan y se reclasifican, recuperando de entre ellas las rechazadas erróneamente. Cabe resaltar que para este esquema de clasificación también se desarrolló un nuevo método de pesado. Gracias a este método se logra describir de mejor manera la contribución de cada atributo en la discriminación entre las categorías.

Los experimentos reportados en este trabajo demuestran la pertinencia del método al alcanzar mejores resultados a los obtenidos con otros métodos del estado del arte. El método propuesto alcanzó una precisión del 95.5 % y un rechazo erróneo del 2.1 % en un *corpus* recopilado en situaciones reales. Cabe remarcar que a diferencia de otros trabajos que usan únicamente transcripciones correctas para su evaluación, en nuestro caso se incluyeron todas transcripciones automáticas.

Abstract

The recognition of spoken languages for the communication between humans and technological systems has gotten more attention in the last years. Specifically, this is true for the case of call centers, where it is desirable for an automatic system to link a user with the corresponding service.

Typically, the use of voice for automatic call routing requires two main steps. In the first step, the elocution or intervention from the user is transcribed using a voice recognition system. Then, in a second step, the automatic transcription is analyzed in order to determinate the user requested service. This second step is the focus of this thesis, and it is defined as a problem of automatic text classification.

It is important to point out that the application of traditional text-classification approaches is not possible for this task; on the one hand, because transcriptions from the elocutions are very short (commonly a phrase of few words), and, on the other hand, because current voice recognizers are not perfect and tend to produce several wrong transcriptions. In addition, and given the functionality of a call center, it is always possible to search help from a human operator. Thanks to this possibility, the interventions that can not be recognized are rejected by the system and are resolved through an operator. Evidently, this rejection implies a cost; the higher the rejection rate the higher the cost.

The current work presents a multi-step classification method that allows finding a

balance between the rejection rate and the precision for an automatic call center. The proposed classification scheme considers two steps. The first step tries to ensure a high precision in the classification of the interventions, by incurring in a high rejection rate. Later, in the second step, the rejected interventions are reclassified with the aim of recovering some of them (which were erroneously rejected in the fist step). It is important to mention that for this classification scheme we also developed a new weighting method. Thanks to this method we could better describe the contribution of each attribute to each one of the categories.

The experiments reported in this work demonstrate the relevance of the proposed method; its results were better than those obtained by other state of art approaches. In particular, the proposed method achieved a precision of 95 % and an erroneous rejection rate of 2.1 % in a corpus gathered from a real scenario. It is important to remark that, in contrast to other previous works that only use correct transcriptions for their evaluation, we considered the complete set pof transcriptions.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La interacción entre el hombre y las computadoras ha cambiado radicalmente a lo largo de las últimas décadas, recibiendo un particular interés la interacción oral. Muchos factores han permitido alcanzar un nivel tecnológico gracias al cual es posible que una computadora reciba órdenes verbales. Esta evolución tecnológica va desde los sistemas de reconocimiento de habla de unas cuantas palabras en los años 70 hasta los sistemas actuales capaces de reconocer habla espontánea con grandes vocabularios. Estos avances no sólo tienen un impacto tecnológico, sino también un fuerte impacto económico que ha contribuido al avance obtenido en esta área.

A pesar de estos logros, aún falta mucho por resolver respecto a las dificultades inherentes al proceso de la comunicación oral. Estos problemas tienen que ver con la cantidad y diversidad de áreas de conocimiento involucradas en el proceso de interpretación del lenguaje humano. En nuestro caso estos problemas se simplifican, pues estamos interesados en abordar un tipo de interacción oral muy particular: el direccionamiento de llamadas en un centro telefónico.

La tarea de direccionamiento de llamadas (o *call routing*) consiste en dirigir una llamada a un destino apropiado dentro de un centro telefónico. Los primeros sistemas

automáticos de direccionamiento de llamadas aprovecharon el sistema de tonos de los teléfonos digitales para permitir la navegación de un usuario entre las diferentes opciones de un simple menú jerárquico. Una vez seleccionada la opción adecuada se enlazaba al usuario con el servicio correspondiente. Sin embargo, el sistema es abrumador pues exige al usuario, no sólo escuchar todas las opciones del menú, sino identificar la opción apropiada dada una necesidad puntual. Por ejemplo, una solicitud simple tal como: *deseo conocer mi estado de cuenta*, obligará al usuario a navegar por varios menús con distintas opciones cada uno. Es por ello que una solución es el uso de sistemas comandados por voz que permita a un usuario interactuar con el sistema de direccionamiento de llamadas usando el habla tal y como lo haría con un ser humano.

1.1. Problemática

Típicamente, el direccionamiento automático de llamadas a través del habla requiere dos pasos. Durante el primero se transcribe la elocución de la persona que llama utilizando un sistema de reconocimiento de habla. El segundo paso toma la transcripción automática y determina el servicio solicitado por el usuario. En este trabajo de tesis nos centramos en el segundo paso, es decir, partimos de la transcripción automática sin considerar el problema del reconocimiento automático del habla.

Una manera de abordar este problema es considerarlo como un problema de clasificación automática de texto. La clasificación automática de texto es un área de investigación ampliamente explorada con avances significativos. Básicamente, un algoritmo de aprendizaje automático crea un modelo probabilístico basado en la frecuencia de los términos en los documentos. Como es de imaginar el rendimiento de la clasificación está estrechamente relacionado con la extensión del documento. En particular, en nuestro caso las transcripciones provienen de elocuciones muy cortas, una frase de unas cuantas palabras con las que un usuario expresa su necesidad. Además, dado que aún

no se cuenta con un reconocedor de habla perfecto, es posible encontrar transcripciones erróneas. Debido a estos problemas es necesario adecuar las técnicas tradicionales de clasificación de textos.

Por otro lado, dada la naturaleza de un centro de llamadas, siempre existe la posibilidad de buscar el apoyo de un operador humano. Gracias a esta posibilidad el sistema puede recurrir a un operador en caso de la incomprensión de una orden. De esta manera, aquellas intervenciones ambiguas o no reconocibles son rechazadas por el sistema automático y resueltas por un operador. Por supuesto, este rechazo implica un costo, mientras más alta sea la tasa de rechazo más alto será el costo, ya que la llamada no se direccionará por medios automáticos. Sin embargo, el no rechazar alguna intervención, también acarrea un alto costo, pues la errónea categorización subsecuente de ciertas llamadas se transformará en un usuario inconforme. De ahí la importancia de buscar un balance adecuado entre estos dos costos.

En este trabajo se abordan los problemas presentes en la categorización de la transcripción automática de una llamada. Por un lado, para mejorar la clasificación de textos cortos, se propone un nuevo esquema de pesado para mejorar la discriminación entre las categorías. Además, para encontrar tanto una baja tasa de rechazo así como una alta precisión, se propone un esquema de clasificación en dos pasos. Donde en un primer paso se busca asegurar una alta confianza en la clasificación de las transcripciones, incurriendo inevitablemente en un alto rechazo. Posteriormente, en un segundo paso, las llamadas rechazadas se revisan con la intención de reclasificar algunas de las rechazadas erróneamente.

1.2. Retos del problema de *call routing*

Los problemas computacionales particulares abordados por el problema del direccionamiento de llamadas (*call routing*) son:

- **Análisis y detección del contexto o intensión basado en poca evidencia**

  Mientras más elementos (atributos) se tengan para representar un texto, mejores resultados se alcanzarán en la clasificación automática. Sin embargo, en nuestro caso el enfoque tradicional resulta deficiente debido a la poca extensión del texto.

  Otros enfoques, como el uso de gramáticas, requieren un modelado completo con el mayor número de casos posibles para lograr un buen desempeño. Estos enfoques requieren demasiado tiempo en su fase de diseño y son dependientes del *corpus* con el que fueron entrenados, así mismo estos enfoques hace difícil la inclusión de nuevas clases.

- **Disminuir la necesidad de un balanceo de clases**

  Debido a que existen en el *corpus* clases mayoritarias, se debe utilizar un método de clasificación aplicable a datos no balanceados. Este problema es común en la vida real para varias situaciones, pues normalmente lo que se quiere clasificar son precisamente los casos extraordinarios.

- **Necesidad de manejar problemas de ambigüedad en la petición**

  La petición del usuario puede ser atendida en más de un lugar, ya sea por la naturaleza de la misma o por errores en el proceso de reconocimiento de voz. Esto obliga a que el clasificador deba ser capaz de seleccionar la instancia que mejor pueda satisfacer al usuario o de rechazar la petición y pedirla nuevamente. Esto último debe ser modelado con cautela para no tener altas tasas de rechazo.

1.3. Motivación

Es un hecho que la importancia de sistemas basados en el reconocimiento automático del habla (ASR) es cada día más creciente (Hernández et al. (1994), Wilpon et al. (1990), Wilpon y Rose (1994)). En los últimos años, se ha dado una proliferación de sistemas basados en esta tecnología para la interacción hombre-máquina mediante la voz, como lo son los sistemas de respuesta vocal interactiva y las aplicaciones para

la automatización de sistemas telefónicos. Los sistemas de reconocimiento de habla, frente a otros sistemas de interacción hombre-máquina, como lo son teclados, ratones, pantallas táctiles, etc., proporcionan una mayor naturalidad, así como utilización por parte de diferentes tipos de usuarios en distintos entornos. En el caso de la automatización de sistemas telefónicos, se tienen exigencias especiales como las limitaciones que impone el canal telefónico, así como la presencia de perturbaciones producidas tanto inherentes al locutor como por factores ajenos a él.

La solución planteada en esta tesis es abordada de dos formas; la primera en la forma de caracterizar los datos de manera automática mediante una función que pudiera ser utilizable en otros problemas similares y la segunda con un esquema de clasificación orientado a resolver los principales problemas en la clasificación de textos provenientes de un reconocedor de voz, como lo son el ruido y el mal reconocimiento.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar un método de clasificación de solicitudes que permita su adecuado direccionamiento en un centro llamadas, conservando un balance entre la precisión del clasificador y la tasa de rechazo.

1.4.2. Objetivos particulares

- Desarrollar un método para la selección de atributos, específicamente orientado a la clasificación de textos cortos.

- Desarrollar un método de pesado de atributos que mejore la clasificación de textos cortos.

- Evaluar y comparar el método de pesado a propuesto con otros en el estado del arte.
- Desarrollar un método de clasificación multi-etapa para llamadas, que logre un balance entre la precisión en la clasificación y la tasa de rechazo.
- Evaluar y comparar el método de clasificación multi-etapa contra métodos del estado del arte.

1.5. Organización de la tesis

En el segundo capítulo se presenta un marco teórico, como un complemento para hacer una mejor introducción al área. El tercer capítulo presenta un análisis del estado del arte y una descripción de los principales enfoques para abordar el problema de *call routing*. El cuarto capítulo describe el método de clasificación y el sistema de pesado propuestos para la resolución del problema de clasificación de intervenciones conversacionales en el marco de *call routing*. El quinto capítulo contiene una descripción del *corpus* utilizado, así mismo los experimentos realizados con diferentes métodos y sus comparaciones con el método propuesto en esta tesis. Finalmente se presenta el análisis de estos resultados, así como las conclusiones y el trabajo futuro en el capítulo sexto.

Capítulo 2

Marco Teórico

En este capítulo se revisarán las bases de la clasificación de textos. Esta introducción a la clasificación de textos es una base importante para esta tesis debido al enfoque planteado en el presente trabajo.

El proceso de clasificación de textos consiste en determinar si un documento pertenece o no a una categoría. Para clasificar los textos de forma automática, es decir mediante un sistema computacional, se requieren varios pasos.

- Extracción de características
- Representación del documento
- Entrenamiento y Construcción del modelo
- Clasificación

Los sistemas informáticos no pueden manipular el texto plano por lo que este debe ser procesado para que los sistemas computacionales puedan manipularlo. A este paso previo para el análisis de los documentos se le llama extracción de características. Este paso recibe su nombre debido a que un documento contiene mucha información que no es útil para determinar su pertenencia a una clase. La información no útil es:

- Signos de puntuación.
- Palabras vacías como son los artículos, conjunciones y muletillas.
- Palabras únicas o poco frecuentes.

Una vez extraídas las características más representativas de un documento se ordena y guarda en una estructura de datos. La forma en que se guarda dicha estructura es conocida como el paso de representación de documentos.Una vez representado el documento, la estructura es usada para entrenar un clasificador. Durante este paso, los resultados del clasificador son usados para construir un modelo. El modelo es puesto a prueba para medir la precisión del clasificador. Como último paso en la clasificación de textos se utiliza el clasificador construido el cual recibe los documentos almacenados en la estructura computacional y se encargara de evaluar la pertenencia de un documento a una clase o varias clases.

A continuación se presentarán los métodos de pesado de atributos, los métodos existentes para la representación de documentos y las técnicas de clasificación de textos más relevantes al presente trabajo.

2.1. Métodos de extracción de características

2.1.1. Eliminación de palabras con base en su frecuencia

Este método para la selección de características es el más simple e intuitivo. Para realizar la selección de las palabras, primero se calcula la frecuencia con la que cada palabra aparece en los documentos del *corpus*. Después se eliminan aquellas palabras que tengan una frecuencia menor a un umbral.

La idea básica es que las palabras poco comunes no contienen información útil para realizar la clasificación o simplemente no afectan el desempeño global.

2.1.2. Ganancia de Información

Definida en teoría de la probabilidad y teoría de la información, la divergencia de KL(Kullback y Leibler (1951)), conocida como divergencia de la información, ganancia de información *(InfoGain)* o entropía relativa, es la diferencia entre dos distribuciones de probabilidad: desde una distribución de probabilidad *"verdadera" P* hasta una distribución de probabilidad arbitraria *Q*. Aunque se infiere frecuentemente como una distancia métrica, la divergencia de *KL* no es una métrica verdadera, pues no es simétrica.

Típicamente, *P* representa datos, observaciones o una densidad de probabilidad calculada precisamente. La medida *Q* típicamente representa una teoría, un modelo, una descripción o una aproximación de *P*.

Para distribuciones de probabilidad *P* y *Q* discretas donde i es el índice de los elementos en la distribución, la divergencia de *KL* como se ilustra en la fórmula 2.1 :

$$D_{kl}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{2.1}$$

Esta medida, en el área de clasificación de textos es el número de bits de información necesarios para la predicción de una clase, basado en la presencia o ausencia de una palabra en un documento.

Dado que $c_1, ..., c_k$ son las posibles clases de un documento. La ganancia de información de una palabra w es definida como:

$$IG(w) = -\sum_{j=1}^{k} P(c_j) log\ P(c_j) + P(w) \sum_{j=1}^{k} P(c_j|w) log\ P(c_j|w) + P(\bar{w}) \sum_{j=1}^{k} P(c_j|\bar{w}) log P(c_j|\bar{w}) \tag{2.2}$$

donde $P(c_j)$ puede ser estimado a partir de los documentos que pertenecen a la clase c_j y $P(w)$ del total de documentos donde la palabra w ocurre. De hecho, $P(c_j|w)$ puede ser caculado del total de documentos de la clase c_j en los que la palabra w ocurre y $P(c_j|\bar{w})$ como el conjunto de documentos de la clase c_j donde la palabra w no ocurre. De esta manera dado que es una medida de la cantidad de información contenida el va-

lor puede ser negativo. Normalmente en clasificación de textos se calcula la divergencia de KL para cada término y se eliminan aquellos por debajo de un umbral. Así este valor sirve tanto para pesar los atributos como para la selección de las más importantes.

2.2. Modelos de representación de documentos

2.2.1. Modelo de espacio vectorial (MEV)

Para la construcción de este modelo un texto es considerado una bolsa de palabras (*Bag Of Words*, propuesto por McCallum (1996)). Así un texto es una colección desordenada de palabras, sin hacer caso de la gramática.

Para este modelo, la unidad de información es la palabra, considerando cada documento como una colección de palabras. Así, en el MEV, cada documento se representa como un vector d_j de términos ponderados w que representan a las palabras contenidas en d y su relevancia ya sea a la clase o al documento, esto en función del pesado que se utilice.

$$d_j = [w_{1j} w_{2j} ... w_{|\psi|j}]^T \tag{2.3}$$

donde $|\psi|$ es el total de términos en el diccionario. De este modo, una colección de $|D|$ documentos queda representada mediante una matriz de términos por documento ($P = [d_1 d_2 ... d_{|D|}]$) (Salton (1989), Salton et al. (1975)). Este enfoque fue usado por primera vez en el sistema SMART (*Salton's Magic Automatic Retriever of Text*) propuesto por Salton (1960) en la Universidad Cornell en Ithaca. Cuando MEV fue presentado por Salton et al. (1975) utilizó el pesado *tfidf* basándose en el trabajo presentado por Jones (1972). Sin embargo pese a ser un modelo bastante utilizado en el área de recuperación de información y clasificación de textos, el modelo MEV presenta algunas limitaciones:

- Los documentos largos tienen una gran dimensionalidad.
- Las palabras de búsqueda deben coincidir con las palabras del documento, usar

sílabas o sub-cadenas de una palabra pueden ocasionar un error en la clasificación.

- Sensibilidad semántica, documentos con contextos similares pero con diferente vocabulario no serán asociados.

2.3. Métodos de pesado de atributos

En la literatura existen varios métodos de pesado de atributos. En este capítulo se revisarán los métodos de pesado más utilizados en la clasificación de textos.

2.3.1. Pesado booleano

Es uno de los más sencillos de entender y utilizar, consiste en crear un vocabulario basado en el *corpus*. El vocabulario puede contener todas las palabras diferentes del *corpus* o sólo las más representativas. Una vez obtenido el vocabulario cada documento es analizado palabra por palabra, dando un valor de 1 si la palabra está en el vocabulario o un 0 si no está en el vocabulario. Construyendo así una matriz (Vocabulario/Documentos). En este pesado una palabra presente tiene el mismo peso aun cuando esté presente en repetidas ocasiones en un documento.

2.3.2. Pesado *tfidf*

El peso *tfidf (term frequency-inverse document frequency)* presentado por Salton et al. (1975), es un peso usado frecuentemente en la recuperación de información y clasificación de textos (Yun-tao et al. (2004)). Este peso es una medida estadística usada para evaluar cuán importante es una palabra de un documento en una colección o *corpus*. La importancia aumenta proporcionalmente al número de veces que una palabra aparece en el documento, pero es disminuida o compensada por la frecuencia de dicha palabra en el *corpus*.

El componente tf está dado por el número de veces que una palabra aparece en el documento. El valor tf se normaliza para prevenir un sesgo hacia los documentos más largos (que puedan tener una frecuencia de palabras más alta independientemente de la importancia real de ese término en el documento), para dar una medida de la importancia del término t_i dentro del documento d_j.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{2.4}$$

donde $n_{i,j}$ es el número de ocurrencias del término t_i en el documento d_j. Y el denominador es el número de ocurrencias de todos los términos en d_j.

El valor *idf* se obtiene dividiendo el total de documentos en el *corpus* por el número de documentos que contienen el término t_i y luego se toma el logaritmo del cociente.

$$idf_i = \log \frac{|D|}{|d_j : t_i \, \epsilon \, d_j|} \tag{2.5}$$

donde $|D|$ es el número total de documentos en el *corpus* y $|d_j : t_i \, \epsilon \, d_j|$ el número de documentos donde aparece el término t_i.

Siendo *tfidf*:

$$tfidf_{i,j} = tf_{i,j} \cdot idf_i \tag{2.6}$$

Por lo tanto, si un alto peso *tfidf* es alcanzado significa que hay una alta frecuencia del término t_i en el documento d_j y una baja frecuencia en los documentos del *corpus*.

Dado que *tfidf* da un peso que representa la relevancia de un término para un documento, entonces documentos pertenecientes a una misma clase tendrán valores *tfidf* parecidos (Yun-tao et al. (2004)). Se reporta que *tfidf* como peso para los atributos de

documentos a clasificar mejora la clasificación. Éste pesado refleja una relación entre la palabra, el documento y el *corpus*, por lo cual muestra muy buenos resultados en conjunto con clasificadores probabilísticos.

2.4. Métodos de Clasificación de Textos

2.4.1. Probabilísticos

Los enfoques de clasificación basados en probabilidad suelen reportar altos grados de precisión, estos clasificadores son muy utilizados en el área de clasificación de textos debido a su buen desempeño y bajo costo computacional (Sebastiani (2002)).

Clasificador Bayesiano

Particularmente, en el área de clasificación de textos uno de los clasificadores basados en enfoques probabilísticos más ampliamente utilizado es el clasificador Bayesiano simple (Minsky y Papert (1969)). Un clasificador Bayesiano es un clasificador probabilístico basado en la aplicación del teorema de Bayes (Barnard (1958)). El teorema de Bayes ofrece un método estadístico para calcular la probabilidad de que un suceso ocurra, partiendo de la probabilidad de que un evento ligado a éste ocurra. Este teorema es de gran utilidad para evaluar una probabilidad *a posteriori* partiendo de probabilidades simples, y así poder revisar la estimación de la probabilidad *a priori* de un evento que se encuentra en un estado o en otro. El teorema de Bayes parte de una situación en la que es posible conocer las probabilidades de que ocurran una serie de sucesos A_i. A esta probabilidad se añade un suceso B cuya ocurrencia proporciona cierta información, porque las probabilidades de ocurrencia de B son distintas según el suceso A_i que haya ocurrido. Conociendo que ha ocurrido el suceso B, la fórmula del teorema de Bayes nos indica como modifica esta información las probabilidades de los sucesos A_i, mostrada en la ecuación 2.7 en la página siguiente.

$$P(f_i|s) = \frac{P(s|f_i)P(f_i)}{P(s)} = \frac{P(s|f_i)}{\sum_{k=1}^{n} P(s|f_k)P(f_k)} \tag{2.7}$$

A pesar de su diseño y simplicidad, a menudo los clasificadores Bayesianos trabajan mucho mejor en situaciones complejas del mundo real, en las que se esperaría lo contrario. Recientemente, el análisis cuidadoso del problema Bayesiano de clasificación ha demostrado que hay algunas razones teóricas sobre la aparente eficacia incongruente de los clasificadores Bayesianos (Zhang (2004)). Una ventaja del clasificador Bayesiano es que requiere una cantidad pequeña de datos para el entrenamiento y así estimar la media y las variaciones de las características las cuales son necesarias para la clasificación. Debido que supone la independencia de las características, sólo las variaciones de las características para cada clase necesitan ser determinadas, en lugar de la matriz de covarianza entera.

Clasificador Bayesiano Simple

El clasificador Bayesiano simple es construido usando un *corpus* de entrenamiento para estimar la probabilidad de cada clase, dados los valores de las características en un nuevo documento. Para su construcción se utiliza el teorema de Bayes para estimar las probabilidades como se muestra en la ecuación 2.8.

$$P(c_j|d) = \frac{P(c_j)P(d|c_j)}{P(d)} \tag{2.8}$$

El denominador en la formula 2.8 no diferencia entre categorías y puede ser eliminado. Más aun, la parte simple del clasificador es la suposición de que las palabras son independientes. De hecho se asume que las características son condicionalmente

independiente. Obteniendo la ecuación 2.9.

$$P(c_j) = P(c_j) \prod_{i=1}^{M} P(d_i|c_j) \tag{2.9}$$

Un estimado de $\hat{P}(c_j)$ para $P(c_j)$ puede ser calculado a partir del número de documentos de entrenamiento que son asignados a la clase c_j:

$$\hat{P}(C = c_j) = \frac{N_{ij}}{N} \tag{2.10}$$

donde N_{ij} es el número de veces que una palabra i ocurre en los documentos de la clase c_j en el *corpus* de entrenamiento. Más aun, un estimado de $\hat{P}(d_i|c_j)$ para $P(d_i|c_j)$ es dado por:

$$\hat{P}(d_i|c_j) = \frac{1 + N_{ij}}{M + \sum_{k=1}^{M} N_{kj}} \tag{2.11}$$

donde M es el número de características por documento.

Pese al hecho de que la independencia condicional generalmente no es cierta para la aparición de palabras en los documentos, el clasificador bayesiano simple tiene un buen desempeño.

Cuando un clasificador Bayesiano simple (Minsky y Papert (1969)) se aplica al texto (Véase sección 2.4.1), la suposición de independencia condicional conduce a la generación de un modelo de bolsa de palabras.

Clasificador VFI *(Voting Feature Intervals)*

Otro clasificador probabilístico el cual también hace suposiciones de independencia entre las palabras de un documento como las del Bayesiano simple es el llamado VFI *(Voting Feature Intervals)* presentado por Demiröz y Güvenir (1997), el cual según los

resultados reportados por sus creadores en determinados problemas supera la precisión de Bayes y sobretodo es más rápido computacionalmente. El clasificador VFI retorna (al igual que Bayes) una distribución de probabilidad sobre todas las clases.

Éste clasificador construye un intervalo de valores para cada característica. Los intervalos son representados con vectores $< lower, count_1, ...count_k >$ donde *lower* es el limite mínimo del intervalo y $count_i$ es el número de instancias de la clase i que caen en ese intervalo, debido a que un intervalo puede representar varias clases sólo guardan el valor mínimo de un intervalo. El cálculo de los votos para cada intervalo se realiza con la fórmula:

$$feature\_vote[f, c] = \frac{interval\_class\_count[f, i, c]}{class\_count[c]} \tag{2.12}$$

donde $interval\_class\_count$ es el número de ejemplos de la clase c que caen en el intervalo i con una característica de dimensión f esto genera un vector de votos $< vote_1, ..., vote_k >$ por lo que al distribución de probabilidad es calculada como:

$$\frac{vote[c]}{\sum_k vote[k]} \tag{2.13}$$

donde c es una clase dada y k recorre todas las posibles clases.

2.4.2. *Boosting*

Los métodos de *boosting* son algoritmos que en el enfoque supervisado conocen la solución *a priori* y utilizaran esto para adaptar su comportamiento. El *boosting* se basa en la pregunta *¿Puede un conjunto de clasificadores débiles crear un sólo clasificador fuerte?* planteada por Kearns (1988). Un clasificador débil es definido como un clasificador que se correlaciona levemente con la clasificación final o verdadera. En cambio, un clasificador fuerte es un clasificador que se correlaciona fuertemente con la clasificación verdadera.

Aunque el *boosting* no está algorítmicamente restringido, la mayoría de los algoritmos que hacen *boosting* consisten en procesos iterativos que aprenden de clasificadores débiles con respecto a una distribución, para finalmente agregarlos a un clasificador final el cual es más fuerte. Cuando se agregan, son almacenados de manera que se relaciona comúnmente con la precisión del clasificador débil. Después de que se agrega un clasificador débil, los datos se vuelven a pesar. Los ejemplos que son clasificados correctamente pierden peso y los que son clasificados de manera errónea aumentan su peso. Así, los siguientes clasificadores se centran más en los ejemplos que los anteriores clasificaron erróneamente.

Hay muchos algoritmos de *boosting*, los originales propuestos por Schapire (1990) y Freund (1990) no eran adaptativos y no podían tomar ventaja completa de los clasificadores débiles. Por tal motivo Freund y Schapire proponen posteriormente *Adaboost* (Freund y Schapire (1997)), el cual es un algoritmo adaptativo que toma ventaja de los clasificadores débiles para formar un clasificador fuerte. El algoritmo de *Adaboost* es normalmente implementado utilizando árboles de decisión, pero puede utilizar diferentes clasificadores base incluyendo el Bayesiano simple. Pese a no haber restricciones en cuanto a la forma de hacer el *boosting*, sólo se consideran algoritmos de *boosting* aquellos algoritmos que cumplen con las normas de aprendizaje PAC (*Probably approximately correct learning*) descritas por Kearns y Vazirani (1994).

Otros algoritmos que son similares en enfoque a los algoritmos de *boosting*, pero que no cumplen con el PAC suelen ser llamados Algoritmos Leveraging (Duffy y Helmbold (1999)), aunque algunas veces también llamados incorrectamente algoritmos de *boosting*. La variación principal entre muchos algoritmos de *boosting* es el método de pesado para los datos de prueba y la hipótesis. *AdaBoost* es muy popular y quizás el más significativo, pues históricamente fue el primer algoritmo que podía adaptarse a los clasificadores débiles que lo componían. Sin embargo, actualmente existen más

algoritmos tales como: *LPBoost* (Demiriz et al. (2002)), *BrownBoost* (Freund (2001)), *LogitBoost* (Friedman et al. (2000)), entre otros.

2.4.3. Máquinas de Vectores de Soporte

Las máquinas de vectores de soporte (SVM, *Support Vector Machines*) son un método de clasificación para aprendizaje supervisado. Una característica especial de las SVM es que reducen simultáneamente el error empírico de la clasificación y maximizan el margen geométrico. Las SVM ven los datos de entrada como dos sistemas de vectores en un espacio *n*-dimensional, un SVM construye un hiperplano de separación en dicho espacio, el cual maximiza el margen entre los dos sistemas. Para calcular el margen, se construyen dos hiperplanos paralelos a cada lado del hiperplano de separación que "empujen" los dos conjuntos de datos, como se ve en la figura 2.1. La idea es que cuanto más grande sea el margen o la distancia entre estos hiperplanos paralelos, será mejor el error de generalización del clasificador.

El algoritmo original propuesto por Drucker et al. (1997) era un clasificador lineal. Sin embargo, en el trabajo de Boser et al. (1992) se sugiere una manera de crear clasificadores no lineales aplicando una modificación al *kernel* (propuesto originalmente por Aizerman et al. (1964)), conocido como hiperplanos del máximo-margen. El algoritmo que resulta es formalmente similar, salvo que cada producto punto es substituido por una función no lineal del núcleo. Esto permite que el algoritmo se ajuste al hiperplano del máximo-margen en el espacio transformado de la característica. La transformación puede ser no-lineal y el espacio transformado arriba podrá ser *n-dimensional*. Así aunque el clasificador es un hiperplano en un espacio de alta-dimensionalidad, la característica puede ser no-lineal en el espacio original de entrada. Si el *kernel* usado es una función radial Gaussiana, entonces el espacio correspondiente de la característica es un espacio de Hilbert, de la dimensión infinita. Donde los clasificadores máximos del margen se regularizan de manera correcta, así que la dimensión infinita no estropea los

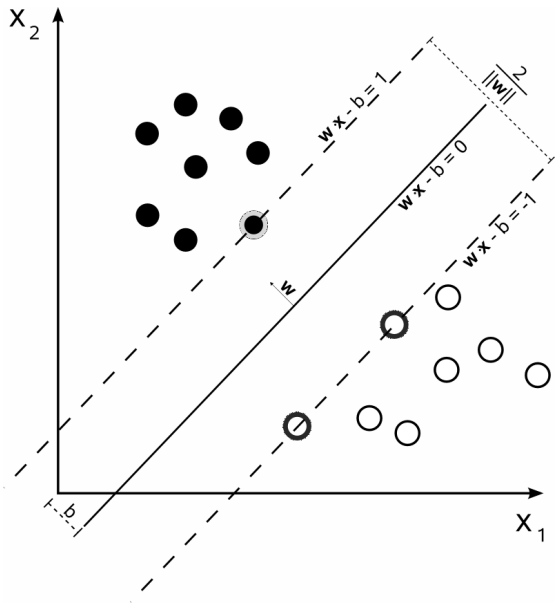

Figura 2.1: Máquina de Vectores de Soporte entrenada para dos clases que son conjuntos linealmente separables por el vector w. La distancia entre los conjuntos es $\frac{2}{||w||}$ la variable b y el vector w buscan maximizar la distancia entre los conjuntos.

resultados. Algunos *kernels* comunes son:

- Polinomio homogéneo:

$$k(x, x^{'}) = (x \cdot x^{'})^{d}$$

- Polinomio (no homogéneo):

$$k(x, x^{'}) = (x \cdot x^{'} + 1)^{d}$$

Los clasificadores basados en SVM son generalmente implementados como clasificadores binarios en cascada, siendo así cada clasificador capaz de reconocer entre una clase y el resto (Burges (1998)). Las SVM tienen un alto desempeño cuando los conjuntos de datos son linealmente separables en alguna dimensión o sí la representación del documento permita una separación.

2.5. Conclusiones

En este capítulo se observó la importancia del pesado de los atributos de un texto. El pesado afecta el desempeño de la clasificación al igual que el modelo que se use para representar un documento.

La selección de atributos de un documento se encuentra ligada a la frecuencia y extensión del mismo por lo que las técnicas de clasificación de documentos aprovechan la extensión del texto para lograr una clasificación.

En el siguiente capítulo se analizarán los trabajos más representativos del estado del arte que intentan resolver específicamente el problema de direccionamiento de llamadas (*call routing*). Los trabajos presentados hacen uso de las técnicas mostradas en este capítulo.

Capítulo 3

Estado del Arte

3.1. Introducción

El interés por la resolución del problema de *direccionamiento de llamadas (call routing)* ha aumentado actualmente, especialmente por el crecimiento de los centros de llamadas o *call centers*. Los centros de llamadas son un negocio en auge, no sólo en los Estados Unidos, sino en países de todo el mundo. Estos emplean agentes en todo el mundo para atender las necesidades de millones de clientes. Esto ha llevado a que la iniciativa privada busque formas más eficientes de manejar el gran volumen de llamadas, particularmente el clasificarlas. De esta manera cada agente se especializa en un área e incluso existen peticiones que pueden ser respondidas por una computadora mediante un sintetizador de voz evitando así la intervención humana. Este interés mezclado entre la comunidad científica y la iniciativa privada ha creado diferentes enfoques para solucionar el problema de *call routing*. Algunos enfoques tienen como objetivos ser rápidos de desarrollar e implementar, otros buscan mejorar la precisión o que sean de bajo costo, ya sea computacional o económico. Los sistemas y artículos de los que se tienen conocimiento y se han publicado hasta la fecha se pueden agrupar en tres enfoques descritos a continuación.

3.1.1. *Keyword Spotting*

El enfoque de *Keyword Spotting* (Kim et al. (2004)) es uno de los primeros que se crearon. Además de ser muy utilizado en sistemas comerciales, ya sea empleandolo al 100 % como un método de resolución o bien como un método para realizar un pesado *a priori*. Es además uno de los esquemas más fáciles de comprender.

El método más sencillo de *Keyword Spotting* consiste en utilizar las palabras clave de cada clase como la única información para decidir el destino que debe asignarse a una instancia. Es fácil ver que una de las más grandes debilidades de este método es la necesidad de reconocer la palabra clave correctamente, pues el método de clasificación depende de ella para su correcto funcionamiento.

Independientemente del sistema de reconocimiento de voz, y asumiendo que se tiene un reconocimiento perfecto, existen usuarios que debido a la riqueza del lenguaje expresan su petición sin utilizar las palabras claves que el sistema tiene registradas. En el trabajo de Wilpon et al. (1990) se utilizan modelos ocultos de Markov (Rabiner (1989)) para subsanar el problema del mal reconocimiento de la palabra clave al ligar el proceso de reconocimiento de voz y la clasificación. Su método, aunque más robusto que otros enfoques simples, sigue requiriendo que el usuario utilice ciertas palabras para poder determinar la clase de la llamada. En sus experimentos utilizan un *corpus* en inglés recolectado por AT&T en su oficina central de Hayward. Debido a ser un enfoque que unifica la tarea de reconocer las palabras clave del hablante para posteriormente realizar el direccionamiento, utilizaron las grabaciones en su *corpus* de entrenamiento. Su *corpus* está dividido en cinco clases y cuenta con un total de 75,000 ejemplos de los cuales separaron 3,689 para *corpus* de prueba, obteniendo así una precisión del 92.48 %. Este trabajo no muestra una tasa de rechazo ya que cuentan con una clase *Operadora* la cual está destinada a contener dichas peticiones.

Al ser este uno de los primeros enfoques, se le han realizado múltiples optimizaciones como las propuestas por Kim et al. (2004), en el cual intentan generar un enfoque que ellos mismos llaman de pseudo *n*-gramas debido a la forma en que pre-procesan la entrada de texto. Su enfoque está basado en los métodos propuestos por Carpenter y Chu-carroll (1998) y Yining et al. (2001), en el cual dada una petición oral, como por ejemplo:

"I want to check on an account"

es procesada para ser vista como:

"$< sw >< sw >< sw >$ *check* $< sw >$ *account"*

En este proceso todas las palabras que no se encuentren entre las palabras claves que el sistema puede reconocer son substituidas por $< sw >$ para luego ser eliminadas de la misma forma que en el trabajo de Salton (1960).

Para el ejemplo anterior, el sistema de Carpenter y Chu-carroll (1998) recibe como entrada el bigrama *"check-account"*. Cada clase es vista como un vector de términos (unigramas, bigramas, trigramas y hasta cuatrigramas), utilizando una representación basada en modelos de espacio vectorial (Salton et al. (1975)). Para obtener el vector con el que tiene mayor parecido una instancia y por ende clasificarla, se utiliza el producto punto entre el vector de entrada y el vector de entrenamiento. Para reducir la dimensionalidad de los vectores se utiliza la fórmula *idf* (Salton y Buckley (1988)) y se eliminan aquellos elementos que se vuelven 0. Su *corpus* consiste de 3,753 instancias de entrenamiento y 389 de prueba divididas en 23 clases. En sus experimentos obtiene una precisión del 75.6 % sin tener rechazo. A diferencia de otros métodos que eliminan de sus resultados errores de clasificación por consecuencia del reconocedor de voz, este trabajo los contempla dentro de las clasificaciones incorrectas del sistema.

Otros trabajos mantienen unido el desempeño del reconocedor de voz con el de la clasificación de llamadas, como es el caso de los sistemas de Kim et al. (2004) y Yining et al. (2001), en los cuales se reporta un error del 8.6 % y del 9.8 % respectivamente.

Como puede verse la debilidad que presentan estos enfoques es su falta de robustez ante una fase de reconocimiento de voz imperfecta, por lo cual siempre se intenta crear un reconocedor que sea muy bueno identificando las palabras claves que necesitan. Aún cuando para el usuario pareciera que son sistemas que responden al habla espontánea no son tan diferentes a aquellos que sólo aceptan palabras sueltas como comandos.

Para poder crear sistemas robustos a una fase de reconocimiento imperfecta y a una gran diversidad de hablantes, se cambió el enfoque del problema. En el nuevo enfoque se utilizan cadenas de palabras o sucesiones para poder ofrecer un mejor desempeño.

3.1.2. Modelos de lenguaje

Otro enfoque utilizado para la resolución de la problemática de *call routing* es el uso de modelos del lenguaje (LM), este enfoque es muy utilizado actualmente en sistemas comerciales debido a la característica de poder ser creado con una alta precisión.

Descripción general de modelos de lenguaje

Los modelos de lenguaje contienen información sobre las probabilidades o grados de pertenencias de una palabra en un determinado dominio, así mismo contiene la probabilidad de ocurrencia de una palabra dada otra palabra. La probabilidad de ocurrencia de una palabra dada otra se denomina *n*-grama, permitiendo que un modelo de lenguaje pueda contener probabilidades para bigramas, trigramas, cuatrigramas, ..., *n*-gramas, de tal modo que refleje de la mejor manera posible el comportamiento que tiene el lenguaje para un dominio en específico, evitando caer en el problema de sobre-entrenamiento (*overfitting*, Tetko et al. (1995)).

Existen también modelos de lenguaje generados como reglas, los cuales pueden ser vistos como autómatas finitos o máquinas de estados finitos. Una máquina de estados finitos es una máquina que contiene estados, arcos que representan transiciones, un estado inicial, un número de estados finitos y un alfabeto (Roche (1997)).

Estos dos enfoques para el uso de modelos de lenguaje, aquellos que contienen probabilidades y los que son un conjunto de reglas, proveen diferentes beneficios y tiene diferentes debilidades. Un modelo basado en reglas si es lo suficientemente extenso, detallado y construido de forma cercana al óptimo, presentará una precisión muy alta, pero normalmente tiene una tasa de rechazo alta para lograr su objetivo. Por el contrario, los modelos de lenguaje probabilístico presentan un rechazo menor o nulo dependiendo la implementación que se utilice, esto sacrificando precisión.

Sistemas de *call routing* basados en modelos de lenguaje

En el trabajo de Garfield y Wermter (2005), se presenta un modelo de lenguaje diseñado como una gramática convertida a expresiones regulares implementada como un autómata finito. La construcción del autómata se realizó a partir de expresiones regulares, las cuales modelan el tipo de expresiones que el sistema puede reconocer del usuario. La construcción de dichas expresiones regulares se realizó de una manera semiautomática. Primero, las instancias del *corpus* fueron etiquetadas simbólicamente para hacer más fácil la identificación de patrones, después mediante técnicas estadísticas extrajeron dichos patrones. Su implementación mostró en algunos casos que ciertas clases contenían expresiones que eran tan descriptivas que podían alcanzar un 100 % de precisión sobre ciertas secuencias individuales y mantener globalmente bajo el nivel de error. Por otro lado, algunas otras instancias generaron expresiones muy generales, lo cual reduce el desempeño; esto es natural debido a la ambigüedad existente en los sistemas de habla espontánea. Su método reporta una precisión del 36 % y un recuer-

do del 66 % en el *corpus* utilizado por Durston y Kuo (2001), el cual contiene 4,000 instancias divididas en 17 clases. En sus experimentos se utilizó 80 % del *corpus* para entrenamiento y construcción de la máquina de estados finitos y el 20 % restante para pruebas.

Un trabajo descriptivo sobre el funcionamiento de esta técnica es el realizado por Huang y Cox (2004). En dicho trabajo proponen el uso de múltiples modelos de lenguaje para detectar información útil dentro de una instancia. La salida de estos múltiples modelos de lenguaje es después examinada y procesada por un modelo oculto de Markov (Rabiner (1989)), el cual decide si están presentes o no secuencias putativas. Huang y Cox (2004), parten de la idea de que un modelo de lenguaje multi-propósito pueda ser en ocasiones ambiguo, especialmente en la fase de decodificación, ya que aunque aparentemente contiene secuencias fonéticas claves de diferentes rutas, puede que aparezcan secuencias no útiles o que no contenga secuencias significativas para alguna clase.

El método de Huang y Cox (2004) consta de los siguientes pasos. Construyen un LM con *n*-gramas utilizando las transcripciones del *corpus* WSJ (*Wall Street Journal*), particularmente utilizaron un valor de *n* igual a 6 para sus experimentos. Después utilizan el LM junto con el reconocedor para crear secuencias de texto (*como puede verse los LM son una extensión al modelo de Keyword Spotting*). Su siguiente paso es construir un nuevo modelo de lenguaje basado en las cadenas de texto que produjo el reconocedor junto con el LM anterior. Este último paso lo repiten hasta alcanzar cierto umbral, el cual es definido manualmente. De esta manera construyen un nuevo modelo de lenguaje, el cual intenta mejorar el reconocimiento sobre aquellas instancias que el modelo anterior no realiza correctamente. De este modo su sistema puede verse como un enfoque de *Boosting* mediante un algoritmo iterativo. Debido a esto, dentro de su esquema de funcionamiento una instancia puede ser reconocida y clasificada por el primer LM o necesitar de más niveles para lograr ser reconocida. En las pruebas reali-

zadas, el nivel máximo que se logró alcanzar para obtener una tasa de clasificación alta, sin caer en el sobre-entrenamiento requirió 18 LM.

Huang y Cox (2004) reportan resultados de clasificación correcta del 72.6 % y de un 86.5 % utilizando múltiples LM, lo que refleja una notoria mejora. El *corpus* de prueba consistió de 15,000 ejemplos, de los cuales utilizaron 4,511 para entrenamiento y 3,518 para pruebas. La razón por lo que no utilizaron todos los ejemplos es que el *corpus* original contenía 61 clases y ellos acotaron el problema a sólo 18 clases, aquellas que según su análisis eran las más diferentes entre sí. Así mismo, reportan que sus *corpus* de entrenamiento y prueba cuentan con un vocabulario 1,608 palabras. De esto, modelos de lenguaje detectaron que existían alrededor de 51 palabras clave capaces de aumentar la clasificación, debido a ser distintivas de una clase. Desafortunadamente detectaron que un gran número de instancias no contenían palabras clave, pero aún así podían ser clasificadas (al menos por un humano) pese a su ambigüedad. El método anterior posee una parte automática para la generación de los múltiples LM. El primero es generado de una manera semiautomática, pues el *corpus* fue escogido, ajustado y analizado para extraer la información más importante y así construir el LM.

En el trabajo de Balakrishna et al. (2006), proponen un método de generación estadística de modelos de lenguaje, para ser utilizado en el reconocimiento e identificación de instancias de sistemas de habla espontánea. Además, utilizan herramientas como *WordNet* (Fellbaum (1998)) y el tesauro de Roget (1852). Su trabajo reduce la tarea manual, normalmente asociada en el desarrollo de aplicaciones IVR (*Interactive Voice Response*), pero hace hincapié en minimizar el WER (*Word Error Rate*).

La forma de representar su SLM es mediante probabilidades, de tal manera que una instancia es analizada en cada una de sus palabras, proporcionando pesos que al final darán una probabilidad o valor de pertenencia para las distintas clases; estás probabilidades son asignadas *a posteriori*. El *corpus* utilizado contiene 20,804 instancias con

55 posibles clases. Después de un análisis se determinó que un total de 23 clases eran suficientes para cubrir las 55 clases originales del *corpus,* obteniendo una precisión del 83.6 % en el *corpus* de prueba.

3.1.3. Clasificación de textos cortos

Uno de los enfoques más nuevos para la resolución del problema de *call routing*, se basa en considerar cada intervención de una conversación como un texto corto. Estos textos cortos son conocidos en la literatura como *utterances* o elocuciones.

La principal diferencia entre el enfoque de clasificación de textos cortos (*utterance classification*) con respecto a los enfoques mencionados anteriormente, es el ver la petición oral como un documento. De este modo la tarea de encaminar las llamadas o conocer la intención del usuario se ve como una tarea de clasificación de textos. Este enfoque es más robusto que los anteriores pues al sacar provecho de la frase completa, no liga una clase con un conjunto reducido de palabras clave.

Uno de los primeros trabajos utilizando este enfoque es el propuesto por Gorin et al. (1997) para mejorar el desempeño del sistema *How may I help you?* de AT&T. Para su trabajo utilizan un *corpus* de 10,000 ejemplos recabados de la interacción entre clientes y agentes humanos. El *corpus* utilizado fue agrupado en 14 clases específicas y una clase extra llamada *other*. El *corpus* fue etiquetado manualmente. El 84 % de los ejemplos del *corpus* correspondían a una sola clase, mientras que el 16 % fue etiquetado como perteneciente a 2 clases y sólo el 0.6 % de los ejemplos fue etiquetado como perteneciente a 3 clases. El multi-etiquetado se da por la ambigüedad presente en las peticiones de los usuarios. Cabe destacar que al realizar las evaluaciones del desempeño de su sistema, cuando se trataba de un ejemplo multi-etiquetado cualquiera de las etiquetas era tomada como válida. Lo anterior debido a que su sistema propone una sola clase para cada ejemplo. Este trabajo reporta la precisión y el rechazo de su sistema, así mismo

realiza un análisis cuantitativo para determinar cuál es el máximo rechazo aceptable para ellos con base a la precisión mínima deseada. Este equilibrio lo encuentran en el 75 % de precisión y en el 69 % de rechazo.

Otro trabajo representativo de este enfoque es el presentado por Alshawi (2003), en el cual se utilizó un *corpus* con 10,470 ejemplos de entrenamiento y 5,005 de prueba divididos en 54 acciones. En este caso sólo se usaron ejemplos que fueron reconocidos correctamente por el sistema de reconocimiento de voz. El proceso de entrenamiento utiliza un modelo parecido al *boosting,* pues consta de un proceso iterativo en el cual a un clasificador se le da un ejemplo, éste lo clasifica y proporciona un porcentaje de confianza. Si la confianza es menor a un umbral preestablecido vuelven a calcular los pesos para incrementar la confianza. Este proceso es repetido hasta que todos los ejemplos del *corpus* de entrenamiento son clasificados con una confianza mínima preestablecida. Con este trabajo Alshawi (2003), enriquece el trabajo propuesto por Gorin et al. (1997) utilizando *n*-gramas para el modelo y la representación de las *utterances*. En este trabajo reportan un máximo de precisión del 97.5 % con un 50.0 % de rechazo y un mínimo de precisión del 80.8 % con un 0.0 % de rechazo.

3.2. Conclusiones

Como puede verse existen muchos trabajos para intentar solucionar el problema de *call routing*. La tabla 3.1 en la página 31 muestra un resumen de las características más destacadas de los trabajos analizados en este capítulo.

Estos trabajos son realizados tanto por grupos de investigación de las áreas de procesamiento de textos así como grupos de análisis digital de señales. Estas áreasabordan el problema de manera distinta. La mayoría de los trabajos con enfoque *keyword spotting* se basan en la mejora o unión del reconocedor y el sistema de clasificación.

Por último, cabe mencionar que el enfoque de clasificación de textos nos brinda un marco de referencia apropiado, bajo el cual es posible integrar aquellas elocuciones mal transcritas por el reconocedor. En este caso, es suficiente incluir una nueva clase donde agrupar todas las transcripcines erróneas.

Esta tesis aborda el enfoque de clasificación de textos cortos y a diferencia de los trabajos presentados en el estado del arte, donde no toman en cuenta las elocuciones mal transcritas para el cálculo de sus resultados (Tang et al. (2003); Wilpon et al. (1990)), esta tesis las considera dentro de los cálculos de precisión y de rechazo. En el siguiente capítulo se detalla la solución propuesta.

Tabla 3.1: Descripción de los principales trabajos del estado del arte

Autores	Entrenamiento	Prueba	Clases	F-Score	Precisión	Rechazo	Pesado	Clasificador
Min Tang, Bryan Pellom, Kadri Hacioglu	668	75	6	-	72.20 % 69.40 % 73.40 % 75.90 %	-	Modelos de Lenguaje generado estadísticamente BOW(bag of words) BOW BOW	LM(medir la probabilidad de un ejemplo pertenezca a un modelo de lenguaje) NaiveBayes TFIDF likehood SVM
Shelia Garfield and Stefan Wermter	2,800	1,200	9	81.59 % 59.60 % 65.02 40.8	-	-	Vector de ocurrencia con la frecuencia de la palabra por clase como entrada Convierten la frase a expresiones regulares	Simple Recurrent Network(SRN) SVM sigmoid Kernel SVM polynomial Kernel Autómata Finito
J. G. Wilpon, L. R. Rabiner, C. H. Lee and E. R. Goldman	71,311	3,689	5	-	92.48 %	-	Booleano	Keyword Spotting
Joo-Gon Kim, Ho-Youl Jung and Hyun-Yeol Chung	3,753	389	23		75.60 %	-	Booleano	keyword Spotting
Q. Huang y S. Cox	4,511	3,518	18	-	86.50 %	-	Booleano	18 Modelos de Lenguaje
Mithun Balakrishna, Cyril Cerovic, Dan Maoldovan, Ellis Cave	20,804	10 Cross	23	-	83.60 %	-	Booleano	Modelos de Lenguaje
A. L. Gorin and B. A. Parker, R. M. Sachs and J. G. Wilpon	8,000	2,000	15		75.00 %	69.00 %	Probabilidad a priori	Clasificador Probabilistico(no mencionan cual)
Hiyan Alshawi	10,470	5,005	54		97.50 %	50.00 %	Probabilidad a priori/N-gramas	Proceso Iterativo para aumentar la confianza(Boosting)

Capítulo 4

Método de clasificación multi-etapa

4.1. Introducción

El método propuesto en esta tesis plantea primero un esquema para el pesado de los términos, el cual cubre una doble función: (i) determinar la importancia de un atributo asociándole un peso; y (ii) este peso servirá para seleccionar los atributos más relevantes. Una vez seleccionados los atributos estos serán utilizados para el entrenamiento de los diferentes clasificadores usados por nuestro método.

El esquema de clasificación consta de dos etapas. La primera etapa tiene como objetivo lograr una alta precisión en la clasificación de las intervenciones, a pesar de generar una alta tasa de rechazo. La segunda etapa busca reducir este rechazo sin afectar la precisión obtenida en el paso previo.

En la figura 4.1, se muestra la arquitectura del método de clasificación multi-etapa. Este esquema de clasificación recibe como entrada la transcripción automática de una petición realizada por un usuario (i.e. una instancia). Durante un primer paso, esta petición es clasificada para ver si pertenece a alguna de las n clases del *corpus*. Si la instancia no es etiquetada como *unknown*, la instancia se considera definitivamente cla-

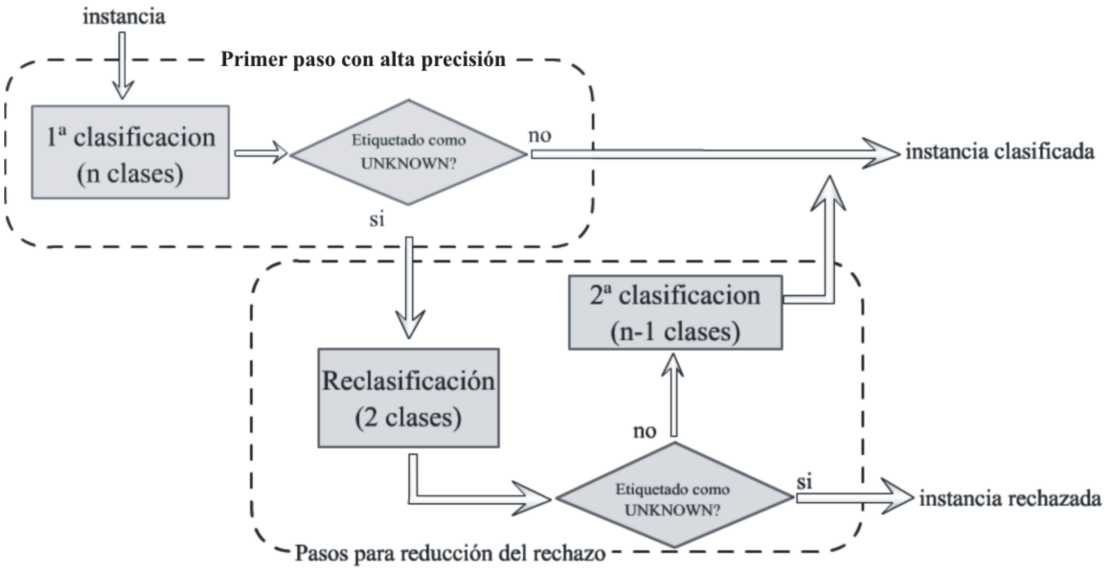

Figura 4.1: Arquitectura general del método de clasificación multi-etapa.

sificada (i.e. instancia clasificada). En caso contrario, si la instancia es etiquetada como *unknown*, ésta es analizada durante un segundo paso. Este segundo paso inicia por evaluar si la instancia puede pertencer a otra clase diferente de *unknown.* Si este análisis determina que efectivamente la instancia es *unknown* la instancia es rechazada. En caso de que la instancia no fuese etiquetada como *unknown* la instancia es reclasificada con un clasificador que sólo conoce las $n - 1$ clases restantes (excluyendo *unknown*). Este último clasificador no rechaza instancias por lo que su salida es considerada la clasificación definitiva de la instancia.

En las secciones siguientes se detalla el pesado de atributos, la selección de atributos relevantes, la representación de los documentos y etapas de clasificación de nuestro método.

4.2. Pesado de atributos

Como se ha discutido en el capítulo 1, la forma en la que se pesan los atributos afecta el desempeño de los clasificadores. Un método sencillo es el pesado booleano,

pero su misma sencillez lo hace deficiente para caracterizar clases que se encuentren muy cercanas entre sí, es decir, que su vocabulario sea bastante similar; por este motivo se decidió que el primer paso fuera el pesado de los atributos.

En este trabajo se parte del enfoque de pesado *tfidf* Yun-tao et al. (2004), el cual ha mostrado muy buenos resultados en las áreas de clasificación de textos y documentos. El enfoque clásico de *tfidf* se basa en la frecuencia con la que aparece una palabra en un documento. En particular, en el caso del problema de *call routing*, la mayor parte de las instancias no contienen palabras repetidas. Esto conlleva a que los valores tf tiendan ha ser muy bajos; además la cantidad de clases y documentos dentro de cada clase generan un valor *idf* muy alto, por lo cual algunas instancias son difíciles de caracterizar. En varios casos los valores obtenidos son muy cercanos a cero. Esto nos lleva a plantear la primera modificación al enfoque tradicional de *tfidf*.

Para efectos de la explicación repetimos las fórmulas dadas en la sección 2.3.2 en la página 11. Las siguientes fórmulas son usadas para calcular el valor *tfidf* tradicional

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{4.1}$$

$$idf_i = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|} \tag{4.2}$$

$$tfidf = tf_{i,j} * idf_i \tag{4.3}$$

En el esquema tradicional *tfidf*, el valor de *tf* calcula la ocurrencia de un término en un documento. Dado que en esta tesis se está trabajando con textos cortos los términos prácticamente no se repiten en un documento. Es por esto que se plantea como primera modificación la fórmula 4.4 para calcular el valor *tf*, donde $C(n_{i,j})$ es la ocurrencia de un término *i* en todos los documentos de la clase *j*. Este nuevo valor *tf* es normalizado

por el número de términos en la clase $\sum_k C_{k,j}$:

$$tf_{i,j} = \frac{C(n_{i,j})}{\sum_k C_{k,j}} \tag{4.4}$$

Ahora bien, el valor *idf* también es modificado para considerar valores a nivel de clase y no de documento. Siendo así $|\{C_j : t_i \in C_j\}|$ el número de clases en las que el término t_j aparece:

$$idf_i = \log \frac{|C|}{|\{C_j : t_i \in C_j\}|} \tag{4.5}$$

De esta manera cada clase es vista como un documento, así nuestro *corpus* de *N* clases con *M* ejemplos por clase, se reduce a *N* clases con un ejemplo por clase. Al utilizar este enfoque, una instancia ya no es vista como un documento, en su lugar los ejemplos son ahora frases dentro de un documento. Esta modificación al pesado *tfidf* permite aprovechar las repeticiones de ciertos términos entre las diferentes clases, mejorando así la caracterización de cada una de las clases.

La modificación propuesta permite caracterizar mejor las clases pese a trabajar con textos cortos. En contraste a la fórmula original que saca provecho de la extensión. Aún así esta fórmula presentaría ciertos problemas. En especial debido a la extensión de algunas clases. Palabras con una muy baja incidencia dentro de una clase, pero que sólo aparezcan en una clase, provocarán un *idf* que generará un valor para maximizar el término, pero el valor *tf* de la palabra sería reducido en exceso debido a la función de normalización en la fórmula *tf*. Es por esto que se propone una función para el cálculo de *tf*, la cual no minimice términos con una baja ocurrencia.

Dado el objetivo perseguido para el cálculo de *tf*, se propone una función que realce la relación entre un término y la clase que lo contiene pero que no penalice términos que aparecen pocas veces. La modificación realizada a *tf* en la fórmula (4.4) afecta a

una clase y presenta la relación de un término con toda la clase. Por lo que podemos ver a *tf* como la probabilidad de que un término pertenezca a una clase y no solamente a una instancia o documento. De este modo *tf* puede ser vista como la probabilidad condicional de que dado un término t_i ocurra la clase c_j. La ecuación (4.6) muestra la forma en la que se calcula la probabilidad condicional de cada término para ser utilizada en la nueva fórmula para el cálculo de *tf*.

$$tf_{i,j} = P(c_j|t_i) = \frac{f_i^{c_j}}{\sum_{\forall c_k \in C} f_i^{c_k}} = p_{t_i}^{c_j} \tag{4.6}$$

En esta nueva fórmula C es el conjunto de clases en el *corpus*. El valor $f_i^{c_j}$ es la frecuencia del término t_i en la categoría c_j. Puede observarse que la nueva fórmula *tf* calculada como una probabilidad condicional genera que mientras más veces un término esté presente en las instancias de una clase, mayor será su peso. De esta manera, $p_{t_i}^{c_j}$ es la probabilidad de que se trate de la clase c_j dado el término t_i. Ahora bien, mientras mayor sea el *corpus* más confiable será el peso asociado a los términos, y por ende el modelo tenderá a un mejor desempeño.

Como puede observarse, bajo este esquema de pesado, un término tendrá diferentes pesos para cada una de las clases, pero un término que apareciera sólo una vez en una clase tendría un valor *tf* muy alto, identificando a ese término como una característica muy importante de esa clase. Esto es de particular importancia para la clase *unknown*. Bajo esta clase se encuentran todas aquellas intervenciones que por errores de reconocimiento son rechazadas. Estas palabras erróneas, que difícilmente tendrán una alta frecuencia, podrán ser tomadas en cuenta y tener un peso asociado alto.

En resumen, la idea intuitiva bajo este esquema de pesado es diferenciar las clases a través de los términos más comunes en ella. Debido a las características que presenta este esquema de pesado, al resaltar los valores de aquellos términos importantes a una

clase y minimizar los que no son útiles, también utilizaremos este peso para seleccionar los atributos más relevantes para la clasificación. A continuación se detalla este proceso de selección.

4.3. Selección de atributos

Como se mencionó en la sección anterior, el peso que calculamos busca diferenciar los términos comunes de cada clase al asignarles pesos diferentes para cada una de ellas. Esto nos es útil durante el paso de selección de atributos.

La selección se realiza conservando sólo aquellos atributos que se encuentren por encima de un umbral μ (el cual se definirá posteriormente). Aún cuando un atributo no sea seleccionado para una clase c_i no significará que será eliminado del vocabulario en este momento. Ya que existe la posibilidad de que en otra clase c_j dicho término tenga un valor mayor a μ. Al momento de construir nuestra representación vectorial todos aquellos atributos que no estén por encima del umbral μ su peso es considerado 0. De esta manera, sólo se eliminarán aquellos términos distribuidos uniformemente en la mayoría de las clases y por ende sin importancia discriminativa. Es importante no perder de vista, que es posible dejar de caracterizar algunas instancias si el valor de μ es muy alto.

Al seleccionar atributos normalizamos los valores calculados anteriormente. Normalizamos de modo que todos los valores se encuentren en un rango de 0 a 1. La normalización realizada se ilustra en la fórmula (4.7). Esta normalización se logra al dividir el peso de un término ($p_i^{c_j}$) entre el peso máximo existente para el término i en cualquier clase ($\underset{\forall c_k \in C}{\max}(p_i^{c_k})$). Con esta normalización aseguramos que exista, al menos

en una clase, un peso de 1 para el término i.

$$w_i^{c_j} = \frac{p_i^{c_j}}{\max\limits_{\forall c_k \in C}(p_i^{c_k})} \tag{4.7}$$

Por último, una vez que hemos terminado nuestra representación vectorial substituyendo por 0 todos aquellos atributos cuyo peso no superan el umbral, la dimensionalidad del modelo vectorial es reducida eliminando todas aquellas columnas (atributos) que sólo contengan ceros.

Como es de esperarse, el umbral μ tendrá un efecto directo en la precisión así como en el rechazo alcanzado. En nuestro caso, este umbral se determinó empíricamente como se observará en el capítulo de resultados.

4.4. Primera clasificación con *N* clases

Una vez calculados los pesos de todos los términos de cada clase y realizado el proceso de selección de atributos estamos listos para el entrenamiento de este primer clasificador.

En este paso se le dan al clasificador las *N* clases que aparecen en el *corpus*. Como ya habíamos mencionado, la clase *unknown* contiene aquellas instancias que queremos rechazar, por lo cual toda instancia que el clasificador etiquete como *unknown* será rechazada. Para lograr esto y tener una alta precisión en las *N-1* clases restantes se debe escoger un valor de μ que elimine el mayor número de atributos o términos capaces de generar ambigüedad en las *N-1* clases que se necesitan clasificar con una alta precisión. De esta manera cuando el clasificador en la fase de pruebas encuentre instancias con algún grado de ambigüedad, los clasificará como *unknown* ya que esta clase por sí misma contiene términos ambiguos.

El enfoque propuesto en este clasificador requiere una alta precisión y a su vez generará rechazo, durante este paso lo que nos interesa es tener una alta precisión. Debido a que los pesos representan probabilidades con respecto de una clase, y el enfoque en la selección de atributos acentúa dichas probabilidades , se considera que los clasificadores con un enfoque probabilístico tendrán un mejor desempeño y un comportamiento más estable en la clase *unknown*.

En la fase de pruebas compararemos diferentes métodos de clasificación para sustentar este argumento. Independientemente de esto, el *corpus* contiene instancias repetidas dentro de las clases debido a que muchas personas preguntan las cosas de una misma manera. Esto es una característica que desea ser explotada para mejorar el desempeño y los clasificadores de enfoque probabilísta toman ventaja de este tipo de casos, por lo cual son la primera elección como clasificadores candidatos a utilizarse en este paso.

4.5. Recuperación de instancias rechazadas

La alta precisión buscada en el paso anterior no sólo rechaza instancias de la clase *unknown*, sino también aquellas instancias ambiguas o demasiado ruidosas. Para buscar una reducción en la tasa de rechazo generada por el clasificador anterior, se propone el uso de un clasificador capaz de reconocer las instancias de clase *unknown* de aquellas que pueden pertenecer a las otras *N-1* clases. Esto se logrará re-etiquetando el *corpus* de manera que sólo contenga 2 clases, la clase *unknown* y la clase de *NO-unknown.*

Con este *corpus* re-etiquetado se calculan los nuevos pesos de cada atributo para cada clase utilizando el mismo esquema de pesado que en el paso anterior, pero ahora calculado sólo para 2 clases. Después se construye el modelo vectorial para entrenamiento, seleccionado aquellos atributos que superan el mismo umbral que se utilizó para el primer clasificador.

Este clasificador generará sólo dos posibles salidas *unknown* (o rechazada) y *NO-unknown* ,es decir, la instancia era ambigua por lo que fue eliminada del primer clasificador con alta precisión. Son las instancias *NO-unknown* las que alimentarán al siguiente clasificador el cual buscará re-clasificarlas para asignarles la clase correcta.

Hay que recordar que a pesar de que las instancias mal clasificadas cayeron en la clase *unknown* esto fue realizado por un clasificador multiclase donde el peso de los atributos variaba con respecto a decenas de clases. En cambio el objetivo de este clasificador binario es distinguir las instancias mal clasificadas, para lo cual los pesos de los atributos han sido recalculados.

Este es un clasificador binario por lo que los enfoques de clasificación basados en máquina de vectores de soporte o clasificadores probabilísticos tendrían un buen desempeño, aunque consideramos que debido al método de pesado la diferencia entre ambos sistemas de clasificación será mínima.

4.5.1. Extracción de instancias ruidosas

Durante este paso las instancias clasificadas como *NO-unknown* por el clasificador binario, son re-etiquetadas a su clase original para así ser alimentadas al tercer y último paso de clasificación, dado que en el paso anterior se extrajeron todas las instancias de clase *unknown*.

4.5.2. Reclasificación con *N-1* clases

Finalmente, las instancias clasificadas como *NO-unknown* se busca que sean reclasificadas como alguna de las otras *N-1* clases, además se da por hecho que en este paso ya no llegarán instancias pertenecientes a clase *unknown*, por lo que este clasificador sólo debe ser capaz de reconocer las *N-1* clases identificables.

Para lograr este objetivo, la forma de entrenamiento de este clasificador deberá variar con respecto del primer clasificador. Este clasificador, aunque es entrenado con las mismas instancias de las primeras *N-1* clases, no se le muestra ninguna instancia de la clase *unknown*. Esto genera la segunda diferencia, la cual consiste en que los pesos calculados en este paso difieran de los obtenidos en el primer pesado de atributos.

Como este clasificador busca sacar provecho del peso asignado sólo sobre *N-1* clases y no va a rechazar instancias, el umbral μ con el cual se entrene para alcanzar su máxima precisión, difícilmente será el mismo que el usado en los pasos previos.

4.6. Resumen

El método propuesto en esta tesis consta de dos partes principales.

- Un esquema de pesado el cual provee un valor útil para la extracción de características, además de ser discriminativo y ayudar a diferenciar entre clases.
- Un método de clasificación multi-etapa, el cual busca reducir el número de instancias que son rechazadas durante la clasificación.

En el siguiente capítulo se detallan las pruebas destinadas a caracterizar los datos. También se muestran los experimentos enfocados a proveer una base para la evaluación del esquema de pesado y el método de clasificación multi-etapa presentados en este capítulo.

Los experimentos descritos nos permiten evaluar el desempeño de nuestro trabajo así como compararlo con otros trabajos los existentes en el área. Estos experimentos demuestran el impacto del pesado en la precisión y el efecto de la clasificación multi-etapa para la reducción del rechazo.

Capítulo 5

Corpus y Resultados

En este capítulo se presentan los resultados obtenidos en esta tesis y su comparación con los métodos existentes actualmente. Para poder presentar los resultados, primero se dará una descripción del *corpus* que se utilizó para realizar las pruebas. Posteriormente se detallan las métricas de evaluación utilizadas en esta tesis. Al haber sentado el entorno de trabajo se explican los experimentos base (o de referencia). Finalmente se presentan los resultados alcanzados demostrando el desempeño del método presentado en este trabajo.

5.1. Descripción del *corpus*

Para esta tesis usamos un *corpus* de 24,638 transcripciones automáticas en español (recolectado por Nuance[1] utilizando la técnica del Mago de Oz). El cual nos brindó ejemplos tales como los mostrados en la tabla 5.1 en la página siguiente. Todas las instancias fueron manualmente etiquetadas y distribuidas en 24 clases de acuerdo al criterio de Nuance. Estos datos fueron obtenidos para el centro de llamadas de una compañía que provee servicios de televisión, internet y telefonía.

[1]Nuance es un fabricante de soluciones para procesamiento de imágenes y voz. http://spain.nuance.com/company/

Clase asignada en el *corpus*	Ejemplo de instancia perteneciente a la clase
Payment	Necesito hacer un **pago**
Operator	servicio al cliente
NewService	con departamento de ventas por favor
NewPhone	Cuanto cuesta un nuevo teléfono
Balance	preguntas acerca de **pago**
TechHelp	No se por que falla
NewInternet	Quisiera contratar internet
NewCable	quiero saber cuanto cuesta el costo del cable
Cancel	quiero cortar la la linea
ChangeService	quiero saber sobre las promociones
TechInternet	ya estoy harto de tratar de que reparen mi internet
Internet	es relacionado con el internet
TechCable	bajar el cable
Billing	necesito aclarar un **pago**
Phone	servicio al cliente telefonos
ChangePhone	quiero contratar larga distancia
Cable	alta definicion cable de alta definicion
Move	necesito cambiar mi direccion para **pago**
PaymentArrangement	un arreglo de **pago**
ChangeCable	yo quiero servicio de alta definicion
TechPhone	necesito mejorar mi servicio de telefono
PPV	pago por evento
Appointment	estoy esperando un tecnico
unknown	El motivo de su llamada

Tabla 5.1: Esta tabla muestra ejemplos de las instancias en las clases del *corpus*. Es fácil observa que palabras representativas como pago, se encuentra en varias clases por lo que un enfoque de *keyword spotting* no resulta eficiente.

Las primeras 23 clases son posibles destinos para una petición del usuario. La clase restante denominada "*unknown*" corresponde a los ejemplos que fueron imposibles de clasificar por el humano. La imposibilidad de clasificar los datos se debió a que la petición fue ambigua, incompleta o un error en el reconocedor de voz. Todos los ejemplos que pertenecen a esta clase deben ser rechazados por el sistema. Por esta razón en las evaluaciones se tomarán en cuenta la precisión promedio, el rechazo, el rechazo correcto y el rechazo erróneo.

5.2. Métricas de evaluación

5.2.1. Precisión promedio

Para calcular la precisión promedio es necesario calcular la precisión para cada clase tal como se muestra en la fórmula 5.1:

$$P_{c_i} = \frac{|TD_{c_i}|}{|TD_{c_i}| + |FD_{c_i}|} \tag{5.1}$$

donde P_{c_i} es la precisión de la clase c_i, $|TD_{c_i}|$ es el total de documentos de la clase c_i que se clasificaron correctamente y $|FD_{c_i}|$ es el número de documentos que se clasificaron erróneamente como de la clase c_i. La precisión promedio del clasificador es el promedio de las precisiones de cada una de las clases.

En nuestro caso la precisión promedio es calculada a partir de las precisiones de las 23 clases posibles para una petición del usuario (excluyendo la clase *unknown*).

5.2.2. Rechazo

El rechazo es el porcentaje de documentos clasificados como *unknown*, y es calculado de la siguiente manera:

$$Rechazo = \frac{|TD_{c_u}| + |FD_{c_u}|}{\sum_i |c_i|} \tag{5.2}$$

donde $|TD_{c_u}|$ es el total de documentos clasificados correctamente como *unknown* y $|FD_{c_u}|$ es el total de documentos erróneamente clasificados como *unknown*, entre el total de documentos del *corpus* $\sum_i |c_i|$.

Rechazo correcto

El rechazo correcto es el porcentaje de documentos correctamente asociados a la clase *unknown*. Para calcularlo se utilizó la fórmula 5.3

$$Rechazo\ correcto = \frac{|TD_{c_u}|}{|c_u|} \tag{5.3}$$

donde $|TD_{c_u}|$ es el total de documentos clasificados correctamente como *unknown* y $|c_u|$ es el total de documentos de la clase *unknown*.

Rechazo incorrecto

El rechazo incorrecto es el porcentaje de documentos erróneamente clasificados como *unknown*. El cálculo de este porcentaje se hizo mediante la fórmula 5.4 :

$$Rechazo\ incorrecto = \frac{|FD_{c_u}|}{\sum_i |c_i|} \tag{5.4}$$

En esta fórmula $|FD_{c_u}|$ es el total de documentos clasificados como *unknown* sin realmente serlos entre el total de documentos del *corpus* $\sum_i |c_i|$.

5.2.3. División del *corpus*

Para la evaluación del método se utilizó un esquema 80-20, es decir, para la fase de pruebas del método y esquema de pesado, el 20 % de los ejemplos de cada clase en el *corpus* fueron seleccionados de manera aleatoria (un total de 4,944 instancias). El 80 % restante fue usado para el entrenamiento de los distintos clasificadores (un total de 19,694 instancias). La distribución de instancias por clase se presenta en la figura 5.1. Como puede observarse, se trata de un *corpus* no balanceado y la forma en que se dividió el *corpus* mantuvo el desbalanceo para así presentar un esquema de pruebas lo

más cercano a la realidad[2].

El esquema de evaluación fue escogido con la intención de aproximarnos a una situación real, donde es posible encontrar en las instancias a clasificar palabras desconocidas por el clasificador. De esta manera en particular nuestros experimentos sobre el *corpus* de prueba y entrenamiento representan este caso. Esta representación se logra debido a que los vocabularios del *corpus* de entrenamiento y prueba difieren entre sí. El *corpus* de prueba cuenta con 334 términos ausentes del *corpus* de entrenamiento, es decir un 13.50 %.

5.3. Experimentos base

Como un primer paso para mostrar el comportamiento del *corpus* se utiliza un pesado booleano y una matriz de ocurrencia (bolsa de palabras binaria). Al vocabulario se agregaron todas las palabras contenidas en el *corpus* de entrenamiento, aún cuando estas aparecieran sólo una vez en toda la colección. Esto sirve como una base específica del *corpus*, dado que aunque se presentan comparaciones de esta tesis con otros trabajos, estos utilizan otros *corpus*. En este primer grupo de experimentos utilizamos diferentes clasificadores, pero utilizando en todos los casos el mismo pesado booleano descrito en el párrafo anterior.

La tabla 5.2 muestra la precisión de los clasificadores usados[3]. Como puede observarse los clasificadores probabilísticos y los métodos de *boosting* obtienen las precisiones más altas. Así mismo puede observarse que un enfoque de clasificación basado en árboles de decisión generados con el algoritmo ID3 es insuficiente para realizar la clasificación, situación esperada debido a que el vocabulario de las clases se encuentra traslapado en varios casos.

[2]También se realizaron experimentos utilizando el esquema de evaluación *10 cross-validation*. Los resultados alcanzados se reportan en el apéndice B.

[3]Los clasificadores utilizados en las pruebas son los ofrecidos por WEKA y Matlab

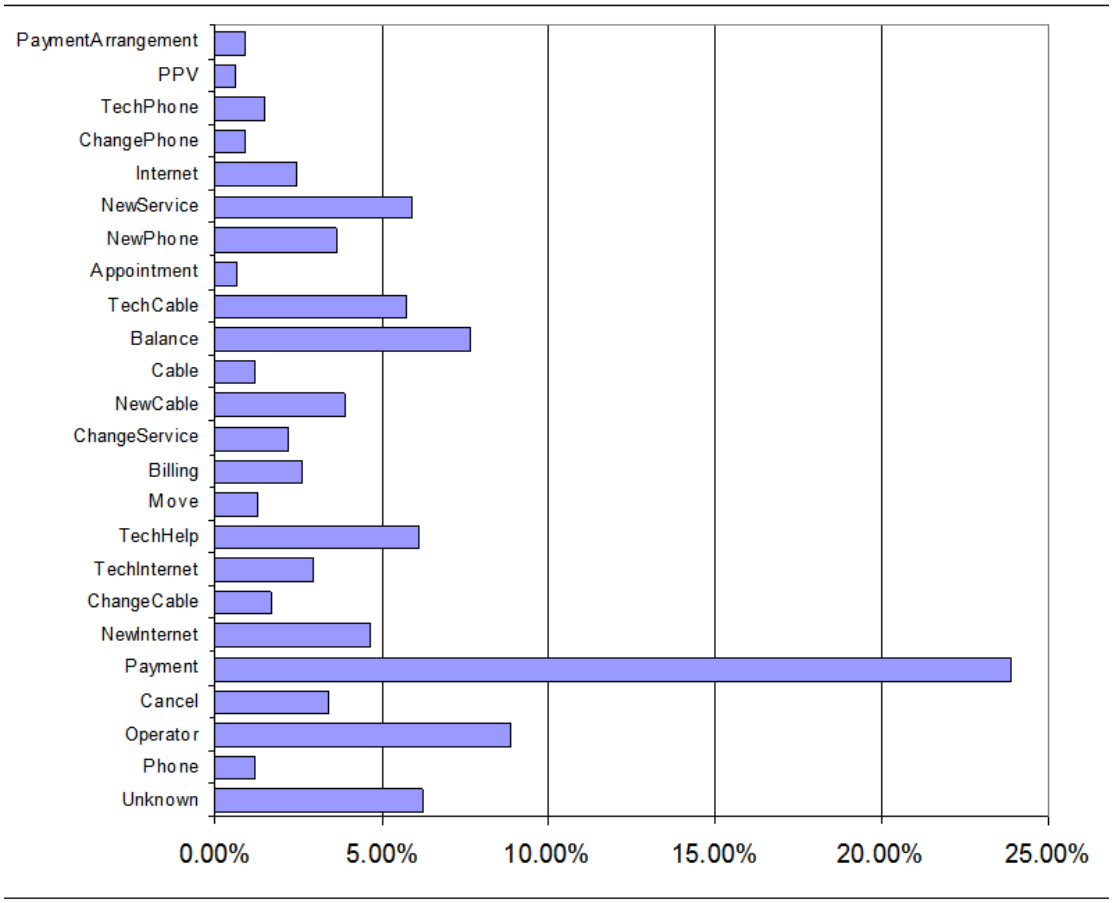

Figura 5.1: Distribución de las clases en el *corpus*. Esta figura muestra el desbalance existente en las clases del *corpus*. La clase mayoritaria *"Payment"* es la más grande con casi un 25 % del total de los ejemplos en el *corpus*. Es importante recalcar que la tercera clase más grande es la clase *unknown*, la cual contiene ejemplos de peticiones imposibles de clasificar por etiquetadores humanos, lo que ilustra el porqué es necesario rechazar instancias correctamente.

Durante la explicación del método propuesto en esta tesis se mencionó que en la primera etapa de clasificación, debido a las características del pesado, un clasificador probabilístico sería el que mejor desempeño diera.

Método	Precisión promedio	Rechazo
NaiveBayes	78.10 %	8.00 %
VFI	63.51 %	0.16 %
LogitBoost	78.71 %	8.20 %
AdaBoost(NaiveBayes seed)	78.87 %	8.10 %
AdaBoost(ID3 *seed*)	[4]no puede realizar clasificación	-
SVM	72.71 %	8.30 %

Tabla 5.2: Experimentos base utilizando todo el vocabulario presente en el *corpus*. Para estos experimentos se utilizó un pesado booleano de los atributos y una matriz $documentos \times vocabulario$ para todos los casos. Todos los métodos recibieron como entrada el mismo *corpus* de entrenamiento y el mismo *corpus* de prueba. Los clasificadores elegidos son los más representativos de los enfoques revisados en el capítulo del estado del arte.

Para demostrar los efectos que la selección de atributos tiene en este problema, realizamos experimentos con el clasificador simple de Bayes (*NaiveBayes*), utilizando el mismo pesado que en los experimentos de 5.2 pero esta vez, eliminamos palabras del vocabulario. En la clasificación de textos suelen eliminarse palabras vacías, o que tengan una frecuencia menor a un umbral. En este problema en particular, los textos son cortos, en general frases pequeñas de un renglón. Esto genera que eliminar palabras por la repetición en un documento no sea una buena opción, debido a que la mayoría de los ejemplos no contienen palabras repetidas. Por esta razón, eliminamos palabras según un umbral de frecuencia, pero calculando las frecuencias sobre todo el *corpus*. La tabla 5.3 muestra el desempeño del clasificador baseyiano simple (*NaiveBayes*) con diferentes umbrales de frecuencia. Como puede observarse el comportamiento del cla-

[4]El método ID3 calcula la entropía de cada término y utiliza la diferencia entre ellas para construir un árbol de decisión, dado que los términos se encuentran repartidos en varias clases la diferencia entre sus entropías no es suficiente para poder realizar una clasificación.

sificador empeoró inmediatamente cuando se eliminaron algunas palabras. Aunque al incrementar el umbral la precisión comenzó a mejorar, nunca alcanzó la precisión promedio al no eliminar atributos. Como mencionamos en el capítulo anterior, la selección de atributos es importante, y definir los criterios de selección depende del problema.

Método	Precisión promedio	Umbral de Frecuencia
NaiveBayes	78.10 %	0
NaiveBayes	68.30 %	1
NaiveBayes	47.53 %	2
NaiveBayes	59.40 %	3
NaiveBayes	61.89 %	4
NaiveBayes	58.91 %	12

Tabla 5.3: En este experimento se eliminaron palabras con base a la frecuencia en la que aparecían en el *corpus*. Esta tabla muestra que la selección de atributos es importante. Incluso eliminar palabras únicas afecto la precisión del clasificador.

En los experimentos realizados hasta este punto se utilizó un pesado booleano, el cual no es suficiente. Para demostrar el efecto del pesado, utilizamos ahora el mismo clasificador simple de Bayes (*NaiveBayes*) con los pesos calculados por la función de ganancia de información (*InfoGain*) alcanzando los resultados reportados en la tabla 5.4. La columna $umbral$ es el valor mínimo de ganancia de información para considerar un atributo.

En un primer experimento utilizamos un umbral de 0, dado que la ganancia de información es una medida asociada a la entropía y sus valores pueden ser negativos, pues un atributo puede no dar información alguna a la distinción entre clases. Como puede observarse el resultado es malo, por lo cual realizamos un segundo experimento. En esta ocasión fijamos el umbral a -1. Aunque el resultado fue ligeramente mejor, sigue resultando mucho menos eficiente que un simple pesado booleano. Este resultado no es extraño pues las clases y en específico las instancias por separado tienen un vocabulario muy similar, por lo cual analizadas de esa forma no ayuda a caracterizarlas.

Pesado	Precisión promedio	Umbral	Rechazo	No. de Atributos
InfoGain	09.20 %	0	8.2 %	213
InfoGain	11.45 %	-1	9.1 %	450

Tabla 5.4: Eliminando palabras con base a *InfoGain* (Ganancia de información) y utilizando su valor como peso. El umbral es el valor mínimo de entropía que un atributo debía tener para ser seleccionado. Como puede verse debido a las características del *corpus* los valores de entropía son bajos pues incluso utilizando un umbral de -1, obtuvimos muy pocos atributos y la precisión se vio afectada.

Debido a los resultados mostrado en la tabla 5.4 se pensó en pesar cada palabra de acuerdo a como ésta se relaciona con el resto de los documentos del *corpus*. De ahí que se tomará el pesado *tfidf*. El cálculo del *tfidf* se realizó por clase para obtener un punto de referencia más adecuado a la forma de ver las instancias que ésta tesis propone. Los resultados, como muestra la tabla 5.5, sea mejores que los de la ganancia de información, aunque no son mejores que los obtenidos con un pesado booleano.

Pesado	Precisión promedio	Rechazo	No. Atributos
tfidf por clase	76.0 %	19.0 %	2451

Tabla 5.5: En este experimento se calculó el *tfidf* por clase (ver fórmula 4.4 en la página 36) para obtener el peso de los atributos.

5.4. Experimentos con el pesado propuesto

En la sección anterior mostramos el desempeño de los métodos tradicionales de clasificación de textos. A continuación mostraremos el desempeño del método de clasificación propuesto en esta tesis.

Para los experimentos se utilizaron los mismos *corpus* de prueba y entrenamiento que los utilizados para las pruebas previas.

5.4.1. Utilizando un paso

El primer experimento tiene por objetivo demostrar el efecto del pesado propuesto. Para ello, se presentan los resultados al usar un clasificador simple de Bayes (*Naive-Bayes*) sin aplicar el esquema multi-etapa propuesto. Como explicamos en el capítulo anterior, el pesado representa la relevancia de una palabra para una clase dada. La tabla 5.6 muestra el comportamiento del clasificador al ir incrementando el umbral para eliminar atributos poco relevantes. Se observa que el aumento del umbral mejora la precisión del clasificador con un aumento en el rechazo. Los resultados obtenidos en estos experimentos demuestran la utilidad de una segunda etapa, la cual este orientada a disminuir el rechazo y buscar un compromiso con la precisión de la primera etapa.

Tomando en cuenta el comportamiento del clasificador, y al observar las matrices de confusión, decidimos utilizar un umbral μ de 0.3 para el primer paso del método.

Umbral μ	Atributos	Vocabulario Promedio por Clase	Precisión promedio	Rechazo
0	2474	316.46	64.9 %	1.0 %
0.1	2473	191.08	71.6 %	0.9 %
0.2	2431	144.04	87.5 %	24.2 %
0.3	2290	121.79	97.4 %	31.5 %
0.4	2074	99.66	99.5 %	40.7 %
0.5	1712	74.08	99.9 %	50.9 %

Tabla 5.6: Comportamiento del clasificador bayesiano simple utilizando el peso propuesto. En cada paso se seleccionan aquellos atributos cuyo peso normalizado sea superior al umbral μ.

5.4.2. Utilizando dos pasos

Para el segundo paso de clasificación el cual consiste en extraer las instancias de la clase *unknown* y tratar de recuperar instancias rechazadas en el primer paso, utilizamos dos clasificadores más. Ambos clasificadores utilizan el mismo pesado que el primer

clasificador y un clasificador Bayesiano simple. Las diferencias radican en que el segundo clasificador es un clasificador binario con el peso calculado sólo para 2 clases: *unknown* y el resto (las otras 23 clases). En la tabla 5.7 observamos los resultados de los 2 pasos, en la cual se aprecia que si bien la precisión disminuyó en un 1.9 % con respecto a utilizar un único paso, el rechazo disminuyó un 23.3 %. Esto es una mejora considerable respecto al desempeño global en la clasificación.

Correctamente Clasificadas			
Primer Paso	Segundo Paso	Precisión promedio	Rechazo
3396	938	95.5 %	8.2 %

Tabla 5.7: Comportamiento del método multi-etapa propuesto.

La tabla 5.7 muestra el número de instancias clasificadas correctamente en cada paso. Dicha tabla muestra la utilidad del uso de un esquema multi-etapa. Se puede ver que el primer paso clasificó 3396 ejemplos correctamente y el segundo paso 938, incrementando el total de ejemplos clasificados correctamente. Esta tabla muestra además una reducción relativa del rechazo del 384.14 % con respecto del uso de un sólo paso como se muestra en la tabla 5.6 en la página anterior.

Por último, realizamos un experimento para demostrar la pertinenecia del método al compararlo con otros métodos de clasificación usados en otros trabajos. Específicamente nos comparamos contra el método *Adaboost* usando nuestro pesado así como *tfidf*. Los resultados se muestran en la tabla 5.8

Esquema de Clasificación	Precisión promedio	Rechazo	Rechazo Correcto	Rechazo incorrecto
Método propuesto	95.5 %	**8.2 %**	**100 %**	**2.1 %**
Adaboost(Pesado propuesto)	**98.9 %**	29.7 %	100 %	23.6 %
Adaboost(*tfidf*)	78.2 %	6.8 %	60 %	4.08 %

Tabla 5.8: Método propuesto comparado con el algoritmo de *Adaboost* usando el pesado propuesto en vez del pesado *tfidf*.

5.5. Análisis de los resultados

Como puede observarse en la tabla 5.8, se presenta una tabla comparativa entre el método multi-etapa propuesto y el método de *Adaboost*. Seleccionamos *Adaboost* por ser un método que utiliza un clasificador base y de ahí intenta construir uno más potente, haciendo así un clasificador final que utiliza varios clasificadores. Esto es una de las características del *boosting* y el levering, característica que el método propuesto también explora al utilizar 3 clasificadores para construir una clasificación final. Como puede observarse las mejoras en el desempeño, mostradas por el método sobre los enfoques estándar de *boosting*, son claras al obtener un mejor compromiso entre la precisión y el rechazo. Así mismo la tabla 5.8 en la página anterior muestra el desempeño de *Adaboost* usando *tfidf*, de esta manera también se ilustra la importancia del peso propuesto en esta tesis.

Con base en lo mostrado en la tabla 5.5 en la página 51 y la tabla 5.6 en la página 52, podemos afirmar que el método de pesado es una aportación clave de este trabajo. Esto en combinación con el uso del esquema multi-etapa propuesto se tiene un método de clasificación con una importante reducción en la tasa de rechazo, como puede verse en las tablas 5.6 en la página 52 y 5.7 en la página anterior.

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Recapitulación

En este trabajo de tesis se abordo el problema del direccionamiento de llamadas (*call routing*), el cual consiste en tomar la transcripción de una petición telefónica y clasificarla. En esta tesis se desarrolló un método basado en clasificación de textos para abordar este problema. El principal objetivo del método fue encontrar un balance apropiado entre la precisión de clasificación y la tasa de rechazo. El método desarrollado utiliza un esquema de clasificación multi-etapa el cual es capaz de lograr un balance adecuado. Con este esquema se alcanzó un 95.5 % de precisión promedio, así como un 100 % de rechazo correcto y sólo un 2.1 % de rechazo incorrecto. Una mejora importante al compararnos con otros métodos (como se puede observar en la tabla 5.8 en la página 53). Aunado a este esquema de clasificación se propuso un método de pesado de los atributos, con el cual no sólo se cuantifica el valor discriminativo de cada atributo, sino también sirve para reducir la dimensionalidad del problema de aprenidzaje.

6.2. Conclusiones

Las conclusiones de nuestro trabajo pueden resumirse en las siguientes:

- Se logró definir un esquema de pesado de atributos propio para este problema, el cual ayudo a enfrentar el desbalanceo de clases así como la pequeña longitud de las instancias a clasificar.

- El esquema de pesado propuesto también permitió la selección de los atributos más relevantes para la clasificación, es decir, permitió la reducción de la dimensionalidad de nuestro problema de aprendizaje.

- Fue posible atribuir un peso relevante a todas las palabras discriminativas incluso a aquellas palabras cuya frecuencia era mínima, por presentarse una sola vez, pero de utilidad para determinar la clase.

- El esquema de pesado propuesto permite determinar la relevancia de una palabra con respecto a una clase dada, es decir, es posible establecer diferentes relevancias según la importancia de la palabra en cada clase.

- El método de clasificación multi-etapa propuesto permitió reducir el rechazo sin sacrificar la precisión, lo cual fue comprobado con un *corpus* recolectado en situaciones reales y de gran tamaño.

6.3. Trabajo Futuro

El método propuesto en esta tesis fue probado con un *corpus* en español para un dominio específico. Como trabajo sería relevante analizar el comportamiento del método con otros *corpus* en otros dominios así como en otros idiomas. Estos experimentos ayudarán a demostrar el alcance del método.

Así mismo el método fue evaluado usando transcripciones automáticas cortas propias de un sistema de direccionamiento de llamadas. Las transcripciones automáticas cortas existen en otros dominios. El control de computadoras por voz o la solicitud de acciones a robots son áreas que requieren de la correcta clasificación de la intención

del usuario. Estas tareas pueden modelarse como peticiones orales a ser clasificadas. La forma en la que está planteado el método y el sistema de pesado lo hace flexible para abordar estos problemas.

Otro trabajo futuro es la evaluación del método de pesado de atributos en la clasificación de documentos ya sean estos cortos o no. Incluso esta evaluación podría extenderse a la clasificación no-temática de documentos, como lo es la atribución de autoría o la clasificación de opiniones.

Apéndice A

Comportamiento del umbral μ

En este anexo se presenta de forma detallada el comportamiento del primer clasificador ha medida que se incrementa el umbral μ. La tabla A.1 muestra el comportamiento del primer clasificador cuando el umbral μ es 0.0. Obsérvese que las clases mayoritarias (“a” y “m”) absorben instancias de todas las clases. La clase *unknown (“x”)* no absorbe las peticiones ambiguas del resto de las clases, incluso, no recupera todos las instancias que corresponden a ella.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
1160	0	0	1	4	0	0	1	1	0	0	0	1	0	0	0	0	0	7	0	0	0	0	0
9	419	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0
15	7	94	45	9	0	0	0	0	1	0	0	6	0	0	0	0	0	25	0	2	0	0	0
38	5	6	110	3	0	0	0	0	0	0	0	1	0	0	0	0	0	7	0	9	0	0	0
82	10	4	7	224	0	0	1	0	0	0	0	30	12	0	0	0	0	8	0	0	0	0	0
35	22	1	0	15	137	0	1	0	0	4	0	55	0	0	0	0	0	28	0	2	0	1	0
41	2	2	5	6	0	110	0	0	0	1	58	5	0	0	0	0	0	0	0	1	0	0	0
35	4	7	4	6	0	0	114	0	0	0	0	22	0	0	0	0	0	1	0	0	0	0	0
6	1	0	1	2	0	0	2	138	0	0	1	6	0	0	0	1	0	9	0	0	0	0	0
35	0	3	2	8	0	0	0	2	40	0	0	8	0	0	0	0	4	7	0	1	0	0	0
15	30	1	0	16	15	0	0	0	0	3	7	55	0	0	0	0	0	3	0	1	0	0	0
6	9	10	0	7	0	0	0	0	0	0	82	7	0	0	0	0	0	0	0	0	0	0	0
19	12	0	0	5	5	0	12	0	1	0	0	225	0	0	0	0	0	1	2	0	1	0	0
25	1	0	3	45	0	0	0	0	0	0	0	18	33	0	0	0	0	4	0	0	1	0	0
12	5	0	24	2	0	0	0	0	0	0	0	4	0	0	3	0	0	6	0	3	0	0	0
11	0	2	8	0	0	0	0	1	9	0	0	2	0	0	9	0	2	0	0	1	0	0	0
11	4	2	0	4	0	0	32	0	0	0	0	5	0	0	0	0	0	2	0	0	0	0	0
10	0	0	0	0	0	0	0	0	2	0	0	1	0	0	0	0	48	1	0	0	0	0	0
30	1	0	0	3	0	0	0	0	0	0	0	1	0	0	0	0	0	9	0	0	0	0	0
21	1	4	2	3	0	0	3	7	9	0	0	17	0	0	0	0	0	1	13	0	3	0	0
6	7	0	13	4	3	0	0	0	0	0	0	35	0	0	2	0	0	2	0	3	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	28	0	0
4	1	0	0	0	3	0	1	7	1	0	0	5	0	0	0	0	0	0	1	0	0	10	0
72	17	6	1	14	0	0	0	1	4	0	0	58	0	0	2	1	0	79	1	2	0	1	48

Tabla A.1: Matriz de confusión sin umbral 0.0

La tabla A.2 en la página siguiente muestra el comportamiento del clasificador uti-

lizando un umbral de μ 0.1. Nótese que ahora las clases mayoritarias ya no absorben a las otras clases. Sin embargo, la clase con el mayor número de palabras no únicas en su vocabulario *(“i”)* es ahora la que absorbe los ejemplos ambiguos del resto de las clases.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
1161	0	0	0	0	0	0	0	10	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0
2	410	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	122	62	0	0	0	0	110	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	147	0	0	0	0	16	0	0	0	0	0	0	1	0	0	0	0	15	0	0	0
1	0	6	0	283	0	0	0	51	0	0	0	16	21	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	152	0	0	101	0	16	0	31	1	0	0	0	0	0	0	0	0	0	0
0	0	0	2	0	0	116	0	3	0	110	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	2	4	0	0	0	180	2	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	167	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	32	77	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	13	0	0	0	11	0	0	6	0	83	0	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	4	0	117	0	0	0	0	0	0	0	0	0	0	0	0	0
0	10	0	0	13	5	0	13	4	0	0	0	238	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	26	0	0	0	34	0	0	0	0	70	0	0	0	0	0	0	0	0	0	0
0	0	0	36	0	0	0	0	13	0	0	0	0	0	0	3	0	0	0	0	7	0	0	0
0	0	0	0	0	0	0	0	16	11	0	0	0	0	0	11	0	1	0	0	6	0	0	0
0	0	0	0	0	0	0	53	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	5	48	0	0	0	0	0	0	0	9	0	0	0	0	0	0
0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0
0	0	0	0	0	0	0	0	35	13	0	0	18	0	0	0	0	0	0	16	0	2	0	0
0	0	0	45	0	1	0	0	6	0	0	0	13	0	0	1	0	0	0	0	9	0	0	0
0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0
0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0
0	0	0	0	0	0	0	0	260	0	0	0	0	0	0	5	0	0	2	0	0	0	0	40

Tabla A.2: Matriz de confusión con umbral 0.1

En la tabla A.3 se muestra el desempeño del clasificador para el umbral μ en 0.2. Como puede observarse, la clase *UNKOWN (“x”)* es ahora la que absorbe las instancias ambiguas de las otras clases. Son estas instancias las que el sistema rechazará y a su vez servirán de entrada al segundo nivel de clasificación.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
1156	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19
0	412	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24
0	0	124	62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	108
0	0	0	147	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32
0	0	0	0	297	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	63
0	0	0	0	0	156	0	0	0	0	21	0	5	0	0	0	0	0	0	0	0	0	0	119
0	0	0	2	0	0	116	0	0	0	110	0	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	181	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12
0	0	0	0	0	0	0	0	146	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21
0	0	0	0	0	0	0	0	0	77	0	0	0	0	0	0	0	0	0	0	0	0	0	33
0	0	0	0	0	14	0	0	0	0	110	0	12	0	0	0	0	0	0	0	0	0	0	10
0	0	0	0	0	0	0	0	0	0	117	0	0	0	0	0	0	0	0	0	0	0	0	4
0	0	0	0	0	0	0	26	0	0	5	0	238	0	0	0	0	0	0	0	0	0	0	14
0	0	0	0	0	0	0	0	0	0	0	0	0	55	0	0	0	0	0	0	0	0	0	75
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	59
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	1	0	0	0	0	0	24
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60
0	0	0	0	0	0	0	0	0	49	0	0	0	0	0	0	0	9	0	0	0	0	0	4
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	19
0	0	0	0	0	0	0	0	0	1	0	0	24	0	0	0	0	0	0	13	0	0	0	46
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	25	0	0	47
0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	9
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	11
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	307

Tabla A.3: Matriz de confusión con umbral 0.2

Observando la tabla A.4 es claro que el umbral μ de 0.3 tiene una mejor precisión y es la clase *unknown* la que absorbe los ejemplos ambiguos del resto de las clases. A su vez esta tabla muestra la necesidad del segundo paso dedicado a recuperar instancias rechazadas en este, puesto que, la clase con el menor número de palabras en su vocabulario es confundida al 100 % con *unknown*.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
1155	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20
0	412	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24
0	0	128	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	166
0	0	0	135	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	44
0	0	0	0	298	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	64
0	0	0	0	0	167	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	134
0	0	0	0	0	0	228	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	181	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12
0	0	0	0	0	0	0	0	146	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21
0	0	0	0	0	0	0	0	0	77	0	0	0	0	0	0	0	0	0	0	0	0	0	33
0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0	0	0	104
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	121
0	0	0	0	0	0	0	0	0	0	11	0	228	0	0	0	0	0	0	0	0	0	0	44
0	0	0	0	0	0	0	0	0	0	0	0	0	67	0	0	0	0	0	0	0	0	0	63
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	59
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	25
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60
0	0	0	0	0	0	0	0	0	33	0	0	0	0	0	0	0	22	0	0	0	0	0	7
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	19
0	0	0	0	0	0	0	0	0	1	0	0	25	0	0	0	0	0	0	13	0	0	0	45
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	72
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	7
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	5
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	307

Tabla A.4: Matriz de confusión con umbral 0.3

Los experimentos mostrados en este apéndice ilustran la repercusión del umbral μ en la precisión, así como también la capacidad de éste de rechazar instancias ambiguas o ruidosas para su posterior reclasificación.

Un umbral μ alto, incrementará la precisión del sistema, pero también aumentará la cantidad de instancias que son rechazadas.

Apéndice B

Comparación del presente trabajo con *10 cross-validation*

En este apéndice se muestra el desempeño del método propuesto utilizando la división 80 %-20 % del *corpus* y utilizando un esquema de evaluación *10 cross-validation*.

Umbral μ	precisión división 80 %-20 %	Precisión *10 Cross Validation*
0	64.90 %	71.60 %
0.1	71.60 %	82.00 %
0.2	87.50 %	97.00 %
0.3	97.40 %	98.60 %
0.4	99.50 %	99.80 %
0.5	99.90 %	100 %

Tabla B.1: Comparación del rechazo obtenido por el clasificador utilizando la división del 80 %-20 % el *corpus* contra el método de evaluación *10 cross-validation*.

En la tabla B.1 puede observarse como al utilizar *10 cross-validation* el trabajo incrementa su precisión más rápidamente que utilizando la división del *corpus*. Este comportamiento se debe principalmente a que *10 cross-validation* utiliza un 10 % diferente del *corpus* para cada iteración. Este tamaño de división reduce el vocabulario desconocido e incluso en ciertos casos no se encontró vocabulario desconocido.

Umbral μ	Rechazo división 80 %-20 %	Rechazo *10 Cross Validation*
0	1.0 %	2.77 %
0.1	0.9 %	2.5 %
0.2	24.2 %	18.60 %
0.3	31.50 %	28.5 %
0.4	40.70 %	36.8 %
0.5	50.09 %	59.6 %

Tabla B.2: Comparación de la precisión obtenida por el clasificador utilizando la división del 80 %-20 % el *corpus* contra el método de evaluación *10 cross-validation*.

La tabla B.2 muestra el rechazo obtenido en *10 cross-validation* y la división 80 %-20 %. En este caso el comportamiento del trabajo muestra que *10 cross-validation* tiene tasas de rechazo que en principio parecieran más altas que las obtenidas por la división del *corpus*, pero comparandolas con la precisión obtenida en el mismo caso, tenemos que, para una precisión de 97.00 % en *10 cross-validation* obtuvimos un 18.60 % de rechazo y en el 80 %-20 % para un 97.40 % de precisión obtuvimos un 37.5 %. Lo que muestra que el tener menos palabras desconocidas influye tanto en la precisión del clasificador como en su tasa de rechazo.

En esta tesis decidimos utilizar la división 80 %-20 % por ser un entorno más próximo a la realidad. Donde la diversidad de usuarios y problemas de reconocimiento, generan que las peticiones contengan palabras desconocidas para el clasificador durante la fase de entrenamiento.

Referencias

Aizerman, M., Braverman, E., y Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. In *Automation and Remote Control*, pages 821–837.

Alshawi, H. (2003). Effective utterance classification with unsupervised phonotactic models. In *Proceedings of HLT-NAACL 2003*, pages 1–7.

Balakrishna, M., Cerovic, C., Moldovan, D., y Cave, E. (2006). Automatic generation of statistical language models for interactive voice response applications. In *ICSLP-INTERSPEECH*, pages 1648–Wed2CaP.12.

Barnard, G. A. (1958). Studies in the history of probability and statistics: Ix. thomas bayes' essay towards solving a problem in the doctrine of chances. In *Biometrika*, volume 45, pages 293–295.

Boser, B. E., Guyon, I. M., y Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In Haussler, editor, *Annual ACM Workshop on COLT*, volume 5, pages 144–152. ACM Press.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. In *Data Mining and Knowledge Discovery*, volume 2, pages 121–167.

Carpenter, B. y Chu-carroll, J. (1998). Natural language call routing: A robust, self-organizing approach. In *ICSLP*.

Demiriz, A., Bennett, K., y Shawe-Taylor, J. (2002). Linear programming boosting via column generation. In *Kluwer Machine Learning*, pages 225–254.

Demiröz, G. y Güvenir, H. A. (1997). Classification by voting feature intervals. In Heidelberg, S. B. ., editor, *Machine Learning: ECML-97*, pages 85–92.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A., y Vapnik, V. (1997). Support vector regression machines. In *Advances in Neural Information Processing Systems*, volume 9, pages 155–161. MIT Press.

Duffy, N. y Helmbold, D. (1999). A geometric approach to leveraging weak learners. In Heidelberg, S. B. ., editor, *Computational Learning Theory*, page 638.

Durston, P. y Kuo, J. (2001). Oasis natural language call steering trial. In *Proceedings of Eurospeech*, volume 2, pages 1323–1326.

Fellbaum, C. (1998). Wordnet:an electronic lexical database. In *MIT Press*.

Freund, Y. (1990). Boosting a weak learning algorithm by majority. In *Annual Workshop on Computational Learning Theory*, pages 202 – 216.

Freund, Y. (2001). An adaptive version of the boost by majority algorithm. In *Machine Learning*, volume 43, pages 293–318.

Freund, Y. y Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. In *Journal of Computer and System Sciences*, number 55, pages 119 – 139.

Friedman, J., Hastie, T., y Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. In *Annals of Statistics*, volume 28, pages 337–407.

Garfield, S. y Wermter, S. (2005). Call classification using recurrent neural networks, support vector machines and finite state automata. In *Knowledge and Information Systems*, pages 131 – 156.

Gorin, A. L., Parker, B. A., Sachs, R. M., y Wilpon, J. G. (1997). How may i help you. In *Speech Communication*, volume 23, pages 113–127.

Hernández, L., Caminero, J., de la Torre, C., y Villarrubia, L. (1994). Estado del arte en tecnología del habla. In *Comunicaciones de Telefónica I+D*, volume 5, pages 3–27.

Huang, Q. y Cox, S. (2004). Automatic call routing with multiple language models. In *Workshop on Spoken Language Understanding for Conversational Systems*, page 2530.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. In *Journal of Documentation*, volume 28, pages 11–21.

Kearns, M. (1988). Thoughts on hypothesis boosting. In *Unpublished manuscript*, http://www.cis.upenn.edu/ mkearns/ visitada el 20 de abril de 2008. Project for Ron Rivest's machine learning course at MIT.

Kearns, M. y Vazirani, U. (1994). An introduction to computational learning theory. In *MIT Press*.

Kim, J., Jung, H., y Chung, H. (2004). A keyword spotting approach based on pseudo n-gram language model. In *SPECOM*, pages 256–259. ISCA Archive.

Kullback, S. y Leibler, R. A. (1951). On information and sufficiency. In *Annals of Mathematical Statistics 22*, pages 79–86.

McCallum, A. K. (1996). Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. In *http://www.cs.cmu.edu/ mccallum/bow*. visitado el 16 de octubre de 2007.

Minsky, M. y Papert, S. (1969). In *Perceptrons; an introduction to computational geometry*, page Chapter 12. MIT Press. Una de las primeras referencias sobre la definicion e implementacion de un clasificador bayesiano ingenuo(NaiveBayes).

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE, 77 (2)*, pages 257–286.

Roche, E. (1997). Compact factorization of finite-state transducers and finite-state automata. In *Nord. J. Comput.*, volume 4, pages 187–216.

Roget, D. P. M. (created 1805, released 1852). Roget's thesaurus. http://thesaurus.reference.com/. English thesaurus.

Salton, G. (1960). Smart (salton's magic automatic retriever of text). disponible para descargar bajo la licencia GNU en ftp://ftp.cs.cornell.edu/pub/smart/. Creado en Cornell University.

Salton, G. (1989). Automatic text processing:the transformation, analysis and retrieval of information by computer. Addison-Wesley.

Salton, G. y Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. In *Information Processing and Management 24(5)*, pages 513–523.

Salton, G., Wong, A., y Yang, C. S. (1975). A vector space model for automatic indexing. In *Communications of the ACM*, volume 18, pages 613–620.

Schapire, R. (1990). Strength of weak learnability. In *Journal of Machine Learning*, volume 5, pages 197–227.

Sebastiani, F. (2002). Machine learning in automated text categorization. In *ACM Computing Surveys*, volume 34(1), pages 1–47.

Tang, M., Pellom, B., y Hacioglu, K. (2003). Call-type classification and unsupervised training for the call center domain. In *Automatic Speech Recognition and Understanding IEEE Workshop*, pages 204– 208.

Tetko, I., Livingstone, D., y Luik, A. (1995). Comparison of overfitting and overtraining. In *J. Chem. Inf. Comput. Sci.*, volume 35, pages 826–833.

Wilpon, J. y Rose, D. (1994). Applications of speech recognition technology in telecommunications. In *Proceedings of ICSLP-94*, pages 667–670.

Wilpon, J. G., Rabiner, L. R., Lee, C. H., y Goldman, E. R. (1990). Automatic recognition of keywords in unconstrained speech using hidden markov models. In *IEEE Trans. Acoust., Speech Signal Processing, vol. 38, no. 11*, pages 1870–1878.

Yining, C., Jing, L., Lin, Z., Jia, L., y Runsheng, L. (2001). Keyword spotting based on mixed grammar model. In *International Symposium on Intelligent Multimedia, Video and Speech Processing*.

Yun-tao, Z., Ling, G., y Young-cheng, W. (June 26, 2004). An improved tf-idf approach for text classification. In *Journal of Zhejiang University SCIENCE ISSN 1009-3095*, volume 2, pages 944– 946.

Zhang, H. (2004). The optimality of naive bayes. In *International FLAIRS conference*. Best paper award winner (second place).