UNIVERSIDAD DE SEVILLA

DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS

Combinación de Sistemas mediante Aprendizaje Automático en Tareas de Procesamiento de Lenguaje Natural

Memoria de Tesis Doctoral para la obtención del grado de Doctor en Informática presentada por

D. Fernando Enríquez de Salamanca Ros

dirigida por el doctor

D. José Antonio Troyano Jiménez

Sevilla, junio de 2011.

A Lourdes, por su apoyo y comprensión y a Nora,
por hacer de nosotros unos padres orgullosos y felices

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Son muchas las personas a las que me gustaría agradecer su contribución directa o indirecta en este trabajo. En primer lugar a mi tutor, José Antonio Troyano, del que únicamente he obtenido apoyo y colaboración desde mis primeros pasos como investigador, así como a mis compañeros Juan Antonio Álvarez, Fermín Cruz, Javier Ortega, Carlos García y Víctor Díaz, por todos los momentos en los que me han sabido escuchar y aconsejar durante este largo proceso.

También tengo mucho que agradecerle a los familiares y amigos en los que uno se apoya en los momentos adversos y con los que siempre se desea compartir los momentos de felicidad. Echando la vista atrás, me acuerdo inevitablemente de las personas de las que tanto he aprendido, con las que me gustaría charlar sobre estas líneas y que por desgracia ya no se encuentran a mi lado. Con la esperanza de que se hubiesen sentido orgullosos del esfuerzo y la dedicación que he depositado en este trabajo, se lo dedico especialmente a mi padre, al que sigo echando muchísimo de menos después de tantos años y al que siento a mi lado especialmente en los momentos difíciles, y a mi abuelo, por el que lamento profundamente no haber logrado acabar este trabajo unos meses antes para compartirlo con él después de tantos años ocupando un espacio muy importante en mi vida.

Por otra parte, debo destacar la figura de mi madre, que además de representar ese papel crucial a la perfección, siempre apoyó incondicionalmente mi afición por la informática, por lo que, al igual que por tantas otras razones, merece esta mención especial. Por último, me gustaría poder dedicar unas líneas llenas de cariño a cada uno de mis hermanos, abuelas, tíos, primos y amigos, pero espero que entiendan que reserve esa explicitud para las dos personas que me llenan de orgullo con sólo cruzarse en mis pensamientos. A mi hija Nora, que desde su nacimiento va salpicando mi vida cotidiana de momentos de felicidad, ahondando cada vez más en mis ansias por darle lo mejor de mí para que disfrute de una vida plena y feliz. Y por encima de todo, a Lourdes, que sufre y disfruta conmigo todo cuanto acontece en mi vida, razón por la cuál nunca dejaré de sentirme un hombre afortunado.

Introducción

La idea principal que se desarrollará a través de los diferentes capítulos que componen este documento es la siguiente:

> La aplicación de técnicas de combinación en la resolución de tareas del PLN constituye una oportunidad para mejorar los resultados, tanto en situaciones obvias en las que contamos con varios clasificadores, como en otros escenarios menos evidentes a priori. En este sentido, además de analizar las distintas alternativas de combinación, creemos que es igualmente importante explorar distintas formas de "generar variabilidad" para obtener el máximo partido de la aplicación de la combinación.

En el Procesamiento del Lenguaje Natural (PLN) existen muchas líneas de investigación basadas en la clasificación de palabras, construcciones gramaticales o documentos de texto en un cierto número de categorías predefinidas. En los últimos años han sido muchos los algoritmos de clasificación que han sido desarrollados mediante técnicas de aprendizaje automático, basándose en múltiples teorías y enfoques distintos. Estos rasgos diferenciadores les otorgan características propias y hacen que se adapten mejor a unas tareas que a otras impidiendo la aparición del algoritmo "perfecto" para cualquier problema.

El reconocimiento de patrones es un área en la que estos algoritmos han cobrado un gran protagonismo y la que la combinación de clasificadores se ha estudiado con gran interés. Además de investigar los diversos métodos de combinación que se utilizan hoy en día, se han buscado los fundamentos teóricos que demuestran las ventajas de su utilización frente a la aplicación de un único clasificador. La idea subyacente no es otra que la de obtener el máximo provecho de los diferentes puntos de vista que pueden aportar distintos clasificadores enfrentados a un mismo problema. El PLN no tardó en hacer uso de estos métodos de combinación para sus propias tareas de clasificación, creándose una sucesión de trabajos que desde finales de los noventa siguen aportando mejoras en los resultados.

No obstante, la cobertura que se ha llevado a cabo en PLN sobre el total de métodos de combinación existentes no ha estado nunca balanceada, evidenciándose una clara tendencia al uso de técnicas de votación y *stacking* frente al uso de otros métodos sobre los que apenas hay trabajos publicados. Como muestra de ello hemos llevado a cabo un estudio bibliográfico en el que se han seleccionado un gran número de trabajos de PLN que hacen uso de la combinación de clasificadores, extrayendo información sobre los clasificadores base utilizados, las técnicas de combinación aplicadas y otros muchos datos. De este estudio se desprende que mientras el tipo de clasificadores es muy variado, en efecto las técnicas de combinación muestran a la votación y el *stacking* como claros favoritos. Las razones pueden ser la simplicidad del primero y el potencial del segundo, que ha demostrado ser capaz de lograr mejoras muy significativas en diversas tareas en las que se ha aplicado.

En nuestro caso no fuimos una excepción, ya que comenzamos nuestros primeros trabajos de combinación llevando a cabo experimentos con votación y *stacking* sobre el reconocimiento de entidades con nombre (NER), en los que generamos diferentes clasificadores llevando a cabo diversas transformaciones sobre un único corpus, sin la necesidad de disponer de recursos adicionales.

Una vez que comprobamos la capacidad de los algoritmos de combinación para extraer conocimiento de la diversidad de opiniones enfrentadas, decidimos llevar a cabo un estudio comparativo de los métodos de combinación más relevantes a un nivel de profundidad que no hemos encontrado en los trabajos publicados hasta la fecha. Para ello se han recopilado corpus de diferentes tamaños, conjuntos de etiquetas o categorías, idiomas y procedencias. También se han propuesto escenarios específicos que sirven a su vez para mostrar aspectos adicionales que pueden ser tenidos en cuenta a la hora de valorar un determinado método, como su robustez o la capacidad de adaptación a variaciones en los datos o los clasificadores base utilizados.

El escenario más habitual para aplicar la combinación suele presentarse con varios clasificadores disponibles y un corpus sobre el que aplicarlos, pero no es el único escenario posible. En este trabajo se han intentado explorar vías que nos permitan crear diversidad cuando los recursos escasean, por ejemplo teniendo un único clasificador y un sólo corpus y generando la variabilidad a partir de ellos. También se han estudiado técnicas que pueden integrar múltiples fuentes de información heterogénea, de manera que corpus dedicados a otras tareas puedan colaborar en la resolución de la tarea objetivo. Y en otros casos la combinación puede ayudarnos simplemente evitándonos la búsqueda del algoritmo o los parámetros de configuración "óptimos", permitiéndonos delegar esta tarea en la propia técnica de combinación. En definitiva, la combinación aparte de ofrecer una gran capacidad para mejorar los resultados de los clasificadores que participan en el sistema, ofrece a su vez soluciones ante

situaciones de incertidumbre o escasez de recursos, convirtiéndose en una herramienta a tener en cuenta para todo aquel que se enfrente a un problema de clasificación.

El documento se estructura de la siguiente forma: en el capítulo 1 se introducirán los aspectos fundamentales de la combinación de clasificadores, en el capítulo 2 se estudiará la influencia que han tenido estos métodos en el área del Procesamiento del Lenguaje Natural, en el capítulo 3 se presentarán los trabajos derivados de la aplicación de técnicas de combinación al reconocimiento de entidades y a la generación de recursos lingüísticos mientras que en el capítulo 4 se llevará a cabo un estudio comparativo de los métodos más relevantes aplicados en diferentes escenarios. Finalmente se presentarán las conclusiones generales.

Capítulo 1

La Combinación de Clasificadores

En multitud de ocasiones nos encontramos con la tarea de asignar a un elemento una categoría de entre varias posibles. Esta tarea tan ligada al ser humano, forma parte de numerosas acciones que llevamos a cabo diariamente y que nos permiten comunicarnos con nuestro entorno. En nuestro intento por avanzar tecnológicamente hacia sistemas cada vez más inteligentes, nos encontramos frecuentemente con algún proceso de clasificación o categorización de instancias, por lo que durante años se ha dedicado mucho esfuerzo a desarrollar técnicas y métodos que den una cobertura teórica y práctica a esta problemática. La identificación de huellas dactilares, el reconocimiento óptico de caracteres o la detección de tumores en pacientes clínicos son ejemplos de tareas asociadas con el reconocimiento de patrones, disciplina que trata este tipo de problemas y que gira en torno al concepto de clasificación. La generalización de las características comunes de este tipo de problemas ha favorecido el desarrollo de un número considerable de métodos de clasificación, basados muchos de ellos en enfoques muy dispares y alcanzando niveles de precisión muy satisfactorios en la mayoría de los casos. Como es lógico, cada alternativa presenta una lista más o menos equilibrada de puntos fuertes y débiles, pero aun existiendo la posibilidad de establecer un ranking de clasificadores, ordenados en función de los resultados aportados, no es lo habitual disponer de un clasificador perfecto que alcance una precisión del 100 % para la tarea encomendada. En algunos casos no podemos ni tan siquiera disponer de esa lista ordenada que nos facilite el nombre del mejor sistema de clasificación, ya sea por variaciones en los resultados según los datos utilizados o por circunstancias propias de la tarea en cuestión, y puede que nos debamos contentar con un conjunto de métodos aplicables sin posibilidad de conocer a priori cuál de ellos deberíamos escoger. En el primer caso,

donde tenemos un candidato destacado del resto, no podemos asegurar que éste no carezca de alguna característica deseable que sí se encuentra presente en el resto de clasificadores. En el segundo caso expuesto, parece lógico pensar que sería mejor poder trabajar con todos los clasificadores, extrayendo lo mejor de cada uno, en lugar de decantarnos por uno de ellos, descartando al resto. Esta es, a grandes rasgos, la idea que motivó la aparición de una nueva línea de investigación consistente en la combinación de sistemas de clasificación. A continuación, iremos detallando poco a poco en qué consiste esta nueva metodología, comentando los aspectos más importantes y describiendo las diferentes alternativas que han ido apareciendo a lo largo de los últimos años.

1.1. La Clasificación

Veamos antes de nada algunos conceptos fundamentales sobre la tarea de clasificación.

1.1.1. Supervisión

Un problema de clasificación comienza con la recolección de los datos y la determinación del conjunto de características que poseen dichos datos, dividiéndose a partir de aquí el proceso en tres subgrupos, que son:

- Con supervisión. Son los métodos basados en la disponibilidad de una colección de datos ya clasificados, a partir de los cuales se extrae un modelo que sirve para predecir la categoría de los datos nuevos que se reciben como entrada. En este tipo de métodos, la secuencia de pasos en líneas generales será la siguiente, pudiendo iterar hasta obtener los resultados deseados:
  - Selección y extracción de características
  - Selección del tipo de modelo de clasificación
  - Entrenamiento y generación del modelo
  - Ejecución del modelo sobre datos sin clasificar
  - Evaluación de los resultados
- Sin supervisión. En estos casos no se dispone de datos clasificados previamente, por lo que los elementos se van relacionando con otros formando grupos o *clusters*, cuyo número suele ser un parámetro del algoritmo utilizado. Estos métodos llamados de *clustering*, suelen seguir a

su vez la siguiente secuencia de pasos, que al igual que en los métodos supervisados, puede iterar hasta lograr los objetivos planteados:

- Selección del método de *clustering*
- Formación de *clusters*
- Evaluación de los resultados

La evaluación de los resultados obtenidos por los métodos de *clustering* han dado lugar a múltiples trabajos de investigación, debido a la inexistencia en muchos casos de un grupo de categorías predefinido sobre los que poder clasificar los datos. Para ello se suele evaluar la semejanza entre los elementos de un mismo *cluster* (que debe ser alta) frente a la semejanza entre elementos de diferentes *clusters* (que debería ser baja), lo cual depende de la medida de similitud aplicada. La evaluación extrínseca por su parte se basa en la ejecución de una tarea, midiendo las mejoras aportadas al resultado final de la tarea completa y también hay otros métodos basados en un *gold standard*, que consiste en aplicar el algoritmo sobre unos datos sobre los que se dispone de una agrupación considerada como "ideal".

- Semi-supervisado. Se trata de una versión mixta, en donde se dispone de una cierta cantidad de datos previamente clasificados (generalmente pequeña), llamada 'semilla' y de una determinada cantidad de datos sin clasificar (habitualmente muy grande). Estos métodos han ido ganando adeptos debido a sus capacidades para aprovechar la inmensa cantidad de información existente en Internet, siendo una misión de especial interés para las tareas que no disponen aún de grandes volúmenes de datos etiquetados manualmente. El método más sencillo, llamado *self-train*, consiste en ir agregando de forma iterativa a la colección de datos de entrenamiento los nuevos ejemplos que se van clasificando, hasta generar un modelo final que sirva para clasificar los ejemplos que representan el objetivo de la tarea. Los objetivos que limitan el número de iteraciones pueden basarse en el tamaño del conjunto de datos de entrenamiento deseado, la precisión del modelo generado en la iteración actual, etc. Los pasos del proceso a grandes rasgos son:

  - Selección y extracción de características
  - Selección del tipo de modelo de clasificación
  - Entrenamiento:

1. Utilizar los datos etiquetados para entrenar y generar del modelo
2. Ejecución del modelo sobre una porción de los datos sin clasificar
3. Agregar los resultados al conjunto de entrenamiento
4. Si no se han alcanzado aún los objetivos marcados, volver al paso 1

- Ejecución del modelo de la última iteración sobre datos sin clasificar
- Evaluación de los resultados

Existen otras técnicas de aprendizaje semi-supervisado que veremos más adelante, prestándoles especial atención en la sección 3.2.2.

1.1.2. Análisis de Características

En cualquier caso, sea cual sea el tipo de método a utilizar, las características que van a ser tenidas en cuenta durante el proceso de clasificación son probablemente la parte más importante del sistema, y de ellas dependerá en gran medida el grado de éxito o fracaso alcanzado una vez completados los experimentos. Estas características dependerán evidentemente de la naturaleza de los datos empleados y pueden dividirse en los siguientes tipos o categorías (se muestran algunos ejemplos entre paréntesis):

1. Cuantitativas o numéricas
   a) Con valores continuos (medidas de longitud, presión)
   b) Con valores discretos (nº de habitantes, resultado deportivo)
2. Cualitativas o categóricas
   a) Ordinales (calificaciones académicas)
   b) Nominales (profesiones, marcas de empresas)

El análisis de características persigue tres objetivos fundamentales, que son la mejora de las capacidades predictivas, aumentar la rapidez del sistema y permitir una mayor comprensión del proceso que genera los datos. En [137] se nos presentan algunas situaciones relacionadas con las características en las que combinar varios clasificadores puede ser la mejor opción:

- Si las características son de tipos muy dispares, puede que debamos optar por usar clasificadores basados en teorías diferentes para tratar cada tipo de características.

- En el caso de estar asociadas con representaciones o aspectos físicos distintos, habría que normalizar las escalas para homogeneizar los valores aunque puede ser más ventajoso dedicar un clasificador para cada tipo de características.

- Si disponemos de un conjunto demasiado extenso de características, la opción más común es dividirlo en vectores de una dimensión menor que la original para evitar problemas habituales de precisión e implementación.

Conceptos como la utilidad, redundancia y relevancia de características han hecho que se destine una gran cantidad de esfuerzo al análisis de estos parámetros buscando corregir ciertos errores que se pueden cometer si intentamos guiarnos únicamente por el sentido común. Hallar una correlación muy alta por ejemplo puede hacernos pensar en que se trata de características redundantes cuando en realidad puede indicar un alto grado de complementariedad.

A grandes rasgos hay dos tendencias principales en el análisis de características, que son la selección y la extracción. La selección puede desglosarse en tres grupos de métodos:

- *Wrappers*: Se utilizan técnicas de aprendizaje para valorar los posibles subconjuntos de características. La búsqueda de estos subconjuntos puede recaer en métodos con *backtracking* como *branch and bound* o la búsqueda *best-first*, existiendo otras opciones con componente aleatoria como los algoritmos genéticos. También existen diversas maneras de valorar o guiar la búsqueda como la validación cruzada, métodos basados en métricas, etc.

- *Filters*: Son métodos genéricos y que gozan de una mayor velocidad de procesamiento de los datos, por lo que en muchas ocasiones se presentan como una fase preliminar o de pre-procesamiento completando una etapa previa a la clasificación. Un caso muy popular es el de *feature ranking*, representado por métodos como *RFE* (recursive feature elimination) en el que se llevan a cabo un cierto número de iteraciones, en cada cuál se ordenan las características y se van eliminando las últimas de la lista.

- *Embedded*: Se trata de métodos donde la selección está incrustada en el propio proceso de entrenamiento del clasificador. Esta característica le otorga una mayor eficiencia al no tener que volver a entrenar un modelo para cada subconjunto de características considerado.

En cuanto a la extracción, puede llevarse a cabo mediante métodos que se encargan de la construcción de nuevas características a partir de las ya existentes, así como de la reducción de la dimensión del espacio original. Suele decirse que el arte del aprendizaje automático comienza con la elaboración de las representaciones de los datos, aunque existen algunos métodos genéricos que se pueden aplicar, como pueden ser el *clustering*, *principal component analysis* (PCA), *singular value decomposition* (SVD), etc.

1.1.3. Datos

Para realizar una tarea de clasificación se requiere una colección de datos Z de tamaño $N \times n$, donde N es el número de ejemplos de que disponemos y n el número de características con las que se describen dichos ejemplos. De esta forma, cada ejemplo de nuestra base de datos es un vector n-dimensional. En el modo supervisado, tal y como vimos en la sección 1.1.1, se realiza una fase de entrenamiento para la adquisición del conocimiento a través de los datos, seguida de una fase de pruebas o de ejecución, donde se aplica ese conocimiento para clasificar nuevos ejemplos. Durante la experimentación y evaluación, se ha de gestionar el volumen total de ejemplos disponibles para cubrir las dos fases del proceso, y existen diversas formas de gestionar estos datos, que podemos resumir de la siguiente forma:

- *Resubstitution* (también llamado *R-method*): se utiliza todo el conjunto Z tanto para el entrenamiento como para las pruebas.

- *Hold-out* (o *H-method*): donde se divide Z en dos partes, empleando una para entrenar y la otra para evaluar

  - *Data shuffle*: es una variación del *hold-out* que consiste en hacer divisiones aleatorias para posteriormente calcular la media de los valores obtenidos.

- *Cross-validation* (*rotation-method*, π*-method*): se divide Z en K partes, utilizando una de ellas para la ejecución y el resto $(K-1)$ para el entrenamiento. Tras repetir el proceso para cada una de las divisiones se utiliza el promedio como estimación final.

- *Leave-one-out* (U-method): se trata de un caso particular en donde el numero de particiones K es igual a N.

- *Bootstrap*: por último esta estrategia consiste en generar un número determinado de versiones de Z mediante el remuestreo con reemplazamiento. Tras llevar a cabo la clasificación con cada conjunto de datos se obtiene la media como estimación final.

1.1.4. Evaluación de Resultados

Existen una serie de herramientas que sirven para analizar los resultados obtenidos por un clasificador, de manera que se puedan detectar rasgos característicos de su comportamiento, compararlo con otros métodos o evaluar su tasa de errores.

Un ejemplo son las matrices de confusión, donde aparecen tanto en las filas como en las columnas las clases o categorías a las que pertenecen los ejemplos clasificados, representándose en la posición a_{ij} el número de ejemplos que pertenecen a la clase c_i y han sido clasificados como pertenecientes a la clase c_j (resultado representado por el término r_j). La tasa de acierto del clasificador podría medirse sumando los elementos de la diagonal y dividiendo entre el número total de ejemplos evaluados. Tomando como ejemplo la siguiente matriz de confusión:

$$\begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \end{array} \begin{array}{c} \begin{array}{ccc} r_1 & r_2 & r_3 \end{array} \\ \begin{pmatrix} 7 & 2 & 0 \\ 1 & 10 & 1 \\ 2 & 0 & 9 \end{pmatrix} \end{array}$$

podríamos cifrar la tasa de acierto del clasificador aplicado en 26/32, que da lugar a un 81,25 %. En clasificadores binarios, los elementos de la matriz de confusión se denominan:

$$\begin{array}{c} \\ c_1 \\ c_2 \end{array} \begin{array}{c} \begin{array}{cc} r_1 & r_2 \end{array} \\ \begin{pmatrix} tp & fn \\ fp & tn \end{pmatrix} \end{array}$$

donde:

$$\begin{array}{lcl} \text{tp} & = & \textit{true positive} \\ \text{fn} & = & \textit{false negative} \\ \text{fp} & = & \textit{false positive} \\ \text{tn} & = & \textit{true negative} \end{array}$$

Si al hecho de asignar una clase c_j a un elemento de la clase c_i le asignamos un valor de 'pérdida', obtendremos una matriz de pérdida de tamaño $c \times c$, donde quedan reflejados los perjuicios sufridos por cada tipo de error que pueda cometer el clasificador. De esta forma, podríamos utilizar una matriz de pérdida como la siguiente para adjudicarle una mayor importancia (una pérdida mayor) a los falsos positivos que a los falsos negativos:

$$\begin{array}{c} \\ c_1 \\ c_2 \end{array} \begin{array}{c} \begin{array}{cc} r_1 & r_2 \end{array} \\ \begin{pmatrix} 0 & 1 \\ 1,5 & 0 \end{pmatrix} \end{array}$$

También hay que resaltar la posibilidad de que la técnica de clasificación empleada considere la opción de abstenerse. Esto ocurrirá en los casos en que no disponga de evidencias suficientes para respaldar una decisión sobre la clase que se le debe asignar a un ejemplo. Esta característica del clasificador se suele representar por una clase adicional, habitualmente denominada *refuse-to-decide*.

Es común que los métodos de evaluación giren en torno a dos conceptos definidos como:

- Precisión (*precision*): es el porcentaje de categorías asignadas por el sistema que son correctas (número de asignaciones correctas partido entre todas las asignaciones realizadas). En ocasiones se utilizan grupos de etiquetas para clasificar elementos compuestos de la base de datos. En estos casos, todo el grupo debe estar clasificado correctamente para considerar el ejemplo correctamente clasificado, distinguiéndose *precision* (tasa de acierto aplicada sobre los grupos) y *accuracy* (tasa de acierto a nivel de los elementos o etiquetas individuales). Siguiendo la nomenclatura utilizada para las matrices de confusión binarias, esta medida se define mediante la ecuación 1.1 en donde cobran una mayor relevancia los falsos positivos (fp).

$$precision = \frac{tp}{tp + fp} \tag{1.1}$$

- Cobertura (*recall*): es el porcentaje de unidades a etiquetar, presentes en la base de datos, que han sido halladas por el sistema (número de asignaciones correctas de entre el número total de asignaciones posibles). En este caso la cobertura se centra en los falsos negativos (fn) y se obtiene de la siguiente forma:

$$cobertura = \frac{tp}{tp + fn} \tag{1.2}$$

El factor F[128], relaciona la precisión y la cobertura en los resultados de la siguiente forma:

$$F_{\beta=1} = \frac{(\beta^2 + 1) \cdot precision \cdot cobertura}{\beta^2 \cdot precision + cobertura} \tag{1.3}$$

El término β suele adoptar el valor 1 para otorgarle la misma importancia a la precisión que a la cobertura, aunque en algunos casos se utilizan otros valores. Con $\beta = 2$ por ejemplo, se valora la cobertura el doble que la precisión, mientras que con $\beta = 0,5$ es al contrario. De esta forma se compensan las carencias en uno u otro sentido penalizando el exceso de exigencia para conseguir una precisión muy alta y el aumento de la cobertura a través de un etiquetado muy poco exigente en cuanto a la calidad. Si el objetivo fuese localizar los círculos en la figura 1.1, ambas situaciones quedan reflejadas en los diagramas 'a' y 'b' respectivamente. Vemos que en 'a' la exigencia ha sido muy alta por lo que estamos seguros de que los elementos seleccionados son válidos, pero por otro lado hemos dejado fuera otros muchos que también lo eran y no han sido seleccionados, provocando un aumento de la precisión y una disminución de la cobertura. En 'b' sin embargo, ocurre lo contrario, ya que al no querer dejar ningún elemento válido sin seleccionar, hemos seleccionado otros muchos que no lo son, provocando un aumento de la cobertura y disminución de la precisión.

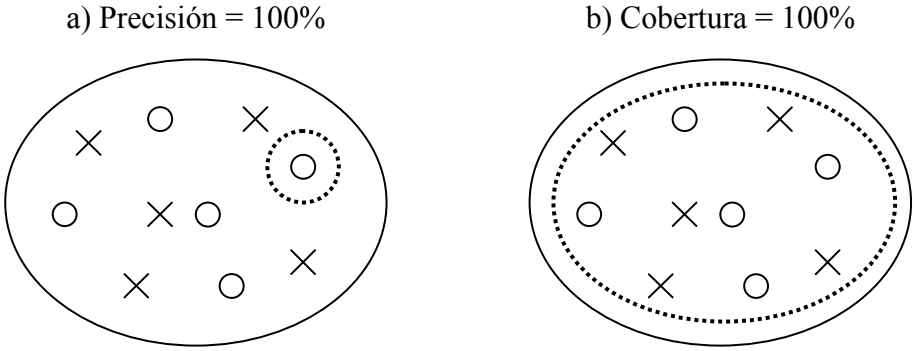

Figura 1.1: Precisión vs. cobertura

Asegurar que los resultados obtenidos con un clasificador son mejores que los de otro dista mucho de ser una tarea trivial, dado que hay múltiples

fuentes de variación que pueden alterar la percepción de calidad aportada por un determinado método. Tal y como se postula en [32], la elección de los conjuntos de datos, tanto de entrenamiento como de pruebas, la aleatoriedad que puede albergar de forma interna el algoritmo en cuestión o los errores que puedan contener los datos de entrada son factores que pueden influir en los resultados de un clasificador y que deben ser tenidos en cuenta a la hora de realizar comparaciones. No obstante, existen métodos que sirven para comparar los resultados de distintos clasificadores, como por ejemplo el llamado test de McNemar. En este test partimos de una tabla (ver tabla 1.1) donde se refleja la relación existente entre los resultados obtenidos por dos clasificadores D_1 y D_2, siendo N_{xy} el número de ejemplos que han sido clasificados por D_1 de forma correcta si $x = 1$ o errónea si $x = 0$ y que a su vez han sido clasificados por D_2 de forma correcta si $y = 1$ o errónea si $y = 0$.

	D_2 correcto	D_2 erróneo
D_1 correcto	N_{11}	N_{10}
D_1 erróneo	N_{01}	N_{00}

Tabla 1.1: Relación entre los resultados de dos clasificadores.

A partir de ahí se calcula la discrepancia entre ambos mediante la fórmula 1.4 y dado que esta función se distribuye aproximadamente como χ^2 con 1 grado de libertad, podemos realizar el test calculando x^2 y contrastándolo con el valor de χ^2 para un nivel de confianza estadística de nuestra elección. Si escogemos por ejemplo un 95 % como nivel de confianza, el valor de x^2 deberá ser mayor que 3,841 para poder considerar que ambos clasificadores tienen niveles de precisión significativamente diferentes, y en caso contrario, no podremos afirmar que uno es mejor que el otro.

$$x^2 = \frac{(|N_{01} - N_{10}| - 1)^2}{N_{01} + N_{10}} \tag{1.4}$$

Otra posibilidad reside en los diversos métodos basados en la variabilidad de los datos de entrenamiento y prueba utilizados. En las técnicas de validación cruzada se realizan diversas particiones para comparar los resultados de los clasificadores enfrentados en varias iteraciones. Las críticas a estos métodos recaen en la reutilización de los datos de entrenamiento y/o de prueba, lo que entra en conflicto con la asunción de independencia del muestreo realizado, aunque no cabe duda de que siempre es mejor basar una

hipotética mejora en la clasificación en alguno de estos métodos que en una única experimentación sobre un escenario particular.

También queremos mencionar un método de evaluación comparativa presentado en [2], llamado *procedimiento de selección multinomial*, en donde se trata de identificar el clasificador que con mayor frecuencia ha acertado con la clase correcta.Analizando un conjunto de datos Z de tamaño N y suponiendo que los clasificadores $D_1, \ldots, D_L$ proporcionan las probabilidades para cada clase posible, el algoritmo sería el siguiente:

1. Para $i = 1, \ldots, c$,

   a) Usar los N_i ejemplos cuya etiqueta correcta sea ω_i e inicializar una matriz T de tamaño $N_i \times L$.

   b) Para cada uno de los ejemplos z_j cuya etiqueta es ω_i, hallar las estimaciones de la probabilidad $P(\omega_i|z_j)$ obtenidas por cada clasificador. En la tabla T, actualizaremos con un 1 la posición $T(j, q)$ donde D_q es el clasificador que ha obtenido un valor mayor para el cálculo de la probabilidad, y añadiremos un 0 en las columnas correspondientes al resto de clasificadores.

   c) Sumando las columnas de T y dividiendo por el número de ejemplos de que disponemos, obtendremos una estimación de la probabilidad de que el clasificador D_k sea el *ganador* para la clase ω_i:

$$\hat{P}(D_k \text{ gana } |\omega_i) = \frac{1}{N_i} \sum_{j=1}^{N_i} T(j, k) \tag{1.5}$$

2. Calculamos el rendimiento global del clasificador D_k sumando las probabilidades acumuladas para cada clase, multiplicadas por la probabilidad de cada clase respectivamente:

$$\hat{P}(D_k \text{ gana}) = \sum_{i=1}^{c} \hat{P}(D_k \text{ gana } |\omega_i)\hat{P}(\omega_i) \tag{1.6}$$

   Y si utilizamos los fragmentos N_i en lugar de Z, la medida global se calcula de la siguiente manera:

$$\hat{P}(D_k \text{ gana}) = \frac{1}{N} \sum_{i=1}^{c} N_i \hat{P}(D_k \text{ gana } |\omega_i) \tag{1.7}$$

Por último, hay trabajos que evitan el uso de la precisión como medida de bondad y comparación entre clasificadores, recurriendo a otras técnicas como las gráficas *receiver operating characteristic* (ROC) en el caso de clasificadores binarios en los que se clasifican un total de P ejemplos positivos y N negativos. Se basan en las tasas de '*true positive*' (tp) y '*false positive*' (fp), cuyos valores son tp/P y fp/N respectivamente. En la figura 1.2 vemos un ejemplo sencillo en el que C_1 sería el mejor clasificador ya que clasifica muchos ejemplos positivos correctamente y en pocas ocasiones clasifica un ejemplo negativo como positivo (al contrario que C_3). Por su parte C_4 sería un clasificador "conservador" al cometer pocos errores pero dejando muchos ejemplos positivos sin ser detectados como tales y C_2 sería de los considerados "liberales", al clasificar muchos ejemplos positivos correctamente pero también muchos negativos como positivos.

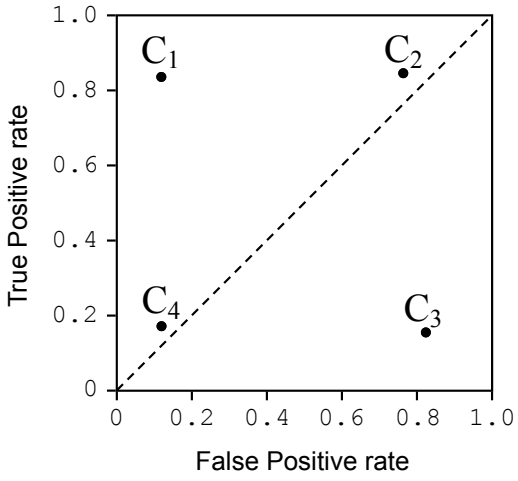

Figura 1.2: Gráficos ROC.

1.1.5. Tipos de Clasificadores

El gran número de métodos existentes para el diseño de un clasificador hace posible su catalogación desde diferentes puntos de vista y mediante diversos criterios, de ahí que no exista consenso entre los investigadores sobre la mejor forma de completar esta taxonomía. En la figura 1.3 mostramos como ejemplo la que ofrece Kuncheva [69], en la que se distinguen principalmente los métodos basados en la aproximación de las funciones de probabilidad o densidad frente a los métodos basados en la aproximación de las fronteras de clasificación o funciones discriminantes.

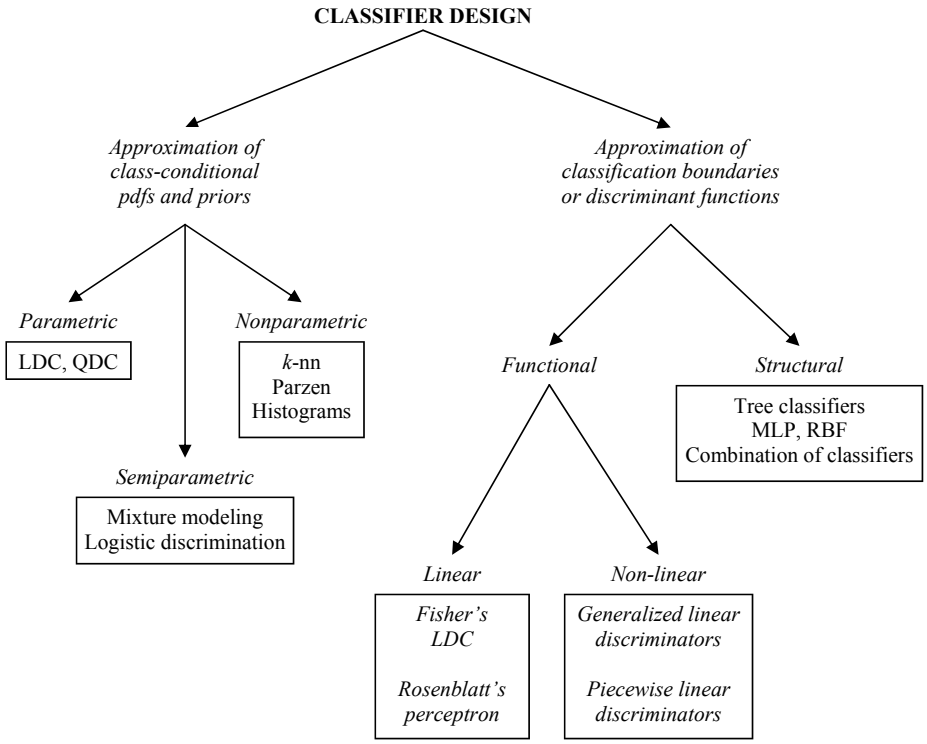

Figura 1.3: Una taxonomía de métodos para el diseño de clasificadores.

Desde otro punto de vista, Lippmann [75] agrupa todos los clasificadores en cinco tipos posibles:

- probabilísticos (LDC, QDC, Parzen)
- globales (MLP)
- locales (RBF)
- del tipo vecinos-más-cercanos (k-nn, LVQ)
- de formación de reglas (árboles de decisión binarios, sistemas basados en reglas)

1.2. Requisitos y Justificación

¿Qué se necesita para tener un mínimo de garantías a la hora de combinar varios clasificadores? ¿Qué puede aportarnos la combinación de clasificadores? ¿Qué razones tenemos para creer que la combinación mejorará los resultados de un único clasificador? Dietterich [33] ofrece respuestas a estas preguntas básicas de las que depende el desarrollo de las técnicas de combinación de clasificadores y su puesta en práctica.

Siguiendo los estudios de Hansen y Salamon [54], existen dos requisitos básicos necesarios y suficientes para que la combinación en general consiga mejorar a los clasificadores individuales que forman parte del sistema, independientemente de que una determinada técnica consiga explotar el potencial que teóricamente tenemos a nuestro alcance:

- Diversidad: Consiste en que los clasificadores que componen el sistema cometan errores diferentes cuando se enfrentan a nuevos datos para clasificar. Se trata de que aporten diferentes puntos de vista del mismo problema, lo cuál se consigue de muy diversas maneras, entre las que se incluyen el uso de diferentes subconjuntos de características por parte de cada clasificador o la implementación de diferentes métodos de aprendizaje que sigan enfoques dispares. De aquí surgen importantes medidas de correlación, diversidad y redundancia, que a menudo son objeto de estudio en busca de la optimización del proceso de combinación aplicado. En [68] se recogen diez medidas estadísticas de la diversidad, además de un estudio interesante sobre su influencia en los resultados, concluyendo con algunas dudas por parte de la autora sobre su aplicabilidad en problemas reales.

- Precisión: Los clasificadores implicados deben disfrutar de una tasa de error inferior a la que proporciona un clasificador aleatorio. En caso contrario sólo conseguiremos reducir la precisión final al no disponer de una base de conocimiento de la que podamos extraer beneficios a través de la combinación.

A la hora de justificar la confianza depositada en la combinación como método capaz de mejorar la clasificación, Dietterich sugiere tres razones, las cuales pasamos a comentar a continuación, aunque no sin antes mencionar el trabajo de Freund et al. [47], en el que se agrega la capacidad de evitar o mitigar los efectos del *overfitting* a través de la combinación.

1. Estadística:

   Desde el punto de vista estadístico, escoger un clasificador para resolver nuestro problema entraña un riesgo, ya que éste puede no ser el más capacitado para el caso particular que estamos tratando. Debido a esto, puede que la combinación no nos proporcione mejores resultados que el mejor clasificador individual de que disponemos, pero sí que elimina o mitiga el riesgo de equivocarnos en la selección. No consiste únicamente en que un clasificador sea peor que otro, ya que si contamos con pocos datos para la fase de entrenamiento, puede que incluso contando con clasificadores teóricamente capaces de acercarse a la misma distancia del objetivo, unos sufran más que otros la falta de datos generando peores resultados. En la figura 1.4 publicada por Dietterich, el clasificador que constituye nuestro objetivo viene representado por un punto rotulado con una 'f', las hipótesis de los clasificadores de que disponemos están representados por puntos rotulados con 'hi' y rodeándolos vemos una línea que delimita el espacio de búsqueda de nuestros clasificadores respecto al espacio de hipótesis completo rotulado con una 'H'. La esperanza reside en que la combinación de esas hipótesis nos acerque al objetivo deseado, eliminando el riesgo de seleccionar la que está más alejada de 'f'.

2. Computacional:

   Por otro lado, aún teniendo un volumen de datos que haga desaparecer el problema estadístico mencionado anteriormente, existe el problema de los máximos locales, en los que pueden incurrir muchos algoritmos en su búsqueda del objetivo. Una vez más, la combinación de los clasificadores puede realizar la búsqueda partiendo de muchos puntos diferentes del espacio de hipótesis y acercarse más a la solución ideal de lo que consiguen acercarse estos clasificadores individualmente (ver figura 1.5).

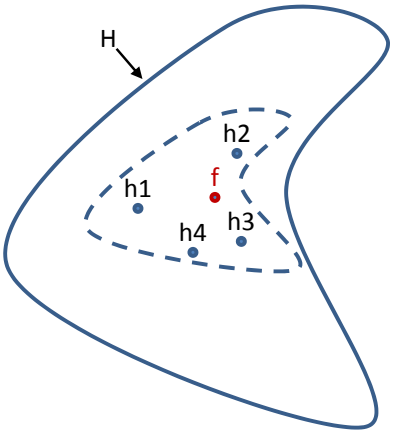

Figura 1.4: Primera justificación para la combinación (estadística).

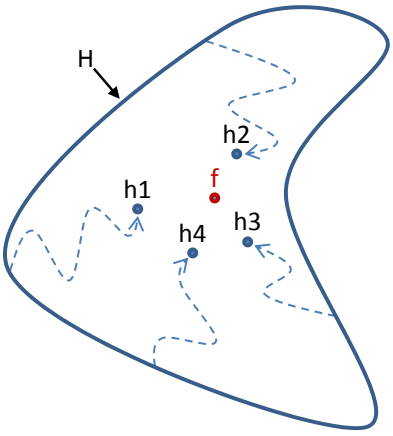

Figura 1.5: Segunda justificación para la combinación (computacional).

3. Representación:

   Finalmente, como tercera razón que justifica la combinación de clasificadores, hay que considerar la posibilidad de que el espacio de búsqueda de los clasificadores no contenga la solución que estamos buscando (ver figura 1.6), mientras que la combinación puede darnos la oportunidad de expandir dicho espacio con la posibilidad que esto acarrea de que consigamos incluir el objetivo en dicho espacio y tengamos opciones de aproximarnos más a él. Aún así, tal y como apunta Dietterich, hay que matizar que existen algoritmos muy flexibles, como las redes neuronales y los árboles de decisión, que pueden explorar el espacio de todas las hipótesis posibles, aunque dado que dispondremos de un conjunto de datos finito, podemos considerar 'H' como el espacio de búsqueda para un determinado conjunto de datos.

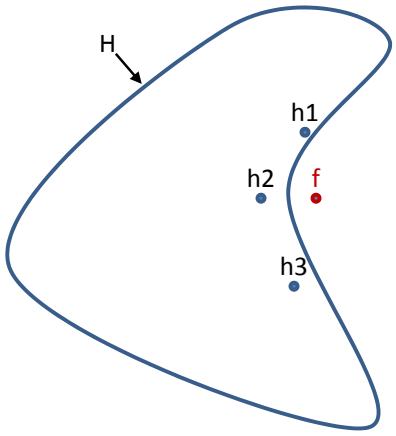

Figura 1.6: Tercera justificación para la combinación (de representación).

1.3. Tipología

Existen diversas maneras de combinar diversos clasificadores, así como múltiples formas de agrupar las técnicas existentes. En esta sección vamos a comentar las más utilizadas actualmente, indicando en qué se basan para distinguir los métodos que entran dentro de cada categoría propuesta. Uno de los conceptos clave relacionados con la combinación de sistemas es la 'diversidad', ya que el éxito de los métodos de combinación dependen en gran medida de la existencia de diferentes puntos de vista del problema. La forma más intuitiva de catalogar los métodos se basa por tanto en la forma en que

intentan obtener la diversidad necesaria, pudiendo optar por crear diferentes particiones de los conjuntos de datos (*bagging*), del conjunto de características (*random subspace method*), del conjunto de clases (*ECOC*), del propio conjunto de clasificadores base (*evolutionary algorithms*) o inyectando aleatoriedad en el proceso, pudiendo centrarse en cualquier fase del mismo. Esto nos lleva a la primera aproximación, basada en la fase donde recae el mayor peso de la labor de combinación.

1.3.1. Enfoques

La combinación se puede llevar a cabo en diferentes puntos del proceso de clasificación y siguiendo diversas estrategias con el fin de aunar los esfuerzos individuales para mejorar el resultado global. Kuncheva [69] lo recoge en la figura 1.7, donde muestra cuatro tipos diferentes de combinación, según el nivel de procesado al que se produce, y aunque en capítulos posteriores hablaremos de algunos métodos que no encajan del todo bien en este esquema (como puede ser el *Error Correcting Output Codes* o *ECOC*), la gran mayoría de técnicas actuales recaen en alguno de estos niveles.

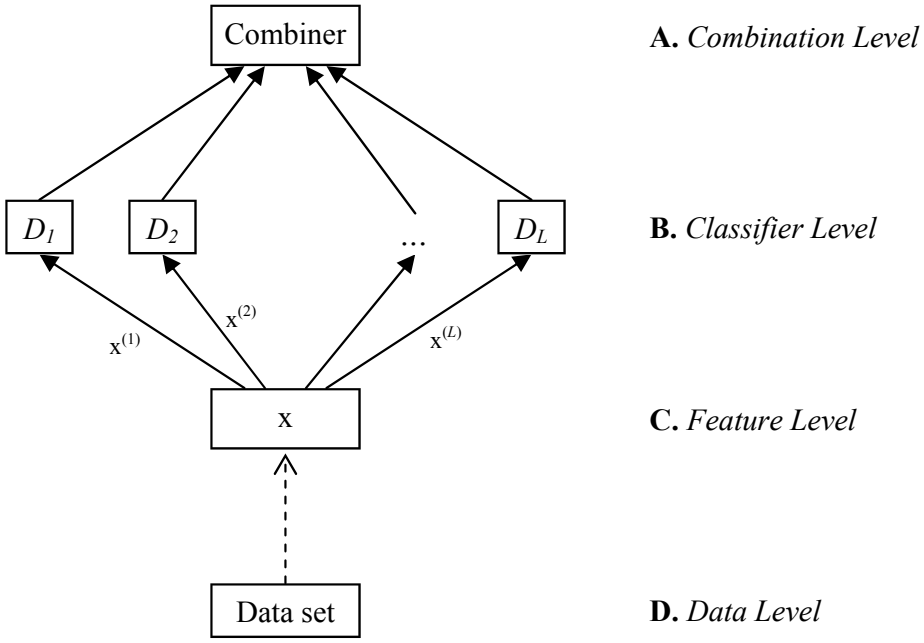

Figura 1.7: Enfoques para la combinación de clasificadores.

Combination Level

A este nivel, la decisión a tomar consiste en escoger el algoritmo que vamos a emplear para llevar a cabo la combinación de todas las opiniones aportadas por cada uno de los clasificadores implicados. Podemos utilizar métodos de votación, aplicar nuevos clasificadores sobre estas opiniones, emplear algoritmos genéticos, etc.

Classifier Level

Si nos centramos en los clasificadores que forman parte del sistema, existe una gran variedad de clasificadores que pueden ser utilizados y combinados. Podemos variar el número de algoritmos, su tipo, versión o configuración utilizada, buscando siempre la mayor variabilidad posible en las respuestas, de manera que la combinación tenga un mayor margen de mejora.

Feature Level

El conjunto de características también influye notablemente en la generación de un modelo de clasificación, de ahí que se puedan utilizar diferentes subconjuntos como entrada para los clasificadores, obteniéndose de esta forma la variabilidad necesaria para llevar a cabo la combinación.

Data Level

Por último, otra manera de conseguir diferentes visiones del problema de clasificación que queremos tratar, es utilizando diferentes conjuntos de datos para cada uno de los clasificadores base. Esta posibilidad está claramente representada por los algoritmos de *bagging* y *boosting* que comentaremos más adelante.

1.3.2. Fusión y Selección

Entre tantos intentos de clasificar los métodos de combinación existentes, ha tomado fuerza en los últimos años la tendencia a dividirlos en dos grandes grupos, donde residen por un lado los basados en la fusión de opiniones, y por otro lado los basados en la selección de opiniones. Los primeros dan por buena la hipótesis de que todos los clasificadores involucrados en el sistema cubren todo el espacio de características o poseen un conocimiento que lo abarca en su totalidad. En el segundo grupo sin embargo, la hipótesis de partida es la opuesta, considerándose que hay clasificadores que poseen un conocimiento mayor que otros sobre una determinada región, convirtiéndose

en el encargado de etiquetar los elementos que recaen dentro de sus límites de actuación. Los procedimientos irán encaminados a delimitar dichas regiones para determinar los clasificadores que deben darnos la clase resultante en cada caso. Una vez más, hay métodos que no encajan a la perfección en este esquema, como pueden ser los métodos en cascada, en los que la salida de un clasificador sirve como entrada a otro clasificador. En estos métodos se suele aplicar un umbral que decide si la respuesta de un clasificador es suficientemente fiable o si debe seguir su camino y actuar como entrada del siguiente clasificador.

1.3.3. Dependencia de los Datos

También podemos basarnos en los requisitos de entrenamiento de los diferentes métodos, ya que los hay que deben procesar datos previamente clasificados para poder combinar los resultados de varios clasificadores, mientras que otros no dependen de esta fase preliminar. Un claro ejemplo es el algoritmo de votación por mayoría, que no requiere entrenamiento alguno, frente al algoritmo de votación ponderada, que debe asignar un valor de influencia a cada clasificador, para lo cuál requiere una fase de entrenamiento o preprocesado de los datos. También suelen referirse en la bibliografía a estos dos tipos de combinación, como métodos dependientes de los datos para los casos en que se requiere entrenamiento, y métodos independientes de los datos para los casos que no requieren entrenamiento. Dentro del primer tipo, también cabe destacar la existencia de métodos implícitamente dependientes, donde los parámetros de la técnica de combinación se entrenan previamente a su aplicación para una entrada x, y de métodos explícitamente dependientes, donde los parámetros dependen directamente de x (se usan pesos que son funciones sobre x).

1.4. Clases y Salidas

Cuando ejecutamos un clasificador obtenemos como salida la clase (o etiqueta asociada a la clase) que se le asigna a cada elemento que se le introduce como entrada a dicho clasificador. Dependiendo del método seleccionado, de la implementación desarrollada o de los parámetros utilizados, la salida del clasificador puede clasificarse en tres tipos según Xu[137], que son:

1. *Abstract level*: Es el nivel que alcanzan los clasificadores que ofrecen como salida una única etiqueta o a lo sumo un subconjunto del conjunto de etiquetas posibles.

2. *Rank level*: Es el nivel en el cuál el clasificador devuelve todas las etiquetas, o al menos un subconjunto de ellas, ordenadas según el orden de preferencia, siendo la primera de la lista la que debe considerarse la primera opción.

3. *Measurement level*: Es el nivel que alcanzan los clasificadores que atribuyen a cada etiqueta o clase un valor que indica hasta qué punto el clasificador considera que el elemento pertenece a dicha clase.

Como puede apreciarse, el nivel de información que nos aporta cada nivel es diferente, de manera que el *abstract level* aporta la menor cantidad de información, mientras que el *measurement level* es el que aporta una mayor cantidad de información. Estos tres tipos de salidas que pueden proporcionar los clasificadores deben ser tenidos en cuenta a la hora de seleccionar el método de combinación o los clasificadores base, ya que pueden ser un requisito indispensable para poder emplear una determinada técnica de combinación.

A estos tres niveles, Kuncheva [69] añade uno adicional, que consiste en el *oracle level*, cuya salida indica únicamente si el clasificador ha etiquetado correctamente el elemento o no. Se trata de un nivel de clasificación artificial, tal y como comenta la autora, ya que sólo es posible su aplicación sobre una colección de datos ya clasificados. Este tipo de información es especialmente útil para la aplicación de la técnica de *stacking*, como veremos más adelante.

1.5. Métodos de Combinación

A lo largo de los años han surgido multitud de métodos de clasificación destinados a dar respuesta a los desafíos propuestos por un gran número de tareas. Basándose en ocasiones en teorías muy dispares, los resultados han hecho que se desarrollen numerosas variantes de los métodos más exitosos, aumentando el número de opciones disponibles actualmente. En esta sección vamos a enumerar los métodos que a nuestro juicio han resultado ser más relevantes, destacando sus aspectos más importantes.

Siguiendo la notación utilizada en trabajos anteriores, hablaremos de métodos que combinan L clasificadores $D_1 \dots D_L$ que asignan a cada ejemplo una etiqueta ω (ω_k si nos referimos al clasificador D_k) de entre c etiquetas posibles. La base de datos completa Z se compondrá de N ejemplos y en ocasiones utilizaremos $\mu_k(x)$ para hacer referencia al soporte recibido por la etiqueta k para ser seleccionada por el método como la etiqueta propuesta para el ejemplo x.

1.5.1. Teoría de Elección Social

Probablemente el método de combinación de clasificadores más utilizado sea la votación, en alguna de sus diferentes versiones. La posibilidad de asemejar el concepto de 'elecciones' al problema de clasificación, el de 'votantes' a los clasificadores implicados y el de 'votos' a las etiquetas propuestas por cada uno de ellos, permite aplicar los métodos de elección contempladas en la Teoría de elección social [3], cuyo objetivo es el estudio de la toma de decisiones colectivas en función de las preferencias individuales.

Votación por Mayoría

El voto por mayoría es la versión más conocida de todos los sistemas de votación y consiste en decretar vencedor a aquel que consiga superar en número de votos a todos sus rivales. Aún así, existen varios criterios que podemos adoptar para poder determinar cuál de los candidatos es el vencedor, entre los que destacamos:

- Unanimidad: se aplica cuando todos los votantes deben coincidir en su elección del candidato vencedor.
- Mayoría simple: se trata del caso en el que un candidato debe superar el 50 % de los votos para proclamarse vencedor.
- Mayoría plural: opción por la que el candidato que obtiene un mayor número de votos es el vencedor, sin necesidad de alcanzar un número mínimo de votos a su favor.

Generalmente, es esta última versión la que se conoce como sistema de votación por mayoría, y si consideramos un sistema con c clases $\omega_1 \dots \omega_c$ y L clasificadores $D_1 \dots D_L$ que generan la matriz D, con sus elementos $d_{i,j} = 1$ si D_i clasifica x en la categoría ω_j y $d_{i,j} = 0$ en caso contrario, la decisión final será ω_k si:

$$\sum_{i=1}^{L} d_{i,k} = max_{j=1}^{c} \sum_{i=1}^{L} d_{i,j} \tag{1.8}$$

La aplicación de este sistema de selección a la combinación de clasificadores no requiere entrenamiento previo, ya que se trata de un sistema independiente de los datos y podemos emplearlo sea cual sea el tipo de clasificador que tengamos incorporado al sistema. Se trata por tanto de un método rápido y sencillo que se ha mostrado muy eficaz en múltiples aplicaciones. Aún así, en la práctica parece deseable distinguir entre los diferentes clasificadores,

dado que suelen proporcionar diferentes niveles de precisión para una misma tarea lo que les atribuye una mayor o menor importancia a la hora de opinar sobre cuál debe ser la categoría final de los datos a clasificar. Esto nos lleva a la votación ponderada que comentamos a continuación.

Votación Ponderada

La votación ponderada consiste en atribuir una serie de pesos a los clasificadores, de manera que su aportación a la decisión final se vea reforzada o disminuida según su precisión, medida generalmente sobre los datos de entrenamiento. El primer paso por tanto será atribuir los pesos a los clasificadores de forma subjetiva o ejecutando los clasificadores sobre un fragmento de los datos de entrenamiento de que disponemos, obteniendo los resultados y contrastándolos con las etiquetas conocidas para obtener un valor de precisión de cada clasificador para la tarea que estamos llevando a cabo. Estos valores darán lugar a los pesos, que se incorporarán a la función 1.8 que determina si la clase w_k debe ser escogida para clasificar el elemento actual, en función de si se cumple:

$$\sum_{i=1}^{L} b_i d_{i,k} = max_{j=1}^{c} \sum_{i=1}^{L} b_i d_{i,j} \tag{1.9}$$

donde b_i representa el peso asociado al clasificador D_i, siendo habitual recurrir a la normalización de manera que:

$$\sum_{i=1}^{L} b_i = 1 \tag{1.10}$$

Métodos Mayoritarios y Métodos Posicionales

Dentro de los métodos de votación existe un gran número de variantes en las que cada votante le otorga una puntuación a cada candidato o a un subconjunto de ellos en lugar de escoger sólo uno, estableciéndose dicha puntuación de muy diversas maneras. En función de esto podemos dividir los métodos en dos grupos, que son los métodos mayoritarios, en los cuales se realizan comparaciones por pares, y los métodos posicionales, en donde la valoración de los candidatos se cuantifica en base a su posición relativa al resto. Los métodos *condorcet* y *borda count* son los métodos más representativos de estos dos grupos respectivamente.

- *Condorcet*: El sistema consiste en que cada votante ordena los candidatos por orden de preferencia, realizándose comparaciones por pares

de cada candidato con el resto, sumándole un punto a aquel que se encuentre mejor situado en la lista. El ganador será aquel que siendo comparado con cada uno de sus rivales haya conseguido sumar más puntos en los duelos directos, es decir, que consiga siempre más victorias que derrotas frente a cada uno de sus oponentes. El método *condorcet* tiene la particularidad de que puede ser elegido un candidato que no haya sido considerado como primera opción por ningún votante. En algunos casos pueden producirse empates, existiendo diversas posibilidades para resolver dicha situación.

- *Borda count*: Este método es el empleado en muchas competiciones deportivas en donde varios participantes compiten en diversas pruebas independientes. En cada una de estas pruebas se ordenan los candidatos según sus resultados y se le otorga al mejor posicionado la máxima puntuación, mientras el resto ve disminuida su puntuación paulatinamente según vamos bajando en la lista. El ganador será el que reúna el mayor número de puntos una vez finalizadas todas las pruebas, las cuales representan los votos en este caso, existiendo la posibilidad de que no resulte vencedor el que haya ganado más pruebas al ser consideradas también el resto de posiciones.

1.5.2. Naive-Bayes

El uso de la regla de Bayes o Teorema de Bayes como método de clasificación ha dado lugar a resultados muy positivos en diversas aplicaciones, máxime cuando se comparan con los alcanzados por métodos mucho más complejos [46]. Esta regla permite obtener la probabilidad de un evento A dado B en términos de la probabilidad del evento B dado A y la probabilidad de sólo A. De esta forma, si tenemos una serie de eventos $A_i \dots A_n$ independientes entre sí con probabilidad distinta de cero y un suceso B, se define la probabilidad a posteriori $P(B|A_i)$ de la siguiente forma:

$$P(B|A_i) = \frac{P(B)P(A_i|B)}{P(A_i)} \tag{1.11}$$

Aplicando este teorema a la combinación de clasificadores podemos obtener un sistema sencillo, que aún con la suposición poco realista sobre la independencia de los atributos, puede obtener resultados que alcancen los objetivos establecidos a priori. En este caso, la probabilidad a posteriori que debemos calcular será la probabilidad de que dadas c etiquetas posibles y dada una serie de observaciones $s = s_1, \dots, s_L$ que representa las etiquetas propuestas por los L clasificadores, la etiqueta real sea ω_k, con $k = 1, \dots, c$:

$$P(\omega_k|s) = \frac{P(\omega_k)P(s|\omega_k)}{P(s)} = \frac{P(\omega_k)\prod_{i=1}^{L} P(s_i|\omega_k)}{P(s)} \tag{1.12}$$

Dado que el denominador no depende de ω_k, podemos eliminarlo quedándonos la posibilidad de llegar a la probabilidad de que el elemento x deba ser etiquetado con ω_k de la siguiente forma:

$$\mu_k(x) \propto P(\omega_k)\prod_{i=1}^{L} P(s_i|\omega_k) \tag{1.13}$$

En la práctica, para una colección de datos Z de tamaño N, se procede calculando la matriz de confusión CM^i de tamaño $c \times c$ para cada clasificador D_i. Cada elemento $cm^i_{k,s}$ de la matriz de confusión representa el número de ejemplos del corpus cuya etiqueta real es ω_k habiendo sido etiquetado por el clasificador D_i con ω_s, mientras que N_s es el número total de elementos de Z cuya etiqueta es ω_s. Estimando la probabilidad $P(s_i|\omega_k)$ como $cm^i_{k,s_i}/N_k$ y la probabilidad a priori de la clase ω_k como N_k/N, podemos calcular la ecuación 1.13 mediante:

$$\mu_k(x) \propto \frac{1}{N_k^{L-1}}\prod_{i=1}^{L} cm^i_{k,s_i} \tag{1.14}$$

1.5.3. Métodos Basados en la Memorización

En los trabajos relacionados con el desarrollo de métodos de combinación de clasificadores podemos encontrar diversos algoritmos que comparten una idea fundamental, que consiste en memorizar el comportamiento demostrado por los clasificadores en los ejemplos de entrenamiento, para luego tomar decisiones sobre los nuevos elementos basándose en el pasado. Entre ellos destacamos aquí el *behavior knowledge space*, que se aplica cuando las salidas de los clasificadores se enmarcan en el *abstract level* según [137] (nivel que alcanzan los clasificadores que ofrecen como salida una única etiqueta o a lo sumo un subconjunto de todas las etiquetas posibles), y el *decision template*, que se puede aplicar cuando las salidas son valores continuos según indica el *measurement level* (cuando los clasificadores atribuyen a cada etiqueta o clase un valor que indica hasta qué punto el clasificador considera que el elemento pertenece a dicha clase).

Behavior Knowledge Space

El *behavior knowledge space* (BKS) [58], perteneciente a la familia de métodos multinomiales, se basa en estimar la probabilidad $P(\omega_k|s)$ para todo $k = 1, \ldots, c$ y para toda combinación de votos posible $s \in \Omega^L$. En la práctica, se ejecutan los clasificadores sobre todos los elementos z_j de la base de datos de entrenamiento Z, generando el vector $s = [s_1, \ldots, s_L]$ para cada uno de ellos. Dichos vectores se almacenan en una tabla indexada junto a las frecuencias en que dicho vector dio lugar a cada una de las etiquetas posibles. Para clasificar un nuevo elemento, se halla el vector s correspondiente a ese elemento y se accede a la tabla indexada, recuperándose la etiqueta más utilizada entre los elementos que ocasionaron el vector residente en dicha posición. Tanto los empates como las celdas vacías suelen resolverse arbitrariamente mediante la elección de una clase al azar, o en el caso de celdas vacías se puede recurrir a una votación para seleccionar la etiqueta definitiva.

Cada entrada $BKS(s)$ de la tabla indexada estará compuesta por 3 datos:

1. $n(s)(k)$: Veces que esa combinación de resultados s obtenidos por los clasificadores va asociada a la clase k.

2. $S(s)$: Número total de ocasiones en que ocurre la combinación de resultados s en la base de datos.

3. R_s: Clase más representativa de la celda asociada a la combinación s.

Partiendo de estos datos se calcula el valor de confianza en cada clase:

$$Belief(\omega_k) = \frac{n(s)(k)}{S(s)} \tag{1.15}$$

Finalmente, el resultado será R_s siempre que $S(s) > 0$ y $Belief(R_s) \geq \alpha$. En caso contrario, el resultado debería ser la abstención ya que al no superar el umbral predefinido α, no se tiene suficiente confianza en la respuesta más probable.

Decision Template

En este método se genera una plantilla (*decision template* o $DT(j)$) para cada clase ω_j, que consiste en una matriz cuyos elementos representan las probabilidades "medias" que fueron asignadas durante el entrenamiento por los diferentes clasificadores a las distintas clases para los ejemplos pertenecientes a la clase ω_j. Posteriormente, se halla la matriz de probabilidades para un nuevo ejemplo y buscamos a continuación la plantilla $DT(j)$ que minimiza la

distancia con ella, siendo la etiqueta ω_j asociada a dicha plantilla la seleccionada. El proceso comenzará por lo tanto con una fase de entrenamiento en la que se generarán las plantillas $DT(j)$ para cada clase ω_j y continuará con la fase de ejecución encargada de etiquetar los nuevos ejemplos:

1. Entrenamiento:

   Para cada ejemplo z_k del conjunto de datos de entrenamiento Z, formamos una matriz $M(z_k)$, cuyos elementos $m_{ij}(z_k)$ se correspondan con la probabilidad otorgada por el clasificador D_i al hecho de que ω_j sea la etiqueta correcta para z_k, y calculamos la plantilla $DT(j)$ para cada clase ω_j mediante:

   $$DT(j) = \frac{1}{N_j} \sum_{z_k \in \omega_j} M(z_k) \tag{1.16}$$

   siendo N_j el número de ejemplos de entrenamiento que pertenecen a la clase ω_j.

2. Ejecución:

   Para un nuevo elemento x, obtenemos su matriz $M(x)$ y calculamos su distancia $\mu_j(x)$ respecto a cada plantilla $DT(j)$, asignando la clase cuyo índice nos ha proporcionado una distancia menor. Como medida de distancia tenemos múltiples posibilidades, entre las que destaca por su popularidad la distancia euclídea:

   $$\mu_j(x) = \sum_{i=1}^{L} \sum_{k=1}^{c} [d_{ik}(j) - m_{ik}(x)]^2 \tag{1.17}$$

   donde $d_{ik}(j)$ se corresponde con el elemento de $DT(j)$ que ocupa la fila i y la columna k.

1.5.4. Singular Value Decomposition

Otra de las vías para llevar a cabo la clasificación de elementos en un espacio multidimensional consiste en reducir dicho espacio creando una aproximación de la matriz original utilizando los autovalores y autovectores en un proceso de factorización. El procedimiento comienza creando una matriz donde se almacenan las etiquetas asignadas por los L clasificadores a los N ejemplos de la base de datos de entrenamiento Z. Dicha matriz se obtiene de la siguiente forma:

- Cada clasificador genera un vector de dimensión c, donde cada posición se corresponde con una de las posibles etiquetas $\omega_1, \ldots, \omega_c$. En dicho vector se almacenará un cero para todos los casos salvo para la posición que se corresponda con la etiqueta propuesta por el clasificador, que contendrá un uno. De esta forma, si $c = 3$ y el clasificador devuelve la etiqueta ω_2, generará el vector $s = [0, 1, 0]$.

- Los vectores generados por todos los clasificadores para un mismo ejemplo se concatenan formando un vector de dimensión $L \cdot c$ y repitiendo este proceso para los N ejemplos de Z (respetando el orden de ejecución de los clasificadores), obtenemos la matriz de resultados M, cuyos elementos normalizaremos dividiendo por la suma de todos los elementos de la matriz, es decir, multiplicando por $1/(N \cdot L)$.

Si ahora sumamos todos los elementos de cada columna en M, obtenemos un vector $x = [x_1, \ldots, x_{L \cdot c}]$, donde $x_k = \sum_{i=1}^{N} m_{ij}$ para $k = 1, \ldots, L \cdot c$. Haciendo lo propio con las filas, obtenemos un vector y en el que se repite N veces el valor $1/N$. Usando estos dos vectores x e y, obtenemos las matrices diagonales M_x y M_y:

$$M_x = \begin{pmatrix} \frac{1}{\sqrt{x_1}} & 0 & \ldots & 0 \\ 0 & \frac{1}{\sqrt{x_2}} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \frac{1}{\sqrt{x_{L \cdot c}}} \end{pmatrix}_{((L \cdot c) \times (L \cdot c))} \tag{1.18}$$

$$M_y = \begin{pmatrix} \frac{1}{\sqrt{N}} & 0 & \ldots & 0 \\ 0 & \frac{1}{\sqrt{N}} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \frac{1}{\sqrt{N}} \end{pmatrix}_{(N \times N)} \tag{1.19}$$

Esto nos permite hallar la matriz residual $A = M_x(M - x^T y)M_y$ a la que aplicamos el método SVD para obtener la descomposición $A = U\Gamma V^T$. Una vez completado el proceso de factorización, obtenemos las coordenadas principales de las N filas y las $L \cdot c$ columnas en forma de las matrices F y G respectivamente:

$$F = M_x \cdot U \cdot \Gamma \tag{1.20}$$

$$G = M_y \cdot V \cdot \Gamma^T \tag{1.21}$$

Para concluir, se aplica la fase de clasificación, que comienza por decidir cuál será la nueva dimensión δ con la que trabajaremos. Este valor será igual o inferior al número de valores singulares positivos de A, que a su vez será menor o igual al mínimo entre N y $L \cdot c$. Una vez decidido dicho valor, procedemos de la siguiente manera para clasificar un nuevo elemento:

1. Fase de entrenamiento:

   - Se entrena un clasificador con la matriz F como entrada, aunque utilizando sólo las δ primeras columnas, y tomando como etiqueta real la correspondiente en Z a cada uno de los N ejemplos representados por las N filas de F.

2. Fase de ejecución:

   - Ejecutamos cada clasificador sobre el nuevo ejemplo para generar el vector de etiquetas binarias s, tal y como explicamos anteriormente para la obtención de las filas de A, siempre respetando el mismo orden de ejecución de los clasificadores.
   - Una vez que tenemos el vector s, que representa al nuevo elemento en el espacio original, calculamos su transformación al nuevo espacio reducido en función de δ de la siguiente forma:

     $$s_\delta = \frac{1}{L} s \cdot G' \cdot \Gamma' \tag{1.22}$$

     donde G' y Γ' representan las versiones reducidas de G y Γ respectivamente, es decir, G' será el resultado de extraer las primeras *delta* columnas de G, mientras que Γ' será la matriz formada por las primeras δ columnas y δ filas de Γ.
   - Ejecutamos el clasificador elegido en la fase de entrenamiento sobre el nuevo vector s_δ.

El proceso de transformación puede simplificarse mediante la unificación de los términos constantes de 1.22, obteniendo la matriz de transformación $T = (1/L) \cdot G' \cdot \Gamma'$, de tamaño $(L \cdot c) \times \delta$.

1.5.5. Combinación Simple con Valores Continuos

Cuando contamos con clasificadores que nos proporcionan información de acorde al *measurement level* [137], podemos combinar las salidas aportadas por los clasificadores en forma de valores continuos mediante diversas reglas o

fórmulas. Se consideran métodos simples de combinación ya que no requieren entrenamiento previo o preprocesamiento de los datos para el propio método de combinación, sino que se aplica directamente sobre los resultados de los clasificadores, tal y como ocurre con la votación por mayoría. Si para un ejemplo x, formamos una matriz M, cuyos elementos m_{ij} se correspondan con la probabilidad otorgada por el clasificador D_i al hecho de que la etiqueta correcta de x sea ω_j, podemos utilizar algunas de las siguientes funciones combinatorias, donde el índice que maximiza el valor de $\mu_j(x)$ nos dará como resultado la etiqueta a seleccionar:

- Media:

$$\mu_j(x) = \frac{1}{L}\sum_{i=1}^{L} m_{i,j}(x) \tag{1.23}$$

  Una variante consiste en la ordenación previa de todos los valores para cada etiqueta, descartando un porcentaje de los valores a ambos lados de la lista. El resultado se obtendría hallando la media sobre los valores restantes.

- Máximo (también se podría usar el mínimo o la mediana):

$$\mu_j(x) = max_i\{m_{i,j}(x)\} \tag{1.24}$$

- Producto:

$$\mu_j(x) = \prod_{i=1}^{L} m_{i,j}(x) \tag{1.25}$$

- Media armónica:

$$\mu_j(x) = \left(\frac{1}{L}\sum_{i=1}^{L}\frac{1}{m_{i,j}(x)}\right)^{-1} \tag{1.26}$$

También se pueden emplear funciones que requieren un entrenamiento previo de la función de combinación o una fase de pre-procesado, como es el caso de la media ponderada. En estos casos, el término $m_{i,j}$ de la ecuación 1.23 queda multiplicado por un peso asociado al clasificador D_i. Estos pesos suelen basarse en la tasa de error estimada en base a los datos de entrenamiento.

1.5.6. Integral Difusa

Podemos ver la combinación de clasificadores desde el punto de vista del análisis de decisiones multicriterio. De esta forma, para tomar una decisión sobre un problema dispondremos de varias alternativas, que atendiendo a diversos criterios aportarán otros tantos valores de preferencia tras aplicar lo que en este contexto suele denominarse 'función de utilidad'. En nuestro caso, los criterios se corresponden con los clasificadores y podemos tomar sus valores de confianza para cada clase (o alternativa) como valores de utilidad de tipo difuso, comprendidos en el rango $[0 \dots 1]$. A continuación llegaría la parte del consenso, en la que se utiliza un operador de agregación para unificar los valores de utilidad que se obtienen al aplicar cada criterio a cada alternativa posible. Este operador que obtiene la decisión final puede ser de diferentes tipos, incluyendo las integrales difusas. Estos operadores se basan en el concepto de medida difusa, introducido por Sugeno en 1974. Si tomamos X como el conjunto de criterios y $\wp(X)$ como el conjunto de las partes de X, podemos definir como medida difusa la función que cumple:

$$\begin{aligned} g : \wp(X) \rightarrow [0 \dots 1], \quad & g(\emptyset) = 0, \\ & g(X) = 1, \\ & A \subset B \Rightarrow g(A) \leq g(B) \quad A, B \in \wp(X) \end{aligned}$$

Utilizando este tipo de operadores de agregación como método de combinación, se tienen en consideración no sólo el conjunto completo de clasificadores base, sino también todos los subconjuntos que se pueden formar con ellos. Una vez obtenidas las medidas difusas para estos subconjuntos se aplicará la integral, cuyas versiones más populares son las de Sugeno y Choquet. Mientras la integral de Choquet puede verse como una media ponderada, la integral de Sugeno se asemeja más a una generalización del concepto de mediana. El algoritmo, adaptado a la combinación de clasificadores tal y como se muestra en [97], es el siguiente:

1. Para poder medir la competencia de un determinado grupo de expertos, se recurre a una medida difusa g, que inicializamos asignando un valor g^j a cada clasificador D_j, basándonos en la probabilidad estimada de que su salida sea la correcta.

2. Calculamos ahora un valor $\lambda > -1$, necesario para hallar los valores de g definitivos, resolviendo:

$$\lambda + 1 = \prod_{i=1}^{L}(1 + \lambda g^i), \qquad \lambda \neq 0$$

3. Para un ejemplo x a clasificar, formamos una matriz M, cuyos elementos m_{ij} se correspondan con la probabilidad otorgada por el clasificador D_i al hecho de que la etiqueta correcta de x sea ω_j. A continuación, para calcular las estimaciones de cada posible etiqueta ω_k, ordenamos la k-ésima columna de M para obtener $[m_{i_1,k}(x), \ldots, m_{i_L,k}(x)]$, siendo $m_{i_1,k}(x)$ el valor máximo y $m_{i_L,k}(x)$ el mínimo.

4. Ordenamos las medidas difusas $g^{i_1}, \ldots, g^{i_L}$ y fijamos el valor $g(1) = g^{i_1}$.

5. Desde $t = 2$ hasta L, calculamos recursivamente:

$$g(t) = g^{i_t} + g(t-1) + \lambda g^{i_t} g(t-1)$$

6. Calculamos la estimación final para la clase ω_k mediante:

$$\mu_k(x) = max_{t=1}^{L}\{min\{m_{i_t,k}(x), g(t)\}\}$$

   Como mencionamos anteriormente, existen varias formas de hallar el valor de $\mu_k(x)$ utilizado como criterio de decisión para seleccionar la categoría a asignar. Además de la que mostramos arriba, que se corresponde con la integral de Sugeno, también podemos recurrir a la integral de Choquet, que calcularíamos mediante:

$$\mu_k(x) = m_{i_1,k}(x) + \sum_{j=2}^{L}(m_{i_{j-1},k}(x) - m_{i_j,k}(x))g(j-1)$$

Hay que destacar que los valores de g pueden variar de una clase a otra, y que además son específicos para el ejemplo x actual.

1.5.7. Dempster-Shafer

Al igual que ocurre con la integral difusa, este método proporciona una vía para considerar todas las combinaciones de clasificadores base, aunque en este caso basándonos en la Teoría de la evidencia desarrollada por Dempster en 1967 y extendida por Shafer en 1976. En la llamada Teoría de Dempster-Shafer (DST), comenzamos por asumir un 'marco de discernimiento' o 'universo de discurso' θ, compuesto por un conjunto de alternativas mutuamente

excluyentes. Cada una de estas alternativas se considerará una hipótesis, aunque también todas las posibles agrupaciones que hagamos de ellas, por lo que el conjunto de las partes de θ, $\wp(\theta)$, será el conjunto de hipótesis finalmente considerado. La base del procedimiento reside en la función llamada 'asignación básica de probabilidad'. Esta función se define como:

$$m : 2^{\theta} \rightarrow [0 \ldots 1], \quad m(\emptyset) = 0, \quad \sum_{A \subseteq \theta} m(A) = 1 \tag{1.27}$$

Su principal característica es que rompe la restricción impuesta por las leyes de probabilidad bayesianas sobre la obligación de que la suma de las probabilidades de las hipótesis individuales sumen uno. De esta forma, la creencia aportada por una evidencia sobre una hipótesis no implica que el resto sea asignada a la negación de la misma, sino que se reparte entre todo el conjunto de hipótesis.

Apoyadas en esta función de asignación básica de probabilidad, surgen un conjunto de medidas de creencia que sirven para valorar los efectos que tienen las evidencias sobre las hipótesis:

- Grado de creencia (*belief function*): A cada función de asignación básica de probabilidad le corresponde una función que determina el grado de creencia y viceversa. Representa nuestra creencia total en la proposición representada por el conjunto de hipótesis dado y se calcula mediante:

$$Bel(A) = \sum_{B \subseteq A} m(B), \quad \forall A \subseteq \theta \tag{1.28}$$

$$m(A) = \sum_{B \subseteq A} (-1)^{|A-B|} Bel(B) \tag{1.29}$$

- Grado de duda: Representa el mínimo grado de creencia en la negación de A dada una determinada evidencia:

$$D(A) = Bel(\neg A) \tag{1.30}$$

- Grado de plausibilidad (o verosimilitud): Representa cuanto deberíamos creer en A si todos los hechos desconocidos apoyasen a A.

$$P(A) = 1 - D(A) \tag{1.31}$$

- Intervalo de creencia: Representa el grado de certeza sobre el impacto de una evidencia sobre la hipótesis A, siendo mayor cuanto menor es el intervalo, que viene determinado por $[Bel(A), P(A)]$. En la figura 1.8 vemos una representación gráfica de este concepto en relación al resto de medidas comentadas.

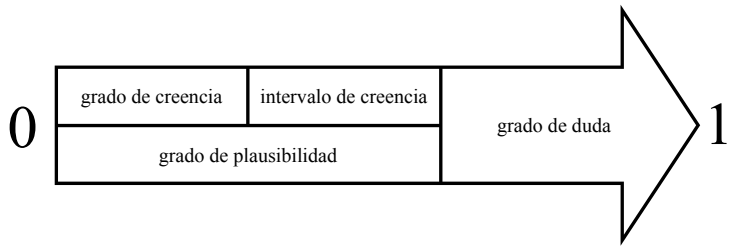

Figura 1.8: Relación entre las medidas utilizadas en DST.

Con estos instrumentos, es posible la combinación de evidencias de manera que se pueda calcular el efecto que van teniendo estas evidencias en el grado de creencia en las hipótesis. El orden en el que se vayan combinando no afecta al resultado final, que se calcula mediante la siguiente ecuación para dos asignaciones básicas de probabilidad m_1 y m_2:

$$m(A) = \frac{1}{1-K} \sum_{B \cap C = A} m_1(B) \cdot m_2(C) \tag{1.32}$$

Donde K es una constante de normalización que sirve para distribuir el valor de creencia en $\emptyset$ por el resto de hipótesis:

$$K = \sum_{B \cap C = \emptyset} m_1(B) \cdot m_2(C) \tag{1.33}$$

Aunque existen distintas adaptaciones de esta teoría a la tarea de clasificación, vamos a exponer el algoritmo de combinación más popular, según las ideas de [102], en el que los pasos a ejecutar serían los siguientes:

1. Calculamos las plantillas de decisión ($DT(j)$) tal y como se explica en el apartado 1.5.3, de manera que obtengamos una plantilla para cada una de las c etiquetas posibles.

2. Para un ejemplo a clasificar x, calculamos la proximidad Φ entre la plantilla $DT(j)$ y las decisiones de un clasificador D_i mediante:

$$\Phi_{j,i}(x) = \frac{(1 + \|DT_i(j) - D_i(x)\|^2)^{-1}}{\sum_{k=1}^{c}(1 + \|DT_i(k) - D_i(x)\|^2)^{-1}1} \tag{1.34}$$

Siendo $DT_i(j)$ la fila i-ésima de la plantilla $DT(j)$ y considerando $D_i(x) = [d_{i1}(x), \ldots, d_{ic}(x)]$ el vector de probabilidades que el clasificador D_i asigna a cada una de las clases $\omega_1, \ldots, \omega_c$ para el ejemplo x. $\|\cdot\|$ puede calcularse como la distancia euclídea o cualquier otra norma.

3. A continuación, una vez obtenidos los valores de proximidad entre los L clasificadores $D_1, \ldots, D_L$ y cada etiqueta ω_j, calculamos los grados de creencia (*belief*):

$$b_j(D_i(x)) = \frac{\Phi_{j,i}(x)\prod_{k\neq j}(1 - \Phi_{k,i}(x))}{1 - \Phi_{j,i}(x)[1 - \prod_{k\neq j}(1 - \Phi_{k,i}(x))]} \tag{1.35}$$

4. Finalmente, calculamos la estimación final $\mu_j(x)$ para cada clase ω_j, de donde obtendremos la categoría seleccionada, que vendrá determinada por el índice que consiga maximizar el valor resultante de la expresión:

$$\mu_j(x) = K\prod_{i=1}^{L} b_j(D_i(x)), \qquad j = 1, \ldots, c \tag{1.36}$$

siendo K una constante de normalización para que $\sum_{j=1}^{c}\mu_j(x) = 1$.

1.5.8. Métodos de Selección de Clasificadores

Los métodos basados en la selección de clasificadores se caracterizan por dividir el conjunto de ejemplos, declarando un experto de entre los clasificadores disponibles para cada una de las regiones creadas. De esta forma, dicho experto será el encargado de etiquetar los ejemplos que pertenezcan a la región que le ha sido asignada.

Clustering

Mediante las técnicas de *clustering*, como puede ser el método de *k-means*, podemos crear las regiones o *clusters*, evitando el peligro de generar regiones para las que no existe un número suficiente de ejemplos. Los pasos básicos que deberemos seguir son:

1. En primer lugar entrenaremos los clasificadores base $D_1, \ldots, D_L$ que queremos considerar para nuestro sistema y elegiremos el número de regiones k que queremos crear.

2. Aplicamos un método de *clustering* para generar las regiones $R_1, \ldots, R_k$ y seleccionamos los centroides $v_1, \ldots, v_k$ de cada una de las regiones, los cuales actuarán como representantes de las mismas.

3. Calculamos ahora la tasa de error de cada clasificador correspondiente a cada región, seleccionando como experto $D_{i,k}$ al clasificador D_i que obtenga mejores resultados sobre la región k.

4. Finalmente, para cada ejemplo x que queramos clasificar, deberemos hallar la región cuyo centroide se encuentre más cerca de x, ejecutando el clasificador asignado a dicha región.

Existen algunas variaciones sobre este algoritmo, como el publicado por Liu [76], en el que cada clasificador D_i propone una división de Z, comenzando por la escisión del número total de ejemplos en dos, dejando a un lado los que ha podido clasificar correctamente (Z_i^+) y por otro a los que no (Z_i^-). A su vez Z_i^+ se divide en función de las clases a las que pertenecen los ejemplos, generando las regiones $Z_i^+(j)$, con $j = 1, \ldots, c$. Z_i^- por su parte se divide en k regiones según el algoritmo de *clustering* utilizado. Una vez creadas las particiones asociadas a cada clasificador, se calcula la tasa de error que obtiene cada clasificador para cada una de sus regiones. A la hora de clasificar un nuevo ejemplo x, se averigua en primer lugar la región a la que pertenece según la división de cada clasificador, aplicándose finalmente el que tenga una tasa de error menor para la región que le ha sido asignada previamente a x.

Mixture of Experts

El método de *mixture of experts* [60] fue ideado para su utilización con redes neuronales. Utiliza un conjunto de clasificadores candidatos a ser seleccionados para emitir la clase final y un clasificador adicional que influirá decisivamente en la selección del candidato. Este clasificador recibe como entrada el ejemplo a clasificar x, emitiendo como salida las probabilidades de acierto estimadas para cada clasificador base D_i. En función de estas probabilidades y de las salidas correspondientes a cada clasificador, se emplea una regla de selección que determina la salida que debe escogerse finalmente. Esta regla de selección puede variar entre muchas posibilidades, aunque destacamos por su popularidad las siguientes:

- Selección estocástica: se selecciona el clasificador base realizando un muestreo respetando las probabilidades de acierto asignadas por el clasificador adicional.
- Todo al ganador: se selecciona la salida aportada por el clasificador base cuya probabilidad de acierto asignada por el clasificador adicional es mayor.
- Ponderación: Las probabilidades de acierto asignadas por el clasificador adicional se utilizan para ponderar las salidas de los clasificadores, calculando la etiqueta que recibe un mayor apoyo por los clasificadores base una vez han sido multiplicadas por los pesos.

1.5.9. Bagging y Boosting

En la figura 1.7 vimos como la combinación se puede afrontar desde diferentes puntos de vista o haciendo uso de diversos enfoques, dependiendo del nivel del proceso en el que generemos la variabilidad necesaria. En esta sección, veremos dos algoritmos que trabajan este aspecto directamente con los datos y empleando un único algoritmo de aprendizaje, generando diferentes versiones del corpus de entrenamiento en el caso del *bagging*, o ponderando la utilidad que tiene cada ejemplo para el proceso de aprendizaje en el caso del *boosting*.

Bagging

El método *bagging*, término que procede de la expresión anglosajona *bootstrap aggregating*, adquirió mucha popularidad tras su publicación por Breiman [16] dada su capacidad para combinar clasificadores inestables. Su aplicabilidad en entornos donde existe una inestabilidad manifiesta procede del mecanismo que emplea para obtener la variabilidad que persigue todo algoritmo de combinación para mejorar las prestaciones de los clasificadores base. En este caso, se consigue dicha variabilidad obteniendo diferentes versiones del corpus de entrenamiento original mediante remuestreo con reemplazamiento, perturbando de esta forma el conjunto de aprendizaje y dando lugar a cambios significativos en los diferentes clasificadores generados. De esta forma, en cada versión nueva de la colección de datos se pueden encontrar varias ocurrencias de un mismo ejemplo o puede que ninguna, manteniendo en todo caso el tamaño original de la colección. Para bases de datos de tamaño considerable, es de esperar que en cada conjunto de datos generado se encuentren aproximadamente el 63 % de los ejemplos contenidos en el conjunto original, siendo el resto repeticiones de estos mismos ejemplos. Una vez

que hemos obtenido las L versiones del conjunto de datos, se generan los L clasificadores utilizando cada versión para el entrenamiento de un clasificador diferente. A la hora de clasificar un elemento nuevo, éste se presenta como entrada para cada clasificador y se obtiene la etiqueta definitiva mediante votación aplicada a las L salidas obtenidas. De esta forma, la técnica *bagging* sólo requiere un bucle inicial para seleccionar los muestreos y un *back-end* que se encarga de la agregación.

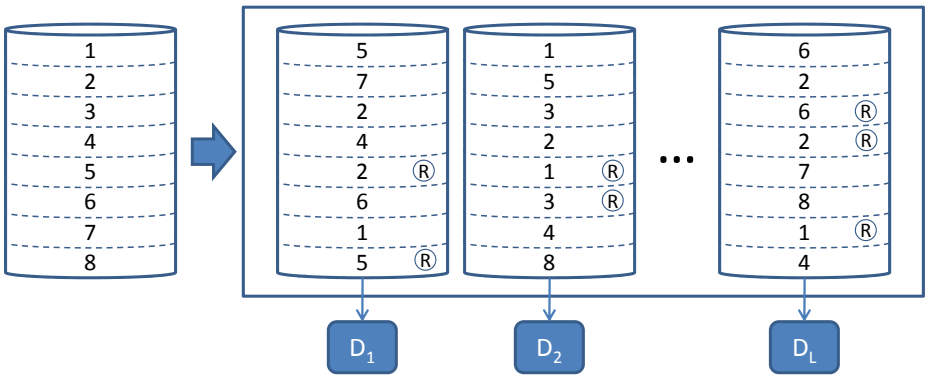

Figura 1.9: Muestreos para la generación de clasificadores (*bagging*).

Como hemos comentado anteriormente, es preciso que los clasificadores base sean inestables, como es el caso de las redes neuronales o los árboles de decisión, cuyo comportamiento sufre grandes alteraciones tras ejercer pequeñas variaciones en los datos de entrada. Con métodos estables, como la regresión lineal o la técnica de los vecinos más cercanos, los resultados pueden degradarse en lugar de mejorar tras la aplicación de *bagging*. También hay que reconocer que la ganancia de precisión hay que enfrentarla con la pérdida de velocidad y en ciertos casos, como ocurre con los árboles de decisión, también se pierde claridad y comprensión.

Los algoritmos llamados *random forest*, se corresponden con la idea de *bagging* aplicada a los árboles de decisión como clasificador base. La forma de obtener la variabilidad puede estar basada en el muestreo sobre los ejemplos del conjunto de datos, tal y como se ha explicado anteriormente, o sobre el vector de características e incluso sobre los parámetros internos del árbol de decisión empleado. Cualquiera de los sistemas obtenidos de alguna de estas formas es considerado un miembro de la familia de los *random forest*.

Boosting

La técnica *boosting* consiste en ponderar los ejemplos del conjunto de datos, de manera que de forma iterativa se vaya generando un clasificador que se vea reforzado para lograr clasificar correctamente los ejemplos en los que ha cometido errores en las primeras iteraciones del proceso. De esta forma, los ejemplos "fáciles" no reclaman la atención del algoritmo de aprendizaje, centrado en mejorar la precisión en los ejemplos más "difíciles" de etiquetar. La idea procede del algoritmo *hedge*(β) [48], cuyo objetivo es la asignación de pesos a un conjunto de clasificadores, en función de la probabilidad estimada de que aporte la etiqueta correcta a un ejemplo nuevo. En este algoritmo, partimos de varios clasificadores y desconocemos cuál de ellos está más capacitado para la tarea que queremos desempeñar, por lo que iniciamos el proceso asignándoles la misma probabilidad de éxito, representada en este caso por los pesos. A medida que vamos clasificando ejemplos y vamos comprobando los errores cometidos, se procede a la modificación de los pesos, que se verán aumentados en el caso de aquellos asignados a clasificadores que aporten la etiqueta correcta en su salida, y viéndose decrecidos aquellos asignados a los clasificadores que han errado en su predicción. Si aplicamos este método durante un número considerable de iteraciones, veremos como el algoritmo va decantándose por uno de los clasificadores participantes, acercándose sus cotas de error a las que obtendría este clasificador en solitario, siendo considerado el más apto para la tarea.

Hedge

	k=1 (z_1)		k=2 (z_2)		...	k=N (z_N)
D	w	err	w	err	...	w
D_1	1/3	0	2/5	0	...	~ 1
D_2	1/3	0	2/5	1	...	~ 0
D_3	1/3	1	1/5	0	...	~ 0

Boosting

	k=1 (D_1)		k=2 (D_2)		...	k=N (D_N)
Z	w	err	w	err	...	w
z_1	1/3	1	2/5	1	...	~ 1
z_2	1/3	1	2/5	0	...	~ 0
z_3	1/3	0	1/5	1	...	~ 0

Tabla 1.2: De *hedge* al *boosting*

El método *boosting* modifica este algoritmo ponderando en este caso los ejemplos del conjunto de datos en lugar de los clasificadores. En el ejemplo orientativo de la tabla 1.2 se aprecia como pasamos, de ponderar los clasificadores en base a los errores cometidos con el ejemplo de la presente iteración, a ponderar los ejemplos en base al clasificador que se genera en la iteración actual. De esta forma, un ejemplo que no consigue etiquetarse correctamente por el clasificador, verá incrementado su peso para la siguiente iteración, reclamando más atención por parte del clasificador. En cada iteración por tanto, se muestrea el conjunto de datos Z de tamaño N, según la distribución de probabilidad representada por los pesos w_j^i de la iteración actual i para cada ejemplo $z_j \in Z$. Los pesos serán inicializados generalmente con el valor $w_j^1 = 1/N$ aunque se les podría asignar diferentes valores si disponemos de algún conocimiento previo sobre la "dificultad" de los ejemplos. Los pasos de la fase de entrenamiento del algoritmo resultante, bautizado como *adaboost* (*adaptive boosting*), serían los siguientes:

1. Inicializar los pesos de manera que $\sum_{j=1}^{N} w_j^1 = 1$ y seleccionar el número de clasificadores a generar L (equivalente al número de iteraciones).

2. Para cada iteración $k = 1, \ldots, L$:

   a) Realizar un muestreo del conjunto de datos Z según la distribución de pesos de la iteración actual, utilizándolo para entrenar el clasificador D_k.

   b) Calcular el error asociado a la iteración k como:

   $$\epsilon_k = \sum_{j=1}^{N} w_j^k l_k^j \tag{1.37}$$

   donde $l_k^j = 1$ si D_k se equivoca al clasificar z_j y $l_k^j = 0$ en caso contrario.

   c) Si $\epsilon_k = 0$ ó $\epsilon_k \geq 0,5$, desechar D_k, reinicializar los pesos y continuar volviendo al paso $2a$, y en caso contrario, calcular:

   $$\beta_k = \frac{\epsilon_k}{1 - \epsilon_k} \tag{1.38}$$

   y actualizar los pesos según la fórmula:

   $$w_j^{k+1} = \frac{w_j^k \beta_k^{1-l_k^j}}{\sum_{i=1}^{N} w_i^k \beta_k^{1-l_k^i}} \qquad j = 1, \ldots, N \tag{1.39}$$

Dado un nuevo elemento x a clasificar, el algoritmo realiza la predicción sobre la etiqueta que debe serle asignada mediante la búsqueda de la clase ω_t que obtiene el mayor valor para la función:

$$\mu_t(x) = \sum_{D_k(x)=\omega_t} ln\left(\frac{1}{\beta_k}\right) \tag{1.40}$$

1.5.10. Selección de Características

Centrándonos en el tercer nivel de la taxonomía mostrada en la figura 1.7, en la cual la combinación se lleva a cabo buscando la variabilidad en el conjunto de características, encontramos diferentes agrupaciones de estas características que pueden dar lugar a los clasificadores a combinar. Además de una posible agrupación aleatoria, como ocurre en el *random subspace method*, o de una agrupación natural, que podrá llevarse a cabo si hay características relacionadas entre sí de manera conceptual, existen otras muchas técnicas de selección de características. El *random subspace method* genera conjuntos de características seleccionadas de forma aleatoria, dando lugar cada uno de ellos a un clasificador base. La combinación de estos clasificadores se suele llevar a cabo mediante algún sistema de votación. Otro algoritmo sencillo, llamado *input decimation* [96], consiste en crear un clasificador especializado en una clase determinada. Si tenemos por lo tanto c clases, el número de clasificadores a generar será $L = c$, cada uno declarado como experto en una clase concreta. Para hallar el conjunto de características a utilizar para un clasificador asociado a la clase ω_j, se calcula la correlación existente entre cada característica del conjunto $F = f_1, \ldots, f_n$ y la clase ω_j, ordenando los resultados para descartar las que presenten una menor correlación. Los pasos del algoritmo son los siguientes:

1. Para cada clase ω_j:

   a) Utilizando el corpus de entrenamiento $Z = z_1, \ldots, z_N$, hallar los valores de correlación entre las características $f_1, \ldots, f_n$ y la clase ω_j mediante:

$$\rho(f_i, \omega_j) = \frac{N \sum f_{i,k}\omega_{j,k} - \sum f_{i,k} \sum \omega_{j,k}}{\sqrt{N \sum f_{i,k}^2 - (\sum f_{i,k})^2}\sqrt{N \sum \omega_{j,k}^2 - (\sum \omega_{j,k})^2}}$$

   donde $k = 1, \ldots, N$ y $\omega_{j,k} = 1$ si $z_k \in \omega_j$ o $\omega_{j,k} = 0$ en caso contrario.

b) Seleccionar el conjunto de características s_j con mayor correlación en valor absoluto y generar un corpus Z_j, en el que sólo aparezcan las características seleccionadas.

c) Generar el clasificador D_j utilizando el corpus Z_j.

2. Dado un nuevo ejemplo x a clasificar, generamos las L versiones de su vector de características, en las que la versión x_j del ejemplo viene representada por el conjunto de características s_j, utilizando dicha versión como entrada para el clasificador D_j.

3. Aplicamos un criterio de selección, como por ejemplo calcular el valor promedio de las probabilidades asignadas por cada clasificador a cada una de las clases, eligiendo aquella que obtenga el mayor valor como etiqueta final para x.

1.5.11. ECOC

El método *error correcting output codes* (ECOC) representa un caso particular de combinación de sistemas, ya que no encaja en la mayoría de las taxonomías empleadas habitualmente para agrupar los métodos existentes. Se trata de un método que realiza la combinación en base a las salidas o clases asignadas a los ejemplos, generando la variabilidad en el propio conjunto de etiquetas $C = \omega_1, \ldots, \omega_c$. La estrategia va encaminada a convertir un problema multiclase en un conjunto de problemas binarios, cada uno de los cuales será el objetivo de uno de los clasificadores que se van a generar, de ahí que este método esté enfocado a problemas con múltiples categorías de clasificación.

La idea consiste en crear particiones binarias del conjunto de clases, creando un clasificador asignado a cada una de estas particiones, de manera que se encargue de clasificar los ejemplos en base ella. Si tenemos c clases, podemos representar una determinada partición asociada al clasificador D_j mediante un vector binario de c posiciones, en donde los ceros y los unos indican el lado de la partición en el que ha recaído una determinada clase. Por ejemplo, un vector $[0, 1, 1, 0]$ para un problema de cuatro clases ($c = 4$), representa una partición que divide C en los conjuntos $\{\omega_1, \omega_4\}$ y $\{\omega_2, \omega_3\}$. Si creamos L particiones diferentes, podemos generar una matriz de códigos en la que aparecen los valores binarios asignados en cada partición (que a su vez lleva asociado un clasificador binario D_j) a cada clase ω_j. El número de particiones que podemos generar con c clases es igual a $2^{c-1} - 1$ tras eliminar las divisiones vacías y tomar en cuenta la duplicidad de las particiones inversas que hace que la partición $[1, 0, 0, 1]$ sea equivalente a $[0, 1, 1, 0]$. En la tabla 1.3

podemos ver una matriz de códigos para $c = 4$, en la que las filas reciben el nombre de *codewords*.

	D_1	D_2	D_3	D_4	D_5	D_6	D_7
ω_1	0	0	0	1	0	0	0
ω_2	0	0	1	0	0	1	1
ω_3	0	1	0	0	1	1	0
ω_4	1	0	0	0	1	0	1

Tabla 1.3: Matriz de códigos para $c = 4$.

Al ejecutar los L clasificadores basados en las diferentes particiones, obtendremos un vector de salidas binarias para un ejemplo x, que una vez comparadas con los *codewords* servirán para hallar la clase más probable mediante alguna medida de distancia (habitualmente se utiliza la distancia de Hamming).

Es evidente que la matriz de códigos que se emplee es de vital importancia para la obtención de buenos resultados mediante ECOC, por lo que existen ciertas directrices que deben contemplarse a la hora de diseñarla. Lo primero es procurar maximizar la distancia entre los *codewords* utilizados, de manera que el sistema sea lo más robusto posible frente a errores puntuales cometidos por alguno de los clasificadores generados. Asimismo, conviene maximizar la distancia entre las particiones seleccionadas, de manera que se vea reducida la correlación entre los errores de clasificación. A estas dos premisas se les suele denominar 'separación de filas' y 'separación de columnas' respectivamente. En la literatura podemos encontrar diversos métodos para la creación de las matrices de códigos, en algunos casos bastante sencillos de implementar, aunque hay trabajos como [50] que muestran evidencias teóricas sobre la conveniencia de utilizar códigos aleatorios. También se ha experimentado con la cooperación con otros algoritmos, como es el caso del *low density parity check* (LDPC), que hace posible emplear la opción de abstención en el método ECOC.

1.5.12. Meta-Aprendizaje

El término 'meta-aprendizaje' suele emplearse para denominar a los sistemas que utilizan las salidas proporcionadas por un método de aprendizaje automático como entrada para otro. En el área del reconocimiento de patrones, habitualmente se utiliza la palabra *stacking* para hacer referencia a estos métodos, ya que se puede interpretar este proceso como una pila, en la que

un algoritmo se nutre de los resultados del algoritmo en el que se apoya, ya sean uno o varios. De todos modos, distinguiremos dos claras vertientes que en ocasiones se encuentran mezcladas bajo la denominación de *stacking* pero que aquí separaremos según el flujo de información:

- *Stacking*: Entendemos que estos métodos son los relacionados directamente con la idea original de Wolpert [134], bautizada como *stacked generalization*, según la cual se deben seguir los siguientes pasos para crear un sistema que combine la salida de n clasificadores:

  1. Generar k particiones Z_i del conjunto de datos de entrenamiento Z.
  2. Para cada clasificador D_j, con $j = 1, \ldots, n$ y cada partición Z_i, con $i = 1, \ldots, k$:
     a) Generar el clasificador $D_j(i)$ entrenando D_j con los datos que resultan de sustraer de Z la partición Z_i $(Z - Z_i)$.
     b) Ejecutar el clasificador $D_j(i)$ sobre la partición Z_i
  3. Entrenar un clasificador D^s cuyos datos de entrenamiento estén compuestos por las salidas aportadas por cada clasificador D_j para cada ejemplo de Z, junto a la salida correcta correspondiente.
  4. Finalmente, se entrenan los n clasificadores con la totalidad de Z para obtener la versión definitiva de los clasificadores base D_j que se utilizarán en la fase de clasificación.

  En la fase se ejecución o clasificación, los clasificadores base procesarán los nuevos ejemplos, generando de esta forma el vector de resultados que D^s se encargará de clasificar para obtener la etiqueta final.

- *Cascading*: En este caso, el proceso consiste en que un clasificador es entrenado con la totalidad del conjunto de datos de entrenamiento Z, y su salida es empleada como entrada para otro clasificador, que a su vez puede enlazarse con otro y así sucesivamente. La forma en que un clasificador hace uso de la información aportada por otro puede variar, siendo la más habitual la concatenación de los resultados en forma de nuevas características. Un ejemplo de esta técnica lo encontramos en [83], donde se afronta el problema de adaptar características nominales para su utilización con algoritmos de clasificación que requieren entradas numéricas. Debido a que pasar los datos al formato binario puede provocar algunos problemas o generar regiones no separables linealmente, se propone recurrir a un primer nivel con un clasificador

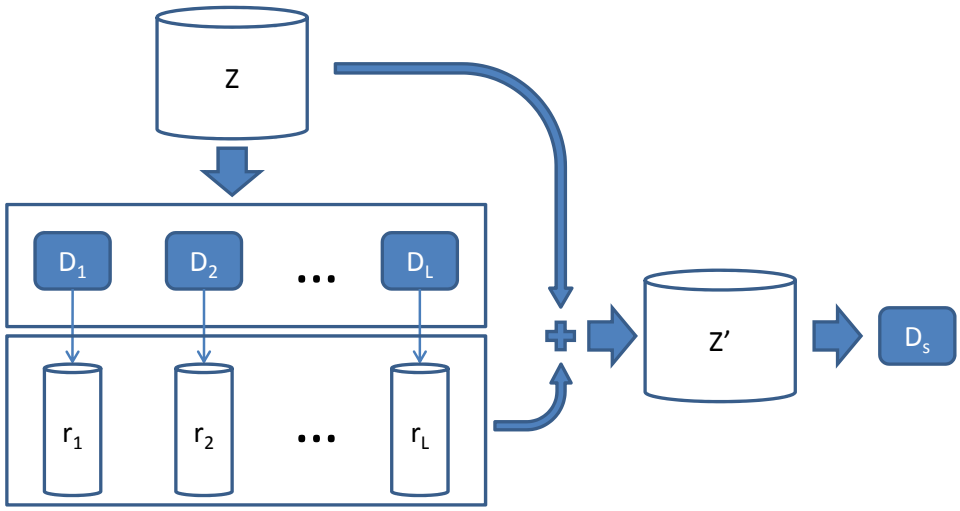

Figura 1.10: Esquema general del método *stacking*

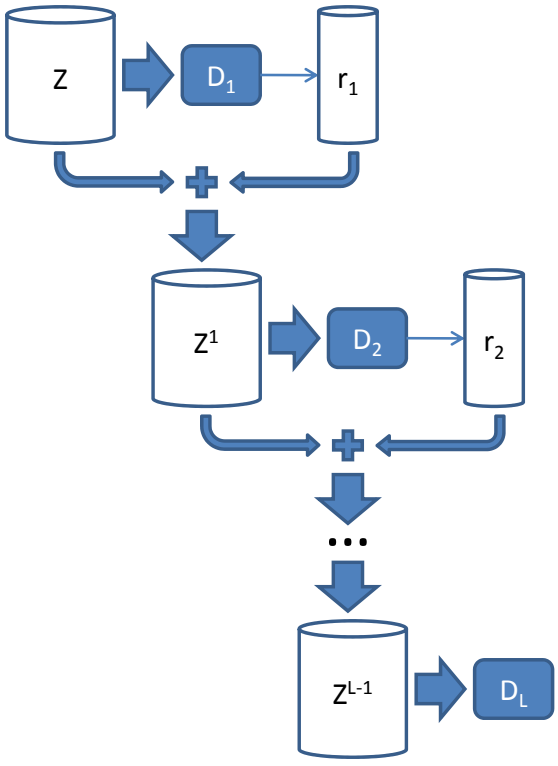

Figura 1.11: Esquema general del método *cascading*

no binario (como puede ser un árbol de decisión) para posteriormente enlazarlo con un segundo nivel ocupado por un clasificador binario (como por ejemplo un SVM). Este segundo nivel acogería como entradas las características originales más el vector de pertenencia a cada clase aportado por el primer nivel, consiguiendo así no verse degradado por el *overfitting*.

La información que un nivel de aprendizaje proporciona al siguiente, sea cual sea el esquema que sigamos de los dos que hemos visto, puede no consistir únicamente (o en absoluto) en las etiquetas o clases asignadas por los clasificadores de dicho nivel. Hay trabajos que utilizan los valores de confianza aportados por dichos clasificadores, mientras que otros consideran estos valores poco fiables y utilizan valores probabilísticos que miden la confianza que se le otorga a las salidas obtenidas. También se pueden agrupar varios conjuntos de datos, como los que acabamos de mencionar, subconjuntos de características empleadas por los clasificadores base o valores nuevos, como por ejemplo la distancia de Monge Elkan entre las distintas respuestas recopiladas.

Capítulo 2

La Combinación y el PLN

Muchas de las técnicas desarrolladas en los últimos años en el área del reconocimiento de patrones y el aprendizaje automático han tenido una repercusión directa sobre la forma de resolver muchas de las tareas que componen la disciplina del procesamiento del lenguaje natural (PLN). Un gran número de estas tareas pueden considerarse tareas de clasificación, donde debemos resolver algún tipo de ambigüedad relacionada al contenido textual de una colección de documentos, pudiendo aplicarse múltiples algoritmos de clasificación, así como técnicas y estrategias de combinación de sistemas como iremos viendo a lo largo de este capítulo. Las partes de que se compone el problema de la combinación de clasificadores se pueden agrupar en dos conjuntos según Xu [137]: dependientes de la aplicación y generales o comunes a varias aplicaciones. En el primer grupo hay que resolver cuestiones como el número de clasificadores a emplear, el tipo de los mismos y el conjunto de características que se van a considerar. En el segundo grupo recaería la decisión de cómo combinar los clasificadores base. En los últimos años son muchos los trabajos que han ofrecido soluciones a estas preguntas para afrontar alguna tarea del PLN haciendo uso de la combinación de sistemas. A lo largo de este capítulo, haremos un repaso a estas propuestas, comenzando por comentar los primeros trabajos que ligaron la combinación de clasificadores y el PLN.

2.1. Orígenes

A partir de los noventa, los investigadores del campo del procesamiento del lenguaje natural empezaron a aplicar con mayor frecuencia técnicas de combinación para obtener mejores resultados en sus tareas. Los trabajos reflejaban mejoras en los resultados respecto a los obtenidos de forma aislada por cualquiera de sus componentes, aunque en general las técnicas utilizadas

para obtener el resultado final no solían ser demasiado elaboradas. Entre los métodos más utilizados durante esos primeros trabajos se encuentran los promedios sobre los resultados obtenidos por los clasificadores del primer nivel, aunque ya empezaban a vislumbrarse el potencial de emplear combinaciones lineales, Bayes o algoritmos de clasificación más complejos para un segundo nivel de aprendizaje.

En [7] se afrontaba la tarea de la recuperación de información combinando diferentes formulaciones booleanas de *querys* sobre una base de datos de documentos. Aunque por la naturaleza de la tarea, los métodos de combinación difieren de los métodos más genéricos aplicables en otros dominios, los resultados, evaluados sobre los recursos de la competición más relevante dedicada a resolver esta tarea, el TREC, fueron prometedores y sirvieron como punto de arranque para otros trabajos realizados posteriormente. Un año más tarde, en [45] también se combinó un sistema basado en conocimiento con otro basado en ejemplos y un sistema de transferencia léxica para resolver la tarea de la traducción automática (MT), concluyendo con la premisa, tan bien expuesta en el título de su trabajo, de que tres opiniones valen más que una. Las puntuaciones que cada sistema asignaba a la traducción que ofrecían para cada segmento de texto se normalizaban para hacerlas comparables entre sí y se emplearon simples medias ponderadas para favorecer a los sistemas que fallaban menos. En su esquema general, que vemos en la figura 2.1, se incluye un algoritmo de *chart walk* que recibe las propuestas de traducción de cada sistema y mediante programación dinámica busca la combinación de segmentos que obtiene mayor puntuación cubriendo todo el texto.

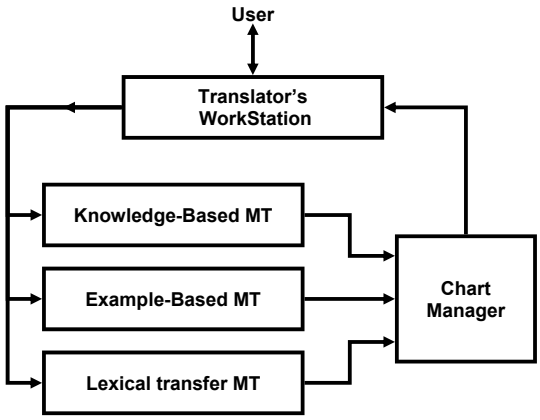

Figura 2.1: Esquema del sistema de MT en [45]

En [59] se aplicaban estas mismas ideas a la tarea de clasificación de documentos, y más concretamente al filtrado de documentos. Utilizando diversos tipos de clasificadores (nearest neighbors, Rocchio, linear discriminant analysis y neural networks) y tras varios experimentos con diferentes métodos de combinación (técnicas de promedios y de regresión), llegaban a la conclusión de que los sistemas más sencillos de promedios ofrecían mejores resultados, aunque la sospecha de que la razón podía estar en los datos utilizados, con una correlación bastante alta, hacía a los autores prever que podrían mejorar en futuros trabajos empleando modelos de inferencia Bayesianos.

En la otra gran rama del PLN, el procesamiento del habla, también se empezó a prestar más atención a los métodos de combinación a raíz de la publicación de trabajos como [40], que dejaban constancia de los beneficios que éstos podían ofrecer. En este trabajo concretamente se realiza una etapa de post procesado del reconocimiento automático del habla, consiguiendo mediante técnicas de votación reducir la tasa de errores cometidos por los diferentes sistemas utilizados. El resultado fue bautizado como ROVER (Recognizer Output Voting Error Reduction) y en la figura 2.2 podemos ver su esquema general.

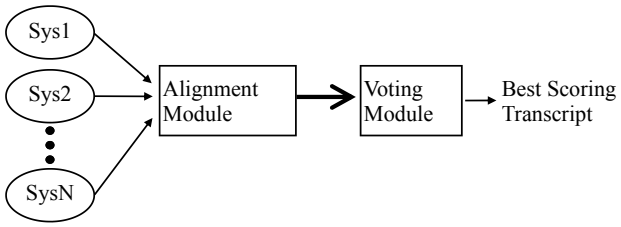

Figura 2.2: Esquema del sistema de ASR en [40]

Todos estos trabajos aportaban una base sólida para confiar en que la aplicación de las técnicas de combinación de sistemas a las tareas del área del PLN, serviría para mejorar la aportación de diversos módulos que componen un sistema complejo compuesto por analizadores sintácticos, reconocedores de entidades, etc, y fue a partir de 1998, con la publicación de los trabajos [53] y [18], cuando un mayor número de investigadores desarrollaron sus trabajos en esta dirección.

Estos dos trabajos evolucionaron prácticamente en paralelo y ambos afrontaban una tarea básica del PLN como es el análisis morfosintáctico o *Part-Of-Speech Tagging* (POS). Si bien van Halteren llevaba desde 1992 realizando experimentos de combinación de clasificadores mediante técnicas de votación, fue en [53] donde se exponían definitivamente los resultados obtenidos con el corpus LOB (Lancaster-Oslo/Bergen) aplicando, desde técnicas de votación

simple como *majority* o *total precision voting*, hasta técnicas de *stacking* empleando clasificadores de segundo nivel, pasando por un sistema de votación por pares que conseguía muy buenos resultados. Como meta-clasificadores a su vez se hicieron pruebas con árboles de decisión y algoritmos de aprendizaje basado en ejemplos. Brill y Wu por su parte, mostraron en [18] conclusiones parecidas tras aplicar técnicas de votación simple y métodos de meta-aprendizaje para extraer el máximo provecho de los contextos utilizados por los clasificadores iniciales. De nuevo, tras diversos experimentos utilizando el corpus Penn Treebank Wall Street Journal, la opción de utilizar un segundo nivel de aprendizaje, en este caso utilizando un algoritmo basado en ejemplos con dos formas diferentes de explotar la información contextual, obtenía los mejores resultados en cuanto a reducción del error.

A partir de estos trabajos se han sucedido las publicaciones que desarrollan métodos de combinación de sistemas para mejorar los resultados en tareas como el análisis sintáctico [55], la desambiguación de significados [98], el reconocimiento de entidades [41], etc, empleando diferentes algoritmos y estrategias que iremos comentando en los siguientes apartados.

2.2. Tareas

En las dos últimas décadas, los esfuerzos para sacar provecho de la variabilidad ofrecida por diferentes sistemas de clasificación, aplicados sobre una misma tarea, han demostrado la validez de esta linea de trabajo, obteniéndose en la mayoría de los casos mejoras respecto a los resultados obtenidos individualmente por los sistemas participantes en el esquema combinatorio. A continuación vamos a repasar los trabajos más destacables en las tareas más representativas del área del PLN, comentando los resultados conseguidos para, finalmente, extraer algunas conclusiones relativas a la conveniencia de aplicar estas técnicas en una determinada tarea.

2.2.1. Etiquetado POS

En la tarea del análisis morfológico, etiquetado gramatical o Part-Of-Speech, el clasificador recibe un texto y debe etiquetar cada una de las palabras que contiene asignándole la categoría gramatical correspondiente (sustantivo, adjetivo, determinante, etc). La información relevante para asignar la etiqueta correcta a cada una de las palabras del texto, reside en las características propias de la palabra, así como en su contexto, considerando como tal el resto de palabras que rodean a la que se está analizando, o incluso a las de otros documentos, que pueden formar parte de la misma colección de

datos que se esté utilizando o no. Los clasificadores por lo tanto difieren en muchos casos los unos de los otros en cuanto a la información que manejan, o también en cuanto a la forma que tienen de emplear dicha información para lograr el objetivo final. La combinación de sistemas intenta obtener beneficio de estas diferencias entre los clasificadores, localizando conflictos que se repiten entre las opiniones que aportan los componentes ante sucesivos datos de entrada, de manera que se puedan detectar patrones de error basándose en la premisa de que no todos cometen los mismos errores ante las mismas entradas. De esta forma, se intentan establecer criterios para elegir la etiqueta correcta en los casos más complicados, para los que no queda claro cuál de los clasificadores aporta la solución correcta una vez comprobado que no hay consenso entre los participantes.

Ya en [53] y [18] se aportaron datos concluyentes sobre las ventajas de combinar diferentes sistemas para la tarea del etiquetado POS, reflejando mejoras considerables sobre los resultados obtenidos por los diferentes clasificadores utilizados de forma aislada. Concretamente, ambos utilizaros sistemas basados en modelos de Markov, en reglas de transformación y en el algoritmo de máxima entropía, añadiendo [53] un etiquetador basado en ejemplos mientras que [18] agrega un sencillo algoritmo de unigramas a la lista de clasificadores empleados. Ambos reportan mejoras aunque [18] evalúa sus algoritmos mediante el corpus Penn Treebank del Wall Street Journal (WSJ) mientras que [53] lo hace mediante el corpus Lancaster-Oslo/Bergen (LOB). En la tabla 2.1 vemos como en [53] los resultados de todos los clasificadores (T=*Trigrams*, R=*Transformation rules*, M=*Memory-based* y E=*Maximum entropy*) son claramente mejorados por todos los sistemas de combinación probados. Entre estos se encuentran diversos algoritmos de votación y también *stacking*, como método de meta-aprendizaje que agrega un segundo nivel de abstracción mediante un clasificador que recibe como entradas las salidas de los clasificadores base.

En [18] por su parte, utilizando el corpus Penn Treebank como mencionamos anteriormente, también se reduce la tasa de error con un algoritmo de votación simple, aunque vuelve a resultar como mejor opción la de agregar un segundo nivel de aprendizaje como podemos ver en la tabla 2.2. En este caso, se emplea un algoritmo basado en ejemplos y se desarrollan dos formas de seleccionar la etiqueta final, seleccionando la etiqueta que más probabilidades tiene de ser la correcta o bien seleccionando el clasificador que más probabilidades tiene de aportar la solución correcta.

No es de extrañar por lo tanto, que de estos trabajos surgiese un gran interés por experimentar con nuevos corpus y clasificadores para comprobar si los resultados eran consistentes en diferentes entornos de ejecución. Como precursor de este tipo de trabajos en la tarea del POS también hay que men-

Single Tagger		
T	96,08	
R	96,46	
M	96,95	
E	97,43	

Simple Voting		
Majority	97,63	+0,20
TotPrecision	97,80	+0,37
TagPrecision	97,68	+0,25
Precision-Recall	97,84	+0,41
Pairwise Voting		
TagPair	97,92	+0,49
Memory-Based		
Tags	97,87	+0,44
Tags+Word	97,82	+0,39
Tags+Context	97,69	+0,26
Decision trees		
Tags	97,78	+0,35
Tags+Context	97,63	+0,20

Tabla 2.1: Resultados de [53]

Single Tagger		
Unigram	93,26	
Trigram	96,36	
Transform.	96,61	
Max. Ent.	96,83	

Combination		
Simple Voting	97,05	+0,22
Context: Pick Tag	97,14	+0,31
Context: Pick Tagger	97,16	+0,33

Tabla 2.2: Resultados de [18]

cionar a [81], en cuyo trabajo se emplearon sólo dos clasificadores base, uno basado en satisfacción de restricciones (concretamente utilizando *relaxation labelling*) y otro basado en árboles de decisión. Los métodos de combinación explorados eran métodos de votación y aunque los resultados no presentaban mejoras sustanciosas, se utilizaron estos métodos para estudiar la posibilidad de ampliar un corpus ya existente aprovechando la mayor fiabilidad que aporta un sistema que tiene en cuenta más de una opinión sobre la etiqueta que debe recibir una determinada palabra. Un año más tarde, en [82] los autores evaluaron mediante el corpus Penn Treebank (en lugar del corpus LexEsp utilizado en el trabajo anterior) un mayor número de métodos para combinar diversas fuentes de información sobre las etiquetas a asignar. Concretamente se utilizaron dos versiones de un clasificador basado en árboles de decisión, que por otra parte se divide en varios clasificadores para encargarse cada uno de ellos de una clase en particular. Las técnicas evaluadas son *bagging*, *feature selection criteria* (FSC) y combinación de características. En el caso de *bagging* se crearon diez réplicas de cada colección de datos de entrenamiento utilizados. En el método FSC se crearon distintos clasificadores empleando diferentes funciones de selección de características dentro de los nodos del árbol. Por último en el apartado de combinación de características se agruparon las características utilizadas para clasificar las instancias de manera que se construyeron combinaciones de ocho árboles. Finalmente, se evalúa también la utilización del método *convex pseudo data* (CPD), que consiste en seleccionar aleatoriamente ejemplos de la misma clase y generar un nuevo ejemplo extrayendo características de los "padres". Los resultados obtenidos igualan los recopilados por [18], alcanzando una mejora del 0,38 % de *accuracy* respecto al mejor clasificador.

En [13] el problema se afronta desde otra perspectiva, mediante la utilización de un sistema de combinación basado en reglas motivadas lingüísticamente. Esto hace de él un sistema pionero al ser un sistema de combinación basado en conocimiento, intentando profundizar en cómo deben utilizarse las diferencias existentes entre los distintos clasificadores para conseguir mejores resultados en la asignación final de las etiquetas. Los experimentos se llevan a cabo en unas circunstancias en las que no se poseen suficientes datos de entrenamiento para poder utilizar métodos basados en ejemplos y en donde se desea aplicar clasificadores ya existentes sobre textos de diferentes categorías a los utilizados durante el entrenamiento. Durante los experimentos se utilizaron manuales técnicos (corpus Scania) y prosa política (*Swedish Statement of Government Policy*), realizándose un estudio sobre las diferencias entre las opiniones aportadas por los etiquetadores. Se establecieron una serie de reglas que especificaban en qué circunstancias se debe escoger la etiqueta seleccionada por los clasificadores menos precisos, en contra de la opinión de

aquellos que, en general, se muestran más efectivos. Estas reglas permiten sacar las mismas conclusiones que se extraen de los trabajos que emplean métodos estadísticos para la combinación de los clasificadores, estableciendo una previsión de mejora en los resultados, que varía en función del tipo de textos que se van a procesar, pero que se presenta como una opción consistente para mejorar los resultados en las circunstancias descritas anteriormente.

El trabajo [10] gira en torno a la eliminación del ruido que se introduce en un corpus etiquetado manualmente. Para reducir la tasa de errores se emplean técnicas de votación sobre clasificadores automáticos basados en ejemplos, en reglas de transformación y en árboles de decisión, evaluando los resultados sobre el corpus Susanne. Se utilizan dos métodos para decidir sobre la eliminación de una etiqueta introducida anteriormente, que consisten en utilizar un filtro por consenso (todos los clasificadores deben coincidir en que la etiqueta es correcta) o por mayoría (la etiqueta se considera errónea si más del 50 % de los clasificadores coinciden en esta afirmación). El primer filtro resulta más conservador y elimina menos ruido aunque también comete menos errores eliminando un menor número de etiquetas correctas. El filtro por mayoría en cambio actúa de manera opuesta, siendo más eficaz eliminando ruido aunque cometiendo más errores durante el proceso.

En el año 2001, Halteren vuelve a dar a conocer nuevos experimentos que profundizan más aún en la línea de investigación que abrió en 1998. En [52] se agregan los corpus Penn Treebank y Eindhoven al corpus LOB utilizado anteriormente, lo que les permite compararse con otros trabajos como [18] en el caso del Penn Treebank, o medir las bondades de su propuesta con un corpus más pequeño y en un idioma distinto del inglés, en el caso del corpus Eindhoven, compuesto por documentos en holandés utilizando el conjunto de etiquetas Wotan. En la tabla 2.3 se muestran los resultados obtenidos con los cuatro corpus (hay dos versiones del corpus Eindhoven utilizando el conjunto de etiquetas original, el Wotan, y una versión simplificada del mismo, el WotanLite). Las siglas WPDV se utilizan para referenciar un sistema de votación llamado *weighted probability distribution voting* (se calculan las sumas ponderadas de probabilidades asociadas a cada clase por cada subconjunto de características del ejemplo a clasificar). *Memory based learning* (MBL) hace referencia a un algoritmo de clasificación basado en ejemplos y DecTrees se corresponde con los árboles de decisión. Por último, el sistema Maccent está basado en un modelo de máxima entropía. Con TagPair se obtiene un sistema a medio camino entre la votación y *stacking* ya que consiste en localizar los casos en los que un etiquetador selecciona la etiqueta t_1 y el otro selecciona t_2, estimándose las probabilidades de que en esta situación la etiqueta real sea t_X. En este trabajo resulta de gran interés también el análisis que se realiza de los resultados, así como los estudios sobre la influencia

del tamaño del corpus de entrenamiento y diversos aspectos de los datos, los conjuntos de etiquetas y los clasificadores empleados.

	LOB	WSJ	Wotan	WotanLite
Best Single Tagger	HMM	MXP	HMM	MXP
	97,55	96,88	92,06	95,56
Voting				
Majority	+0,21	+0,10	+0,45	+0,45
TotPrecision	+0,40	+0,19	+0,52	+0,58
TagPrecision	+0,27	+0,11	+0,45	+0,42
Precision-Recall	+0,39	+0,17	+0,44	+0,66
TagPair	+0,43	+0,23	+0,66	+0,72
Stacked Classifiers				
WPDV(Tags)	+0,51	+0,27	+0,80	+0,77
WPDV(Tags+Word)	+0,52	+0,29	+0,79	+0,78
WPDV(Tags+Context)	**+0,59**	**+0,35**	+0,97	**+0,86**
MBL(Tags)	+0,50	+0,26	+0,66	+0,74
MBL(Tags+Word)	+0,47	+0,24	+0,39	+0,74
MBL(Tags+Context)	+0,55	+0,23	+0,69	+0,75
DecTrees(Tags)	+0,46	+0,26	+0,57	+0,75
DecTrees(Tags+Context)	+0,48	+0,24	-	+0,70
Maccent(Tags)	+0,48	+0,22	+0,70	+0,73
Maccent(Tags+Word)	+0,47	+0,21	+0,57	+0,71
Maccent(Tags+Context)	+0,57	+0,22	**+1,19**	+0,81

Tabla 2.3: Resultados de [52]

Con una serie de experimentos con el *Stockholm-Umeå corpus* en sueco, [115] muestra como el método *stacking* genera un número menor de tipos de errores, lo que puede facilitar ciertas tareas de post-procesado para intentar eliminar todos los errores posibles manualmente tras un procesado automático. Para estos experimentos se utilizaron métodos de aprendizaje automático presentes en anteriores trabajos, como modelos de Markov (TnT y Granska), reglas de transformación (fnTBL), árboles de decisión (TreeTagger), máxima entropía (Mxpost) y modelos basados en ejemplos (Timbl). Además se añade el clasificador Stomp, que trabaja emparejando secuencias de palabras entre los datos de entrenamiento y los de test. En la tabla 2.4 vemos los resultados obtenidos por todos estos clasificadores individualmente, además de los datos recopilados sobre tiempos de entrenamiento y ejecución.

Sjöbergh crea diferentes conjuntos de clasificadores para investigar sobre la repercusión que tiene añadir nuevos clasificadores a un conjunto ya selec-

Tagger	Accuracy (%) (all words)	Accuracy (%) (unknown words)	Training time	Tagging time
Baseline	87.3	25.4	34 s	13 s
fnTBL	95.6	79.8	2 h	2 min
Granska Original	96.0	89.5	6 min	41 s
Granska	95.4	88.4	6 min	41 s
Mxpost	95.5	85.1	13 h	4 min
Stomp	93.8	63.3	0	2.5 min
Timbl	94.7	79.1	8.5 min	1 h
TnT	95.9	88.5	20 s	8 s
TreeTagger	95.1	77.5	35 s	5 s

Tabla 2.4: Resultados de los clasificadores empleados en [115]

cionado para emplear un método de votación, así como el efecto que produce el número total de clasificadores en los resultados obtenidos tras combinar sus salidas. De ahí extrae algunas conclusiones interesantes, indicando que generalmente es beneficioso aumentar el número de sistemas participantes, aunque no siempre es así, ya que si se agrega un sistema muy eficaz, habiendo ya otro sistema muy parecido en el conjunto, los resultados globales pueden verse mermados (TnT y Granska son similares y provocan este efecto en sus experimentos). Agregar clasificadores que no son demasiado buenos también queda reflejado en sus resultados que puede ser ventajoso en el sistema final (introducir Stomp en el esquema de ejecución puede dar mejores resultados que introduciendo fnTBL, aunque este último sea mucho mejor en términos de precisión). En la tabla 2.5 aparece el mejor resultado obtenido mediante votación tras experimentar con diversos conjuntos de clasificadores. También aparece la mejora conseguida mediante *stacking* utilizando dos clasificadores diferentes para el segundo nivel de aprendizaje. La versión con Relief-F (un clasificador basado en la selección de características) alcanza el mejor resultado, aunque tuvo que probarse con un conjunto de datos de entrenamiento reducido ya que su consumo de tiempo hacía impracticable ejecutarlo sobre el corpus completo. Un intento por emplear votación ponderada, así como algunos experimentos enfocados a crear un *stacking* centrado únicamente en los casos más difíciles, no consiguieron mejorar las cotas reflejadas en la tabla.

Centrándose en explorar diversos métodos de votación, [51] incluye el uso de algoritmos genéticos para seleccionar los pesos que deben asociarse a los diferentes clasificadores que forman parte del proceso. El trabajo se evalúa haciendo uso del corpus *SUSANNE*, considerado por los autores como

Tagger	Accuracy (all words)
Best tagger	96.0
Best voting	96.6
Timbl (stacking)	96.7
Relief-F (stacking)	96.8

Tabla 2.5: Resultados de la combinación de clasificadores en [115]

suficientemente diferente al Penn Treebank para poder validar los resultados obtenidos en anteriores trabajos con esta nueva colección de documentos. También se propone una métrica para establecer el nivel de confianza que se puede depositar en las salidas que ofrece un sistema de combinación de clasificadores en los casos en que no se puedan obtener valores de precisión sobre un conjunto de datos disponible. Estas métricas están basadas en las coincidencias entre los sistemas participantes y hacen creer en una relación existente entre la precisión y el acuerdo entre etiquetadores, lo que permitiría conocer estos valores de confianza para textos sin etiquetar.

Al igual que hiciera Sjöbergh con el sueco, en [77] se utilizan técnicas de combinación de sistemas para el etiquetado POS de textos en islandés. Concretamente hace uso del corpus *Icelandic Frequency Dictionary* (IFD) y experimenta con varios sistemas de clasificación típicos, además de con Ice, un clasificador basado en reglas lingüísticas. También investiga no sólo la combinación, sino la integración de clasificadores, entendiendo como integración la capacidad de un etiquetador de utilizar una característica o funcionalidad de otro. Por ejemplo se utilizó Ice para crear el etiquetado inicial empleado por TBL para desarrollar su algoritmo o para completar el lexicon que más tarde adoptaría TnT durante su ejecución. También integraron su reimplementación de TnT con Ice, obteniendo en todos los casos mencionados mejoras que van desde el 3 % al 9 % de reducción de error respecto a las versiones originales. En cuanto a la combinación, los experimentos se limitaron a la votación y a reglas motivadas lingüísticamente como métodos aplicados a los clasificadores básicos. Entre los resultados destaca un 20.7 % de reducción de error respecto al mejor participante logrado por votación por mayoría.

En [70] se afronta la tarea una vez más con un idioma menos prolífico que el inglés en cuanto a disponibilidad de recursos como es el idioma polaco. Aquí de nuevo se emplean métodos de votación sobre diversos etiquetadores ya conocidos evaluando sobre el corpus *Frequency Dictionary of Contemporary Polish* (FDCP) ligeramente modificado. También se proponen dos métodos basados en la división por temática y por atributos que

no dieron mejores resultados que los métodos tradicionales de votación. Estas divisiones fueron posibles ya que el corpus empleado tiene separados sus documentos en diferentes temáticas y además se compone de un conjunto de etiquetas suficientemente grande como para que sea posible la división por atributos, donde se crean réplicas del corpus especializadas en un tipo concreto de etiquetas.

2.2.2. Desambiguación de Significados

La desambiguación de significados (WSD - *Word Sense Disambiguation*) ha sido otra de las tareas del PLN que más interés ha despertado entre los que investigan con métodos de combinación de clasificadores. Ya en [98] sorprenden las mejoras obtenidas con un sistema de combinación bastante sencillo, en el que básicamente se varía el tamaño de las ventanas de contexto para obtener la variabilidad necesaria partiendo de un clasificador basado en el algoritmo Naïve Bayes. Los experimentos se llevaron a cabo con los sustantivos *line* e *interest* y entre los sistemas de votación probados, los más sencillos son los que finalmente dieron mejores resultados. En la tabla 2.6 podemos comprobar como el primer clasificador que aparece, que es precisamente el producto de las investigaciones desarrolladas en este trabajo, sobrepasa a otros sistemas anteriores. Este clasificador se compone de nueve versiones diferentes del clasificador bayesiano, a las que se le aplica el algoritmo de votación por mayoría, después de que un intento de aplicar diferentes pesos a los votos no consiguiese mejorar los resultados obtenidos por este sencillo método.

En ese mismo año, [63] reunió todos los sistemas participantes en el ejercicio Senseval, evento que gira en torno a la tarea del WSD, realizando experimentos sobre la aplicación de métodos de votación para combinar todos estos clasificadores. Se realizaron pruebas sobre distintos conjuntos de sistemas, que van desde el que considera todos los participantes, hasta los que sólo tienen en cuenta los que cumplen alguna condición (e.g. son supervisados o están entre los mejores clasificados en la competición). A todos ellos se les aplicaron tres métodos de votación:

- *UNANIMOUS*, donde se asigna una etiqueta sólo si todos los sistemas coinciden en asignarle la misma etiqueta (o se abstienen).
- *ABSOLUTE MAJORITY*, que asigna una etiqueta si esta recibe más votos que todas las demás juntas.
- *WINNER*, que se queda con la etiqueta que simplemente recibe mayor número de votos

interest

	accuracy	method
Naive Bayesian Ensemble	89 %	ensemble of 9
Ng & Lee, 1996	87 %	nearest neighbor
Bruce & Wiebe, 1994	78 %	model selection
Pedersen & Bruce, 1997	78 %	decision tree
Pedersen & Bruce, 1997	74 %	naive bayes

line

	accuracy	method
Naive Bayesian Ensemble	88 %	ensemble of 9
Towell & Voorhess, 1998	87 %	neural net
Leacock, Chodorow, & Miller, 1998	84 %	naive bayes
Leacock, Towell, & Voorhees, 1993	76 %	neural net
Leacock, Towell, & Voorhees, 1993	72 %	content vector
Leacock, Towell, & Voorhees, 1993	71 %	naive bayes
Mooney, 1996	72 %	naive bayes
Mooney, 1996	71 %	perceptron

Tabla 2.6: Comparativa de resultados en [98]

El autor muestra su satisfacción al comprobar que en todos los casos se mejoran los resultados alcanzados individualmente por los sistemas participantes, y vislumbra una gran utilidad en el empleo de estas técnicas combinatorias para diversos cometidos, entre los que se puede encontrar la creación de recursos lingüísticos de alta calidad que puedan ser utilizados para la fase de evaluación en eventos como Senseval, que congregó en este caso a un gran número de clasificadores permitiendo la evolución de los sistemas destinados a resolver una tarea compleja del PLN como es el WSD.

Además de completar un estudio bibliográfico bastante exhaustivo en torno a esta tarea, [116] presenta una estructura mucho más compleja, como podemos ver en la figura 2.3, en la que se utilizan tres tipos diferentes de módulos. El primer tipo son los filtros, que descartan significados que se consideran suficientemente improbables en el contexto actual. En segundo lugar están los etiquetadores parciales, que son capaces de aportar información acerca del sentido correcto que se debería asignar, aunque son incapaces de asegurarlo con un nivel suficiente de confianza. Por último se necesita un último tipo de módulo que se denomina extractor de características, el cual se encarga de representar el contexto adecuado de las palabras para que su significado pueda ser desambiguado. En el esquema utilizado existe un único filtro (filtro POS), tres etiquetadores parciales (*simulated annealing*, *subject codes* y *selectional restrictions*) y un único extractor de características (*collocation extractor*). A la hora de combinar estos módulos, se utiliza un método basado en ejemplos (TiMBL) que una vez más obtiene mejores resultados que estos sistemas ejecutados individualmente.

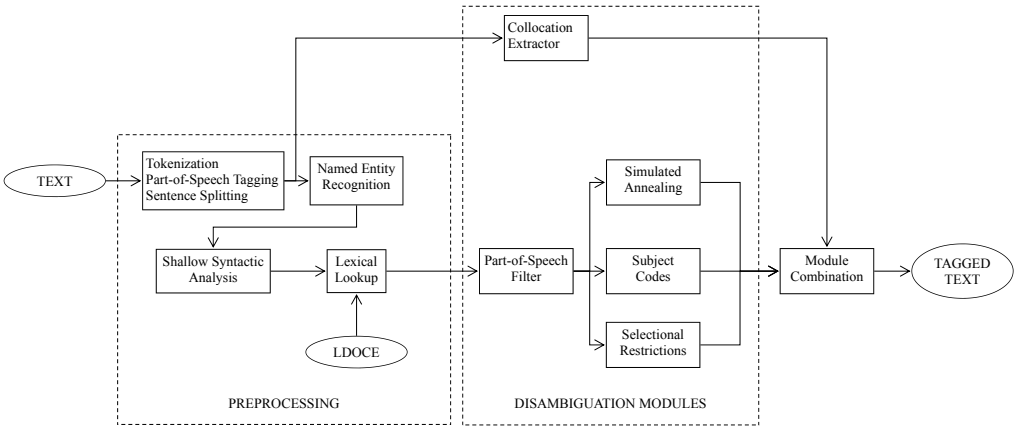

Figura 2.3: Arquitectura del sistema presentado en [116]

En [42] se afronta con determinación el tema de la combinación de sis-

temas aplicada al WSD, desarrollando el trabajo basándose en el sistema que los autores presentaron en nombre de la Johns Hopkins University al certamen Senseval-2. Este sistema combinó cuatro clasificadores (Naïve Bayes, Coseno, *Bag-of-word* Naïve Bayes y listas de decision no jerárquicas) obteniendo buenos resultados en la citada competición. En este trabajo se examina un conjunto diverso de clasificadores entre los que se encuentran métodos muy populares y otros más novedosos, concretamente se utilizan los ya mencionados del sistema original, además de BayesRatio, algoritmos basados en reglas de transformación y clasificadores MMVC (Maximum Variance Correction) entre otros, pudiéndose comprobar en la figura 2.4 los resultados que aportan al evaluarse individualmente con los datos en cuatro idiomas utilizados en Senseval-2.

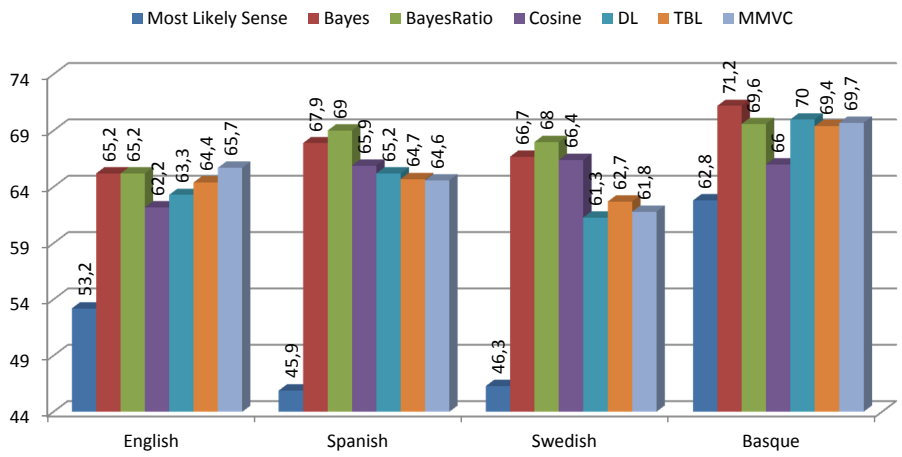

Figura 2.4: Rendimiento de los clasificadores utilizados en [42]

Utilizando votación por mayoría (aquí llamada *count-based voting* o CBV) se mejoran en torno a un 1 % los resultados en todos los idiomas, aunque estos resultados mejoran aún más si se tienen en cuenta las distribuciones de probabilidad de los clasificadores individuales mediante interpolación (lo que los autores llamaron *enhanced CBV*) o desarrollando sistemas más complejos de votación. En la figura 2.5 vemos las mejoras logradas para el idioma inglés respecto al mejor resultado individual, manteniéndose esa dinámica para el resto de idiomas. Poco después, en [44], el autor añade técnicas de *stacking* a los experimentos realizados anteriormente, lo que arroja ligeras mejoras que sirven para colocar estas cifras por encima de los mejores resultados conocidos hasta la fecha.

En [64] se muestra como la combinación de sistemas que alcanzan resul-

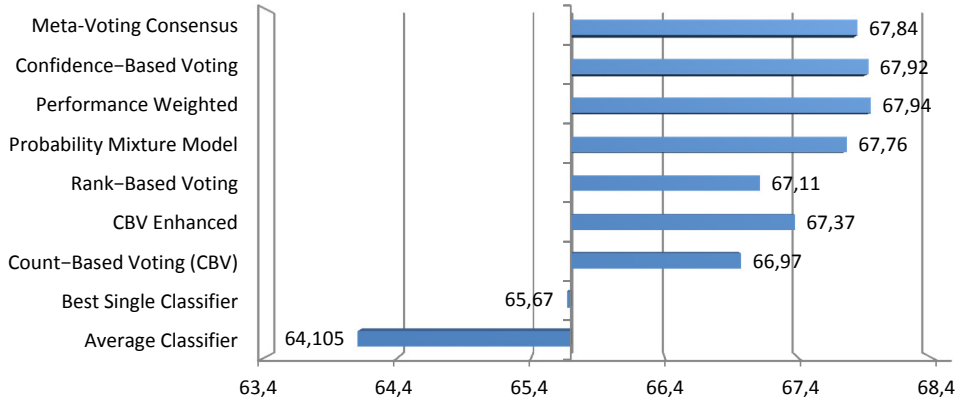

Figura 2.5: Mejoras de los métodos de combinación para el inglés en [42]

tados mediocres puede lograr medidas de precisión a la altura de los mejores sistemas publicados para esta tarea. En este caso en particular, se recopilan 23 trabajos de alumnos matriculados en el curso de PLN ofrecido por la Universidad de Stanford durante la primavera de 2000, generando un conjunto de clasificadores muy heterogéneo. La combinación se realiza en varios niveles, escogiendo entre los distintos subconjuntos posibles de clasificadores del primer nivel, así como eligiendo el clasificador de segundo nivel a utilizar entre tres posibilidades: votación por mayoría, votación ponderada o máxima entropía (ver figura 2.6). La selección en los distintos niveles se realiza estableciendo un ranking de las posibilidades mediante la evaluación de una porción del corpus reservada a tal efecto.

Como presentación de un nuevo método de combinación, [131] genera diferentes clasificadores utilizando el algoritmo de Naïve Bayes y variando el tamaño de la ventana contextual utilizada. De esta forma, se genera una trayectoria de selección de significados asociada a la secuencia de contextos utilizados para generar los clasificadores. Finalmente, se emplea el método de los vecinos más cercanos con estas trayectorias para seleccionar el significado que debemos asignar en los textos de prueba. Los resultados obtenidos con datos extraídos del *Chinese People Daily* mejoran los alcanzados por otros algoritmos implementados por los autores para su comparación, mostrando una gran eficiencia y robustez.

El trabajo [73] por su parte, adapta la Teoría de Dempster-Shafer a la combinación ponderada de clasificadores de WSD, obtenidos mediante las distintas representaciones posibles del contexto, a través de las características utilizadas para generar el modelo. Tras evaluar con los datos disponibles en la

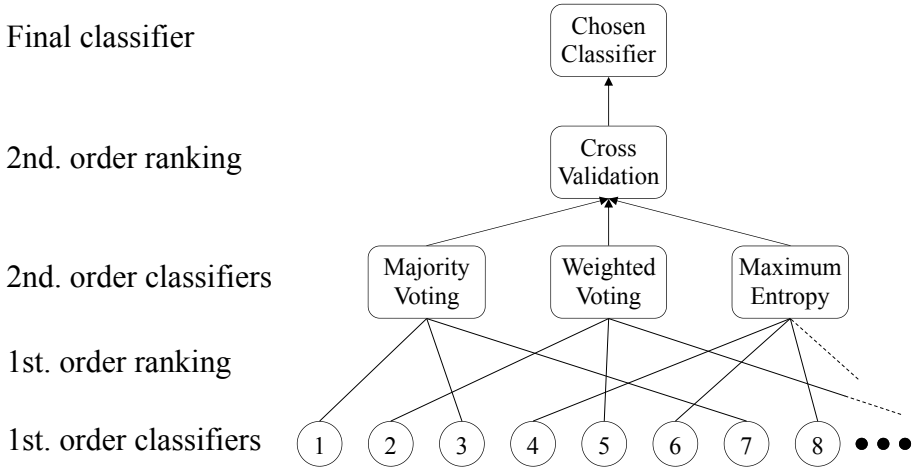

Figura 2.6: Esquema del sistema presentado en [64]

página personal de Pedersen[1], los cuales fueron utilizados en varios trabajos anteriores, los resultados mejoran las cifras conocidas hasta la fecha, dejando muestras del beneficio que puede obtenerse aplicando técnicas de combinación provenientes de otras áreas.

Especial atención merece también el trabajo reciente de Cuong, en cuya Tesis Doctoral [28], enfocada hacia esta tarea, dedica gran parte de su esfuerzo a experimentar con diversos métodos de combinación para estudiar las posibilidades de mejora que ofrecen sobre sus resultados. Entre los métodos considerados destacamos el *Meta-Stacking*, que coincide con un esquema de *cascading* de dos niveles. Se diferencia del modelo de *stacking* habitual por utilizar como entradas para el segundo nivel de aprendizaje los resultados obtenidos por diferentes métodos de combinación aplicados sobre los clasificadores base, en lugar de usar directamente las etiquetas proporcionadas por éstos (ver figura 2.7). Las conclusiones que extrae de los resultados el autor se resumen en tres:

1. El *stacking* puede mejorar los clasificadores individuales.

2. Los resultados del *stacking* de dos niveles superan los obtenidos por el *stacking* de un sólo nivel.

3. A la hora de generar los clasificadores base, se obtienen mejores resultados de *stacking* empleando diferentes algoritmos de aprendizaje que generando diferentes conjuntos de características.

[1] `http://www.d.umn.edu/~tpederse/data.html`

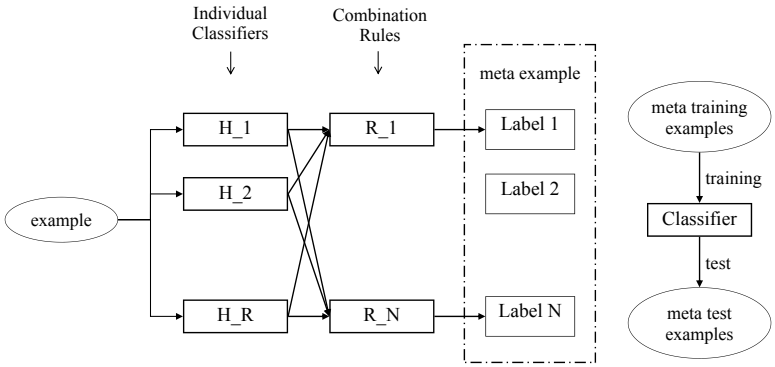

Figura 2.7: Modelo de Meta-*Stacking*.

2.2.3. Reconocimiento de Entidades

Al igual que hicieran en [42] y [44] con la tarea de WSD, Florian y sus colaboradores vuelven a confiar en la colaboración entre clasificadores para mejorar los resultados de una tarea importante del PLN como es el Reconocimiento de Entidades con Nombre (NER - *Named Entity Recognition*)[41]. En esta ocasión y empleando los recursos disponibles en la competición CoNLL-2002, se obtienen mejoras mediante la combinación de un sistema basado en reglas de transformación (fnTBL) y el sistema Snow (Sparse Network of Winnows), que genera una red en donde las etiquetas son representadas como funciones lineales compartiendo un espacio de características común. En este caso, la combinación se hace de forma escalonada, sirviendo el resultado de aplicar fnTBL como punto de partida para ejecutar Snow, el cual consigue mejorar alrededor de 2 puntos el valor $F_{\beta=1}$. También se hace uso de un algoritmo de *forward-backward* para optimizar el proceso global y obtener aún mejores resultados que los iniciales, alcanzando una mejora final de cinco puntos sobre los efectos de aplicar únicamente fnTBL.Aunque en dicho trabajo se hace referencia a este esquema como *stacking*, está más próximo al concepto de *cascading*, en donde la salida de un clasificador se utiliza como entrada al siguiente, en lugar de crear un modelo nuevo mediante un segundo nivel de aprendizaje sobre los clasificadores base.

En [125] se utilizan también los datos de la competición CoNLL-2002, aunque en este caso, el clasificador principal está basado en la versión extendida del algoritmo *adaboost* para el manejo de problemas con múltiples clases. El esquema desarrollado representa un *stacking* múltiple que, tal y como describen sus autores, se diferencia de sus predecesores en los siguientes aspectos:

1. Se emplean más de dos niveles de aprendizaje.

2. Se utiliza un solo algoritmo y se entrena un sólo clasificador en cada nivel.

3. Se utiliza la etiqueta que rodea al objetivo asignada por el clasificador del nivel inferior.

4. Se utilizan tanto las etiquetas anteriores como posteriores como características.

5. Se utilizan las etiquetas predichas en lugar de las correctas en la fase de entrenamiento.

Tras cuatro iteraciones de este algoritmo se alcanzan valores $F_{\beta=1}$ de 67,35 y 58,83 para los datos en español y holandés respectivamente (*esp.a* y *ned.a*).

En [127] se aplica *stacking* sobre el algoritmo de máxima entropía, empleando diferentes longitudes de los contextos anterior y posterior de la palabra que está siendo etiquetada. Utilizando listas de decisión (una lista ordenada de reglas de decisión) como algoritmo de clasificación del segundo nivel de aprendizaje, se consiguen sobrepasar los mejores resultados conocidos para esta tarea en japonés conseguidos por sistemas basados en máxima entropía. En este caso se utilizan los datos de IREX (*Information Retrieval and Extraction Exercise*), por lo que es difícil comparar con los trabajos mencionados anteriormente, aunque la mejora alcanzada con las técnicas de combinación es ostensible.

Por su parte [61] realiza un estudio sobre los sistemas de reconocimiento de entidades aplicados al idioma chino, dedicando buena parte del trabajo a la aplicación de técnicas combinatorias sobre cuatro clasificadores distintos (modelos de Markov, reglas de transformación, máxima entropía y *robust risk minimization*). La mejora alcanza una reducción del 10 % del error sufrido en la medida $F_{\beta=1}$ tal y como se aprecia en la tabla 2.7. Primero se experimentan con métodos de votación simples y también se ejecutan dos modelos (*Model 1* y *Model 2*) en los que se les permite a los clasificadores dar valores parciales a clasificaciones alternativas mediante la probabilidad $P_i(C|w, C_i)$. Por último, se prueban diversos conjuntos de características mediante *stacking* y se muestra también una cota superior representada por un oráculo, que selecciona la etiqueta correcta siempre que al menos un clasificador del primer nivel la haya asignado.

Este mismo grupo realiza nuevos experimentos en [43] para resolver esta tarea en el marco de CoNLL-2003, donde la competición vuelve a solicitar la participación de sistemas centrados en el reconocimiento de entidades,

	Precision	Recall	F-Measure
Best Classifier	84.53	78.64	81.48
Equal weight voting	85.2 ± 0.03	81.7 ± 0.02	83.3 ± 0.02
Weighted voting	86.04	80.63	83.24
Model 1	83.07	82.10	82.58
Model 2	86.3	77.55	81.69
RRM	89.01	79.61	84.05
RRM+flags	88.96	79.84	84.18
RRM+IOB1+IOB2	88.5	80.46	84.29
Oracle	91.6	92.3	91.95

Tabla 2.7: Resultados obtenidos en [61].

tal y como ocurriera en la anterior edición. El esquema presentado en [61] se ejecuta ahora sobre los datos aportados por la organización de la competición, constituidos por colecciones de documentos en inglés y alemán. En la tabla 2.8 vemos los resultados aportados en ambos trabajos sin la utilización de recursos adicionales como gazetteers, etc, donde se aprecia la consistencia de las mejoras en ambas colecciones de datos. Finalmente, se aportan también los resultados tras incorporar información adicional al sistema, proveniente de recursos externos en forma de diversos gazetteers y las salidas de dos reconocedores de entidades adicionales, alcanzando un valor de $F_{\beta=1} = 93{,}9$ (lo que supone una reducción del error del 21 %).

	Jing [61] (Chino)	Florian [43] Inglés
Best Classifier	81.48	89.94
Equal weight voting	83.3 ± 0.02	91.23 ± 0.08
Weighted voting	83.24	91.56
Model 1	82.58	90.9
Model 2	81.69	91.64
RRM	84.29	91.63

Tabla 2.8: Comparación de resultados entre [61] y [43].

En [135] se hace referencia una vez más al *stacking* como el entrenamiento de dos o más clasificadores secuencialmente, con cada clasificador incorporando los resultados del anterior de algún modo. Los clasificadores utilizados

en este caso son *adaboost* (su versión *MH* destinada a resolver problemas multiclase), *support vector machines* (SVM) y TBL (reglas de transformación). Los experimentos realizados se enmarcan de nuevo en la competición CoNLL-2003 y comienzan combinando dos ejecuciones de *adaboost*, en donde primero se identifican las entidades, y posteriormente se utiliza esta información para clasificar las entidades localizadas, lo cuál aporta una mejora de algo más de medio punto. A continuación se emplean los resultados del esquema que acabamos de describir para crear una buena situación de partida para aplicar TBL, constituyendo finalmente el mejor sistema base de los considerados en este trabajo. Este último clasificador se une a dos variantes de *adaboost* y a un modelo SVM para la aplicación de un método de votación simple, y finalmente, la salida de este esquema de votación se utiliza como entrada a *adaboost* creando el esquema *stacked-voted-stacked* definitivo. Con este sistema se consigue un valor $F_{\beta=1} = 89{,}51$ que supera en casi cinco puntos al mejor sistema de la edición anterior de esta competición y supone una reducción del error del 19.7 %. Especial mención merece el comentario de los autores que asegura que la mayoría de los sistemas de combinación probados no obtuvieron mejoras, debiéndose el éxito de su esquema final al análisis detallado del problema.

Otro modo de afrontar el problema reside en la combinación de diferentes estrategias de clasificación, tal y como se aprecia en [38], donde se combinan técnicas de aprendizaje automático con otras basadas en reglas para detectar entidades en un idioma complejo como es el chino. El resultado es un sistema híbrido que busca un equilibrio entre las virtudes y los defectos de ambos enfoques, además de hacer uso de técnicas de *bootstrapping* para generar nuevos patrones y lexicones que logran mejorar el número de entidades extraídas.

En [66] se muestran, además de la aplicación de diversos métodos de votación a un conjunto heterogéneo de clasificadores, la aportación del sistema resultante a diversas tareas finales, como la recuperación de información geográfica o el modelado conceptual. Utilizando modelos de Markov, máxima entropía y algoritmos basados en ejemplos, se obtiene un reconocedor de entidades llamado NERUA (*Named Entity Recognition system of the University of Alicante*), evaluándolo sobre los datos de las competiciones CoNLL-2002 (español y holandés) y HAREM-2005 (portugués). Los resultados lo sitúan en tercera posición (de 15 participantes) y quinta (de 13 participantes) respectivamente, destacando la escasa aportación de recursos externos al resultado final y la facilidad con la que se pudo adaptar el sistema al portugués, siendo inicialmente concebido para el español.

En [78] se hace uso de una técnica semiautomática, llamada *active learning* (también conocida como *selective sampling*), para seleccionar los ejemplos que deben ser anotados manualmente durante el funcionamiento del

sistema. La clasificación de organizaciones y nombres de personas se realiza en dos fases. El objetivo es que si en una primera pasada una cierta palabra de un nombre fue etiquetada con un alto grado de confianza, eso pueda influir en sucesivas ocurrencias de esa misma palabra en otras zonas del texto. Asimismo, si una palabra aparece unida a otra formando un nombre compuesto etiquetado en la primera pasada con mucha confianza, esto puede provocar que una ocurrencia de una sola de las palabras aparecidas de forma aislada se pueda reconocer como un nombre en el texto durante la segunda fase de ejecución. Los corpus utilizados para la evaluación consisten en artículos de periódicos griegos publicados entre 2000 y 2002 y artículos financieros también recopilados de la prensa griega y etiquetados manualmente.

Una de las aplicaciones relacionadas con la tarea del reconocimiento de entidades reside en el área de los textos biomédicos. El aumento del número de documentos que se generan periódicamente dentro de esta rama de conocimiento, hace imprescindible el uso de técnicas de procesamiento automático para filtrar y gestionar tal cantidad de información. El reconocimiento de entidades en este entorno se ha convertido en una tarea importante dentro de la extracción de información o la minería de datos biomédica, como nuevas ramas de la minería de datos y la búsqueda de conocimiento. Las entidades a identificar en este caso son proteinas, ADN, ARN, células, etc, cuyas características morfológicas varían bastante respecto a las entidades objeto de búsqueda en el NER convencional. El trabajo [130] constituye un buen ejemplo de la aplicación de las técnicas de combinación vistas hasta ahora a esta tarea de análisis de textos biomédicos, centrándose principalmente en el método *stacked generalization* que representa el concepto de *stacking*. Los clasificadores utilizados son *generalized winnow*, conditional random fields (CRF), SVM y máxima entropía, mientras que los corpus sobre los que se evalúa el sistema provienen de la tarea del JNLPBA-2004, extraídos del corpus GENIA. En paralelo se propone la utilización de *bagging* asociado al método *stacking* (acuñando el término *bag-stacking*) para la combinación de clasificadores homogéneos. En este esquema el método de votación por mayoría empleado por *bagging* se sustituye por un modelo de aprendizaje obteniendo una mezcla de ambas técnicas. Se obtienen tres variantes del *bag-stacking* según las características utilizadas en el meta-nivel durante la fase de entrenamiento:

- *Class-bag-stacking*. Se incluyen la clase correcta y las clases predichas por los clasificadores del primer nivel.
- *Class-attribute-bag-stacking*. Se incluyen la clase correcta y las clases predichas por los clasificadores del primer nivel, agregando además los vectores de características

- *Class-confidence-attribute-bag-stacking*. Se incluyen la clase correcta, las clases predichas por los clasificadores del primer nivel junto con sus valores de confianza asociados, y añadiendo finalmente los vectores de características.

Para la combinación de los clasificadores heterogéneos mencionados anteriormente también se crean dos versiones según las características utilizadas en el segundo nivel de aprendizaje:

- *Class-stacking*. Se incluyen la clase correcta y las clases predichas por los clasificadores del primer nivel.
- *Class-attribute-stacking*. Se incluyen la clase correcta y las clases predichas por los clasificadores del primer nivel, agregando además los vectores de características

Conscientes de la importancia que tiene en cualquier sistema que afronte esta tarea, el trabajo discute la aportación de diversos subconjuntos de características tras un análisis detallado de un gran número de características posibles de diversos tipos. En la tabla 2.9 podemos ver finalmente los resultados que superan ampliamente al mejor participante en JNLPBA, que obtuvo 69.4 % de precisión y 76 % de cobertura.

Experiment	P	R	F
Class-bag-stacking	70.21	74.56	72.32
Class-attribute-bag-stacking	71.68	76.10	73.82
Class-confidence-attribute-bag-stacking	72.61	76.86	74.67
Class-stacking	72.36	75.20	73.75
Class-attribute-stacking	75.57	79.68	77.57

Tabla 2.9: Resultados obtenidos en [130].

Otro trabajo interesante sobre el reconocimiento de entidades biomédicas es [34], donde se mezclan clasificadores con buena precisión con otros que alcanzan buenos porcentajes de cobertura. En él se hace uso de los algoritmos genéticos para medir la diversidad en base al rendimiento combinado obtenido con los datos de entrenamiento. La selección de los clasificadores se hace mediante votación dinámica, en donde se elige un subconjunto de predicciones (en lugar de clasificadores) en función de la fiabilidad que presenta cada clasificador respecto a cada una de las clases o etiquetas del problema. Los cromosomas se representan por medio de vectores binarios de dimensión $L*c$,

donde L es el número de clasificadores considerados y c es el número de etiquetas. Las c primeras posiciones se corresponden con las predicciones del primer clasificador para las c etiquetas posibles, y así sucesivamente, donde un 1 habilita al clasificador para votar si su predicción es la que corresponde a dicha posición del vector y un 0 lo bloquea. Los resultados avalan este esquema, ya que además de superar los resultados de los clasificadores base, se mejora también un sistema de votación estático en donde se bloquean los clasificadores en función de su redundancia para todos los elementos. También en [141], donde se combinan mediante votación un clasificador SVM y dos basados en modelos de Markov, se obtienen resultados muy positivos. Además se comprueba que el mal uso de ciertos recursos puede dar lugar al empeoramiento de los resultados, algo que ocurre en este caso al incluir un módulo basado en un diccionario abierto de gran tamaño.

2.2.4. Clasificación de Documentos

Otra de las tareas del PLN que mejor ha incorporado las técnicas de combinación de sistemas es la clasificación de documentos. La explosión demográfica de la web hizo patente la necesidad de clasificar de forma rápida y eficaz la inmensa cantidad de información que se genera, de forma que se haga accesible de manera eficiente y útil para colaborar en todo tipo de sistemas que incorporen alguna forma de análisis de textos. Una gran variedad de subtareas se han ido beneficiando de los avances logrados en la clasificación de documentos, como iremos viendo en los múltiples trabajos que comentamos a continuación.

Ya en 1996, [59] y [72] presentaban sistemas que combinaban varios algoritmos para mejorar la precisión con la que se lograba clasificar una colección de documentos. En [59] se afronta la tarea del filtrado de documentos, en la que se realiza una consulta que representa los documentos que son de interés para el usuario y el sistema selecciona aquellos que concuerdan con dicha consulta. En él se aplican técnicas como la de los vecinos más cercanos, Rocchio o *linear discriminant analysis* utilizando métodos muy sencillos de combinación como son el cálculo de promedios o la regresión logística (que aporta mejoras respecto al anterior). Utilizando el corpus Tipster y tras probar otros métodos más sofisticados sin éxito, los autores señalan la homogeneidad entre los sistemas utilizados como posible causa de la falta de mejoría en los resultados obtenidos. El trabajo [72] por su parte, se centra en una tarea circunscrita al dominio médico consistente en la asignación automática de códigos ICD9 (*International Statistical Classification of Diseases and Related Health Problems*) a los informes de altas de pacientes tras una hospitalización. Estos informes son generalmente dictados y representan una fuente de

recuperación de fondos para un centro médico, por lo que el desarrollo de esta tarea es de gran importancia para la comunidad médica. En este caso cada etiqueta ICD9 es una categoría que se puede asignar a un documento y una vez más se requiere un gran número de documentos etiquetados para generar modelos de inferencia y poder clasificar nuevos documentos. Utiliza un sistema probabilístico de recuperación de información llamado inquery y mediante combinaciones lineales (sumas ponderadas) se combinan los métodos de vecinos más cercanos, *relevance feedback* y un clasificador bayesiano. En todos los casos se mejoran los resultados obtenidos por los clasificadores ejecutados individualmente.

Otro trabajo que gira en torno al filtrado de documentos de forma similar a [59] es [140], en donde se experimenta con dos tipos de clasificadores combinados para salvar las diferencias de comportamiento de un clasificador adaptativo en el tiempo. Si bien Rocchio funciona bien en fases tempranas de la ejecución donde hay pocos datos de entrenamiento, la regresión logística puede dar mejores resultados en fases más avanzadas, por lo que se procede a combinar ambos métodos mediante inferencia Bayesiana. De esta forma se pretende encontrar el punto óptimo para combinar el sesgo, como medida de cuanto es capaz de aproximarse a la mejor solución el algoritmo de aprendizaje y la varianza, como medida de cómo de sensible es el algoritmo de aprendizaje a los datos de entrenamiento. Este trabajo también incorpora la interacción con el usuario como también ocurría en [72], ya que un sistema adaptativo de filtrado de información recibe el *feedback* del usuario periódicamente, indicándole si el documento seleccionado es válido o no, lo que aporta datos de entrenamiento para el proceso. Nuevamente, los resultados mejoran los clasificadores individuales y rivalizan con los mejores sistemas presentados en una competición internacional como es el TREC-9.

También la eliminación del correo no deseado forma parte del conjunto de subtareas englobadas en la clasificación de documentos. [105] la afronta aplicando *stacking* sobre dos clasificadores, uno bayesiano y otro basado en ejemplos, escogiendo un clasificador de este último tipo como "presidente", dentro del esquema de comité. Se experimentan dos variantes del *stacking* llamadas *hold-out stacking* y *cross-validation stacking*. Esta última, que obtiene resultados ligeramente superiores, sigue las indicaciones del *stacked generalization* original de Wolpert. La versión *hold-out stacking* no reentrena los clasificadores con todos los ejemplos y en lugar de generar un único clasificador de segundo nivel, genera tantos como particiones del conjunto de datos, calculando el promedio de todos ellos. La combinación resulta efectiva aún teniendo en cuenta el reducido número de clasificadores base empleado, lo que constituye una linea de trabajo futuro para sus autores. En [80] se combinan 53 filtros de correo no deseado participantes en la competición de

TREC-2005, empleando cálculos de promedios, votación y *stacking* mediante *SVM* y regresión logística. Si bien la votación ofrece muy buenos resultados, el método *stacking* por regresión es el que aporta un incremento mayor en las cifras con un gran volumen de datos de entrenamiento. También se ofrece un estudio en el que se demuestra que se puede seleccionar un subconjunto de filtros que mantienen un alto nivel de precisión sin la necesidad de ejecutar los 53 sistemas en paralelo.

En [71] se presenta como novedad un método capaz de proponer diferentes algoritmos para clasificar los documentos, dependiendo de sus características representadas por valores estadísticos. El algoritmo llamado MUDOF (*Meta-learning Using Document Feature characteristics*) detecta propiedades específicas de cada categoría, lo que le permite seleccionar el algoritmo más adecuado para cada una de ellas. Tras realizar pruebas con un conjunto de artículos de noticias de Reuters del año 1987, los resultados muestran mejoras respecto a los algoritmos de clasificación utilizados como componentes. Concretamente en la tabla 2.10 podemos ver los resultados obtenidos, evaluados mediante MBE (*micro-averaged recall and precision break-even point measure*) y ABE (*macro-averaged recall and precision break-even point measure*), donde Rocchio y WH son clasificadores lineales, KNN es el algoritmo de los vecinos más cercanos basado en instancias, SVM pertenece a la familia de los clasificadores lineales y tanto GISR como GISW provienen del enfoque de la generalización de instancias.

Algorithms	ABE	MUDOF+(%)	MBE	MUDOF+(%)
MUDOF	0.656	-	0.857	-
RO	0.578	13.495	0.776	10.438
WH	0.649	1.079	0.820	4.512
KNN	0.607	8.072	0.802	6.858
SVM	0.640	2.500	0.841	1.902
GISR	0.625	4.960	0.830	3.253
GISW	0.655	0.153	0.845	1.420

Tabla 2.10: Resultados obtenidos en [71].

Los indicadores de confianza forman la base de los trabajos [8] y [9], donde se usan estas variables, que proporcionan señales sobre el rendimiento de los clasificadores en diferentes situaciones, para crear un método probabilístico de combinación. En este artículo se toman ideas prestadas sobre señales sensibles al contexto de la comunidad investigadora de la visión artificial, mostrando su adaptabilidad al mundo de la clasificación de textos.

Se utilizan algoritmos diversos como naïve Bayes, clasificadores basados en unigramas, máquinas de vectores de soporte y árboles de decisión para comprobar las mejoras obtenidas al combinar las características habituales con los indicadores de fiabilidad aquí introducidos. Estos indicadores se agrupan en cuatro tipos que hacen referencia a los siguientes parámetros:

1. La cantidad de información presente en el documento original. Por ejemplo la longitud del documento, ya que hay algoritmos que funcionan mejor que otros en función del tamaño de los documentos a clasificar.

2. La información perdida o en desacuerdo entre la representación utilizada y el documento original. Las características eliminadas en selección de características es un ejemplo claro de este tipo de indicadores.

3. La sensibilidad de las decisiones respecto a cambios en las evidencias. La varianza de unigramas representa un ejemplo de esta clase de indicadores, ya que una varianza alta aumenta las probabilidades de que un cambio pequeño en el contenido del documento se traduzca en un cambio en la decisión final.

4. Algunas estadísticas básicas relacionadas con la votación. Sería el caso del porcentaje de acuerdo entre los componentes.

El sistema presentado recibe el nombre de Strive (*Stacked Reliability Indicator Variable Ensemble*) ya que, según sus autores, puede entenderse como una extensión del esquema de *stacking* habitual al que se añaden los indicadores de confianza en el segundo nivel de abstracción. Además de llevar a cabo experimentos con diferentes corpus y Strive usando 49 indicadores de los cuatro tipos descritos anteriormente, también se prueban diferentes métodos de combinación que son superados por éste, como son:

- *best by class*, consistente en seleccionar un clasificador para cada clase en función de los resultados obtenidos con un conjunto de validación

- votación por mayoría rompiendo los empates votando junto al método anterior, adoptando el nombre de *majority BBC*

- *stacking*, con árboles de decisión o SVM como meta-clasificadores, dando lugar a *Stack-D* y *Stack-S* respectivamente

Tanto [49] como [19],[99] y [100], se centran en la clasificación de páginas web, una de las aplicaciones que recientemente ha despertado más interés

dentro de la clasificación de documentos. En [49] se propone utilizar pequeños fragmentos de texto de las páginas que apuntan a la página que queremos clasificar en lugar de extraer características del texto presente en la propia página. De esta forma se obtiene una predicción por cada hipervínculo que apunta al objetivo, obteniéndose un conjunto de predicciones sobre el que podemos aplicar técnicas de combinación para clasificar la página deseada. En este trabajo se presentan diferentes formas de combinar las predicciones así como diversos métodos para identificar el texto relevante en cada página, llevando a cabo los experimentos sobre páginas relacionadas con departamentos de informática. Qi trabaja en la misma linea que Fürnkranz aunque realiza algunas modificaciones, empleando medias ponderadas en lugar de métodos de votación y ampliando el contexto utilizando páginas *parent*, *child*, *sibling* y *spouse* en lugar de sólo *parent* (ver figura 2.8). Calado utiliza los seis tipos de vecinos en dos pasos y genera una red Bayesiana como método de combinación de la información de los vínculos y las respuestas de clasificadores base como naïve Bayes, SVM y KNN. En [100] se discute en profundidad sobre los diferentes trabajos y las variaciones entre ellos, además de constituir un estudio exhaustivo sobre la clasificación de páginas web en general.

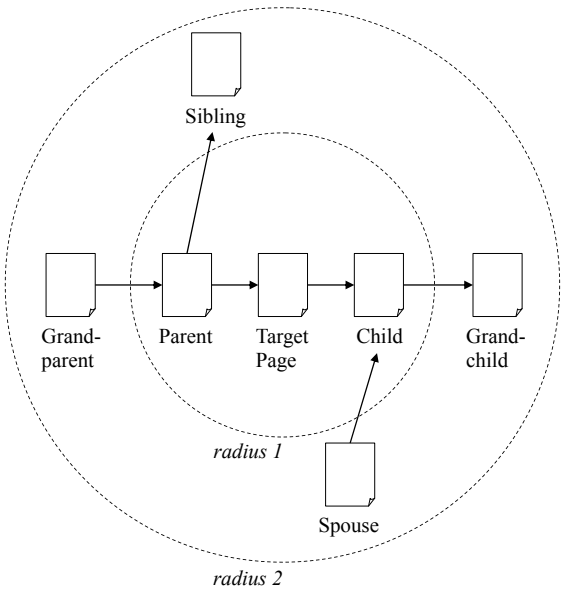

Figura 2.8: Vecinos de una página con un radio igual a dos.

En [112] se desarrolla una metodología para ajustar los parámetros de un clasificador o meta-clasificador a los objetivos de una aplicación. Para ello

se utilizan SVM y un método de separación de centroides, experimentando con la clasificación de documentos e investigando además, hasta qué punto es beneficioso dividir un conjunto de documentos para utilizar métodos combinatorios con las partes. Asimismo se desarrollan estimadores para predecir el error y las pérdidas de una determinada configuración de un metaclasificador. Esta dedicación a los aspectos más ingenieriles tiene como fin a largo plazo alcanzar un mejor entendimiento de cómo incorporar, adaptar y configurar métodos de aprendizaje a sistemas más inteligentes para la búsqueda y organización de la información. Las pruebas se realizan sobre una colección de *newsgroups* y la IMDB (*Internet Movie Database*).

En [57] se presenta un sistema de clasificación compuesto por un clasificador, encargado de dar una respuesta inicial de suficiente confianza para todas las categorías (llamado generalista), al que le sigue un conjunto de expertos encargados de corregir los errores cometidos por el clasificador inicial. Para conseguirlo, se generan grafos de confusión que indican las áreas sobre las que deben especializarse estos clasificadores expertos, de manera que puedan subsanarse los errores reordenando las clases propuestas inicialmente. El sistema recibe el nombre de EDGE (Error-Driven Generalist+Expert), y se compone de una primera parte en la que se ejecuta el 'generalista' y se crea un grafo de confusión (los nodos representan conceptos y los arcos la existencia de confusión entre los conceptos conectados) mientras que en la segunda fase se entrena un 'experto' para cada grupo de conceptos altamente conectados (sólo para las instancias pertenecientes a algún concepto en el grupo). La clasificación por tanto consiste en recopilar los votos del 'generalista' y de algunos de los expertos para mejorar los resultados. Los resultados prueban la validez de este enfoque, especialmente apto para colecciones con un número muy alto de categorías.

La aparición de nuevos algoritmos de clasificación también abre las puertas a nuevos sistemas de combinación, y este es el caso de [114], en donde se hace uso de los *relevance vector machines* (RVM) [119] aprovechando las ventajas que aportan frente a los habituales SVM. De hecho, su incapacidad para escalar a problemas grandes, principal inconveniente de los RVM, es lo que se combate mediante la creación de múltiples clasificadores sobre pequeñas porciones de datos para luego combinar mediante técnicas de votación.

2.2.5. Recuperación de Información

El concepto de combinar múltiples representaciones, tanto de consultas como de textos, así como el uso de diferentes técnicas de recuperación de información, ha sido objeto de estudio por parte de los investigadores desde hace muchos años. Montague [90] recoge en la tabla 2.11 los diversos términos

que se utilizan para referenciar esta tarea en función del tipo de documentos y las consultas que se realizan, aunque la tarea principal consiste en la detección de aquellos documentos que son susceptibles de interés para el usuario, una vez que éste ha proporcionado los términos que a su juicio mejor los representa.

DOCSET	QUERIES	TASK
Stable	Incoming	Doc. Retrieval Cross Language Retrieval Question Answering
Incoming	Stable	Routing or Filtering Categorizing
Stable	NONE	Clustering
Incoming	NONE	Topic Detection

Tabla 2.11: Subtareas de la Recuperación de Información.

En [7] se presenta una forma de combinar distintas formulaciones de los *querys* para lograr una mejora en los resultados de esta tarea. En este trabajo se hace referencia a cinco consultas distintas, formuladas por diez personas expertas en la búsqueda de documentos, sobre diez temas de interés, expresados de forma compleja y extensa en torno a una necesidad de información. En [6] sin embargo se combinan los resultados obtenidos por diferentes sistemas de recuperación, guiados cada uno por algoritmos diferentes. En este caso además, se centra la atención en la forma de combinar los resultados, o mejor dicho, las estimaciones de relevancia adjudicadas a cada documento, empleando combinaciones lineales (aunque se mencionan resultados positivos con métodos no lineales como las redes neuronales) y se evalúan positivamente los resultados con dos aplicaciones reales. Un aspecto interesante observado en [7] y corroborado en [6], es el hecho de que un método que no obtiene buenos resultados por sí mismo, puede mejorar los resultados de otros métodos al ser combinado con ellos, lo cuál representa uno de los pilares sobre los que se sustenta la investigación sobre la combinación de sistemas en general. El trabajo [110] se decanta por la combinación de diferentes paradigmas en lugar de combinar los resultados de diferentes consultas similares, tras realizar diferentes experimentos en ambas direcciones.

El término *metasearch* comienza a emplearse con asiduidad y aparece en trabajos como [91] y [90], donde se enfrentan las dos clases principales de métodos de votación provenientes de la teoría de elección social, *condorcet*

y *borda*, resultando ser mucho más productivo el primer método que el segundo. En [91] se detallan muchos conceptos como la posibilidad de trabajar con sistemas que aportan estimaciones de relevancia o solamente un ranking de documentos, las diferencias entre *external metasearch* e *internal metasearch* (ver figura 2.9) o las diferentes formas en que se pueden relacionar las colecciones de documentos que forman la entrada a cada componente del sistema:

- disjuntas, como en el caso de la recuperación de información distribuida,
- solapadas, como en el caso del *external metasearch*, o
- idénticas, como se supone que ocurre en el caso del *internal metasearch*.

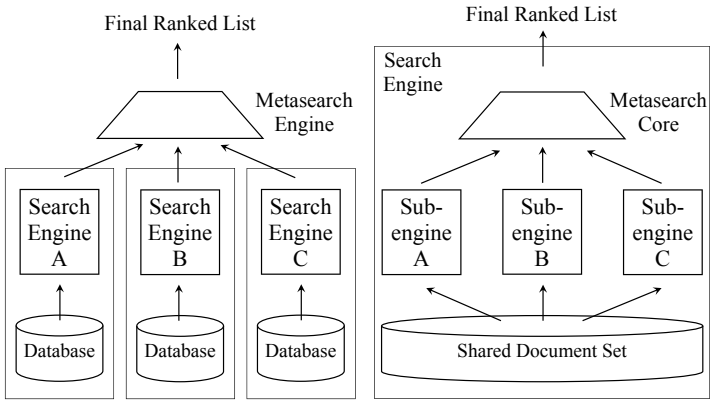

Figura 2.9: Esquema de *external metasearch* e *internal metasearch*.

El estudio que se realiza sobre la aplicación de los conceptos de la teoría de elección social merece especial atención, así como el uso de los grafos *condorcet*, en donde cada candidato tiene un vértice asociado y existe un arco de un candidato x a otro y en el caso de que x recibiese al menos tantos votos como y al enfrentarse uno contra el otro, es decir, si existen tantos votantes o más que colocan a x por encima de y frente a los que colocan y por encima de x. A estos grafos que contienen al menos un arco entre cada par de vértices se les llama *semi-completos* y el ganador del método *condorcet* será el candidato con $n-1$ arcos de salida. El hecho de que se puedan producir empates, obliga a desarrollar métodos costosos como hallar las componentes fuertemente conectadas, lo que elimina los ciclos y genera una jerarquía de grupos de candidatos empatados permitiendo su ordenación, aunque los autores proponen algoritmos más eficientes para resolver estos aspectos sin ni siquiera

tener que construir el grafo completo. Tras experimentar con diferentes sistemas enviados a varias ediciones del TREC (*Text REtrieval Conference*), el algoritmo de fusión mediante *condorcet* se muestra como la mejor opción de combinación en casi todos los casos probados.

Otra duda interesante es la que afronta [79], intentando dilucidar si compartir información en etapas intermedias de diferentes sistemas podría mejorar el rendimiento a nivel global. Concretamente, se unieron investigadores partícipes en el desarrollo de siete sistemas de recuperación de información para analizar el intercambio y combinación de información de las dos etapas de la técnica de *blind feedback*, en la que los resultados de una primera etapa de recuperación sirven para mejorar la eficacia de una segunda etapa final, entre los distintos sistemas. Los experimentos demostraron una vez más los beneficios de compartir información, en esta ocasión a nivel interno, entre diferentes sistemas de recuperación.

2.2.6. Análisis Sintáctico

El análisis sintáctico o *parsing* constituye una de las tareas más importantes del PLN debido al valor de la información que aporta a un gran número de aplicaciones finales. No obstante, su complejidad también hace de ella una de las tareas más desafiantes para los investigadores, representando asimismo una gran oportunidad para obtener beneficios de los sistemas de combinación existentes. En [55] se afronta este reto con éxito presentando varias formas de combinar las salidas de tres sistemas de análisis sintácticos existentes, sobrepasando los mejores resultados que habían sido publicados hasta la fecha para el corpus Penn Treebank. Empleando métodos de votación y naïve Bayes para la combinación, se llevan a cabo experimentos utilizando un primer enfoque llamado *parse hybridization*, en donde se mezclan subestructuras creadas por los diferentes analizadores para los constituyentes. Dos reglas guían el trabajo llevado a cabo en esta dirección: las hipótesis que son compartidas unánimemente por los participantes deben respetarse tras la combinación y la técnica de combinación no debe crear subestructuras para las que no existe ninguna evidencia que las justifique. Un segundo enfoque consiste en preservar la estructura completa del árbol, seleccionando para cada frase, cuál de los árboles sintácticos aportados por los analizadores debe utilizarse, algo que los autores llamaron *parser switching*. Ambos enfoques, así como las variantes presentadas para cada una, presentan mejores resultados que los sistemas originales ejecutados individualmente, tal y como apreciamos en la tabla 2.12.

Un año más tarde los mismos autores presentan otro trabajo en el que aplican las técnicas de *bagging* y *boosting*. En [56] se consigue una mejora

Reference / System	Precision	Recall	(P+R)/2	F
Average Individual Parser	87.61	87.83	87.72	87.72
Best Individual Parser	89.61	89.73	89.67	89.67
parser switching				
Parser Switching Oracle	93.78	93.87	93.82	93.82
Similarity Switching	90.04	90.81	90.43	90.43
Bayes Switching	90.78	90.70	90.74	90.74
parse hybridization				
Maximum Precision Oracle	100.00	95.91	97.95	97.91
Constituent Voting	92.42	90.10	91.26	91.25
Naive Bayes	92.42	90.10	91.26	91.25

Tabla 2.12: Resultados de [55].

equiparable a la que se obtendría duplicando el tamaño del corpus, además de facilitar la localización de lo que los autores consideran etiquetas inconsistentes a través de la revisión de los errores cometidos por las técnicas utilizadas. En cuanto a cuál de las dos técnicas responde mejor, hay que decir que *bagging* presentó mejores resultados, aunque la distribución generada por *boosting* como un efecto colateral fue lo que proporcionó la oportunidad de utilizar estas herramientas con el fin de detectar posibles inconsistencias en el corpus, presentándose incluso una herramienta semi-automática para llevar a cabo esta tarea.

También existen otras tareas relacionadas con el *parsing* que han sido objeto de experimentación con técnicas de combinación. Es el caso del *probabilistic partial parsing*, donde un analizador selecciona como salida las partes del árbol a las que se le atribuye un alto nivel de fiabilidad, permitiéndose así mantener un equilibrio entre precisión y cobertura. El trabajo [118] por ejemplo muestra cómo aplicar técnicas de toma de decisiones basadas en comités para esta tarea. En este trabajo se presenta un conjunto de funciones de pesos y una función de combinación con las que se experimenta haciendo uso de cinco analizadores sintácticos ejecutados sobre un corpus en japonés. Los resultados muestran cómo en muchas de las pruebas realizadas se superan los valores alcanzados por el mejor sistema individual.

Otra tarea ligada al *parsing* es el análisis de dependencias, en donde se trata de reconocer de forma automática la estructura de dependencias de cada frase. Estas estructuras están compuestas por relaciones binarias y simétricas entre pares de palabras, entre las que una queda subordinada a la otra en base a la relación que las une. Generalmente, se representan mediante grafos

dirigidos en los que los nodos se corresponden con las palabras y los arcos vienen etiquetados mediante las relaciones a las que representan. En [139] se combinan por primera vez analizadores de dependencias que usan diferentes estrategias, y se hace mediante diferentes técnicas, llegando a alcanzar una reducción del error del 13 %.

Por último, la identificación de sintagmas nominales simples (*base noun phrase identification*) es otra parte importante de esta linea de investigación. Consiste en hallar aquellos sintagmas nominales que no contienen otro sintagma nominal en su interior, y esta es una tarea más en la que la combinación de sistemas se ha mostrado efectiva, tal y como se aprecia en [106] y [107]. En el primer trabajo se afronta la tarea en el marco de la competición CoNLL-2000, empleando combinaciones internas con un algoritmo basado en ejemplos y estudiando la división de la tarea en dos fases, la localización de los sintagmas y la clasificación de los mismos. La diversidad se consigue creando diferentes representaciones internas de los sintagmas analizados. En el segundo trabajo, realizado por el mismo grupo de investigadores, se combinan siete algoritmos de aprendizaje diferentes mediante cinco variaciones del sistema de votación. Además se llevan a cabo experimentos combinando diferentes conjuntos de etiquetas con cada algoritmo o utilizando un segundo nivel de aprendizaje a modo de *stacking*, superándose los resultados obtenidos previamente con las mismas técnicas utilizadas en [106], ejecutadas sobre el corpus del Wall Street Journal.

2.2.7. Extracción de Información

La extracción de información hace referencia a la localización de fragmentos de texto que son de interés, en muchos casos con el objetivo de rellenar una plantilla predefinida o incluso para crear estas plantillas de datos estructurados. En [113] se combinan tres sistemas basados en diferentes técnicas, como son las máquinas de estados (concretamente los modelos de Markov), la inducción de reglas y el método *boosting* aplicado a la extracción de información. La combinación a través de la votación se muestra beneficiosa en la mayoría de los casos, aunque es *stacking* el que alcanza mejores resultados sobre diferentes dominios, tal y como demuestran los experimentos realizados sobre tres colecciones de datos distintas. Merece especial atención el esfuerzo realizado para adaptar la tarea a los requerimientos de los sistemas de combinación utilizados, al no tratarse de una tarea que pueda considerarse de forma natural como de clasificación. El concepto de *merged template* puede dar ideas sobre cómo adaptar otras tareas que en principio no parezcan susceptibles de ser afrontadas desde el punto de vista de la combinación de sistemas.

En [39] se combina ingeniería del conocimiento y aprendizaje automático para crear el sistema de extracción TEG (*Trainable Extraction Grammar*). Utiliza un corpus de entrenamiento etiquetado manualmente para mejorar un sistema basado en un conjunto pequeño de reglas de extracción expresadas mediante una gramática SCFG, obteniendo mejoras en la extracción de entidades y relaciones. Más tarde, los mismos autores presentan URES [103] que persigue evolucionar TEG de manera no supervisada, eliminando el esfuerzo manual que aún existe en este sistema. Siguiendo con la subtarea de extraer relaciones, [5] combina el método "tradicional" y el "abierto", generando un sistema híbrido. Los métodos tradicionales reciben como entrada ejemplos de las relaciones que deben buscar, basándose por lo tanto en la generación de un modelo y la clasificación de nuevos fragmentos de texto como instancias de dichas relaciones o no. Los métodos abiertos de extracción en cambio, no requieren información preliminar sobre las relaciones que estamos buscando, y simplemente se basan en patrones léxicos y sintácticos para reconocer la existencia de una relación binaria en el texto. Cuando el número de relaciones que son objeto de estudio es muy grande, los métodos abiertos son de gran interés, aunque si el número de relaciones decrece, la cobertura lograda también, independientemente de que logre mantenerse la precisión. Este trabajo emplea la técnica de *stacking* para lograr mejorar un 10 % los resultados del método tradicional, lo que demuestra la posibilidad de obtener beneficios de la combinación de ambos enfoques.

2.2.8. Traducción Automática

Otra tarea importante es la traducción automática o *machine translation*, donde también se ha experimentado exitosamente con técnicas de combinación, como por ejemplo en [45]. En este trabajo se combinan tres sistemas diferentes, estando uno de ellos basado en el conocimiento, otro en ejemplos y el último en un sistema de transferencia léxica reforzado con análisis morfológico, módulos de síntesis y diversas bases de datos. Los tres componentes van aportando traducciones de palabras o estructuras y el sistema va almacenando esta información en una tabla para después, mediante un algoritmo *chart-walk* que emplea programación dinámica, encontrar la mejor traducción posible. El método consiste básicamente de una matriz bidimensional donde el elemento (i, j) es la mejor puntuación que cubre la entrada desde la palabra i hasta la palabra j. En la implementación desarrollada por los autores, las mejores opciones del sistema se le presentan a un traductor humano para que elija la más conveniente, convirtiéndose en un sistema semi-automático de traducción. Otros trabajos han aplicado técnicas de combinación a esta tarea desde entonces, realizándose de dos formas principalmente tal y como

comenta [142], en serie o de forma competitiva (ver figura 2.10). En cualquier caso se ha demostrado que en muchos casos se consigue mejorar los resultados obtenidos por los sistemas implicados individualmente.

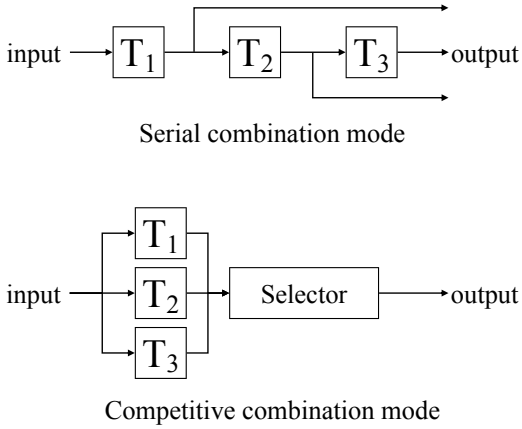

Figura 2.10: Formas de combinar sistemas de TA.

2.2.9. La Comprensión del Lenguaje Hablado

La comprensión del lenguaje hablado o *spoken language understanding* reúne las dos tendencias principales dentro del área de procesamiento de lenguajes, como son el reconocimiento de voz y el procesamiento de textos en lenguaje natural. Siendo un problema tremendamente complejo, se divide generalmente en múltiples subtareas especializadas en algún aspecto del proceso. En el apartado de reconocimiento de voz existen trabajos especializados en dicha subtarea que emplean métodos de combinación, como es el caso de [40]. Se trata de un trabajo en el que la combinación se afronta como un proceso aplicado tras el reconocimiento por varios sistemas independientes, aplicando técnicas de votación, redes de transición entre palabras (*word transition network* o WTN) y diversos métodos para alinear las diversas aportaciones de los componentes, demostrándose una vez más que se consiguen reducir las tasas de error respecto a los sistemas participantes.

Otro ejemplo es la clasificación de tareas (*task classification*), que consiste en identificar el tema sobre el que trata una consulta del usuario. En [136] se combinan varios clasificadores estadísticos para resolver este problema en el contexto de una aplicación de información al usuario sobre transporte público. Tras experimentar con naïve Bayes, modelos de n-gramas, y SVM como algoritmos de clasificación, se consiguen mejorar los resultados obtenidos de

forma aislada por cada uno de ellos, aún dándose la circunstancia de que los resultados de SVM superan ampliamente los obtenidos por el resto de algoritmos. Las pruebas con métodos de votación fueron mejoradas a su vez por técnicas de *stacking*, en las que se utiliza un segundo nivel de aprendizaje mediante árboles de decisión y máxima entropía, siendo este último el que consigue reducir el error en mayor proporción.

2.2.10. Etiquetado de Roles Semánticos

La atención que recae sobre el análisis semántico de textos ha ido incrementándose considerablemente en los últimos años, lográndose importantes mejoras en esta complicada tarea. La combinación también ha dado frutos en este escenario, como demuestra su presencia en los sistemas mejor considerados en la competición de CoNLL-2005, dedicada al SRL (*Semantic Role Labeling*). Sin embargo, destaca por su extenso estudio bibliográfico y experimentación empírica el trabajo de Surdeanu et al. [117], en el que se analizan diferentes modelos de inferencia, entre los cuales se encuentra uno basado en la satisfacción de restricciones y otros que hacen uso del meta-aprendizaje con clasificadores discriminativos mediante conjuntos de características. Tras los experimentos se concluye que la combinación debe llevarse a cabo a nivel de argumentos y no a nivel de soluciones completas. De hecho, esta es la característica fundamental que aleja este enfoque del *reranking*, en donde se selecciona el mejor sistema de entre todas las soluciones completas de las que se dispone. Los experimentos realizados en este trabajo, ponen en práctica tres estrategias de combinación diferentes:

- En la primera simplemente se recogen los valores probabilísticos asociados a los diferentes argumentos del texto y se aplican las restricciones sobre las estructuras semánticas a componer sobre la frase.

- En la segunda se aplica un clasificador previamente entrenado en forma de filtro sobre cada tipo de argumentos, decidiendo de esta manera si el candidato debe considerarse como un argumento final o no. A continuación los candidatos que superaron el paso anterior son combinados de manera que se respeten las restricciones, completando así un esquema en cascada.

- La tercera estrategia es global, y consiste en aplicar diversas funciones de ranking (una por cada tipo de argumento) sobre los candidatos, de manera que la estructura correcta quede situada en la parte alta de la lista de resultados.

Las dos últimas, que obtienen los mejores resultados, hacen uso del aprendizaje, asignando puntuaciones a los diferentes candidatos mediante diversos clasificadores para posteriormente seleccionar la mejor solución global, aunque los resultados muestran la complementariedad que ostentan al obtener mejores valores de precisión y cobertura respectivamente. Los autores también destacan la mejora respecto a los sistemas de *reranking* y los modelos individuales, que son superados en dos puntos aproximadamente tras llevar a cabo la combinación sobre los corpus WSJ y Brown.

2.2.11. Minería de Opiniones

Otra de las líneas de trabajo que está creciendo en interés por parte de los investigadores es el análisis de textos subjetivos. El objetivo en este caso es clasificar el contenido textual en función de si emite una opinión positiva o negativa en relación a un tema predefinido. Gracias a estas herramientas se ha abierto un amplio abanico de aplicaciones que principalmente sirven para extraer conocimiento de los grandes volúmenes de datos que pueden encontrarse en Internet, o bien se encuentran en manos de organizaciones de diversa índole. Las encuestas de opinión, los contenidos de blogs y las páginas de análisis de productos son objeto de aplicación de estas técnicas que pretenden automatizar la extracción de conocimiento relativo a la aprobación o rechazo de la población en lo referente al tema en el que están basados los datos, que pueden ser políticas de gobierno, estrategias de empresa o productos de cualquier tipo expuestos a la valoración de los usuarios. Existe un número cada vez mayor de sistemas de recomendación en la web, cuyo objetivo es facilitarle al usuario los servicios que más se acerquen a sus intereses, como puede ser el caso de un sistema de distribución de videos que intenta ofrecer las películas que mejor se adapten a nuestros gustos. La mayoría de estos sistemas cataloga los usuarios y analiza semejanzas entre ellos para poder establecer vínculos entre las opiniones que aportan y poder así predecir las opiniones que tendrían los clientes sobre un determinado producto. En la línea de este ejemplo encontramos el trabajo de Tsutsumi [126], que analiza documentos con opiniones de usuarios sobre películas para clasificarlos con polaridad positiva o negativa, de manera que se pueda disponer de una medida de calidad de dichas películas o bien sirvan para generar perfiles de usuarios. En este caso se combinan SVM, máxima entropía y cálculo de puntuación (analizando la polaridad de las palabras) mediante técnicas de votación o bien un segundo nivel de aprendizaje gestionado por SVM.

En la figura 2.11 podemos ver el esquema general del sistema presentado por Tsutsumi et al., consiguiendo una mejora ostensible de los resultados como se muestra en la tabla 2.13.

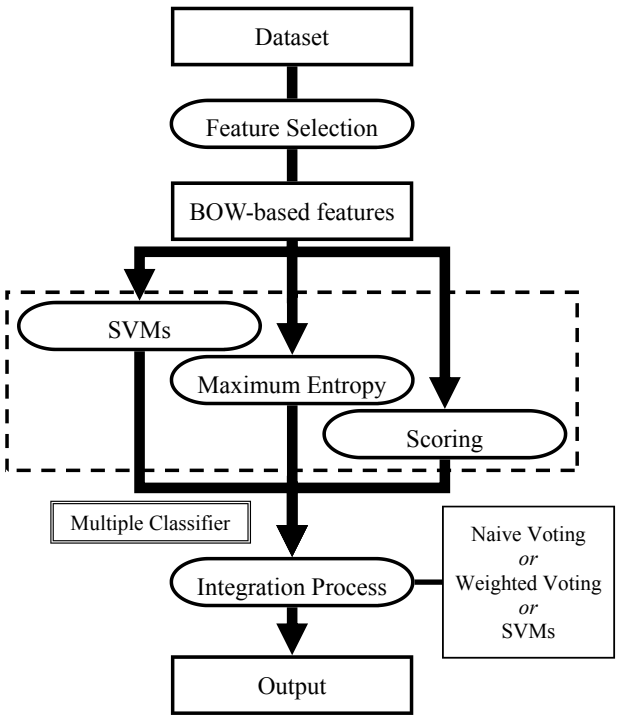

Figura 2.11: Esquema del sistema presentado en [126].

Method		Accuracy
Single	SVM	82.2 %
	ME	80.5 %
	Scoring	83.4 %
Multi	Naive Voting	85.8 %
	Weighted Voting	86.4 %
	SVM	87.1 %

Tabla 2.13: Resultados obtenidos en [126].

En [74] por su parte, se generan diferentes clasificadores utilizando de nuevo un corpus de opiniones sobre películas y distintos conjuntos de características para entrenar máquinas de vectores de soporte. A continuación, se emplean técnicas de selección para extraer un grupo de estos clasificadores y finalmente se realizan experimentos empleando diversos métodos de combinación mediante reglas basadas en fórmulas sencillas, entre los que destacan los resultados de la regla de la suma. Aún teniendo en cuenta la sencillez de los componentes empleados, queda patente una vez más la superioridad del esquema combinado frente a los clasificadores individuales.

2.3. Análisis Global

En este apartado, comentaremos la metodología que hemos seguido para poder hacer un juicio de valor sobre el éxito alcanzado en la aplicación de técnicas de combinación al PLN. Posteriormente, comentaremos algunas conclusiones que hemos extraído del análisis de los datos recopilados mediante dicha metodología.

2.3.1. Metodología

Para poder llevar a cabo este trabajo de investigación, se ha comenzado con la recopilación de artículos publicados en muy diversos foros, haciendo uso de las mejores herramientas y bases de datos como Scopus, Web of Knowledge o la ACM Digital Library. Tras una búsqueda exhaustiva de los artículos que aplican técnicas de combinación al PLN, se procede a seleccionar aquellos que consideramos más influyentes en esta línea de investigación y que aportan una mayor cantidad de información sobre técnicas, clasificadores, corpus de textos y resultados obtenidos, guiados por el contenido, además de por las citas y referencias que aparecen en dichos trabajos.

Finalizando esta tarea de recopilación de información, se ha procedido al estudio de estos trabajos, generando un recurso semiestructurado en forma de tabla, donde se rellenan una serie de campos de interés así como un apartado dedicado expresamente a dejar constancia de los aspectos más importantes que aparecen en los trabajos y que más adelante deben ser tenidos en cuenta a la hora de extraer conclusiones.

Los campos que forman parte de este recurso “intermedio” son muy diversos y se resumen en la siguiente lista:

- Datos referenciales como el año, publicación, autores, etc.

- Marcadores sobre números de citas, referencias que nos han llevado de un trabajo a otro, etc.
- Información sobre las tareas y subtareas del PLN que se han llevado a cabo.
- Relación de clasificadores, algoritmos de aprendizaje y técnicas de combinación que se han puesto en práctica y se han evaluado.
- Listado de corpus de textos que han servido para realizar las pruebas.
- Datos numéricos extraídos de los resultados aportados, centrados específicamente en los valores de mejora tras aplicar la combinación respecto a la ejecución aislada de los clasificadores base.
- Anotaciones sobre los aspectos más relevantes del trabajo realizado.

Esta metodología ha dado como resultado una base de datos bastante dispersa en la que aparecen múltiples variantes de cada uno de los campos mencionados anteriormente. Si bien esto dificulta la posibilidad de extraer conclusiones en términos absolutos para una tarea o una técnica de combinación concreta, creemos que por otro lado permite obtener una visión global de lo que la combinación de clasificadores está aportando al mundo del PLN en general y lo que creemos que puede llegar a suponer en un futuro, algo que trataremos de exponer a continuación.

2.3.2. Conclusiones

Lo primero que habría que destacar, una vez contemplado el efecto que ha producido la incorporación de las técnicas de combinación de sistemas al PLN, es sin duda el alto grado de mejora obtenido a lo largo de tantas y tan variadas tareas enmarcadas dentro de esta área de conocimiento. Desde la década de los noventa, en donde predominaban los trabajos referentes a la recuperación de información combinando diferentes fuentes de búsqueda (por ejemplo en [6]) o distintas representaciones de los *querys* (como en [7]), el número de trabajos ha seguido creciendo hasta nuestros días, adaptándose a otras muchas tareas como hemos visto en este capítulo.

Tal y como vemos en la figura 2.12, es a principios de esta última década cuando se produce un incremento más llamativo en el número de publicaciones, quizás provocado por los primeros estudios exhaustivos enfocados hacia tareas de etiquetado secuencial muy populares, en las que se encuentran trabajando un gran número de investigadores y que constituyen el objetivo de

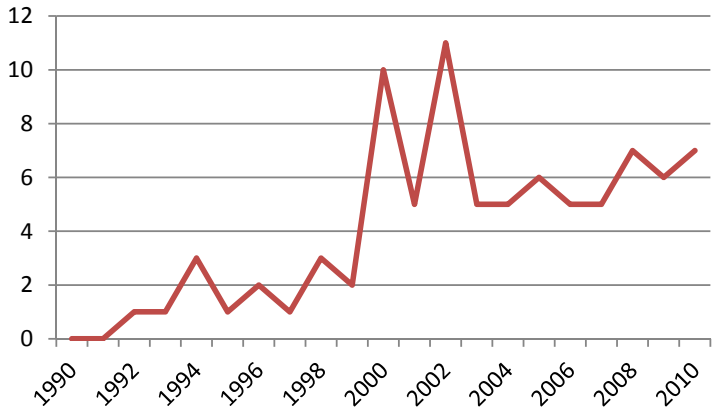

Figura 2.12: Publicaciones seleccionadas por año.

diversas competiciones internacionales como las de CoNLL. Podemos considerar en esta línea los trabajos de [53] y [18], que sin duda se encuentran entre las causas del aumento de interés comentado. La tarea más prolífica según nuestro estudio sigue siendo la clasificación de documentos, quizás por la menor complejidad existente a la hora de combinar los resultados de diversas herramientas y también por la abundante bibliografía existente en esta área.

El amplio abanico de escenarios sobre los que se han ido evaluando las técnicas de combinación en las diversas tareas, muestra sin duda alguna su gran capacidad de extraer conocimiento de diversas fuentes, y los resultados muestran como se consiguen mejoras empleando un amplio rango de clasificadores, técnicas de combinación y corpus. Las colecciones de datos son un claro ejemplo de ello, ya que son muchos los corpus diferentes utilizados en los trabajos que obtienen mejoras, y teniendo en cuenta las diferencias existentes en cuanto a calidad, tamaño y dominio, podemos considerarlo un indicio importante sobre la aplicabilidad de este enfoque. En favor de una mejor situación para comparar resultados, quizás sería conveniente tener un amplio número de trabajos realizados mediante la utilización de un mismo corpus, y aunque esto no ocurre en gran medida como acabamos de mencionar, bien es cierto que existen colecciones de datos que destacan por una mayor utilización por parte de los investigadores. Entre ellos se encuentran el Wall Street Journal, Reuters o los corpus ligados a las diversas competiciones periódicas, como TREC o CoNLL. Enfocar el trabajo hacia la combinación de sistemas puede comenzar por la decisión de seleccionar un conjunto de datos que permita compararnos con un mayor número de experimentos o abrir

la posibilidad de aplicación sobre una nueva tarea, idioma o dominio en el que podamos estar interesados. En cualquier caso, existen razones suficientes para ser optimistas tomando cualquiera de las dos opciones.

En cuanto a las técnicas empleadas, contemplamos una distribución menos equilibrada, destacándose claramente las técnicas de votación y de *stacking* al resto de posibilidades, aunque también hay que tener en cuenta la gran variedad de algoritmos que pueden emplearse a nivel interno por *stacking* y la existencia de múltiples métodos de elección de candidatos dentro de las técnicas de votación. En este apartado, cobra importancia el conocimiento sobre la tarea y los datos manejados, ya que tal y como comentan los autores en [135], el éxito puede depender de la correcta selección y utilización de los clasificadores y las técnicas de combinación existentes.

Por otro lado, los clasificadores vuelven a componer una larga lista de posibilidades, ya que cualquier algoritmo de clasificación es susceptible de ser incluido como componente de un sistema de combinación. La proliferación de implementaciones publicadas de los diversos métodos de clasificación existentes que se ha venido experimentando en los últimos años, ha facilitado una gran heterogeneidad en los trabajos publicados, aunque los más utilizados son los modelos de Markov, naïve Bayes, SVM y máxima entropía, junto con métodos basados en ejemplos y en reglas de transformación. Nuevamente, cada uno tiene sus propias características, siendo necesario conocer de antemano para lograr diversificar el espacio de búsqueda y lograr maximizar las capacidades de los métodos de combinación.

En la figura 2.13 se aprecia el reparto en cuanto a las tareas que han recibido más atención en los trabajos seleccionados. En cuanto a las técnicas de clasificación y de combinación apreciamos un uso dispar, ya que en lo referente a los algoritmos de clasificación, si bien hay algunos métodos que destacan ligeramente del resto, existe un mayor equilibrio en cuanto a la frecuencia de uso. En los métodos de combinación sin embargo, los métodos de votación y *stacking* acaparan la mayor parte de los trabajos, dejando entrever una posible falta de experimentación con el resto de métodos que hemos comentado y que podrían ofrecer mejoras en múltiples tareas de interés para el ámbito del PLN.

En cuanto a los resultados presentados en los trabajos, resulta difícil establecer comparaciones debido a la gran variedad de métodos de clasificación y combinación, además de las tareas y los datos, contabilizándose cerca de cincuenta corpus distintos. Aún así hemos confeccionado la tabla 2.14 donde se reflejan las mejoras mínima, máxima y media alcanzadas en los trabajos que aplican combinación de sistemas a alguna tarea del PLN. Además, mostramos estos mismos valores para los casos en los que se ha aplicado un método de votación o *stacking* (las técnicas de combinación más utilizadas

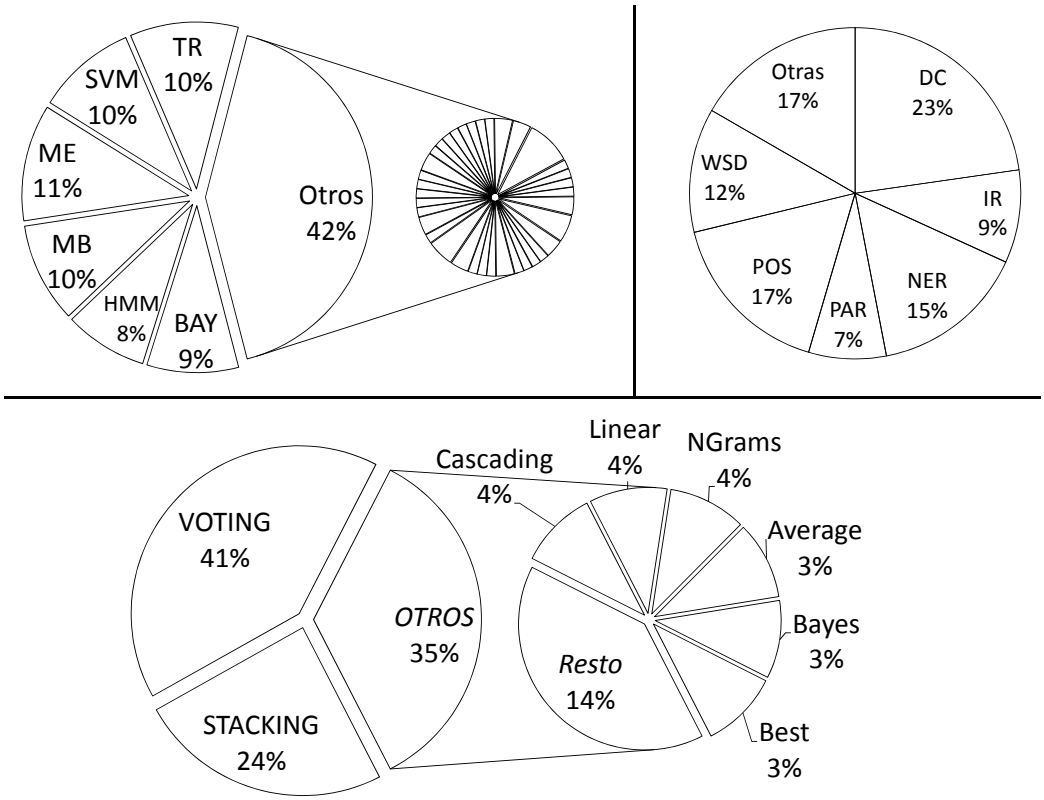

Figura 2.13: Clasificadores, tareas y métodos de combinación.

según se aprecia en la figura 2.13) así como para el grupo formado por el resto de técnicas aquí comentadas. Lejos de querer darle importancia a los valores concretos, pretendemos interpretar estos resultados como claros indicios sobre las capacidades de mejora que pueden lograrse en muchas tareas del PLN, así como mostrar la valía de muchos métodos ya existentes que gozan de mucha menos popularidad que los métodos de votación y *stacking*. Respecto a las tareas, podemos apreciar diferencias significativas entre los valores de mejora, lo cual es fruto, desde nuestro punto de vista, de la dificultad de la tarea y del margen de mejora de los clasificadores individuales. Resulta más difícil mejorar resultados en tareas como el etiquetado POS, donde partimos de resultados muy buenos antes de aplicar la combinación.

	mínimo	máximo	media
DC	0,01	8,10	2,02
NER	1,30	6,41	3,52
PAR	0,03	2,30	1,12
POS	-0,58	1,75	0,75
WSD	1,70	7,00	3,34
Voting	-0,58	6,20	1,53
Stacking	0,03	9,02	2,45
Otros	0,02	8,10	2,62

Tabla 2.14: Resumen de los resultados de las referencias seleccionadas.

Capítulo 3

Trabajos Iniciales

Antes de decidirnos por estudiar en profundidad las características de los diferentes métodos existentes para la combinación de clasificadores, llevamos a cabo varios trabajos en los que se aplicaban algunas de las técnicas más conocidas. Estos trabajos giraban en torno al reconocimiento de entidades y posteriormente en torno a la generación de recursos lingüísticos. Los resultados fueron muy positivos en cuanto a la primera tarea y bastante dispares en la segunda, aunque el potencial de técnicas de combinación como *stacking* quedaba fuera de toda duda, por lo que decidimos que era conveniente estudiar con más detenimiento ésta y otras técnicas destinadas a mejorar los resultados de la clasificación a través de la combinación. A continuación describiremos las tareas afrontadas en esta primera etapa experimental junto con los resultados obtenidos.

3.1. El Reconocimiento de Entidades

Tras explicar brevemente algunas características de la tarea NER, así como algunas aproximaciones iniciales, veremos cómo se combinaron dos etiquetadores con el objetivo de mejorar los resultados. La idea principal es explotar al máximo los recursos de los que ya disponemos, como es el caso de la propia base de datos de entrenamiento. La aplicación de diferentes transformaciones al corpus y al conjunto de etiquetas nos permitió generar la variabilidad necesaria para que los métodos de combinación lograsen mejorar los resultados sin requerir la inclusión de recursos adicionales. Concretamente en NER se consiguió alrededor del 5 % de mejora que acercaba el resultado final a las mejores cifras logradas en la competición CoNLL 2002 dedicada a esta tarea.

3.1.1. Definición de la Tarea

La historia de esta tarea viene marcada por diversas conferencias y competiciones que sirvieron para despertar el interés de los investigadores. El primer gran evento que dio a conocer la tarea NER fue MUC (*Message Understanding Conference*) que, con varias ediciones celebradas en los años noventa, puso de manifiesto el interés de esta tarea, encargándose de definirla y sentando las bases para su desarrollo y evaluación. Con el apoyo de la agencia de defensa estadounidense DARPA, MUC giraba en torno a la extracción de información en textos de interés militar, apareciendo por primera vez en su sexta edición, celebrada en 1995, la tarea de localizar entidades con nombre. Según [67] este tipo de entidades son las que vienen referenciadas por uno o más "designadores" rígidos, principalmente nombres propios. En MUC la labor se centraba en encontrar entidades de un tipo que recibió el nombre de 'enamex' y que incluía los nombres de organizaciones, localizaciones y personas, así como de tipo 'timex', que incluye expresiones temporales y 'numex', que hace referencia a cantidades y porcentajes. El interés se extendió a otras lenguas, quedando este hecho reflejado en la celebración de conferencias como MET (*Multilingual Entity Task Conference*), celebrada a finales de los noventa introduciendo esta tarea para los idiomas chino y japonés o HAREM (*Evaluation contest for named entity recognizers in Portuguese*) en 2004 y 2006, que afrontaba su aplicación al portugués. Aunque quizás fueron las ediciones de CoNLL (*Conference on Computational Natural Language Learning*) de los años 2002 y 2003 las que, centrándose en un formato multilingüe, lograron extender aún más si cabe el interés por esta tarea. En ambas ediciones, las entidades a reconocer consistían en nombres de organizaciones, personas, localizaciones y por último una categoría más, llamada "miscelánea", que contempla otros tipos de entidades a las que no se le asocia una categoría propia. La evaluación se realizó con textos en español y holandés en 2002 y en inglés y alemán en 2003.

Al igual que en otras tareas del PLN, en ocasiones se decide separar el proceso en dos etapas, tal y como hacen en [20]. La primera etapa se encargará de localizar en el texto aquellas secuencias de palabras que constituyen una entidad a juicio del etiquetador. La segunda etapa en cambio tratará de clasificar estas entidades previamente localizadas asignándoles una de las clases posibles, como pueden ser las de localización, organización o nombre de persona. En definitiva, el proceso general, que podríamos denominar "extracción de entidades con nombre" (NEE), se dividiría en dos etapas: el "reconocimiento de entidades con nombre" (NER) y la "clasificación de entidades con nombre" (NEC), donde $NEE = NER + NEC$.

Los corpus utilizados también varían de formato, pudiendo estar etiqueta-

dos por ejemplo en XML o según el esquema BIO, utilizado en las conferencias CoNLL y del cuál podemos ver un ejemplo en 3.1. Este esquema etiqueta el texto palabra a palabra asignando una etiqueta que puede ser una *O*, si la palabra no pertenece a ninguna entidad o una etiqueta compuesta por una *B* o una *I*, seguida de un guión y concluyendo con la categoría de la entidad a la que pertenece, representadas por las cadenas *LOC* para localizaciones, *ORG* para organizaciones, *PER* para los nombres de personas y *MISC* para el resto. La diferencia entre la *B* y la *I* al comienzo de la etiqueta radica en que se emplea la *B*, si es la primera palabra de la entidad, o la *I* en cualquier otro caso (una palabra que pertenece a una entidad y que no es la primera de las que forman parte de ella).

Word	Tag
El	O
presidente	O
del	O
COI	B-ORG
,	O
Juan	B-PER
Antonio	I-PER
Samaranch	I-PER
,	O
se	O
sumó	O
...	...

Figura 3.1: Corpus NER en formato BIO.

En cuanto a los métodos empleados en la resolución de esta difícil tarea, desde los noventa han sido muchos los trabajos dedicados a ella, empleándose múltiples estrategias y algoritmos. Una de las primeras publicaciones es [101], en donde se presenta un sistema que extrae nombres de compañías aparecidas en documentos de texto. Al igual que en la mayoría de los trabajos aparecidos a principio de los noventa, se trata de un sistema basado en reglas diseñadas manualmente. Esta tendencia sin duda fue cambiando hacia trabajos que delegan la extracción de conocimiento en diversos algoritmos de aprendizaje automático, algo que hoy en día tienen en común la inmensa mayoría de las publicaciones sobre esta tarea. Desde el punto de vista del aprendizaje, se han desarrollado sistemas basados en algoritmos no supervisados o semi-supervisados, como en [92], aunque la mayoría hace uso

del aprendizaje supervisado. Entre las técnicas más empleadas se encuentran los modelos de Markov[11], los modelos de máxima entropía[14], los CRF (*conditional random fields*)[84] o los SVM (*support vector machines*)[4]. El conjunto de características empleado también se ha demostrado que influye mucho en el éxito o fracaso final, conclusión que puede extraerse de trabajos como [108]. Por otro lado, estrategias como el reconocimiento simultaneo de diferentes tipos de entidades, introducida en [25], han demostrado ser útiles para la mejora de los resultados, enfrentando cada tipo a todos los demás. En [27] se aplicaron estas ideas sobre varios idiomas diferentes. De hecho, el número de idiomas utilizados para el desarrollo de sistemas NER y los conjuntos de entidades objeto de estudio han sido muy numerosos, tal y como podemos comprobar en [93].

Otro apartado importante en toda tarea es el proceso de evaluación de los sistemas que la afrontan. En este caso la evaluación se ha llevado a cabo de diferentes maneras en los diversos foros que se han organizado en torno al NER. Si bien se han utilizado bases de datos previamente etiquetadas contra las que poder contrastar los resultados, la flexibilidad con la que se juzga una determinada entidad no siempre ha sido la misma. En el MUC por ejemplo se dividió el etiquetado en dos aspectos a evaluar, 'TYPE' y 'TEXT'. Estos aspectos hacen referencia a la categoría asignada a la entidad reconocida y al texto asociado a la misma respectivamente. De esta forma, un sistema puede etiquetar de forma correcta los dos aspectos, uno sólo o ninguno en lo referente a cada entidad, pudiendo obtener dos, uno o cero puntos por ella. En CoNLL y en IREX (*Information Retrieval and Extraction Exercise*) sin embargo se exige un reconocimiento exacto del texto que se asocia a la entidad reconocida, mientras que en ACE (*Automatic Content Extraction Program*) se desarrolló un sistema de evaluación complejo que permite valorar de diferentes formas los posibles errores que se pueden cometer (ver tabla 3.1), lo cuál aporta gran flexibilidad aunque en ocasiones dificulta la interpretación de los resultados.

3.1.2. Sistema Base

Como primera aproximación al problema implementamos un etiquetador basado en los modelos de Markov, capaz de generar un modelo a partir de un corpus de entrenamiento. Este corpus contiene las entidades ya etiquetadas y clasificadas, y será a partir de esta información desde donde se extraerá el conocimiento lingüístico implícito para etiquetar posteriormente nuevas frases sin etiquetar. En este caso se escogieron los corpus empleados en el certamen internacional CoNLL-2002. Concretamente la distribución en español cuyas especificaciones se muestran resumidas en la tabla 3.2.

Etiquetado correcto		Resultado obtenido		Fallos cometidos
desde	O	desde	B-PER	Reconocimiento incorrecto (entidad inexistente)
Juan	B-PER	Juan	O	Reconocimiento incorrecto (entidad no reconocida)
Juan	B-PER	Juan	B-PER	Categoría correcta
Madrid	I-PER	Madrid	O	Límites incorrectos
Real	B-ORG	Real	B-LOC	Categoría incorrecta
Madrid	I-ORG	Madrid	I-LOC	Limites correctos
Real	B-ORG	Real	O	Categoría incorrecta
Madrid	I-ORG	Madrid	B-LOC	Límites incorrectos

Tabla 3.1: Posibles errores en el etiquetado NER

	Nº Tokens	Nº Entidades
TRAIN	264715	18794
TEST	51533	3558

Tabla 3.2: Especificaciones del corpus *CoNLL-2002*.

Este sistema nos servirá para introducir la tarea en sí, además de fijar una base con la que poder comparar nuestros resultados posteriores aplicando la técnica de *stacking*.

Al dividir el problema en dos fases, reconocimiento y clasificación, implícitamente facilitamos la flexibilidad de poder escoger la herramienta más apropiada para cada etapa. En nuestro caso, mientras los modelos de Markov ocultos (HMM) demostraron comportarse satisfactoriamente en la etapa de delimitación de entidades, en la etapa de clasificación pudimos experimentar con diferentes técnicas con el objetivo de aprovechar al máximo la información contenida en el corpus y alcanzar mejores resultados.

Reconocimiento

Respecto a la fase de reconocimiento, podemos adaptar un corpus con facilidad si se encuentra etiquetado según el esquema *BIO*, un requisito que cumple el que hemos utilizado proveniente de CoNLL 2002. Este esquema permite obtener un corpus apropiado excluyendo las categorías de las etiquetas asignadas. De esta forma se obtiene de forma muy sencilla el principal

recurso necesario para aplicar el sistema, reduciendo el número de etiquetas a sólo tres (B, I, O) como se muestra en la figura 3.2.

Word	Tag
El	O
presidente	O
del	O
COI	B-ORG
,	O
Juan	B-PER
Antonio	I-PER
Samaranch	I-PER
,	O
se	O
sumó	O
...	...

Word	Tag
El	O
presidente	O
del	O
COI	B
,	O
Juan	B
Antonio	I
Samaranch	I
,	O
se	O
sumó	O
...	...

Figura 3.2: Corpus original y su versión etiquetada para el NER.

Para esta fase de reconocimiento, los modelos de Markov presentaban una efectividad demasiado baja en la localización de entidades desconocidas, es decir, aquellas que no están presentes en el corpus de entrenamiento que ha generado el modelo. Para mejorar esta disfunción enriquecimos el modelo original incluyendo nuevas características mediante la transformación del corpus [104]. A continuación indicamos las características y transformaciones consideradas:

- Se sustituye cada palabra del corpus por un token representativo. Los tokens considerados se muestran en la tabla 3.3.
- Se excluyen de esta transformación aquellas palabras que aparecen frecuentemente alrededor o dentro de una entidad.

Clasificación

En cuanto a la segunda fase del proceso, y tras obtener un conjunto de entidades aún sin clasificar, podemos extraer de ellas una serie de características que nos sirvan de ayuda a la hora de decidir el tipo de entidades a las que pertenecen. Cada entidad dará lugar a un vector de características, generándose una base de datos a la que podemos aplicar diversos algoritmos

Token	objeto de aplicación
may	palabras que comienzan con mayúsculas
min	palabras que comienzan con minúsculas
tmay	palabras que sólo contienen mayúsculas
mypto	mayúscula seguida de un punto
abrev	abreviaturas en general
1apal	palabras que comienzan una frase

Tabla 3.3: Sustitución de palabras por tokens aplicada al corpus.

de clasificación. En [31] aplicamos máquinas SVM [129] ya que presentan un rendimiento muy bueno aún cuando el número de componentes que constituyen los vectores de la base de datos es muy alto. El conjunto de características utilizadas en nuestro sistema fue muy amplio y cada una de ellas requirió un proceso diferente para su cálculo. En general, se trata de calcular una puntuación que refleje la importancia de cada característica en un determinado ejemplo. Las características utilizadas en los experimentos se agrupan en las siguientes categorías:

- Ortográficas: Se tiene en cuenta si una entidad contiene palabras que comienzan en mayúsculas, o palabras constituidas exclusivamente por mayúsculas, dígitos o números romanos. Se incluyen también en este grupo otras características como la longitud en palabras de una entidad, la posición relativa dentro de la frase y si contiene comillas o no.

- Sufijos y Prefijos: Se calculan a partir del corpus de entrenamiento los sufijos y prefijos de dos y tres letras relevantes para cada categoría.

- Contextos: En una ventana de tres palabras alrededor de una entidad se calculan las palabras relevantes para cada categoría según los ejemplos del corpus de entrenamiento.

- Palabras significativas: Para cada categoría se calcula el conjunto de palabras significativas eliminando palabras huecas y en minúsculas de los ejemplos del corpus de entrenamiento.

- Listas externas: Se comprueba si una determinada entidad presenta alguna palabra perteneciente a una serie de listas generadas con información externa al corpus de entrenamiento. Se tienen en cuenta listas de nombres, apellidos, países, ciudades y disparadores internos de cada categoría (por ejemplo calle para la categoría lugar).

Para el cálculo de las características 2, 3 y 4 se disponía de una serie de listas generadas automáticamente a partir del corpus de entrenamiento. Los elementos de las listas con mayor frecuencia de aparición tenían asociado un mayor peso a la hora de computar el valor atribuido a su característica.

En [121] se aplicaron las implementaciones de la herramienta Weka [133] para los árboles de decisión, así como para los métodos *bagging* y *boosting* en lo que supuso nuestra primera toma de contacto con técnicas de combinación.

Resultados

A continuación se muestra un resumen de los experimentos realizados junto con el valor $F_{\beta=1}$ (ver ecuación 1.3) obtenido en cada uno de ellos:

1. Como *baseline* para el resto de experimentos se descargó toda la responsabilidad del proceso sobre un modelo de Markov oculto. Sin aplicar ninguna técnica adicional (transformación del corpus,...) se obtuvo un valor $F_{\beta=1} = 67,87$.

2. Restringiendo el trabajo del HMM a la fase de reconocimiento de entidades se obtuvo un resultado de $F_{\beta=1} = 72,66$, que tras aplicar las transformaciones que se relacionan en la tabla 3.3, ascendió hasta alcanzar $F_{\beta=1} = 85,95$.

3. Tomando de partida el resultado ofrecido por el modelo de Markov oculto en la fase de localización de entidades, se realizaron diversos experimentos referentes a la fase de clasificación. La aplicación de los SVM dio lugar a un valor $F_{\beta=1} = 70,41$ final para la tarea completa, mientras que los árboles de decisión no consiguieron pasar de $F_{\beta=1} = 67,93$.

4. Finalmente las técnicas combinatorias de *bagging* y *boosting* ofrecieron valores de $F_{\beta=1} = 70,07$ y $F_{\beta=1} = 69,10$ respectivamente.

Para más información sobre los experimentos realizados durante esta fase de investigación, se pueden consultar los trabajos publicados [31] y [121].

3.1.3. Variabilidad Mediante Transformaciones

Llegados a este punto, y tras comprobar la eficiencia de métodos de combinación como *bagging* y *boosting*, decidimos en [120] ir más allá explorando los medios de combinar diferentes visiones de un mismo problema.

En esta ocasión nos centramos únicamente en la fase del reconocimiento de entidades (NER), tomando como base la respuesta de dos etiquetadores

independientes ampliamente utilizados, como son TnT [15] y TBL [17]. Ambos ofrecen visiones radicalmente diferentes, lo cuál aporta mayor riqueza a la infraestructura que ha de extraer la máxima información posible de ellos. El etiquetador TnT está basado en modelos de Markov de segundo orden utilizando trigramas, enriquecidos con algoritmos de interpolación lineal para aliviar el problema de la dispersión de los datos, así como técnicas de análisis de sufijos. La herramienta TBL en cambio, se inspira en métodos de aprendizaje basados en transformaciones. Se trata de recopilar un conjunto de reglas que aporten el etiquetado inicial para luego, en sucesivas iteraciones, ir mejorando el comportamiento transformando ese conjunto de reglas de manera que se disminuya el error de las evaluaciones anteriores. Si bien TBL ofrece mejores resultados finales, algo que se repetirá en todos los experimentos que llevamos a cabo, el coste computacional es mucho mayor comparado con TnT. Mientras TBL requiere de varios minutos para crear el modelo partiendo del corpus de entrenamiento de CoNLL-2002 (ver las características del corpus en la tabla 3.2), TnT está listo para ser ejecutado tras unos pocos segundos.

Partiendo de los modelos aportados por estas dos herramientas, podemos ampliar el número de modelos intuitivamente de dos formas, una es ofreciéndoles a TnT y a TBL nuevos corpus de entrenamiento con los que poder generar nuevos modelos, y la otra es añadiendo nuevos etiquetadores que puedan ofrecer nuevas opiniones a valorar en paralelo junto a las dos que ya tenemos. Para este trabajo recurrimos sin embargo a las transformaciones sobre el corpus de entrenamiento.

Transformaciones

Basándonos en los resultados obtenidos previamente con la reducción del vocabulario, introdujimos dos ideas nuevas como son el cambio del conjunto de etiquetas y la incorporación de información externa en el vocabulario a través de la etiqueta POS de cada palabra. En total, las tres transformaciones aplicadas, de las que se muestra un ejemplo en la figura 3.3, fueron:

- *Reducción del vocabulario y los disparadores*: Al sustituir las palabras por tokens en el caso de encajar con un patrón conocido (ver la tabla 3.3), conseguimos reducir ampliamente el tamaño del vocabulario que ha de manejar el modelo. Además, hacemos frente al problema de las palabras desconocidas, que son aquellas que no aparecen en el corpus de entrenamiento, y que suponen un serio escollo para el modelo, al no poseer información suficiente sobre la etiqueta a asignar. Al sustituir el lexema por un token podemos aprender de sus características

tipográficas pero habrá casos en los que la información que aporta el lexema es demasiado importante para deshacerse de ella. Esto es lo que ocurre con las palabras que aparecen normalmente antes o después de una entidad, y que llamaremos 'disparadores'. La estrategia por tanto consiste en generar una lista de disparadores basándonos en las palabras que cumplen este requisito en el corpus de entrenamiento, para, a la hora de etiquetar el texto nuevo, recorrer en primer lugar estas listas por si la palabra actual puede considerarse un disparador. Si es así, la palabra permanecerá inalterada, mientras que si no aparece en ninguna lista de disparadores, se emparejará con los patrones habituales para la reducción del vocabulario (representados mediante expresiones regulares).

- *Cambio del conjunto de etiquetas*: El cambio consiste en agregar una etiqueta especial para aquellas palabras que finalizan una entidad. Le asignamos por tanto una etiqueta *E* a las palabras que finalicen una entidad compuesta (con más de una palabra), y la etiqueta *BE* a las que sirvan para iniciar y finalizar una entidad de una sola palabra. Tras ser aplicada esta transformación, el corpus estará etiquetado con el siguiente conjunto de etiquetas una vez eliminada la información referente a las categorías de las entidades, ya que estamos trabajando en la fase de reconocimiento: *B*, *I*, *O*, *BE*, y *E*.

  Esta transformación, a diferencia de la reducción de vocabulario, afecta únicamente a las etiquetas y no a las palabras del corpus. Con ella se intenta extraer del corpus información sobre las palabras que usualmente finalizan una entidad, pudiéndose reflejar en el modelo características especiales para estas palabras que sirvan para reconocer nuevas ocurrencias en los textos a etiquetar.

- *Etiquetas POS*: La última transformación que aplicamos, y la única que requirió de conocimiento externo para generar un nuevo modelo, fue la concatenación de la etiqueta POS a las palabras del corpus. Se realiza la concatenación ya que la etiqueta POS por sí sola no aporta información suficiente para detectar entidades en el corpus. Si concatenásemos directamente, veríamos ampliado el número de palabras del vocabulario de manera excesiva; de ahí que antes de concatenar se aplicase la reducción del vocabulario tal y como vimos anteriormente. De esta forma el tamaño del vocabulario se fija entre el que tiene originalmente el corpus y el reducido por sustitución de palabras por tokens.

  Para las pruebas se obtuvieron las etiquetas POS entrenando la herramienta TnT con el corpus CLiC-TALP para más tarde aplicar el

etiquetador generado sobre el corpus de entrenamiento del CoNLL-2002 habitual. El corpus CLiC-TALP [22] está compuesto por textos en español con más de cien mil palabras y extraídos de revistas, libros de texto y novelas entre otros.

Word	Tag
El	O
presidente	O
del	O
tmay	B
,	O
may	B
may	I
may	I
,	O
se	O
min	O
...	...

a) Reducción del vocabulario del corpus.

Word	Tag
El	O
presidente	O
del	O
COI	BE
,	O
Juan	B
Antonio	I
Samaranch	E
,	O
se	O
sumó	O
...	...

b) Ampliación del conjunto de etiquetas.

Word	Tag
El_det_	O
presidente_noun_	O
del_prep_	O
_tmay__noun_	B
,_punt_	O
_may__noun_	B
_may__noun_	I
_may__noun_	I
,_punt_	O
se_pron_	O
_min__verb_	O
...	...

c) Concatenación de etiquetas POS a las palabras.

Figura 3.3: Resultado de las transformaciones del corpus de la figura 3.2.

Resultados de las Transformaciones

Las tres transformaciones llevadas a cabo sobre el corpus de entrenamiento (ver figura 3.4), permitieron generar modelos que mejoraban los resultados aportados tanto por TnT como por TBL inicialmente. La tabla 3.4 muestra todos los valores de $F_{\beta=1}$ obtenidos por los diferentes modelos, así como la precisión (precision) y cobertura (recall) por separado. El modelo *Modif-V* se corresponde con el generado tras la reducción del vocabulario, el modelo *Modif-E* con el generado tras la ampliación del conjunto de etiquetas y el modelo *Modif-P* con el generado tras concatenar las etiquetas POS. El *baseline* muestra los resultados tras generar el modelo con el corpus de entrenamiento original sin modificaciones.

Tomando estos datos como punto de partida, era de suponer que combinar toda esta información en un sistema que considere todas las propuestas y no sólo una, debía suponer una mejoría mayor aún en los resultados finales,

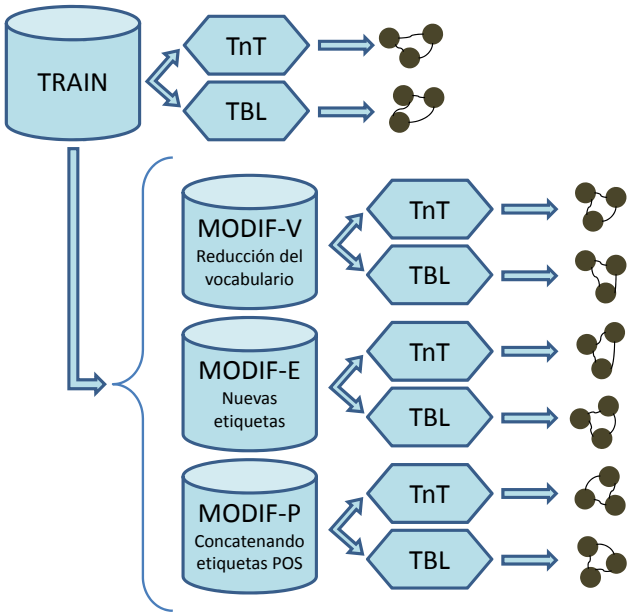

Figura 3.4: Obtención de nuevos modelos transformando el corpus.

como de hecho ocurrió.

Combinación

La votación es posiblemente la primera técnica que nos viene a la mente cuando se trata de combinar diferentes propuestas para dar una respuesta única ante un problema. Aún por simple que parezca, los resultados que aporta son, en muchos casos, superiores a los de sistemas mucho más complejos. En [120] llevamos a cabo experimentos combinando, mediante votación sim-

	TnT			*TBL*		
Modelo	precision	recall	$F_{\beta=1}$	precision	recall	$F_{\beta=1}$
baseline	84,39 %	86,12 %	85,25	85.34 %	89.66 %	87.45
Modif-V	85,19 %	88,11 %	86,63	87.72 %	88.48 %	88.10
Modif-E	86,21 %	87,47 %	86,83	86.78 %	89.07 %	87.91
Modif-P	85,33 %	88,09 %	86,69	89.14 %	89.29 %	89.22

Tabla 3.4: Resultados de las transformaciones del corpus CoNLL2002.

ple y ponderada, las propuestas generadas por todos los modelos obtenidos previamente. Aunque los resultados alcanzados mediante estos dos métodos no fueron finalmente los mejores, veremos que no fueron para nada despreciables.

Los resultados fueron superiores a todos los obtenidos anteriormente alcanzándose un valor de 89,97 con la votación normal y un valor de 90,02 con la votación ponderada.

Aunque el sistema de votación se comporta extraordinariamente bien a pesar de su simplicidad, existen otras maneras de consensuar una respuesta conjunta entre varios etiquetadores independientes. En este caso se experimentó con la técnica de *stacking* reuniendo las etiquetas propuestas por los etiquetadores, concretamente TnT y TBL en sus diferentes versiones, para generar una base de datos de entrenamiento sobre la que se aplica un algoritmo de aprendizaje automático.

Los patrones utilizados se presentan en el formato *arff* empleado por la herramienta Weka, del cuál presentamos un pequeño ejemplo en la figura 3.5. Al comienzo del fichero se especifica el nombre de la base de datos que estamos definiendo, seguido de un listado con todos los atributos o componentes de los vectores de características que vamos a generar. Para este trabajo se empleó un atributo por cada modelo que íbamos a combinar, es decir, el generado por el corpus original y los generados con cada transformación aplicada a TnT y TBL, sumando en total ocho modelos. Junto a cada atributo se indica su tipo asociado, que en nuestro caso será un enumerado con todas las posibles etiquetas de la palabra referenciada. Finalmente, se introduce la sección de datos con todos los vectores que componen la base de datos, representando cada uno a una palabra del corpus.

Los resultados de *stacking* con árboles de decisión (89,72), mejoraron los resultados obtenidos por los modelos individualmente (el mejor obtuvo 89,22), aunque no superaron los obtenidos por el sistema de votación (90,02). Aunque esto demostró que el sistema de votación es muy eficiente, aún quedaban dos características del *stacking* por explotar: la capacidad de añadir y mezclar información heterogénea a la base de datos de entrenamiento y la posibilidad de aplicar diferentes técnicas y esquemas de aprendizaje.

Al igual que en la generación de los vectores de características empleados para las máquinas SVM, podemos utilizar información de diferente índole para aportarle al modelo el conocimiento que requiere para aprender el etiquetado correcto en cada situación. Una de las consideraciones más útiles en el etiquetado de textos es el contexto de las palabras, es decir, tener en cuenta las etiquetas recibidas por las palabras que rodean a la palabra analizada en cada momento. En una base de datos como la figura 3.5, esto se traduce en ampliar el número de atributos para no sólo considerar la etiqueta actual

```
@relation collaboration
@attribute TnT            {O, B, I}
@attribute TnT-VOC-RED    {O, B, I}
@attribute TnT-NEW-TAGS   {O, B, I}
@attribute TnT-POS        {O, B, I}
@attribute TBL            {O, B, I}
@attribute TBL-VOC-RED    {O, B, I}
@attribute TBL-NEW-TAGS   {O, B, I}
@attribute TBL-POS        {O, B, I}
@attribute ACTUAL-TAG     {O, B, I}
@data
O, O, O, O, O, O, O, O,     O
B, B, B, B, B, B, B, B,     B
I, I, I, O, O, O, I, I,     I
I, I, I, B, B, B, I, I,     I
O, O, O, O, O, O, O, O,     O
B, B, B, B, B, B, B, B,     B
I, I, I, I, I, I, I, I,     I
O, I, I, I, I, I, O, O,     I
B, I, I, I, I, I, B, B,     I
O, O, O, O, O, O, O, O,     O
O, O, O, O, O, O, O, O,     O
B, B, B, O, O, B, B, B,     O
...
```

Figura 3.5: Base de datos de entrenamiento en formato '*arff*'.

de cada modelo, sino una ventana de n etiquetas previas y posteriores a la palabra actual asignadas por cada modelo. Hemos registrado en la tabla 3.5 los resultados obtenidos para un tamaño de ventana de una etiqueta (tres etiquetas por cada modelo, en total veinticuatro) y de dos etiquetas (cinco etiquetas por cada modelo, en total cuarenta), llamando a sendos experimentos *Stacking-1* y *Stacking-2* respectivamente. Incrementar el tamaño de la ventana más aún dispara el número de atributos de la base de datos sin que se mejoren los resultados.

Por otro lado, podemos aprovecharnos de la posibilidad de aplicar sobre el algoritmo de aprendizaje utilizado, diferentes esquemas y técnicas de aprendizaje compuestas, como por ejemplo *bagging*. Esta técnica ha demostrado aportar en la mayoría de los casos mejores resultados que el algoritmo de base en solitario, aunque el coste computacional que requiere no resulta

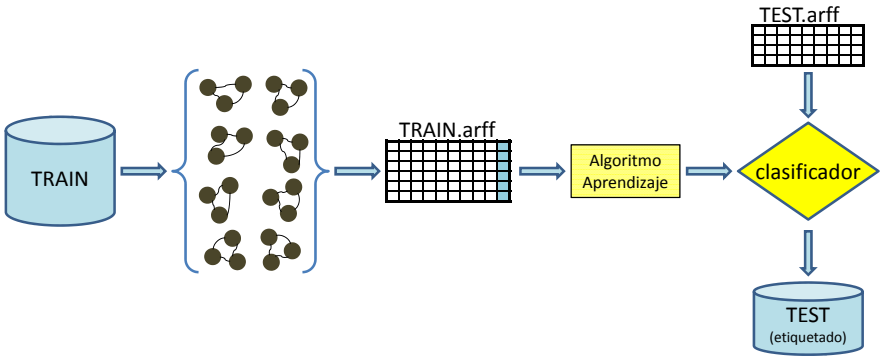

Figura 3.6: Esquema de la combinación de modelos mediante *stacking*.

despreciable.

Los resultados finales para los métodos utilizados para combinar los sistemas se muestran en la tabla 3.5 junto a las mejoras que suponen respecto al mejor resultado individual, un 89,22 conseguido por el modelo *Modif-P* con TBL. En esta tabla vemos que conseguimos alcanzar un valor máximo de 90,90, siendo este un valor comparable al de los mejores sistemas reconocedores de entidades, como por ejemplo el ganador de la competición CoNLL-2002 que alcanzó un valor $F_{\beta=1} = 91,66$ en la subtarea NER.

	precision	recall	$F_{\beta=1}$	$\Delta F_{\beta=1}$
Votación	89,67 %	90,28 %	89,97	0,75
Votación ponderada	89,43 %	90,62 %	90,02	0,80
Stacking	88,93 %	90,53 %	89,72	0,50
Stacking-1	89,54 %	90,92 %	90,23	1,01
Stacking-2	90,18 %	90,78 %	90,48	1,26
Stacking-2 y Bagging	90,69 %	91,12 %	90,90	1,68

Tabla 3.5: Resultados de la combinación de modelos aplicada al NER.

Después de los experimentos con la subtarea del reconocimiento de entidades, exploramos en [122] los efectos de estas técnicas sobre la tarea completa.

Para la fase de clasificación de las entidades en las categorías *LOC*, *ORG*, *PER* y *MISC*, recurrimos a las máquinas SVM, que ofrecieron los mejores resultados en los experimentos anteriores. Generamos los vectores analizando los mismos tipos de características consideradas en el sistema base, añadiendo la longitud y posición relativa de las palabras pero sin utilizar listas externas

de entidades.

El resultado obtenido, $F_{\beta=1} = 70,47$, supone una mejora de más de cuatro puntos sobre el sistema base con el que comenzamos estos experimentos, lo que sirvió, tal y como comentábamos al inicio del capítulo, para hacer crecer nuestro interés en las técnicas de combinación como *stacking*.

3.1.4. Experimentos adicionales

En [123] probamos algunas alternativas que nos parecían interesantes aunque no aportaron mejorías significativas al modelo. Un ejemplo es la ampliación del conjunto de etiquetas BIO no solo con las etiquetas *BE* y *E*, sino con la división de la etiqueta *O* en las etiquetas:

- BEF: para las palabras que aparecen justo antes de una entidad.
- BET: para las palabras que aparecen justo entre dos entidades.
- AFT: para las palabras que aparecen justo a continuación de una entidad.
- O: para el resto de palabras que no forman parte ni se encuentran adyacentes a ninguna entidad.

Este cambio no aportó mejora alguna al modelo inicial, al diversificar en exceso las etiquetas a reconocer y complicar el modelo sin aportar suficiente información para compensarlo. Se obtuvo un valor $F_{\beta=1} = 84,51$ frente al 84,59 de la transformación sólo con *BE* y *E*.

Asimismo se hizo un estudio más exhaustivo sobre los diferentes algoritmos de aprendizaje automático que se pueden aplicar directamente a *stacking*, considerando además del método *bagging* que ya conocemos, las siguientes técnicas disponibles en Weka: *decision table* [65],*part* [132],*ripper* [24] y *random tree* [133]. Los resultados de estas técnicas no mejoraron los obtenidos previamente por *bagging*.

3.2. La Creación de Recursos Lingüísticos

A lo largo de la historia del procesamiento del lenguaje natural, han ido surgiendo múltiples algoritmos que dan respuesta a las diversas problemáticas a las que se enfrentan los investigadores. Muchos de ellos son algoritmos de aprendizaje supervisado que basan sus resultados en la disponibilidad de grandes recursos lingüísticos de los que extraen el conocimiento necesario para las múltiples tareas a desempeñar. La existencia de estos recursos

determina por tanto la calidad de los resultados en términos de cobertura y precisión de las respuestas que ofrecen, el rendimiento general del sistema y en ocasiones, ambas cosas. A lo largo de este capítulo vamos a mostrar diversos aspectos que hacen referencia al esfuerzo necesario para la creación de estos recursos, y que por lo tanto justifican los intentos de desarrollar métodos que alivien esta tarea, así como diversas propuestas que se han mostrado para intentar solventar esta cuestión. Estas propuestas pueden considerarse alternativas al problema y lo afrontan de muy diferentes maneras, algunas de las cuales ya han sido empleadas en sistemas de combinación, mientras que otras quizás puedan ser adaptadas en un futuro próximo. Finalmente mostraremos nuestra aportación, presentada en [35] y [36], la cuál se centró en aprovechar las virtudes de las técnicas de *bootstrapping*, asociándolas al método de combinación de *stacking*.

3.2.1. El Problema y Algunas Aproximaciones

Sin duda alguna el mayor problema que surge a la hora de afrontar la creación de recursos lingüísticos es el esfuerzo que se requiere para obtener corpus de suficiente calidad para que sean útiles al aplicar los algoritmos que los necesitan. Generalmente, un algoritmo de aprendizaje supervisado que hace uso de un corpus etiquetado para una determinada tarea, exige un número altísimo de palabras o frases etiquetadas para ofrecer resultados que puedan ser considerados de calidad. De todas formas, esto dependerá del algoritmo en cuestión y de la tarea que se esté afrontando, aunque también es cierto que aumentar el tamaño del corpus manteniendo la calidad del mismo, nunca irá en detrimento de los resultados.

Si nos centramos en una tarea ampliamente conocida dentro del procesamiento del lenguaje natural, como es la desambiguación de significados, podemos hacernos una idea de este esfuerzo que estamos comentando. Se trata de una tarea que afronta el problema de seleccionar el significado de una palabra en un texto de entre todos los significados que posee. La ambigüedad es muy común aunque los humanos estamos tan acostumbrados a ella y tenemos tal capacidad de resolverla basándonos en el contexto de las palabras, que casi pasa desapercibida ante nuestros ojos. Para esta tarea se han desarrollado múltiples algoritmos con muy buenos resultados, aunque la disponibilidad de corpus etiquetados sigue constituyendo un problema. El estudio [94] asegura que para obtener una precisión buena se necesitan al menos 500 ejemplos por cada una de las palabras ambiguas a tratar (esta es una cifra que representa la media ya que hay diferencias considerables de una palabra a otra). A un ritmo de un ejemplo etiquetado por minuto y considerando la existencia de unas 20000 palabras ambiguas en el vocabulario

inglés común, esto nos conduciría a unas 160000 horas de etiquetado, que resultarían en nada más y nada menos que 80 años de dedicación exclusiva para una persona que lleve a cabo esta tarea de etiquetado. Si además le añadimos el hecho de que las tareas de etiquetado suelen ser llevadas a cabo por lingüistas entrenados o expertos, no cabe duda de que se trata de un proceso realmente caro y generalmente prohibitivo en la inmensa mayoría de los casos.

Todo esto supone una limitación y termina por reducir el número de ejemplos disponibles, afectando a la tarea en general y posiblemente al desarrollo de nuevas vías de investigación que puedan aportar mejoras en los resultados.

Existen diversas formas de generar recursos lingüísticos de manera automática o semiautomática, de las cuales destacamos a continuación algunas de las más representativas de cuantas se han puesto en práctica para afrontar esta difícil tarea.

Empleando Búsquedas en la Web

Una de las vías que han surgido para intentar paliar los efectos del enorme esfuerzo requerido para la creación de recursos, es el uso de la Web. El contenido de la Web puede ser considerado un enorme corpus que puede ser explotado para diversas tareas, si bien presenta una estructura y unos contenidos tan heterogéneos que no siempre se sabe muy bien cómo sacarle partido a toda la información que posee.

En [85] podemos apreciar un magnífico ejemplo de cómo se puede hacer uso de la *World Wide Web* para obtener recursos lingüísticos a través de los sistemas de búsquedas que tenemos a nuestra disposición. La tarea que se afronta en este trabajo es la desambiguación de significados y el sistema propuesto hace uso de diversos recursos disponibles como el corpus SemCor [89] y la base de datos léxica WordNet [88].

El Crowdsourcing

El *crowdsourcing* es un término acuñado recientemente y que guarda una estrecha relación con el fenómeno del outsourcing. Este último está basado en la delegación de ciertas tareas en determinadas entidades externas para ahorrar costes y simplificar el proceso de desarrollo en un proyecto. La nueva mano de obra en este entorno es posible que se encuentre en el trabajo disperso y anónimo de multitud de internautas que desarrollan tareas de mayor o menor valor para una organización que sepa llamar su atención de alguna de entre tantas formas posibles. Esta forma de recopilar el esfuerzo y orientarlo hacia la consecución de algún objetivo relacionado con el desarrollo de

alguna tarea en concreto se denomina *crowdsourcing* [1]. Un buen ejemplo es el llamado Turco Mecánico de Amazon, a través del cuál todo el mundo puede cobrar una pequeña cantidad de dinero por realizar tareas muy simples sin necesidad de una gran preparación previa.

El sistema *Open Mind Word Expert* [87] también encaja en esta descripción ya que se trata de un sistema web enfocado a la generación de grandes corpus anotados semánticamente con la ayuda de los usuarios de Internet. El trabajo de anotar se distribuye entre los miles de etiquetadores humanos potenciales, lo que nos lleva hacia una disminución radical del coste y la duración del proceso de etiquetado. Lo que hace este sistema es permitir a los contribuyentes etiquetar palabras con sus significados de WordNet. Para cada palabra ambigua para la que se desea obtener un corpus etiquetado semánticamente, se escogen una serie de frases en lenguaje natural que poseen alguna aparición de dicha palabra y se les presenta a los usuarios. Con esa colección de frases se entrena al sistema que, tras emplear un algoritmo de active learning (comprobando el grado de desacuerdo entre dos clasificadores diferentes), detecta los casos "difíciles de etiquetar" que son mostrados a nuevos usuarios en una segunda fase del proceso.

La Combinación de Recursos Existentes

Otra estrategia que podemos encontrar en la bibliografía para generar corpus es la combinación de recursos ya existentes, de manera que se enriquezcan unos con otros aumentando su valor al ser considerados de forma global. Un ejemplo muy clarificador lo podemos encontrar en [111], donde se combinan *FrameNet*, *VerbNet* y *WordNet*.

La construcción de recursos lingüísticos requiere un gran esfuerzo humano y cada recurso está pensado para solucionar un determinado tipo de problemas, mostrando virtudes en ciertos aspectos y desventajas en otros. De esta forma, la combinación de estos recursos puede dar lugar a una base de conocimiento más extensa y más rica.

Importando Recursos Cercanos

Cuando queremos afrontar la tarea de crear un recurso lingüístico, una posibilidad que tenemos al alcance de nuestra mano en muchos casos, es adaptar otro recurso "cercano" al que deseamos crear. Es la opción elegida por ejemplo en [21], donde se construye un reconocedor de entidades con nombre para el catalán partiendo de recursos en español. Se emplean dos vías para lograrlo: en primer lugar creando los modelos en español para posteriormente

[1] Del inglés 'crowd' que significa multitud y 'source' que significa fuente

traducirlos al catalán, y en segundo lugar crear los modelos de forma bilingüe directamente.

La cercanía en este caso se presenta ya que se trata de dos lenguas románicas que poseen estructuras sintácticas similares y cuyo entorno social y cultural se solapan en gran medida, haciendo que exista un gran número de entidades que aparecen en los corpus de ambas lenguas. Estas características hacen que los recursos en español sean aprovechables para llevar a cabo tareas sobre el catalán como puede ser el reconocimiento de entidades con nombre.

3.2.2. Técnicas de Bootstrapping

En trabajos como [21] se pone en práctica otra técnica de obtención de recursos muy interesante. Se trata de la técnica de *bootstrapping*, que trata de obtener una gran cantidad de material partiendo de una pequeña "semilla". En la tarea de la creación de corpus etiquetados, el objetivo será obtener un gran número de frases etiquetadas de forma automática partiendo de un número muy reducido de frases etiquetadas manualmente (con un coste bajo debido al reducido número de frases inicial).

Existen múltiples técnicas de *bootstrapping*, que difieren en la forma de aumentar la semilla, el manejo de las frases nuevas etiquetadas o las técnicas de selección en caso de utilizarse alguna. En cualquier caso todas responden a la definición:

> "la elevación de un pequeño esfuerzo inicial hacia algo más grande y más significativo".

Algunos de los esquemas de ejecución más populares dentro de las conocidas como técnicas de *bootstrapping* son:

- *Self-train*: Un corpus es utilizado para crear un modelo que se aplica a un conjunto nuevo de frases que tras ser etiquetadas pasan a formar parte del corpus original para volver a generar un nuevo modelo y avanzar de esta forma iterativamente.

  Esta es la definición de *self-training* que generalmente se adopta (ver [23]) aunque existen otras como la que aporta [95], donde se describe como el entrenamiento de un comité de clasificadores utilizando *bagging* para finalmente utilizar la votación por mayoría para seleccionar las etiquetas finales.

  Entre las técnicas de *bootstrapping* que aquí recogemos, ésta es la más sencilla de implementar. Se trata de un esquema de muy bajo coste pero

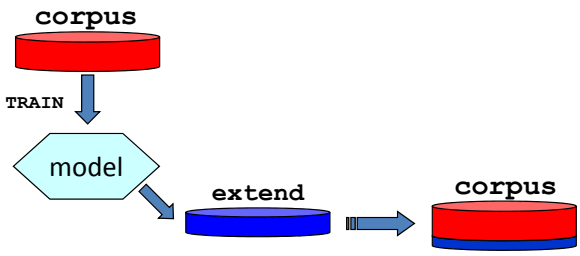

Figura 3.7: Esquema de ejecución para el *self-train*.

desafortunadamente la calidad del corpus decrece a un ritmo que hace inviable la obtención de un tamaño grande cumpliendo unos requisitos mínimos de calidad. La razón radica en la imposibilidad del etiquetador de aprender de él mismo, haciéndose cada vez más influyentes los errores que va cometiendo en las sucesivas iteraciones. Si existen situaciones que no aparecen en el corpus de entrenamiento, el etiquetador nunca aprenderá a responder de forma correcta ante estas situaciones sin recibir ayuda externa.

- *Collaborative-train*: Se emplea un mismo corpus para obtener diferentes modelos empleando diferentes técnicas de aprendizaje. Posteriormente se introduce una fase de selección entre las diferentes opiniones que surgen de aplicar estos modelos al conjunto de frases nuevas y las etiquetas seleccionadas sirven para aumentar el corpus original y proseguir con la siguiente iteración.

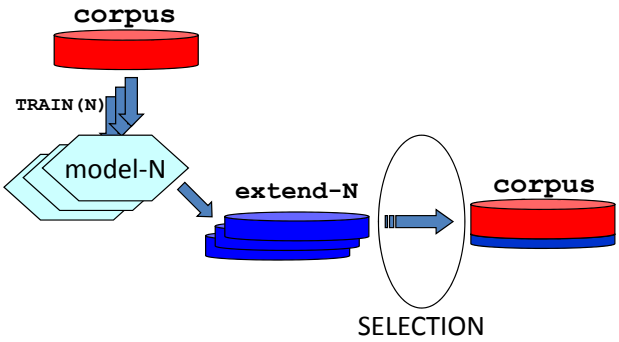

Figura 3.8: Esquema de ejecución para el *collaborative-train*.

Existen muchos métodos de selección que se pueden aplicar, desde un sistema de votación a técnicas de aprendizaje más complejas como *stacking*.

- Co-train: Dos corpus inicialmente iguales sirven para crear dos modelos de diferentes características y los resultados de aplicar estos modelos a un conjunto de frases nuevas se "cruzan", es decir, las frases etiquetadas por el primer modelo sirven para aumentar el corpus que sirvió para crear el segundo modelo y viceversa.

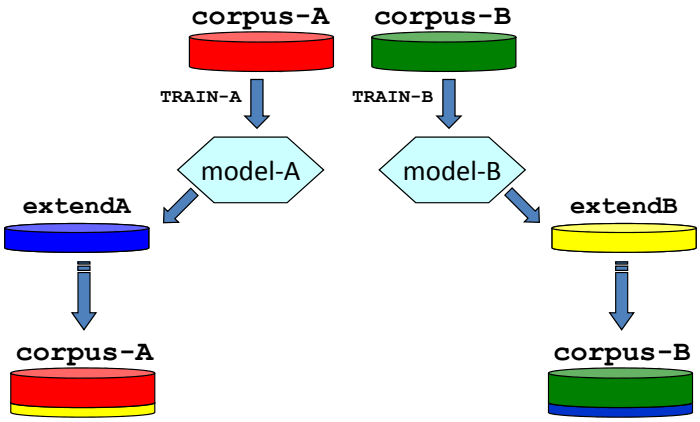

Figura 3.9: Esquema de ejecución para el *co-train*.

La técnica de *co-training* [12] permite que distintos etiquetadores puedan cooperar aportando así la ayuda externa de la que carece el *self-training*. Para aportar información adicional a un etiquetador sobre situaciones que no sabe manejar, se introduce otro etiquetador (o más). Cada etiquetador representa una visión diferente del problema, y la mezcla de estas vistas enriquece las capacidades globales del sistema [1].

En [86] se aplica *co-train* y *self-train* para la desambiguación de significados. En este trabajo se comentan algunos aspectos importantes a tener en cuenta en este tipo de técnicas como son la distribución de clases entre los datos etiquetados o la importancia de la correcta configuración de los distintos parámetros que rigen el funcionamiento de estos métodos. De hecho, este último problema acapara el esfuerzo de gran parte de este trabajo, en el que se definen dos tipos de ajustes: uno óptimo donde se configuran los parámetros en función del conjunto de test que se va a utilizar para la evaluación de los resultados, y otro empírico, en el que se intentan hallar estos parámetros sin hacer uso de este conjunto de frases de test.

Otro aspecto importante a tener en cuenta es que se hace prácticamente imposible mejorar el resultado de un clasificador si los resultados que alcanza son demasiado buenos. En estos casos la aplicación de estas técnicas

se limitará a introducir ruido y empeorar la calidad del resultado final. Es por lo tanto necesario reservar este tipo de técnicas a trabajos "difíciles" como puede ser aumentar un corpus que sólo contiene un número limitado de frases inicialmente, teniendo en cuenta que si el tamaño inicial es suficiente para obtener buenos resultados, difícilmente podremos mejorarlos aplicando *bootstrapping*.

3.2.3. Aportaciones a la Generación de Corpus

Nuestro trabajo se encaminó a obtener un método para incrementar el tamaño de un corpus pequeño de manera totalmente automática o con un mínimo esfuerzo, hasta que adquiera el número deseado de frases. El contenido que se añade al corpus se obtiene de cualquier fuente como puede ser Internet, de la cual se puedan extraer frases sin etiquetar para ser analizadas. Si consideramos el pequeño corpus etiquetado como la semilla, nuestro método hace que evolucione hasta lograr el tamaño deseado. El proceso se basa en la opinión de varios etiquetadores mediante la técnica de *co-training* y de la aplicación de un segundo nivel de aprendizaje mediante *stacking*. Esta última es la técnica que nos sirvió para decidir cuales de las nuevas frases etiquetadas serían seleccionadas para pasar a formar parte del corpus.

Hay muchos trabajos que aplican técnicas de *bootstrapping* a tareas del PLN (por ejemplo [138] y [26]) aunque uno de los que más llamó nuestra atención y que consideramos fuente de inspiración en algunos de nuestros experimentos fue [23]. En él se usa la técnica del *co-training* aunque empleando un método de selección basado en acuerdos y que trabaja seleccionando aleatoriamente un subconjunto de frases nuevas, calculando una tasa de acuerdo entre los diferentes etiquetadores.

El método de selección seguido por [23], mostrado en la figura 3.10, es muy costoso aun habiendo reducido significativamente el número de búsquedas al explorar sólo n subconjuntos aleatoriamente en lugar de analizar todas las posibles particiones. A pesar de este esfuerzo computacional no mejoraban los resultados del método ingenuo sin selección.

La principal motivación de nuestro trabajo fue incluir un criterio de selección más inteligente, lo que perseguimos mediante el uso de algoritmos de aprendizaje automático sobre los ejemplos producidos por los distintos etiquetadores siguiendo el esquema de *stacking*. Esto nos permitió de entrada reducir el coste computacional de manera muy significativa respecto al trabajo antes mencionado.

Los experimentos que llevamos a cabo se realizaron tomando como tareas a resolver el etiquetado *POS*, asignándole a cada una de las palabras de un texto la categoría gramatical correspondiente (sustantivo, adjetivo, de-

C es un grupo de frases etiquetadas por et_1
U es un grupo de frases sin etiquetar
Inicialización:
$c_{max} \leftarrow \emptyset$
$A_{max} \leftarrow 0$
Repetir n **veces**
    $c \leftarrow$ muestra aleatoria de C
    $et'_2 \leftarrow$ etiquetador et_2 enriquecido con c
    $A \leftarrow$ acuerdo entre et_1 y et'_2 sobre U
    Si $(A > A_{max})$
        $A_{max} \leftarrow A$
        $c_{max} \leftarrow c$
    Fin si
Fin repetir

Figura 3.10: Algoritmo de selección usado en [23].

terminante, etc) y el análisis sintáctico superficial o *chunking*. Los recursos necesarios se extrajeron del corpus Penn Treebank y la competición CoNLL-2000 respectivamente.

Nuestro sistema se basaba en intentar tomar ventaja de la combinación de las diferentes vistas presentadas por la técnica de *co-training*, introduciendo un esquema por acuerdos mediante el esquema del *stacking*. También estudiamos los efectos que produce en los resultados la inserción de una fase manual tal y como propone el *active learning*. Este método introduce una fase manual para etiquetar únicamente aquellas frases que presentan dudas para los sistemas automáticos que se están utilizando, que en esta ocasión son las frases que producen desacuerdo entre los diferentes etiquetadores.

Como base del modelo de colaboración ([35] y [36]), se seleccionaron tres etiquetadores reentrenables, TnT [15] que está basado en modelos de Markov, TreeTagger [109] que utiliza árboles de decisión y MBT [29] que emplea un método de aprendizaje basado en ejemplos. También se probaron otros etiquetadores que finalmente no introducían mejoras en el sistema por diversas razones. En algunos casos no se comportaban bien al principio del proceso (con muy poca información de partida), otros introducían demasiado retardo en el sistema en cada iteración mermando en exceso la eficiencia temporal y otros no cumplían con el requisito de aportar una visión diferente del problema al estar basados en métodos similares a los ya presentes.

Baseline

El primer punto a tratar era la creación de un *baseline* que sirviese para comparar los resultados, implementando un sistema sencillo que aportase una solución con un bajo coste. Este sistema lo llamamos *naive co-training* y se muestra en la figura 3.11. Tenemos tres etiquetadores para combinar y tres corpus que se utilizarán para entrenarlos. Cada corpus se extenderá introduciendo frases nuevas etiquetadas por los otros dos. Del conjunto de frases nuevas extraídas de la fuente se escogerá por tanto la mitad con el etiquetado propuesto por un etiquetador y la otra mitad con el propuesto por el segundo etiquetador, pasando a formar parte todas ellas del corpus de entrenamiento del tercero.

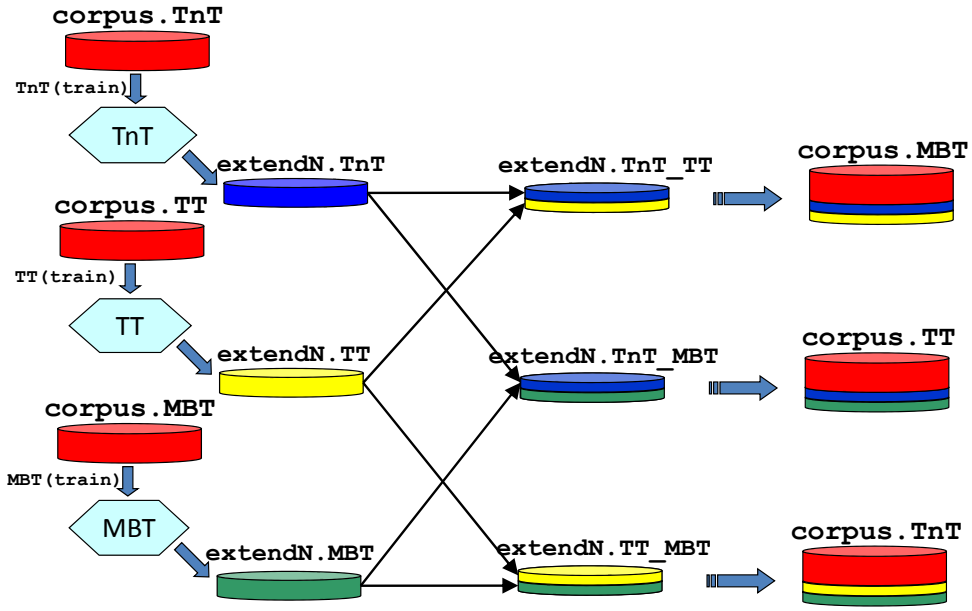

Figura 3.11: Esquema de ejecución para el *co-train* "ingenuo".

En este esquema todas las frases nuevas se introducen en los corpus ya que no existe un criterio de selección entre ellas, provocando un crecimiento homogéneo y constante.

Co-train con Stacking como Método de Selección

Tras establecer un punto de partida con el *naive co-training*, se mejoró la precisión del corpus al final del proceso empleando un esquema de *stacking*,

que aportó un método de arbitraje y consenso entre las diferentes opiniones introducidas por los dos etiquetadores que se combinan en cada ocasión.

Dado que este sistema requería más información para responder correctamente, aplicamos la fase de *stacking* tras cinco iteraciones del método *naive*, ya que resultaba muy difícil para *stacking* extraer información de las pocas frases que contiene el corpus semilla inicial.

Para generar la base de datos de entrenamiento de *stacking*, tomamos las frases etiquetadas desde el comienzo, es decir, las que inicialmente constituyeron el corpus semilla y que al comenzar la fase de *stacking* se extraen del corpus de entrenamiento. Estas frases, que contienen las etiquetas correctas ya que no han sido introducidas de forma automática por el sistema, constituyen el oráculo y se considera la fuente de conocimiento segura en la que se apoya el *stacking*.

Durante el proceso, cada etiquetador se ejecuta sobre el oráculo y sobre el conjunto de frases nuevas que se espera terminen extendiendo los corpus (extraídas en cada iteración de una fuente de frases sin etiquetar). El resultado aportado por dos etiquetadores sobre el oráculo, más las etiquetas reales de que disponemos, componen la base de datos de entrenamiento y el resultado de esos dos etiquetadores sobre el conjunto de frases nuevas constituyen la base de datos de test. El proceso se ilustra en la figura 3.12.

Para aplicar la segunda fase de aprendizaje que representa la técnica de *stacking*, utilizamos un algoritmo de aprendizaje automático basado en árboles de decisión tras ejecutar diferentes algoritmos disponibles y comprobar que éste ofrecía los mejores resultados en la mayoría de las pruebas.

El árbol de decisión es construido a partir de la base de datos de entrenamiento, y es posteriormente usado para decidir las etiquetas de la base de datos de test que ha sido creada a partir de frases nuevas. El árbol proporciona para cada palabra la etiqueta más probable y su correspondiente probabilidad. Si esta probabilidad sobrepasa para cada palabra de una frase un umbral definido previamente, la frase es seleccionada para ampliar el corpus, y en caso contrario se rechaza.

La Fase de Revisión

El esquema que sigue el método *naive* es muy simple y muy rápido pero introduce gran cantidad de errores en los corpus que afectan en las sucesivas iteraciones a los resultados aportados por el sistema. Para intentar paliar este problema antes de que se aplique la fase de *stacking* que sigue a la de *naive*, se incorporó una fase intermedia que revisaba las frases introducidas anteriormente corrigiendo en la medida de lo posible los errores cometidos. La fase de revisión, mostrada en la figura 3.13, consiste en repetir el etiquetado

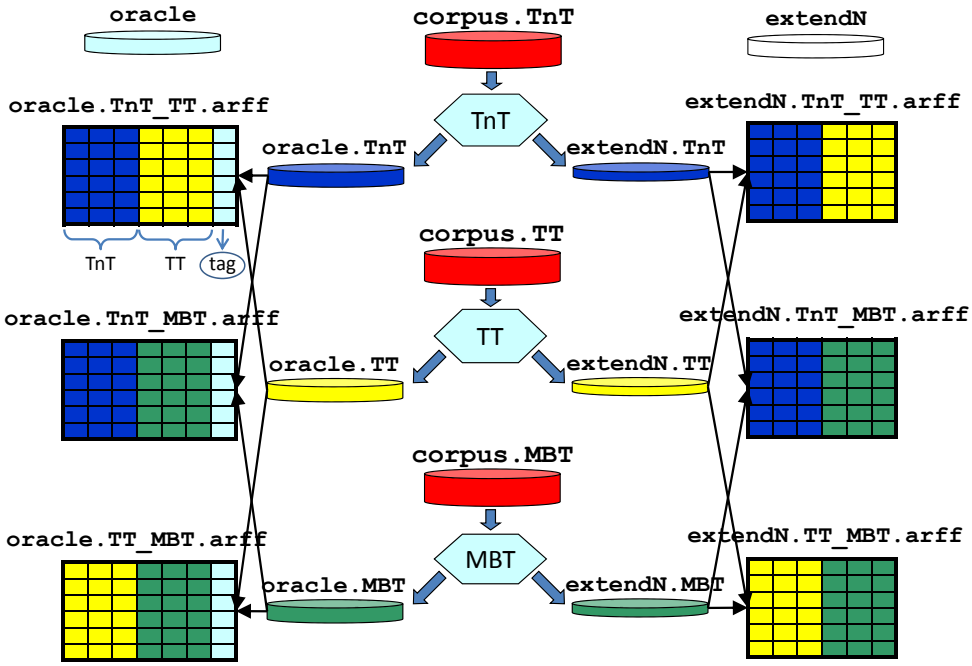

Figura 3.12: Esquema de ejecución para el *co-train* con *stacking*.

de todas las frases introducidas por el método *naive*, pero con el método *stacking*, aprovechando que existen más ejemplos que los disponibles en el corpus semilla del principio.

Aunque durante la revisión no se introducen frases nuevas (puede que incluso se descarten algunas que se habían introducido), los resultados a lo largo del resto del experimento se ven afectados de forma positiva, como comprobaremos en la siguiente sección.

Además de este tipo de revisión, también implementamos otro modo diferente que simula la participación de un experto etiquetando manualmente las palabras que generen más dudas al sistema durante esta fase. En los casos en los que la etiqueta no supere el umbral de fiabilidad especificado al comienzo del experimento, en lugar de ser rechazada se incorporará junto con la etiqueta real en lugar de la propuesta. Se trata de una simulación de etiquetado manual que podemos realizar al obtener las frases nuevas de un corpus etiquetado en lugar de haber sido extraídas de una fuente sin etiquetar. Esto nos da una idea del esfuerzo necesario para mejorar los resultados etiquetando manualmente un cierto número de palabras que en apariencia resultan ser las más difíciles de resolver.

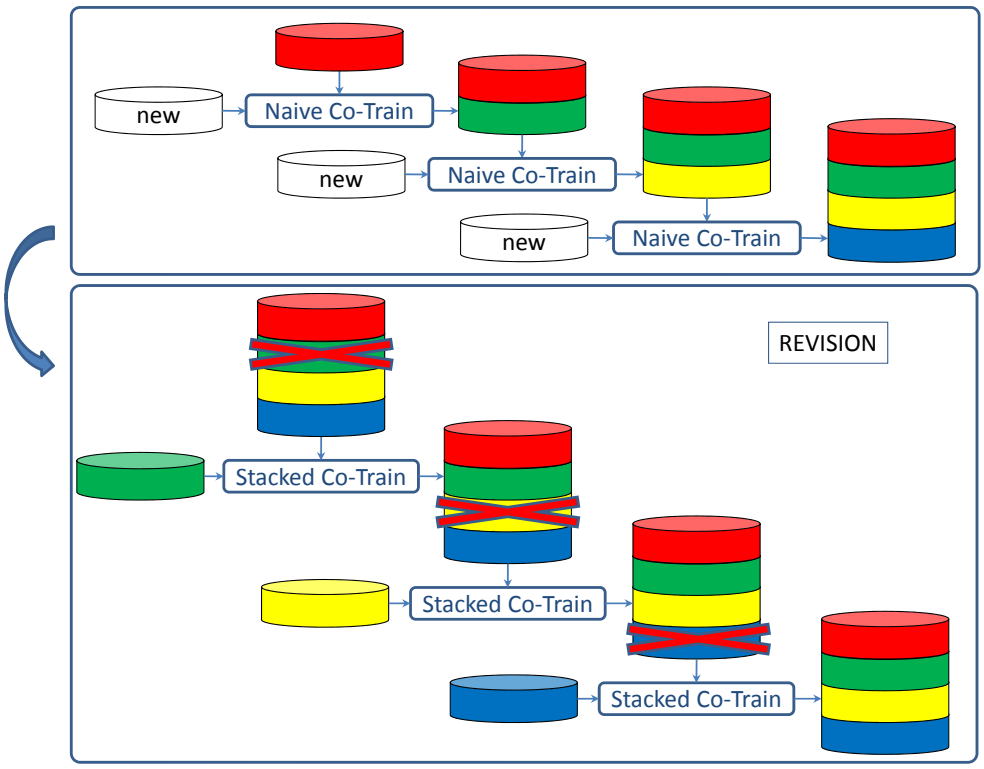

Figura 3.13: Fase de revisión

Resultados Obtenidos

Después de dedicarle tiempo al estudio de las distintas aproximaciones, lo que dio lugar a la publicación [37], comenzamos las pruebas de nuestra propuesta con el corpus Penn Treebank afrontando la tarea del etiquetado POS. En la tabla 3.6 se muestran los resultados obtenidos en cuanto al tamaño alcanzado y la precisión (o *accuracy*) con dicho corpus. Tanto para el método *naive* como para el basado en *stacking*, se muestran los valores de precisión de los corpus al comienzo del experimento y al finalizar, así como la diferencia alcanzada entre estos valores. Todos estos valores se calculan para un corpus unificado generado a partir de los tres corpus que devuelve el sistema. El criterio de selección para mezclar las frases de los tres corpus se basa en las probabilidades de las etiquetas devueltas en cada caso, seleccionando la de mayor probabilidad de acierto (en el caso del método *naive* no tenemos esta probabilidad por lo que se escoge el del mejor etiquetador de entre los tres). El corpus semilla lo componen cincuenta frases y se extraen doscientas frases nuevas en cada iteración como candidatas a extender el corpus hasta alcanzar las cinco mil frases.

	semilla	*Ingenuo*	*Stacking*
TnT	82,27 %	82,43 % (+0,16)	82,69 % (+0,42)
TT	77,21 %	81,11 % (+3,9)	81,4 % (+4,19)
MBT	74,86 %	80,43 % (+5,57)	81,01 % (+6,15)

Tabla 3.6: Resultados con *POS Tagging*.

Los resultados con el etiquetado POS no parecen demasiado esperanzadores ya que la mejora lograda no resulta muy espectacular, aunque al afrontar una tarea más difícil por su mayor dependencia contextual, como es el análisis sintáctico superficial, la situación varía. Como podemos ver en la tabla 3.7, los resultados con el corpus CoNLL-2000 para la tarea de *chunking* muestran una caída sustancial de la precisión del sistema *naive* mientras que el sistema basado en *stacking* se muestra mucho más eficiente, siendo capaz de reponerse ante la caída inicial provocada por las primeras iteraciones del método *naive*. Esto parece indicar, que el sistema de *stacking* es capaz de responder ante la mayor dificultad de esta tarea, al poder manipular de forma más eficiente la información que subyace en el contexto de las etiquetas. Puede aprender ciertas reglas de los ejemplos que poseemos al principio del proceso para responder de forma razonable ante los casos nuevos que van apareciendo en las sucesivas iteraciones.

	semilla	*Ingenuo*	*Stacking*
TnT	76,5 %	70,18 % (-6,32)	76,99 % (+0,49)
TT	68,98 %	69,79 % (+0,81)	72,71 % (+3,73)
MBT	74,19 %	70,04 % (-4,15)	76,14 % (+1,95)

Tabla 3.7: Resultados con *Chunking*.

La figura 3.14 muestra la forma en que evolucionan los corpus unificados generados por ambos sistemas, desde la semilla hasta alcanzar el objetivo de las cinco mil frases.

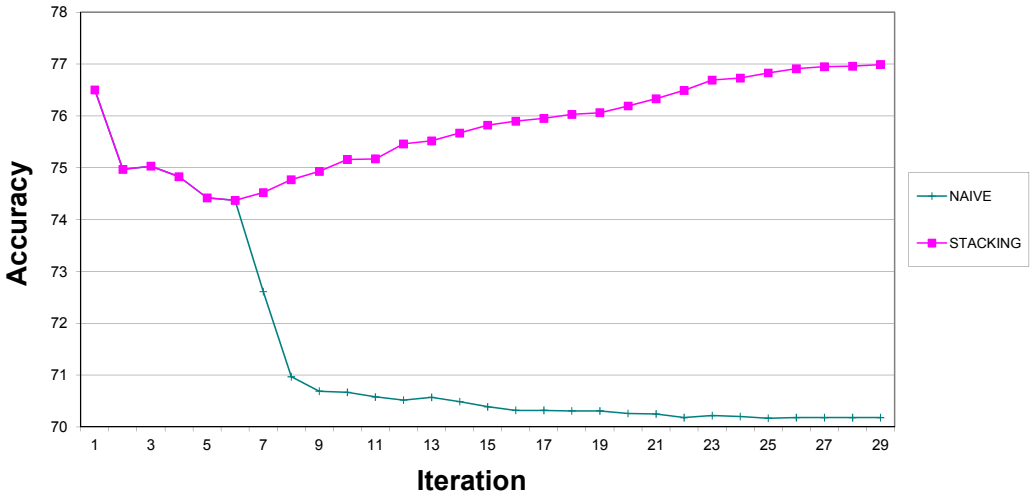

Figura 3.14: Evolución de los resultados en *chunking*.

Finalmente probamos el comportamiento de nuestro sistema introduciendo la fase de revisión en sus dos versiones (automática y mediante active learning simulado) tras la fase inicial de *naive*. Empleamos el corpus de *chunking* al comprobar que es ésta la tarea en la que la aportación de *stacking* resulta más interesante. En la tabla que aparece a continuación observamos una respuesta positiva del sistema y los efectos en los resultados de etiquetar únicamente el 1,58 % de las palabras del corpus (propuestas por el sistema de revisión en los casos en los que la fiabilidad es demasiado baja).

Aunque no existe mucha diferencia entre los resultados de la revisión automática y la activa, observamos que la base de conocimiento generada es mucho más sólida en el segundo caso, quedando patente al introducir una

	semilla	*Automático*	*Activo*
TnT	76,5 %	78 % (+1,5)	78,16 % (+1,66)
TT	68,98 %	73,56 % (+4,58)	73,89 % (+4,91)
MBT	74,19 %	77,7 % (+3,51)	77,84 % (+3,65)

Tabla 3.8: Resultados con *active learning*.

segunda fase de revisión automática al final del experimento. Los resultados se incrementan en dos puntos respecto al logrado por el mejor sistema (TnT) de forma aislada con el corpus semilla inicial, mejorando los resultados obtenidos con ambas revisiones automáticas como se aprecia en la figura 3.15.

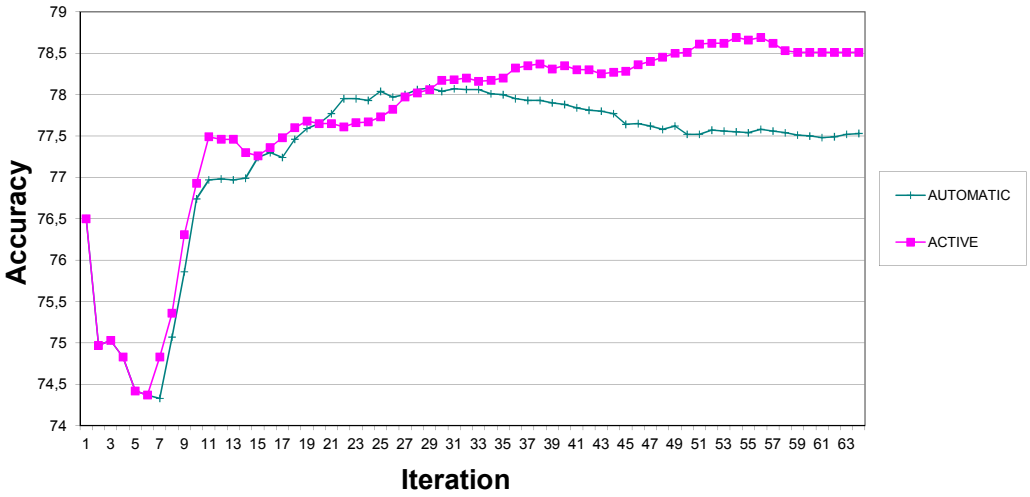

Figura 3.15: Evolución de los resultados con el *active learning*.

La tabla 3.9 junto con su gráfica asociada en la figura 3.16, muestran la comparativa entre los valores alcanzados por el método *naive* en la tarea de *chunking* y el sistema de *stacking* con una breve fase de revisión activa aplicada sobre las cinco primeras iteraciones de *naive* finalizada con una revisión automática de todo el proceso.

La diferencia es obvia siempre considerando el mínimo esfuerzo empleado en este experimento para etiquetar el 1,58 % de forma manual durante la revisión activa.

	semilla	*Ingenuo*	*Stacking*
TnT	76,5 %	70,18 % (-6,32)	78,51 % (+2,01)
TT	68,98 %	69,79 % (+0,81)	74,06 % (+5,08)
MBT	74,19 %	70,04 % (-4,15)	78,3 % (+4,11)

Tabla 3.9: *Co-train* ingenuo vs. *co-train+stacking+active learning*.

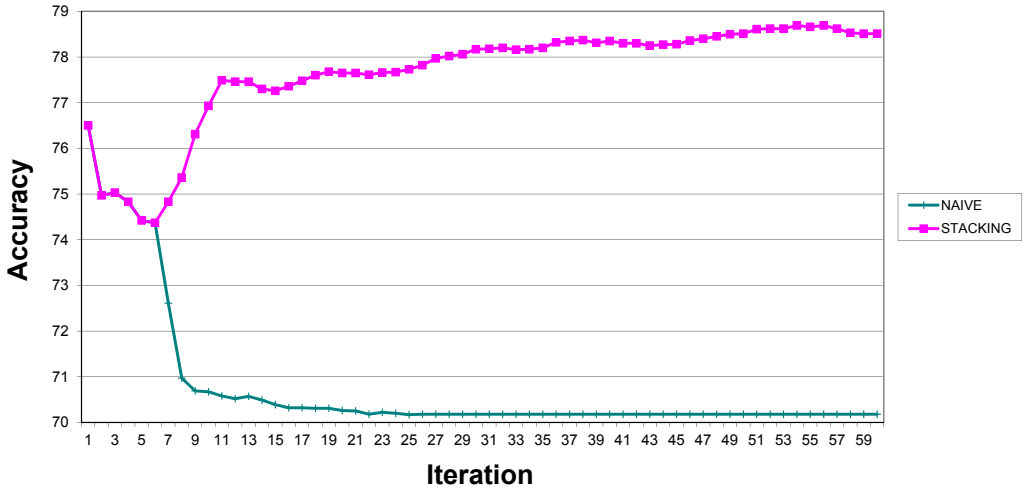

Figura 3.16: Comparativa *co-train* ingenuo vs. con *stacking*+fase activa.

3.3. Conclusiones

A lo largo de este capítulo hemos relatado nuestros primeros experimentos que aplicaban el concepto de combinación de clasificadores a dos tareas muy diferentes.

En primer lugar encontramos una forma de explotar al máximo los recursos disponibles para afrontar la tarea del reconocimiento de entidades. Se consiguió generar la variabilidad necesaria para aplicar los métodos de combinación mediante transformaciones al corpus y al conjunto de etiquetas entre otros, minimizando la necesidad de recursos externos para mejorar los resultados de la clasificación. Las mejoras de alrededor del 5 % nos permitieron acercarnos incluso a los mejores resultados obtenidos en la competición CoNLL 2002 dedicada a esta tarea. En [124] se completa una recopilación de los experimentos realizados y los resultados obtenidos.

La generación de recursos en cambio se mostró como una tarea más com-

plicada en la que los resultados finales no fueron tan satisfactorios, aunque la capacidad de mejora de las técnicas de combinación sirvieron, al igual que los resultados en NER, para animarnos a profundizar más en estos métodos. A partir de aquí ésta se convirtió en nuestra principal línea de investigación, dando lugar al estudio de los métodos de combinación existentes y su aplicación al PLN tal y como hemos visto en los capítulos anteriores, y a un análisis comparativo que expondremos a continuación en el siguiente capítulo.

Capítulo 4

Estudio Comparativo

A lo largo de todo el proceso de análisis de la bibliografía que hemos llevado a cabo, se han recopilado un gran número de datos sobre los resultados que aportan los diferentes métodos de combinación a diversas tareas del PLN. Sin embargo, somos conscientes de la gran variedad de implementaciones y conjuntos de datos utilizados en los trabajos analizados, por lo que se consideró necesaria una fase de experimentación propia que ha dado lugar a este capítulo.

4.1. Metodología

En este estudio no sólo pretendemos determinar cómo se comportan los métodos de combinación en una tarea concreta, sino que procuramos aportar una mayor cantidad de información. Si bien podemos encontrar trabajos como [18], [53] o [52] en los que se analizan varias técnicas de combinación, en este estudio se analiza la combinación aplicada al PLN de una manera que, según nuestra experiencia, no es comparable a ningún trabajo anterior por varias razones que pasamos a comentar.

En primer lugar enfocaremos los experimentos principalmente al etiquetado POS por la disponibilidad de clasificadores ya contrastados y la dificultad que presenta para la combinación debido a los buenos resultados que se obtienen de partida, pero aportando también resultados sobre otras tareas del PLN. La información de estas tareas adicionales es de gran utilidad a la hora de valorar el comportamiento de los diferentes métodos analizados de manera global.

En segundo lugar utilizaremos un total de nueve corpus de POS y otros cuatro destinados a evaluar otras tareas, lo que supone una gran diversidad de datos en cuanto a número de ejemplos, idiomas, conjuntos de etiquetas utili-

zadas, etc, permitiendo comprobar la robustez y consistencia de los métodos evaluados.

En tercer lugar también se han desarrollado implementaciones propias de los distintos métodos de combinación, intentando respetar al máximo las ideas básicas que los sustentan, sin recurrir a optimizaciones *ad-hoc* que puedan desvirtuar las aportaciones en cuanto a lo que el potencial de cada algoritmo se refiere. Hemos implementado al menos un representante de cada una de las familias de métodos de combinación existentes en el estado del arte, salvo por aquellos que requieren entradas del *measurement level*, ya que los clasificadores base no las proporcionan.

Por último, presentamos en el apartado 4.3.4 un número significativo de situaciones o escenarios en donde la combinación se presenta como una opción prometedora, capaz de aportar mejoras o una mayor robustez al sistema de clasificación final. En algunos de estos casos la posibilidad de combinar pasa habitualmente inadvertida por una aparente falta de diversidad o escasez de recursos. En esta línea se han realizado pruebas con diferentes tamaños de corpus, alterando la calidad de los clasificadores, combinando distintos parámetros de los clasificadores base o aprovechando información heterogénea agregándola al sistema de clasificación.

4.2. Clasificadores Base

Se han seleccionado algunas herramientas de etiquetado secuencial diseñadas para resolver tareas de PLN y que cumplen con los requisitos esenciales para aplicar la combinación, es decir, la variedad para aportar diferentes perspectivas del problema, la precisión para que sus resultados sean útiles para el conjunto y la eficiencia para que los métodos de combinación se puedan aplicar sin penalizar en exceso el tiempo de ejecución.

A continuación se explican brevemente los fundamentos teóricos en los que están basados y los parámetros de configuración más significativos. Para todos ellos se ha creado una interfaz homogénea que permite ejecutarlos de forma similar, haciendo uso de ficheros de configuración individualizados que permiten especificar ciertos parámetros que afectan al modo en el que son ejecutados, como la ruta y el nombre de los corpus a utilizar.

4.2.1. TnT

El clasificador TnT [15] (Trigrams'n'Tags)[1] es un etiquetador estadístico desarrollado por Thorsten Brants para la tarea de POS. Se trata de un

[1] http://www.coli.uni-sb.de/ thorsten/tnt

etiquetador muy eficiente entrenable fácilmente para cualquier lenguaje, conjunto de etiquetas o dominio, ya que según el creador no está optimizado para ningún idioma en particular y sí para ser entrenado mediante una gran variedad de corpus y para conseguir una gran velocidad de ejecución. Está basado en modelos de Markov de segundo orden haciendo uso del algoritmo de Viterbi y dispone de diversos métodos de suavizado y de tratamiento de las palabras desconocidas.

4.2.2. TreeTagger

TreeTagger [109] es un etiquetador probabilístico enfocado también a la tarea de POS creado por Helmut Schmid, de la Universidad de Stuttgart[2]. Se distingue de TnT y de los clasificadores basados en n-gramas en general por el uso de árboles de decisión binarios para hallar las probabilidades de las diferentes secuencias de etiquetas posibles (probabilidades de transición). El objetivo es evitar los problemas que tienen estos métodos para estimar pequeñas probabilidades de forma precisa a través de un número limitado de datos de entrenamiento. En la figura 4.1 vemos un ejemplo de estos árboles.

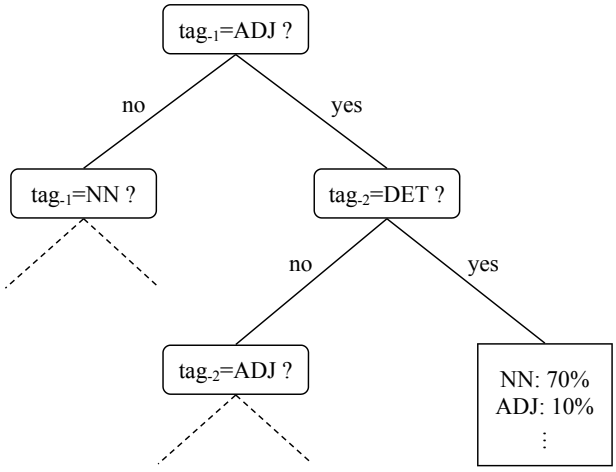

Figura 4.1: Árbol de decision creado por TreeTagger para un ejemplo.

La probabilidad de cada etiqueta se obtiene siguiendo el camino correspondiente a través del árbol hasta encontrar una hoja. La herramienta muestra por defecto únicamente la etiqueta propuesta en cada caso, que será evidentemente la que obtenga una probabilidad mayor.

[2]http://www.ims.uni-stuttgart.de/Tools/DecisionTreeTagger.html

4.2.3. MBT

La herramienta MBT [30] (Memory-Based Tagger)[3] es de nuevo un etiquetador POS desarrollado en esta ocasión por los grupos ILK y CNTS de las Universidades holandesas de Tilburg y Antwerp respectivamente. Hace uso del aprendizaje basado en memoria (*Memory-Based Learning*), que es una adaptación del algoritmo de los vecinos más cercanos (k-NN) utilizado en la clasificación de patrones. Para poder hacer uso de MBT es necesaria la instalación previa del software TiMBL, ya que se trata de una extensión de la funcionalidad ofrecida por éste último. La etiqueta de una palabra en un determinado contexto se extrapola de los casos más similares almacenados en memoria. Este enfoque se sustenta en la asunción de que el razonamiento está basado en la reutilización directa de experiencias pasadas en lugar de la aplicación de conocimiento inducido, como ocurre por ejemplo en los árboles de decisión.

4.2.4. FV

Nuestra aportación a la lista de clasificadores base utilizados consiste en la implementación de un clasificador basado en la generación de vectores de características (*feature vectors*) y su posterior clasificación mediante SVM. Este etiquetador aporta una variante más en cuanto a los fundamentos teóricos en los que se basan los diferentes clasificadores además de estar desprovisto de cualquier optimización ligada estrechamente al POS. Esta característica introduce un rasgo distintivo más que aumenta la variedad de que se dispone, lo que representa un factor primordial en los esquemas de combinación. Para la aplicación de SVM se ha delegado en el software SVM^{light}[62][4], que implementa las máquinas de vectores de soporte en C. Las características implementadas aparecen listadas a continuación y son las más utilizadas en la literatura.

- *Lexical features*: Se recogen mediante expresiones regulares algunos patrones característicos de ciertos tipos de términos o palabras.
- N-gramas: Unigramas, bigramas y/o trigramas que forman el contexto de una determinada palabra.
- Prefijos y sufijos: Prefijos y sufijos de tamaño 1, 2 y 3.
- Longitud de palabra: Característica que recoge el tamaño, en número de caracteres, de la palabra actual.

[3] http://ilk.uvt.nl/mbt/
[4] http://www.cs.cornell.edu/People/tj/svm_light/

- Fin de frase: Característica que recoge el término con el que acaba la frase, habitualmente un punto o un cierre de interrogación.

Para completar el vector se utiliza un esquema de ventana deslizante de tamaño configurable. Los parámetros que se pueden especificar en el fichero de configuración son:

- WINDOW: Tamaño de la ventana deslizante
- FEATURES: Lista de *features* a considerar
- LEXFEAT: Nombre del fichero con las expresiones regulares asociadas a las *lexical features*
- RS: Porcentaje de *random subspace* a aplicar para generar subconjuntos aleatorios de características

4.3. Aplicando Combinación al POS

Entre todas las tareas relacionadas con el etiquetado secuencial de textos, hemos seleccionado el etiquetado POS debido fundamentalmente a los buenos resultados que aportan los clasificadores base utilizados. Esto nos permite poner a prueba los métodos de combinación en un escenario en el que la dificultad se presupone elevada debido al poco margen de mejora con el que contamos inicialmente. No obstante, también se han realizado pruebas en otras tareas que comentaremos más adelante, con las que pretendemos comprobar si se confirman los resultados de esta sección y aportar más información que pueda arrojar luz sobre qué línea seguir a la hora de elegir un método de combinación en concreto.

4.3.1. Corpus Utilizados

En este primer apartado se darán a conocer los corpus utilizados en la fase de experimentación, indicando su procedencia y una pequeña descripción de su contenido. En la tabla 4.1 se muestra un resumen en el que aparecen el número de frases y palabras que contienen, así como el conjunto de etiquetas utilizado, que puede no coincidir con el conjunto de etiquetas original presentando pequeñas modificaciones debido a diversos motivos (por corrección de errores detectados en el corpus, para evitar incompatibilidades con la implementación de los clasificadores, para aumentar comprensibilidad,etc). En los casos en los que sólo se disponía de un conjunto de datos se extrajeron un 10 % de los ejemplos para utilizarlos como *test*, dejando el resto para el entrenamiento.

CORPUS	Idioma	Tags	TRAIN		TEST	
			frases	palabras	frases	palabras
Brown	ING	83	14101	1048112	1566	113080
CoNLL'00	ING	34	8936	211727	2012	47377
CoNLL'02	HOL	13	15806	202931	2895	37761
CoNLL'07	ESP	15	2949	75822	563	19206
	EUS	23	2595	40032	580	10096
Floresta	POR	24	8340	195538	926	17113
Susanne	ING	131	5754	141140	830	15482
Talp	ESP	11	3492	91400	389	9071
Treebank	ING	37	25117	766463	1513	46461

Tabla 4.1: Resumen de los corpus utilizados.

Brown

El corpus Brown[5] se define como:

> A Standard Corpus of Present-Day Edited American English, for use with Digital Computers.

Fue desarrollado en el Departamento de Lingüística de la Universidad de Brown por W. N. Francis y H. Kucera en 1964, aunque ha sido revisado y ampliado posteriormente en varias ocasiones. Además de transformar el formato al de una palabra por línea por razones de implementación, se han unido todas las secciones de las que está compuesto el corpus y se ha simplificado ligeramente el conjunto de etiquetas original.

CoNLL 2000

Se trata del corpus utilizado en la competición CoNLL en su edición del año 2000[6], dedicada a la tarea de *chunking*. Los datos estaban compuestos originalmente por tres columnas separadas por espacios quedando cada palabra representada por una línea del corpus. La primera columna es la que contiene la palabra, la segunda la etiqueta POS y la tercera la etiqueta de *chunking* correspondiente. La existencia de la columna POS nos permite utilizar este corpus para la tarea del POS ignorando la columna de *chunking*. No

[5] http://www.hit.uib.no/icame/brown/bcm.html
[6] http://www.cnts.ua.ac.be/conll2000

obstante, también hemos experimentado con algún caso particular de combinación en el que se intenta aprovechar la posibilidad de contar con esta información heterogénea.

CoNLL 2002

La edición del año 2002 de la competición CoNLL[7] se dedicó a la tarea NER y se desarrolló un corpus en español y otro en holandés. En este último existe una columna con la etiqueta POS, por lo que ofrece la posibilidad de utilizar este corpus para dicha tarea y no únicamente para el reconocimiento de entidades. Los datos provienen de cuatro ediciones del periódico belga "De Morgen"del año 2000 y el etiquetado fue llevado a cabo por miembros de la Universidad de Antwerp.

CoNLL 2007

En su edición del año 2007, la competición CoNLL[8] estuvo centrada en el análisis de dependencias. Para ello se utilizaron corpus de los que se hicieron públicos un subconjunto de los mismos distribuidos bajo los términos de licencia Creative Commons. En este caso sólo hemos hecho uso de la información referente al etiquetado POS en los idiomas español y euskera.

Floresta

El proyecto Floresta[9] ha producido un corpus analizado sintácticamente en portugués, del cuál hemos utilizado la información morfosintáctica para la evaluación de la tarea de etiquetado POS.

Susanne

El corpus Susanne[10] fue creado en la Universidad de Sussex por el equipo de Geoffrey Sampson. Está basado en 64 de los 500 textos de los que se compone el corpus Brown y parte de un conjunto de etiquetas muy extenso. Además de transformar el formato al de una palabra por línea, se ha llevado a cabo una pequeña simplificación del enorme conjunto de etiquetas inicial por razones de implementación.

[7] http://www.cnts.ua.ac.be/conll2002
[8] http://nextens.uvt.nl/depparse-wiki/SharedTaskWebsite
[9] http://www.linguateca.pt/Floresta/
[10] http://www.grsampson.net/Resources.html

CLiC-TALP

El corpus CLiC-TALP[11] es un corpus en español de 100.000 palabras analizado morfológicamente y desambiguado manualmente. Fue desarrollado por el Servei de Tecnologia Lingüística (STeL), adscrito a la Facultad de Filología de la Universidad de Barcelona.

Penn Treebank

Finalmente, el Penn Treebank está disponible a través del Linguistic Data Consortium de la Universidad de Pennsylvania y se compone de un millón de palabras provenientes de material extraído del Wall Street Journal durante 1989. Considerando que los corpus de análisis sintáctico se dividen en dos grandes grupos, aquellos que se basan en el etiquetado de las diferentes categorías sintácticas, y los que se centran en las estructuras de dependencias, el Penn Treebank es uno de los más populares del primero de ellos.

4.3.2. Resultados Iniciales

Con el objetivo de conocer el punto de partida para los diferentes métodos de combinación en los que se basa esta experimentación, se han ejecutado los clasificadores base de forma independiente y haciendo uso de los parámetros por defecto, obteniendo los resultados de la tabla 4.2.

Corpus	FV	MBT	TnT	TT
brown	96,18	95,82	**96,55**	95,64
conll00Pos	96,41	96,8	**97,32**	96,41
conll02nedPos	95,01	95,79	**96,16**	88,53
conll07esp	95,35	95,01	**95,98**	95,44
conll07eus	91,27	90,59	93,73	**94,13**
floresta	96,52	95,81	**97,02**	96,66
susanne	92,26	91,16	**93,61**	91,27
talp	94,59	94,8	**95,82**	95,62
treebankWSJ	**96,28**	95,67	96,21	95,52

Tabla 4.2: Resultados de los clasificadores base.

Como se puede apreciar, TnT obtiene en la mayoría de los casos el mejor resultado, lo que deberá traducirse en un mayor peso a la hora de considerar las diferentes opiniones durante el proceso de combinación. También es

[11] http://clic.fil.ub.es/demos/

importante el hecho de que entre el resto de clasificadores no se mantiene siempre el mismo orden en cuanto a los mejores resultados obtenidos y que las diferencias oscilan en cantidades significativas cuando cambiamos de un conjunto de datos a otro.

4.3.3. Aplicación de los Métodos

Una vez detallados los corpus y clasificadores base disponibles para experimentar con la combinación de sistemas, nos centramos ahora en los métodos de combinación que se van a poner en práctica. Al igual que ocurría con el clasificador FV, decidimos implementar nosotros mismos los algoritmos de combinación para evitar las posibles optimizaciones que puedan residir en implementaciones creadas por terceros, buscando así la máxima correspondencia entre los resultados y el potencial de cada algoritmo original. Esto nos permite además obtener un nivel de homogeneidad y flexibilidad que facilita crear esquemas de clasificación combinada más complejos como puede ser el método *cascading*. A continuación se muestra una breve descripción de los métodos, y en cada caso se aporta una tabla en la que se muestran las mejoras que se aportan respecto al mejor resultado obtenido por un clasificador base involucrado en el esquema de combinación, considerando cada corpus por separado.

Votación (VT)

Los métodos de votación relacionan el concepto de 'elecciones' al problema de clasificación, el de 'votantes' a los clasificadores implicados y el de 'votos' a las etiquetas propuestas por cada uno de ellos, permitiendo aplicar los métodos de elección contempladas en la Teoría de elección social [3], cuyo objetivo es el estudio de la toma de decisiones colectivas en función de las preferencias individuales.

Como representantes de los diferentes métodos de votación existentes se han implementado las dos versiones más populares que son la votación por mayoría plural y la votación ponderada. Hay que destacar el requisito establecido por otras variantes como la mayoría simple o por unanimidad de que se permita la opción *reject*, al existir la posibilidad de que no se elija ningún candidato, algo que no siempre es factible en las tareas que estamos tratando. Para estimar los pesos asociados a los diferentes participantes en la votación ponderada se ha optado por el método *hold-out* (ver el apartado 1.1.3).

Si combinamos los cuatro clasificadores aplicando las dos variantes de votación a los diferentes corpus se obtienen los resultados de la tabla 4.3,

en la que se aprecia un mejor rendimiento del método simple (VT) frente al ponderado (VTw). Esto se debe a que el número de clasificadores no es muy grande, lo que provoca que el mejor de ellos obtenga una posición de privilegio (mayor coeficiente) y al resto les cueste unir votos suficientes para cambiar la etiqueta aportada por dicho clasificador. En muchos casos no se logra mejorar sino tan solo igualar el resultado del mejor clasificador base pero esto no debe considerarse un defecto ya que puede ser muy útil en ciertas situaciones. Por ejemplo cuando desconocemos cual de nuestros clasificadores es el mejor y en situaciones "conservadoras" en las que deseamos asegurarnos un resultado que no empeore el que obtendríamos utilizando únicamente el mejor clasificador base disponible.

CORPUS	VT	VTw
brown	0,49	0,24
conll00Pos	0,45	0,00
conll02nedPos	0,68	0,00
conll07espPos	0,98	0,00
conll07eusPos	0,16	0,00
floresta	0,63	0,00
susanne	0,71	0,00
talp	0,76	0,33
treebankwsjPos	0,45	0,12
PROMEDIO	0,59	0,08

Tabla 4.3: Resultados de votación.

Bayes (BAY)

Los métodos basados en el Teorema de Bayes hacen uso de la regla que permite obtener la probabilidad de un evento A dado B en términos de la probabilidad del evento B dado A y la probabilidad de sólo A. De esta forma, si tenemos una serie de eventos $A_i \dots A_n$ independientes entre sí con probabilidad distinta de cero y un suceso B, podemos definir la probabilidad a posteriori $P(B|A_i)$, tal y como presentamos en el apartado 1.5.2. La implementación que se ha llevado a cabo es una aplicación directa de esta regla. La probabilidad a posteriori que debemos calcular es la probabilidad de que dadas c etiquetas posibles y dada una serie de observaciones $s = s_1, \dots, s_L$ que representa las etiquetas propuestas por los L clasificadores, la etiqueta real sea ω_k, con $k = 1, \dots, c$.

La ejecución de este método con los cuatro clasificadores sobre los distintos corpus arroja los resultados reflejados en la tabla 4.4. Apreciamos de nuevo una mejora significativa en todos los corpus utilizados, por lo que debemos considerar este método como una opción sencilla pero muy rentable en cuanto a los resultados que obtiene.

CORPUS	BAY
brown	0,39
conll00Pos	0,37
conll02nedPos	0,52
conll07espPos	0,88
conll07eusPos	0,43
floresta	0,55
susanne	0,67
talp	0,96
treebankwsjPos	0,27
PROMEDIO	0,56

Tabla 4.4: Resultados de Bayes.

Behavior Knowledge Space (BKS)

Los métodos basados en la memorización consisten en recordar el comportamiento demostrado por los clasificadores en los ejemplos de entrenamiento, para luego tomar decisiones sobre los nuevos elementos basándose en el pasado. Como representante en este caso de los métodos de memorización, se ha llevado a cabo la implementación del método *behavior knowledge space* siguiendo las directivas explicadas en el apartado 1.5.3.

Una vez más, se ha ejecutado este método con los cuatro clasificadores base sobre todos los corpus comentados anteriormente, dando lugar a los valores mostrados en la tabla 4.5. Las mejoras son una vez más generalizadas en todos los corpus y de mayor envergadura aún respecto a los anteriores métodos, llegando a un máximo de $1,36$ para el corpus susanne, que se caracteriza por su dificultad al contar con más de un centenar de etiquetas posibles.

Bagging (BAG)

Entre los métodos que generan variabilidad en los datos destaca *bagging* (BAG), que combina diferentes versiones del corpus creadas mediante un

CORPUS	BKS
brown	0,63
conll00Pos	0,65
conll02nedPos	0,67
conll07espPos	1,10
conll07eusPos	0,51
floresta	0,72
susanne	1,36
talp	1,08
treebankwsjPos	0,47
PROMEDIO	0,80

Tabla 4.5: Resultados de *behavior knowledge space*.

muestreo con reemplazamiento. Centrados en este método de combinación se han efectuado varios tipos de experimentos:

- BAG: *bagging* aplicado sobre un clasificador. Hemos seleccionado el clasificador TnT al ser el que mejor resultados obtiene para comprobar si el método es capaz de mejorarlos. El esquema de ejecución se muestra en la figura 4.2, en donde vemos que el corpus de entrenamiento original T genera n versiones de sí mismo $(TS^1, ..., TS^n)$, que sirven para generar los n modelos a través del clasificador c, cada uno de los cuales aporta un etiquetado para el corpus de test t.

- BAGm: *bagging* aplicado sobre varios clasificadores, de manera que compartan los *bags* generados en cada iteración. En este caso se ha experimentado con TnT, TreeTagger y MBT, excluyendo FV al ser el que más tiempo de ejecución consume, algo que en un proceso iterativo como este perjudica en gran medida la eficiencia del sistema. El esquema de ejecución se muestra en la figura 4.3, en donde se aprecia como los diferentes muestreos del corpus de entrenamiento T se utilizan para crear tantas versiones del corpus de test t como clasificadores base estemos utilizando $(c^1, ..., c^n)$.

- BAGmw: *bagging* aplicado sobre la base de datos generada con las salidas de todos los clasificadores mediante la herramienta Weka. El esquema de ejecución en este caso coincide con el de *stacking* pero seleccionando *bagging* como clasificador de Weka. De hecho se trata de un esquema de *cascading*, ya que se ejecutan los clasificadores base

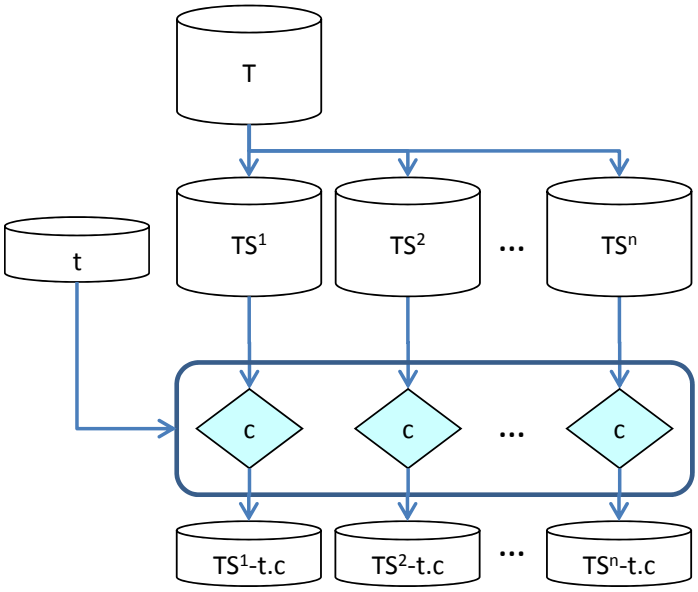

Figura 4.2: Esquema de ejecución del *bagging* "simple" (BAG).

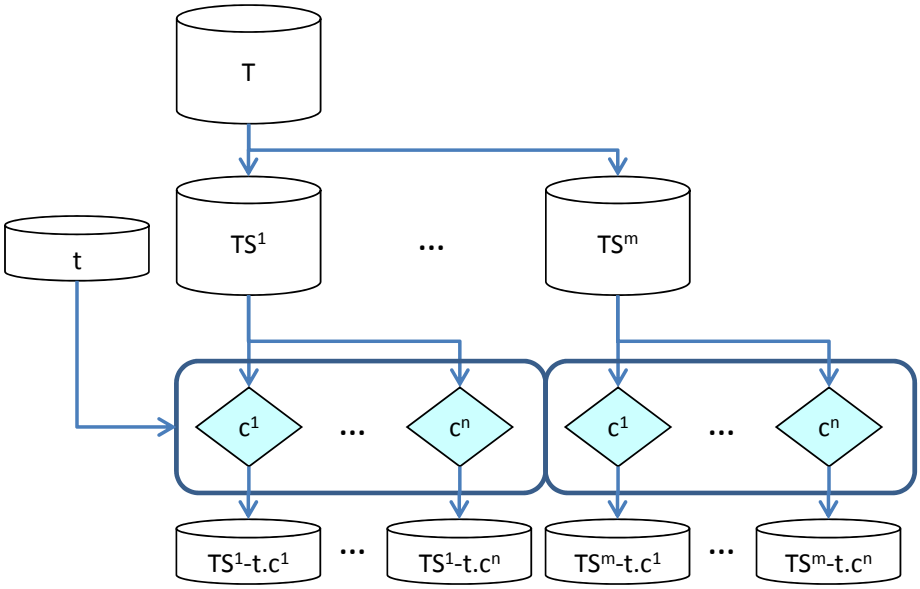

Figura 4.3: Esquema de ejecución del *bagging* "múltiple" (BAGm).

para generar la base de datos y Weka utiliza un clasificador propio para posteriormente aplicar el método de *bagging*.

CORPUS	BAG	BAGm	BAGmw
brown	0,03	0,22	0,32
conll00Pos	0,00	0,17	0,55
conll02nedPos	0,00	0,24	0,51
conll07espPos	0,03	0,15	0,90
conll07eusPos	0,04	0,25	0,22
floresta	-0,01	0,26	0,36
susanne	0,00	0,25	0,81
talp	0,01	0,49	0,75
treebankwsjPos	0,01	0,14	0,35
PROMEDIO	0,01	0,24	0,53

Tabla 4.6: Resultados de *bagging*.

Los resultados, obtenidos con 30 iteraciones, muestran que no hay mejoras para el esquema "simple", probablemente debido a que TnT genera clasificadores de similares características para todos los *bags*. En cambio sí existen mejoras en los esquemas más complejos como se aprecia en la tabla 4.6.

Tanto BAG como BAGm se han implementado no como métodos independientes sino como modificadores de los clasificadores participantes. Este modificador creará diferentes versiones del *tagger*, a las que posteriormente se les puede aplicar cualquier método de combinación. En el caso de *bagging*, se entiende que tras crear los conjuntos de datos mediante remuestreo con reemplazamiento, se debe aplicar la votación simple, aunque en nuestro caso esta es una decisión libre, pudiéndose aplicar otro sistema de combinación diferente. Si aplicamos el método de *stacking* en lugar de la votación obtendremos los resultados mostrados en la tabla 4.7, que como se ve son ligeramente superiores a los del método aplicado habitualmente.

Error Correcting Output Codes (ECOC)

Otro de los sistemas de combinación implementados es el método *Error Correcting Output Codes*, que genera particiones binarias del conjunto de etiquetas y asigna un clasificador binario a cada una de estas divisiones para finalmente tomar la decision sobre la etiqueta final combinando las salidas de estos clasificadores binarios (ver el apartado 1.5.11).

CORPUS	BAGm-SG
brown	0,25
conll00Pos	0,28
conll02nedPos	0,37
conll07espPos	0,64
conll07eusPos	0,22
floresta	0,23
susanne	0,30
talp	0,62
treebankwsjPos	0,22
PROMEDIO	0,35

Tabla 4.7: Resultados de *bagging* combinando con *stacking*.

El funcionamiento de este método depende en gran manera de la matriz que codifica las particiones binarias que están siendo tenidas en cuenta, siendo posible tanto su creación aleatoria como el despliegue exhaustivo de todas las particiones binarias posibles del conjunto de etiquetas. Por desgracia, el gran número de etiquetas presentes en la tarea que nos atañe impide la utilización de matrices exhaustivas, ya que su tamaño crece exponencialmente con el tamaño del conjunto de etiquetas. Quizás esta sea la razón por la cual los resultados obtenidos no han sido los esperados habiendo aplicado la mejor matriz de códigos obtenida aleatoriamente tras cien iteraciones. Como se aprecia en la tabla 4.8, no se han podido obtener mejoras en ninguno de los corpus presentes en este estudio utilizando el clasificador TnT, seleccionado de nuevo por ser el que genera mejores resultados.

Stacking (SG)

El método *stacking* es el más popular de los basados en el meta-aprendizaje, y se nutre de los resultados de los clasificadores base para generar una nueva base de datos sobre la que aplicar un segundo nivel de aprendizaje. La implementación que se ha llevado a cabo se corresponde con las ideas originales sobre el método *stacked generalization*, que permite aprovechar al máximo la información contenida en el corpus de entrenamiento. Los diferentes clasificadores participantes se entrenan y ejecutan sobre distintas particiones del corpus de entrenamiento para formar la base de datos de entrenamiento en forma de conjunto de vectores de etiquetas, y finalmente sobre el corpus de test para generar la base de datos de test. Estas dos bases de datos se

CORPUS	ECOC
brown	-4,35
conll00Pos	-2,44
conll02nedPos	-1,22
conll07espPos	-2,60
conll07eusPos	-1,39
floresta	-2,82
susanne	-5,73
talp	-3,70
treebankwsjPos	-2,95
PROMEDIO	-3,02

Tabla 4.8: Resultados de *ECOC*.

generan en el formato *arff* utilizado por la herramienta Weka, que se encarga del aprendizaje de segundo nivel que determina finalmente la etiqueta a seleccionar de entre las propuestas en cada vector.

Los valores obtenidos sobre los cuatro clasificadores base son los reflejados en la tabla 4.9. Del resultado obtenido por este método se observa un gran potencial ya que parece aprovechar la capacidad que le otorga el segundo nivel de clasificación para reconocer patrones en los que la etiqueta propuesta por un clasificador en inferioridad numérica pueda ser seleccionada frente al resto.

CORPUS	SG
brown	0,64
conll00Pos	0,71
conll02nedPos	0,66
conll07espPos	1,34
conll07eusPos	0,49
floresta	0,78
susanne	1,26
talp	1,10
treebankwsjPos	0,59
PROMEDIO	0,84

Tabla 4.9: Resultados de *stacking*.

Selección de Características

Otro punto en el proceso de clasificación en donde se puede generar variabilidad es a la hora de generar los vectores de características que definen un ejemplo concreto dentro de la base de datos. Para cubrir este tipo de métodos que realizan selección de características se han llevado a cabo dos tipos de experimentos.

La primera aproximación consiste en la implementación del método *random subspace method*, aunque no se ha implementado como un método independiente sino como un modificador aplicable al clasificador FV, seleccionando los subconjuntos de características de forma aleatoria. En la lista de clasificadores a combinar se puede entonces especificar FV con el modificador RS y el número de iteraciones a realizar junto al porcentaje de probabilidad que se desea utilizar para saber si se selecciona una característica o no. Generando 30 versiones del clasificador FV seleccionando las características a considerar en cada versión con un 50 % de probabilidades para cada una, los resultados arrojados por este método no mejoran en ningún caso a los que se obtienen mediante el clasificador FV con todas las características, registrándose pérdidas de $0,39$ puntos de media. La explicación a nuestro modo de ver reside en la correlación existente entre los diferentes tipos de características que hace que eliminar aleatoriamente elementos de diferentes grupos reduzca las probabilidades de extraer conocimiento del conjunto de datos por parte de los métodos de combinación. Esta creencia nos lleva a la segunda vía para experimentar con este tipo de combinación que explicamos a continuación.

La segunda aproximación consiste en realizar agrupaciones naturales de las características que utiliza el clasificador FV para luego proceder a su combinación. De esta forma hemos generado las siguientes versiones del clasificador FV:

- FV: Versión "completa" que hace uso de todas las características posibles indicadas anteriormente.
- FVb: Incluye todas las características salvo las *lexical features*.
- FVc: Equivalente a la version "completa" sin la información de bigramas ni tri-gramas aunque sí con uni-gramas.
- FVd: Equivalente de nuevo a la version "completa" eliminando en este caso la información de prefijos y sufijos.

Los resultados iniciales que aportan como clasificadores base figuran en la tabla 4.10, junto a las mejoras obtenidas combinando las tres versiones

reducidas (C-A) o combinando las cuatro versiones de FV (C-B). En ella vemos que la selección de características puede aportar mejoras aunque debe considerar la relación existente entre los diversos tipos de características contempladas. Para la combinación se ha optado por utilizar *stacking* dado que, tal y como se ha visto en el apartado anterior, hace gala de una gran eficacia y robustez.

CORPUS	FV	FVb	FVc	FVd	C-A	C-B
brown	96,18	95,69	95,89	94,35	0,35	0,36
conll00Pos	96,41	94,70	96,17	93,87	0,23	0,30
conll02nedPos	95,01	94,84	94,60	93,11	0,49	0,49
conll07espPos	95,35	94,82	95,17	91,75	0,03	0,05
conll07eusPos	91,27	91,41	91,17	84,78	0,12	0,11
floresta	96,52	95,35	96,27	93,95	0,09	0,11
susanne	92,26	91,21	91,98	88,90	0,08	0,20
talp	94,59	94,61	94,42	90,62	0,11	0,20
treebankwsjPos	96,28	94,70	96,07	95,00	0,19	0,19
PROMEDIO	94,87	94,15	94,64	91,81	0,18	0,22

Tabla 4.10: Resultados con las versiones de FV.

Cascading (CASC)

Otro método implementado es el llamado *cascading*, aunque de forma diferente al resto ya que se encuentra implícito en el esquema de implementación que se ha seguido para todo el sistema, a través de las interfaces estandarizadas que permiten encadenar los combinadores de muchas formas diferentes. Esto hace posible utilizar la salida de un sistema de combinación como entrada para otro como si de un clasificador base se tratase.

Para comprobar los resultados que se pueden obtener encadenando métodos, se han ejecutado diferentes esquemas de combinación en tres niveles. Los resultados de los cuatro clasificadores base se utilizan como entrada para los diferentes métodos de combinación y las salidas de éstos de nuevo sirven como entrada para un método de combinación diferente. En la figura 4.4 se muestra el esquema con *stacking* como método de combinación de segundo nivel (CASC-SG en la tabla 4.11) y el resto de experimentos siguen el mismo esquema intercambiando el método que ocupa este segundo nivel de clasificación.

En los resultados se aprecia que las mejoras son muy significativas para

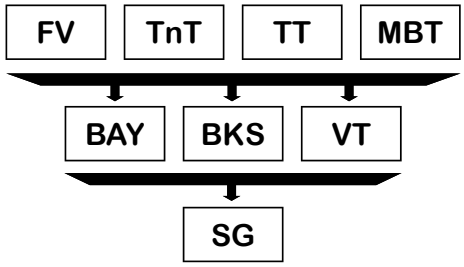

Figura 4.4: Esquema de ejecución de *cascading*

todos los esquemas que se han probado, llegando a superar incluso las mejores cifras obtenidas hasta el momento. Además hay que destacar la robustez de este sistema que mantiene niveles muy buenos de precisión independientemente del clasificador que ocupa el nivel superior del esquema.

CORPUS	CASC-BAY	CASC-BKS	CASC-SG	CASC-VT
brown	0,69	0,66	0,68	0,67
conll00Pos	0,67	0,67	0,66	0,66
conll02nedPos	0,67	0,66	0,72	0,67
conll07espPos	1,23	1,23	1,18	1,18
conll07eusPos	0,52	0,52	0,50	0,59
floresta	0,77	0,81	0,73	0,71
susanne	1,33	1,35	1,15	1,52
talp	1,14	1,08	1,12	1,18
treebankwsjPos	0,57	0,56	0,49	0,55
PROMEDIO	0,84	0,84	0,80	0,86

Tabla 4.11: Resultados de *cascading*.

4.3.4. Experimentos adicionales

Una vez aplicados los métodos de combinación y comprobando las mejoras que pueden aportarnos, hemos querido profundizar aún más diseñando algunos experimentos adicionales, planteando escenarios y esquemas que puedan ser de utilidad en la práctica y respondiendo algunas preguntas sobre lo que ocurriría si se diesen ciertas circunstancias. Para ello hemos utilizado los métodos que se han comportado mejor en las pruebas realizadas en el apartado anterior.

Eliminando el Mejor

¿Qué ocurre si se elimina al mejor clasificador base del esquema de combinación? Hasta ahora hemos llevado a cabo los experimentos con los cuatro clasificadores base y cuando había que escoger uno se ha elegido TnT por ser el que con mayor frecuencia aporta el mejor resultado. Ahora haremos lo contrario, eliminándolo del sistema para ver si la combinación sigue proporcionando mejoras respecto a los resultados individuales de los clasificadores restantes. Tras eliminar el mejor participante para cada corpus se obtienen los resultados de la tabla 4.12, donde comprobamos que la mejora es aún mayor que cuando contábamos con todos los clasificadores.

CORPUS	BAY	BKS	SG	VT
brown	0,64	0,86	0,94	0,66
conll00Pos	0,84	0,98	1,11	0,91
conll02nedPos	0,99	1,04	1,00	0,99
conll07espPos	1,51	1,43	1,51	1,22
conll07eusPos	0,28	0,64	0,68	0,29
floresta	0,86	0,98	0,96	0,82
susanne	1,35	2,19	1,95	1,24
talp	0,93	1,09	1,11	0,85
treebankwsjPos	0,07	0,15	0,31	0,08
PROMEDIO	0,83	1,04	1,06	0,78

Tabla 4.12: Resultados eliminando el mejor clasificador.

La combinación por lo tanto puede ayudar aún más cuando no disponemos de los mejores clasificadores para la tarea con la que estamos trabajando. De hecho en la tabla 4.13 comprobamos que casi siempre se mejoran significativamente los resultados que habríamos obtenido ejecutando el mejor clasificador disponible para cada corpus, lo que hace pensar que es mejor combinar varios clasificadores "regulares" que ejecutar uno sólo, aunque sea mejor que el resto.

La Calidad del Corpus

¿Cómo afecta la calidad del corpus a los resultados de la combinación? Para responder a esta pregunta hemos creado tres versiones adicionales del corpus Penn Treebank disminuyendo el número de palabras que contiene. De esta forma contamos con las siguientes versiones:

CORPUS	BAY	BKS	SG	VT
brown	0,27	0,49	0,57	0,29
conll00Pos	0,32	0,46	0,59	0,39
conll02nedPos	0,56	0,61	0,57	0,56
conll07espPos	0,97	0,89	0,97	0,68
conll07eusPos	-0,12	0,24	0,28	-0,11
floresta	0,50	0,62	0,60	0,46
susanne	0,00	0,84	0,60	-0,11
talp	0,73	0,89	0,91	0,65
treebankwsjPos	0,00	0,08	0,24	0,01
PROMEDIO	0,36	0,57	0,59	0,31

Tabla 4.13: Resultados respecto al mejor clasificador (eliminado).

- treebankwsjPos: versión completa con 766463 palabras.
- treebankwsjPos200k: versión reducida con 198550 palabras.
- treebankwsjPos100k: versión reducida con 95924 palabras.
- treebankwsjPos50k: versión reducida con 47739 palabras.

Tras ejecutar los diferentes métodos de combinación se obtienen los resultados de la tabla 4.14, donde apreciamos que incluso en la versión más reducida se siguen obteniendo mejoras respecto a los clasificadores, teniendo en cuenta que aportan muchos más errores al sistema debido a la escasez de datos de entrenamiento. También es interesante ver cómo *stacking* es capaz de obtener mejoras mayores con el corpus más reducido que con el resto, lo cual hace pensar que es bastante robusto y mientras los clasificadores bajan su calidad debido a la falta de datos, el resultado de la combinación no disminuye en la misma cuantía. Además hemos añadido los resultados obtenidos sin el mejor clasificador, que en este caso es FV, contrastando los datos del apartado anterior y comprobando que se cumplen las previsiones de mejoras aún mayores en este caso.

Información Heterogénea

En este apartado vamos a intentar explotar una característica propia de *stacking*, que consiste en integrar información heterogénea al proceso de aprendizaje para intentar así extraer conocimiento desde fuentes adicionales. Esta fase de experimentación consiste en varios esquemas de ejecución que

CORPUS	BAY	BKS	SG	VT	SG-sinFV
treebankwsjPos	0,27	0,47	0,59	0,45	0,31
treebankwsjPos200k	0,33	0,57	0,56	0,47	0,84
treebankwsjPos100k	0,34	0,49	0,65	0,41	1,06
treebankwsjPos50k	0,25	0,22	0,73	0,36	1,22
PROMEDIO	0,30	0,44	0,63	0,42	0,86

Tabla 4.14: Resultados con versiones reducidas del corpus.

generan la base de datos de *stacking* con las salidas de los clasificadores junto a otros valores de diferentes tipos.

Una primera aproximación sería añadir información léxica sobre las palabras del corpus que se está etiquetando. Concretamente hemos utilizado las siguientes expresiones regulares para detectar las características léxicas:

- Todo minúsculas: `^[a-z]+$`
- Empieza con mayúscula: `^[A-Z].*$`
- Todo mayúsculas: `^[A-Z]+$`
- Abreviaturas: `^[A-Z].*\.$`
- Siglas: `^[A-Z]\.([A-Z]\.)+$`
- Números: `^[0-9]+$` `^[0-9]+[,.][0-9]+$`
- Rango: `^[0-9]+"-"[0-9]+$`
- Cantidad: `^[0-9]+[,.]?[0-9]+[%$]$`
- 4-Números: `^[0-9][0-9][0-9][0-9]$`
- Hora: `^[0-9][0-9]:[0-9][0-9]$`
- Puntuación: `^[\-.,;\!?¿¡"/()\[\]{}]+$`

Los resultados de *stacking* utilizando información léxica se muestran en la tabla 4.15 junto a los originales, en donde se puede comprobar que efectivamente los resultados, que ya eran buenos, mejoran con la nueva información introducida.

CORPUS	SG	SG-lex
brown	0,64	0,72
conll00Pos	0,71	0,77
conll02nedPos	0,66	0,70
conll07espPos	1,34	1,48
conll07eusPos	0,49	0,51
floresta	0,78	0,79
susanne	1,26	1,45
talp	1,10	1,10
treebankwsjPos	0,59	0,67
PROMEDIO	0,84	0,91

Tabla 4.15: Resultados de *stacking* con información léxica.

Otra aproximación consiste en añadir etiquetas obtenidas a través de otros corpus distintos del que se está etiquetando, o incluso etiquetas de otras tareas diferentes al POS, como por ejemplo NER o *chunking*. Para llevar a cabo estos experimentos hemos implementado la posibilidad de especificar una lista de clasificadores junto a los corpus con los que deseamos sean entrenados antes de ser ejecutados sobre el corpus objetivo. De esta manera, además de obtener las etiquetas habituales obtenidas mediante el corpus de entrenamiento actual, se obtienen las etiquetas proporcionadas por los clasificadores a través de otros corpus, pudiendo ser estos clasificadores los mismos o diferentes. En la figura 4.5 vemos el esquema de ejecución siendo T el corpus de entrenamiento, t el de test y los c^i los clasificadores base utilizados, mientras que los T^j son los corpus de entrenamiento que aportan la información heterogénea generando los clasificadores que al ser ejecutados sobre el mismo corpus t aportan las etiquetas adicionales.

Siguiendo esta idea hemos empezado utilizando únicamente el clasificador TnT y realizando los siguientes experimentos:

- SG-H1: utilizamos los corpus Brown, CoNll 2000 y Penn Treebank, tomando en cada caso a los otros dos como fuente de información heterogénea.

- SG-H2: Se añaden al esquema H1 las siguientes fuentes de información heterogénea:

  - El corpus susanne (ver el apartado 4.3.1).

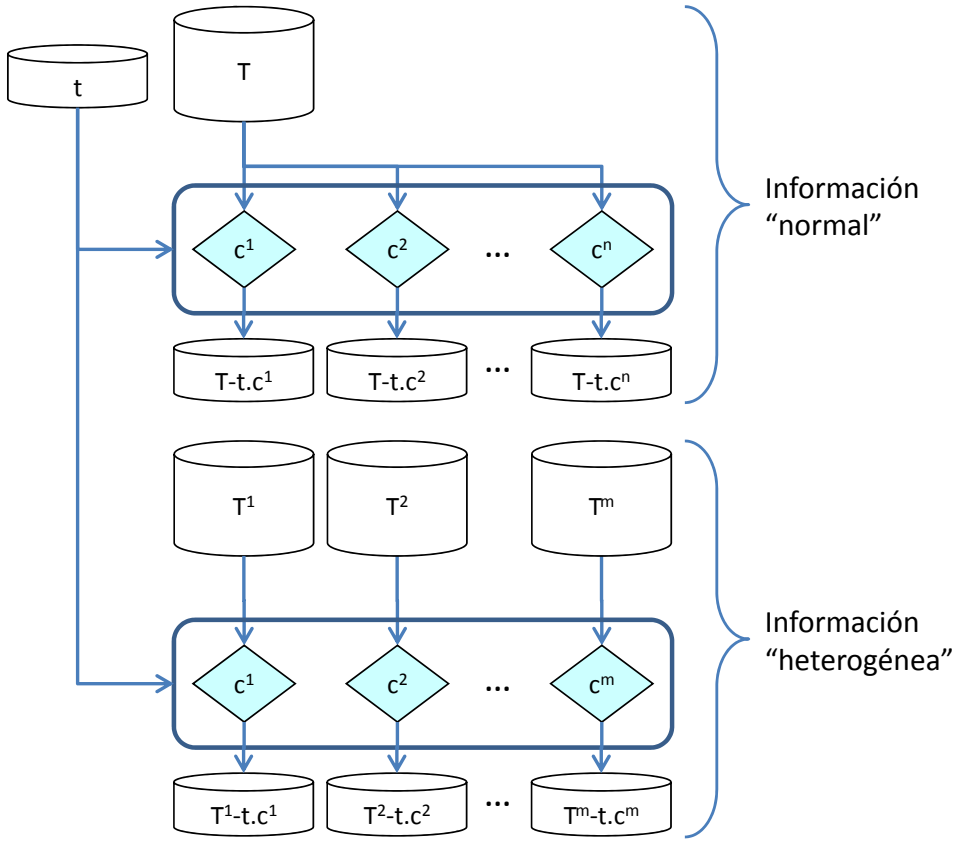

Figura 4.5: Esquema de ejecución de *stacking*+info. heterogénea

- El corpus BioCreAtIvE[12] (*Critical Assessment of Information Extraction Systems in Biology*) que proviene de la Biblioteca Nacional de Medicina de los Estados Unidos y que entre otras etiquetas contiene la de POS, que es la que utilizamos.
- El corpus CoNll 2000 de *chunking* (ver el apartado 4.3.1).
- Una porción del corpus IE-ER de NER proveniente del NIST 1999 (*Information Extraction: Entity Recognition Evaluation*)[13]. Concretamente hemos trabajado con los datos que se pusieron a disposición de los investigadores para las pruebas de desarrollo de sus sistemas.

Los resultados aparecen en la tabla 4.16 donde también aparecen los resultados de TnT que sirven como referencia.

CORPUS	TnT	SG-H1	SG-H2
brown	96,55	0,21	0,25
conll00Pos	97,32	0,39	0,42
treebankwsjPos	96,21	0,35	0,43
PROMEDIO	96,69	0,32	0,37

Tabla 4.16: Resultados de *stacking* con TnT+información heterogénea.

A continuación repetimos las pruebas empleando esta vez los clasificadores TnT, TreeTagger y MBT, obteniendo los resultados que se muestran en la tabla 4.17 bajo el nombre de SGm-H1 y SGm-H2 en donde también se muestran las mejoras que obtiene *stacking* sin añadir la información heterogénea.

CORPUS	SG	SGm-H1	SGm-H2
brown	0,34	0,41	0,45
conll00Pos	0,49	0,51	0,51
treebankwsjPos	0,31	0,45	0,44
PROMEDIO	0,38	0,46	0,47

Tabla 4.17: Resultados de *stacking* múltiple+información heterogénea.

[12] http://www.mitre.org/public/biocreative/
[13] http://www.itl.nist.gov/iad/894.01/tests/ie-er/er_99/er_99.htm

Por último hemos querido añadir también la información léxica tal y cómo explicábamos al principio de este apartado, dando lugar a los experimentos SGmL-H1 y SGmL-H2 que mostramos en la tabla 4.18.

CORPUS	SG	SGmL-H1	SGmL-H2
brown	0,34	0,44	0,47
conll00Pos	0,49	0,61	0,54
treebankwsjPos	0,31	0,48	0,50
PROMEDIO	0,38	0,51	0,50

Tabla 4.18: Resultados de *stacking* múltiple+información het. y léxica.

Los resultados nos muestran que *stacking* es capaz de hacer buen uso de la información adicional que vamos introduciendo en la base de datos y que redunda en una mejora de los resultados apreciable. Sin embargo, es cuando repetimos los mismos experimentos sobre las versiones reducidas del Penn Treebank cuando comprobamos la importancia de esta característica, ya que los beneficios son aún mayores cuando el corpus original es pequeño (tablas 4.19 y 4.20).

CORPUS	TnT	SG-H1	SG-H2
treebankwsjPos	96,21	0,35	0,43
treebankwsjPos200k	95,48	0,70	0,70
treebankwsjPos100k	94,82	1,11	1,16
treebankwsjPos50k	93,88	1,79	1,80
PROMEDIO	95,10	0,99	1,02

Tabla 4.19: Resultados añadiendo información het. al corpus reducido (a).

Parámetros de ejecución

Finalmente vamos a terminar planteando otra situación propicia para hacer uso de los métodos de combinación. Se trata de cuando un clasificador dispone de parámetros de configuración de cuyo valor depende el resultado obtenido. La incertidumbre provocada por esta situación cuando no hay indicios sobre los valores óptimos a utilizar se une a la posibilidad de que los valores óptimos no sean constantes y dependan por ejemplo del corpus utilizado. Si el objetivo es conseguir un clasificador robusto que se adapte a

CORPUS	SG	SGm-H1	SGm-H2	SGmL-H1	SGmL-H2
treebankwsjPos	0,31	0,45	0,44	0,48	0,50
treebankwsjPos200k	0,26	0,73	0,85	0,88	0,88
treebankwsjPos100k	0,32	1,11	1,21	1,11	1,14
treebankwsjPos50k	0,25	1,69	1,74	1,88	1,88
PROMEDIO	0,28	0,99	1,06	1,09	1,1

Tabla 4.20: Resultados añadiendo información het. al corpus reducido (b).

diferentes situaciones puede que la combinación de los diferentes valores aporte la mejor solución. Para poner a prueba esta hipótesis hemos seleccionado el clasificador base TnT, que entre otros parámetros nos permite configurar el modelo de Markov utilizado en cuanto al grado y tipo de suavizado aplicado.

En cuanto al grado del modelo TnT permite crear modelos de grado 1, 2 ó 3 mediante el parámetro `-n<num>` , siendo 3 el valor por defecto. Las diferencias en cuanto a los resultados obtenidos en función del valor utilizado pueden consultarse en la tabla 4.21.

CORPUS	TnT-n1	TnT-n2	TnT-n3
brown	92,3	96,22	96,55
conll00Pos	94,79	97,04	97,32
conll02nedPos	94,6	95,54	95,81
conll07espPos	93,36	95,55	95,98
conll07eusPos	92,24	93,62	93,73
floresta	94,27	96,88	97,02
susanne	87,85	93,2	93,61
talp	92,92	95,05	95,82
treebankwsjPos	93,1	95,89	96,21
PROMEDIO	92,83	95,44	95,78

Tabla 4.21: Resultados en función del parámetro $-n$ de TnT.

Y si lo que estamos interesados en cambiar es el algoritmo de suavizado de la matriz de transición, podremos hacerlo a través del parámetro `-d<num>`, que ofrece las siguientes posibilidades:

- -d1: sustituir los ceros por 0,5
- -d2: sumar 0,5 a todas las frecuencias

- -d4: realiza un suavizado mediante interpolación lineal

siendo `-d4` el valor por defecto. Las diferencias en cuanto a los resultados obtenidos en función del valor utilizado pueden consultarse en la tabla 4.22.

CORPUS	TnT-d1	TnT-d2	TnT-d4
brown	96,56	96,55	96,55
conll00Pos	97,25	97,3	97,32
conll02nedPos	95,74	95,73	95,81
conll07espPos	95,83	95,84	95,98
conll07eusPos	93,96	93,93	93,73
floresta	97	97	97,02
susanne	92,84	93,02	93,61
talp	96,06	96,05	95,82
treebankwsjPos	96,16	96,16	96,21
PROMEDIO	95,71	95,73	95,78

Tabla 4.22: Resultados en función del parámetro $-d$ de TnT.

A partir de aquí se han combinado los resultados proporcionados por los tres posibles valores de 'n' y 'd' mediante *stacking* obteniéndose los valores finales que se aprecian en la tabla 4.23. En dicha tabla aparece la mejora respecto a la mejor versión de TnT de entre las participantes en la combinación (A) y también la mejora respecto a la versión correspondiente al valor por defecto del parámetro estudiado (B). Se observa que apenas hay diferencias entre estas dos referencias ya que el valor por defecto casi siempre es el que aporta mejores resultados, aunque hemos destacado dos casos en los que no es así, y donde se obtiene una "recompensa" adicional por utilizar combinación respecto a optar por los valores por defecto.

4.3.5. Análisis Global

A la vista de todos los resultados obtenidos parece bastante claro que *stacking* es el método de combinación que generalmente ofrece los mejores resultados pero es importante destacar que el rendimiento ha sido muy bueno en general. También es digno de reseñar el rendimiento de algoritmos como la votación o el basado en el Teorema de Bayes, que con una implementación sencilla son capaces de obtener resultados muy buenos. Por otro lado, los métodos de ECOC y *random subspace* han sido los únicos que no han podido mejorar respecto a los clasificadores participantes, que parecen no aportar

CORPUS	TnT-n(A)	TnT-d(A)	TnT-n(B)	TnT-d(B)
brown	0,14	0,12	0,14	0,13
conll00Pos	0,36	0,31	0,36	0,31
conll02nedPos	0,32	0,22	0,32	0,22
conll07espPos	0,47	0,09	0,47	0,09
conll07eusPos	0,24	0,08	0,24	**0,31**
floresta	0,16	0,13	0,16	0,13
susanne	0,39	0,34	0,39	0,34
talp	0,24	0,05	0,24	**0,29**
treebankwsjPos	0,22	0,19	0,22	0,19
PROMEDIO	0,28	0,17	0,28	0,22

Tabla 4.23: Resultados combinando diferentes parámetros de TnT.

información suficiente para que se pueda extraer el conocimiento necesario sobre la tarea que nos ocupa. Las principales conclusiones que extraemos del trabajo realizado pueden resumirse de la siguiente manera:

- Los métodos de votación y Bayes ofrecen unos resultados muy satisfactorios a pesar de su sencillez considerando además que la votación no requiere ni siquiera de una fase de entrenamiento. El *behavior knowledge space* por su parte alcanza resultados cercanos a los mejores de todo el estudio, apoyado en un esquema que también es conceptualmente sencillo pero a todas luces tremendamente eficaz.

- El método *stacking* es sin duda el que más argumentos facilita ser considerado la mejor opción tras este estudio, ya que ha demostrado rentabilizar muy bien el aumento de complejidad que conlleva. Aporta, además de la robustez y flexibilidad que le permite adaptarse a clasificadores y corpus de calidad reducida, la posibilidad de incorporar información heterogénea que aumenta aún más el potencial de mejora que nos ofrece.

- Finalmente, el método *cascading* ha demostrado ser capaz de aprovechar las virtudes de varios métodos de combinación manteniendo las mejores cifras de todo el estudio tras probar varios esquemas de ejecución.

4.4. Aplicando Combinación a Otras Tareas

Una vez comprobada la utilidad de los métodos de combinación con la tarea de POS, vamos ahora a repetir algunos experimentos con otras tareas de etiquetado secuencial como NER, Bio-NER y *chunking*. El objetivo no es otro que confirmar la capacidad de mejorar a los clasificadores individuales en otras tareas diferentes. Esto nos permite también evaluar los métodos con un problema que no es aquel para el que se desarrollaron los clasificadores base (salvo FV que no ha sido optimizado para ninguna tarea en concreto). La medida que vamos a utilizar será el valor $F_{\beta=1}$ en lugar del *accuracy*, ya que de esta forma se tiene en cuenta tanto la precisión como la cobertura a la hora de detectar las palabras que forman parte de una entidad o un *chunk*.

4.4.1. Corpus Utilizados

Comenzamos al igual que hicimos con el apartado dedicado al POS con la descripción de los corpus que utilizaremos en esta ocasión, de los que mostramos sus características principales en la tabla 4.24.

CORPUS	Idioma	Tarea	Tags	TRAIN		TEST	
				frases	palabras	frases	palabras
Coling'04	ING	BIO	11	18546	492551	3856	101039
CoNLL'00	ING	CHK	23	8936	211727	2012	47377
CoNLL'02	ESP	NER	9	8323	264715	1915	52923
	HOL	NER	9	15806	202931	2895	37761

Tabla 4.24: Resumen de los corpus NER/*chunking* utilizados.

Coling 2004

Este corpus fue creado para la tarea evaluada en el *International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*[14]. Contiene más de 2000 abstracts de MEDLINE extraídos del corpus de entidades bio-médicas GENIA, que a su vez es un producto del proyecto del mismo nombre, relacionado con el *Information Mobility Project* (CREST, JST) y el *Genome Information Science Project* (MEXT). Del corpus GENIA se dejaron únicamente cinco clases de entidades que son: DNA, RNA, protein,

[14] http://www.genisis.ch/ñatlang/JNLPBA04/

cell_line y cell_type, siendo un corpus etiquetado en el formato IOB mediante una palabra por línea.

CoNLL 2000

Como vimos en el apartado 4.3.1 se trata del corpus utilizado en la competición CoNLL en su edición del año 2000, dedicada a la tarea de *chunking*. En este caso se utiliza la columna de *chunking* descartando las etiquetas POS.

CoNLL 2002

El corpus CoNLL 2002 también aparece en el apartado 4.3.1 y se trata de un corpus desarrollado para la tarea NER en dos idiomas, español y holandés. Está compuesto por dos columnas separadas por un espacio, en donde la primera contiene las palabras y la segunda la etiqueta NER en formato IOB. Las entidades consideradas son cuatro: nombres de personas (PER), organizaciones (ORG), localizaciones (LOC) and nombres de categoría "miscelánea" (MISC). Los datos en español provienen de noticias de la agencia EFE del mayo del año 2000 y fueron etiquetados por miembros de la Universidad Politécnica de Cataluña y la Universidad de Barcelona. La parte holandesa consiste en cuatro ediciones del periódico belga "De Morgen"del año 2000 y el etiquetado fue llevado a cabo por miembros de la Universidad de Antwerp.

4.4.2. Aplicación de los Métodos

En la tabla 4.25 mostramos los resultados obtenidos con los métodos de combinación que mejor se han comportado para la tarea POS. Las etiquetas se corresponden con el método *stacking*, votación (VT) y *cascading* (C-x), con el que se ha experimentado colocando en el segundo nivel de combinación un método x diferente en cada caso.

Comprobamos que las mejoras también se producen en tareas muy diferentes al POS y para las cuales no fueron optimizados los clasificadores base. Esta es una situación que se presenta a menudo en la investigación, en donde la disponibilidad de herramientas optimizadas no siempre está garantizada y para la que los métodos de combinación pueden considerarse una opción muy a tener en cuenta. Aunque los resultados en estos casos hacen uso del valor $F_{\beta=1}$ en lugar del *accuracy* empleado en la tarea del etiquetado POS, cabe destacar la cuantía de las mejoras obtenidas, superando con creces los valores iniciales aportados por los clasificadores.

CORPUS	SG	SG-l	VT	C-bay	C-bks	C-sg	C-vt
coling04	2,99	1,75	0,57	1,10	2,66	2,45	1,89
conll00Chunk	1,24	1,40	0,00	0,89	1,27	0,88	0,94
conll02esp	2,32	1,18	1,10	1,02	1,04	2,43	1,76
conll02nedNer	4,03	4,37	2,72	4,00	3,92	2,40	4,21
PROMEDIO	2,65	2,18	1,10	1,75	2,22	2,04	2,20

Tabla 4.25: Resultados de combinación con NER, Bio-NER y *chunking*.

Conclusiones

El Procesamiento del Lenguaje Natural dirigido hacia el tratamiento de textos trata problemas muy diversos que debido a su complejidad suelen fraccionarse en tareas independientes, las cuales aportan una vez resueltas parte de la información necesaria para conseguir los objetivos finales. Estas tareas son en muchos casos consideradas problemas de clasificación, ya que consisten en asignar una determinada categoría a una palabra, frase, fragmento de un texto o a todo un documento. A raíz de ello se ha hecho un uso intensivo de los múltiples algoritmos y métodos de clasificación que se han ido desarrollando en los últimos años, avanzando en la resolución de estas tareas, con muy buenos resultados en unos casos y no tan satisfactorios en otros.

Por otro lado, en el área del Reconocimiento de Patrones se ha dedicado mucho esfuerzo al estudio de otro tipo de métodos, que intentan aprovechar las diversas aproximaciones llevadas a cabo por los métodos de clasificación. Se trata de algoritmos que combinan las categorías propuestas por los clasificadores para intentar aprovechar las virtudes de cada uno de ellos. Entre los métodos de combinación se encuentran casos muy populares como la votación, y otros no tan conocidos aunque con resultados muy convincentes según demuestran los trabajos publicados.

En este trabajo se han estudiado los métodos de combinación desde diferentes puntos de vista, contemplando tanto los aspectos teóricos fundamentales, los datos históricos y las implicaciones prácticas derivadas de la experimentación propia. Las principales aportaciones se pueden resumir de la siguiente forma:

1. *Estudio teórico y análisis bibliográfico sobre la implicación de los métodos de combinación en el área del PLN.* Tras un estudio exhaustivo sobre las numerosas alternativas que se nos presentan a la hora de elegir un método de combinación y las bases teóricas que apoyan su uso cómo vía para lograr optimizar los resultados de clasificación, hemos querido comprobar el nivel de influencia que ha tenido esta línea de investigación en el PLN. Analizando un gran número de trabajos donde se hace uso de algún mecanismo de combinación, hemos podido constatar

que, mientras el uso de métodos de clasificación es muy variado, lo que demuestra el alto nivel de conocimiento sobre las diversas alternativas existentes, en el caso de los métodos de combinación está bastante sesgado hacia la votación y el método *stacking*, destacando el primero por su sencillez y el segundo por su gran potencial.

2. *Aplicación de la combinación a dos casos particulares.* Siendo el meta-aprendizaje en el que se basa la técnica de *stacking* el método con los resultados más prometedores según la bibliografía, nos propusimos en primer lugar emplearlo para mejorar el etiquetado de entidades según dicta la tarea de NER, con el objetivo de evitar el uso de un gran número de recursos externos como es habitual en este tipo de sistemas. Los resultados fueron muy satisfactorios ya que la combinación logró aprovechar la diversidad generada mediante transformaciones en el corpus y el conjunto de etiquetas entre otras acciones. Asimismo se intentó aplicar este tipo de conocimientos a otra tarea diferente como es la generación de recursos, con la esperanza de que la unión de técnicas de *bootstrapping* con la combinación mediante *stacking* entre otros métodos, fuera capaz de generar recursos con una calidad superior a la obtenida mediante las técnicas de *bootstrapping* de forma aislada. Los resultados en este caso no fueron tan convincentes ya que, aunque se consiguieron mejoras significativas en tareas como el *chunking*, en otras tareas no se lograron los resultados esperados quizás debido a la excesiva dificultad del problema planteado.

3. *Estudio comparativo de diversas alternativas de combinación empleando diferentes tipos de clasificadores y corpus de características muy variadas.* Se ha llevado a cabo una fase de experimentación para comprobar la efectividad de los métodos de combinación más conocidos para resolver la tarea del POS, representada por un buen número de corpus con características muy dispares. Se han mostrado resultados que apoyan la aplicación de estos métodos para obtener mejoras considerables y se han presentado situaciones concretas en las que es especialmente recomendable su uso si se desean alcanzar los mejores resultados posibles con los recursos que tengamos a nuestra disposición. En general, la combinación se ha mostrado como una opción fiable y rentable en cualquier caso pero especialmente cuando contamos con clasificadores de baja calidad o corpus de pequeño tamaño e incluso cuando los clasificadores disponen de parámetros configurables y desconocemos los valores óptimos. Entre todos los métodos ha destacado el meta-aprendizaje basado en *stacking*, que ha hecho gala de una gran robustez y flexibilidad,

adaptándose a los escenarios más complicados como la baja calidad de los elementos involucrados en la combinación y permitiendo la incorporación de información léxica o de cualquier otra índole en la fase de clasificación. La implementación que hemos llevado a cabo para probar estos métodos ha permitido experimentar fácilmente con varios niveles de combinación, creando esquemas de *cascading* que han aportado también muy buenos resultados. Además de la tarea POS, seleccionada por partir de unos valores de *accuracy* muy altos y por extensión difíciles de mejorar, también se han mostrado las mejoras obtenidas en otras tareas como NER y *chuncking* e incluso Bio-NER. Hay que destacar que de los 360 experimentos de combinación realizados, más del 90 % han superado los valores obtenidos por los clasificadores base y en la mayoría de los casos de forma muy significativa.

4. *Desarrollo de diversas estrategias para generar variabilidad.* Más allá de los casos más evidentes en los que se puede aplicar la combinación de forma inmediata al disponer de varios clasificadores, hemos experimentado con situaciones en las que es necesario generar la variabilidad en un paso previo a la combinación. Un ejemplo lo encontramos en la aplicación de *stacking* al NER, en donde dicha variabilidad se obtiene a través de diversos tipos de transformaciones aplicadas sobre un único corpus generando distintas versiones del mismo, cada una dando lugar a un clasificador nuevo que pueda ser combinado con el resto. Presentamos también junto al estudio comparativo otras situaciones que permiten generar variabilidad. Una de ellas es la existencia de varios tipos de características asociadas a las palabras del corpus, lo cual permite utilizar diferentes subconjuntos de ellas para generar distintas versiones de un mismo clasificador. También existe la posibilidad de generar clasificadores diferentes empleando distintos valores para los parámetros de configuración, lo cuál permite también evitar una búsqueda de los parámetros que optimizan los resultados del clasificador a generar.

Por todo ello creemos que los métodos de combinación representan una gran oportunidad para mejorar los sistemas actuales y que el desarrollo de herramientas que saquen provecho de sus virtudes puede impulsar los resultados obtenidos en tareas de clasificación y por extensión, a las tareas del PLN que se basan o hacen uso de ellas.

Bibliografía

[1] S. Abney. Bootstrapping. *In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 360–367, 2002.

[2] S. G. Alsing, K. W. Bauer, and J. O. Miller. A multinomial selection procedure for evaluating pattern recognition algorithms. *Pattern Recognition*, 35:2397–2412, 2002.

[3] K. Arrow. *Social Choice and Individual Values*. Wiley, New York, 1951.

[4] M. Asahara and Y. Matsumoto. Japanese named entity extraction with redundant morphological analysis. *In Proc. Human Language Technology conference - North American chapter of the Association for Computational Linguistics*, 2003.

[5] M. Banko and O. Etzioni. The tradeoffs between open and traditional relation extraction. *Proceedings of ACL-08: HLT*, pages 28–36, 2008.

[6] B.T. Bartell, G.W. Cottrell, and R.K. Belew. Automatic combination of multiple ranked retrieval systems. *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 173–181, 1994.

[7] N.J. Belkin and J.P. Callan. The effect of multiple query representations on information retrieval system performance. *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 339–346, 1993.

[8] P.N. Bennett, S.T. Dumais, and E. Horvitz. Probabilistic combination of text classifiers using reliability indicators: models and results. *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 207–214, 2002.

[9] P.N. Bennett, S.T. Dumais, and E. Horvitz. The combination of text classifiers using reliability indicators. *Information Retrieval*, 8(1):67–100, 2005.

[10] H. Berthelsen and B. Megyesi. Ensemble of classifiers for noise detection in pos tagged corpora. *TDS '00: Proceedings of the Third International Workshop on Text, Speech and Dialogue*, pages 27–32, 2000.

[11] D.M. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: a high-performance learning name-finder. *In Proc. Conference on Applied Natural Language Processing*, 1997.

[12] A. Blum and T. Mitchell. Combining labelled and unlabeled data with co-training. *11th Annual Conference on Computational Learning Theory*, pages 92–100, 1998.

[13] L. Borin. Something borrowed, something blue: Rule-based combination of pos taggers. *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, 2000.

[14] A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. Nyu: Description of the mene named entity system as used in muc-7. *In Proc. Seventh Message Understanding Conference*, 1998.

[15] T. Brants. Tnt. a statistical part-of-speech tagger. *In Proceedings of the 6th Applied NLP Conference (ANLP00)*, pages 224–231, 2000.

[16] L. Breiman. Bagging predictors. Technical Report 421, Department of Statistics, University of California, Berkeley, 1994.

[17] E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21:543–565, 1995.

[18] E. Brill and J. Wu. Classifier combination for improved lexical disambiguation. *Proceedings of the 17th international conference on Computational linguistics*, pages 191–195, 1998.

[19] P. Calado, M. Cristo, E. Moura, N. Ziviani, B. Ribeiro-Neto, and M.A. Gonçalves. Combining link-based and content-based methods for web document classification. *Proceedings of International Conference on Information and Knowledge Management*, pages 394–401, 2003.

[20] X. Carreras, L. Márquez, and L. Padró. Named entity extraction using adaboost. *COLING-02: proceedings of the 6th conference on Natural language learning*, pages 1–4, 2002.

[21] Xavier Carreras, Lluís Màrquez, and Lluís Padró. Named entity recognition for catalan using spanish resources. *10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, 2003.

[22] M. Civit. Guía para la anotación morfosintáctica del corpus clic-talp. *X-TRACT Working Paper WP-00/06*, 2000.

[23] S. Clark, J. R. Curran, and M. Osborne. Bootstrapping pos taggers using unlabelled data. *In Proceedings of CoNLL-2003*, pages 49–55, 2003.

[24] W. W. Cohen. Fast effective rule induction. *In Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, 1995.

[25] M. Collins and Y. Singer. Unsupervised models for named entity classification. *In Proc. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

[26] S. Cucerzan and D. Yarowsky. Bootstrapping a multilingual part-of-speech tagger in one person-day. *In Proceedings of the 6th Workshop on Computational Language Learning*, 2002.

[27] Silviu Cucerzan and David Yarowsky. Language independent named entity recognition combining morphological and contextual evidence. *In Proceedings of 1999 Joint SIGDAT Conference on EMNLP and VLC*, 1999.

[28] Le Anh Cuong. *A Study of Classifier Combination and Semi-Supervised Learning for Word Sense Disambiguation*. PhD thesis, Japan Advanced Institute of Science and Technology, 2007.

[29] W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. Mbt: A memorybased part of speech tagger-generator. *In Proceedings of the 4th Workshop on Very Large Corpora*, pages 14–27, 1996.

[30] W. Daelemans, J. Zavrel, A.v.d. Bosch, and K.v.d. Sloot. Mbt: Memory-based tagger, reference guide. Technical Report 03-13, ILK, 2003.

[31] Víctor J. Díaz, J. A. Troyano, F. Enríquez, J. Barroso, and Luisa Romero. Aplicación de modelos de markov y máquinas svm al reconocimiento de entidades. *X Conferencia de la Asociación Española Para la Inteligencia Artificial (CAEPIA) y V Conferencia en Transferencia de Tecnología en Inteligencia Artificial (TTIA)*, 2:55–58, 2003.

[32] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 7(10):1895–1924, 1998.

[33] T. G. Dietterich. Ensemble methods in machine learning. *In J. Kittler and F. Roli, editors, Multiple Classifier Systems, Lecture Notes in Computer Science*, 1857:1–15, 2000.

[34] N. Dimililer, E. Varoğlu, and H. Altinçay. Vote-based classifier selection for biomedical ner using genetic algorithms. *IbPRIA 2007, Part II, LNCS*, 4478:202–209, 2007.

[35] F. Enriquez, J. A. Troyano, F. Cruz, and F. J. Ortega. Ampliación automática de corpus mediante la colaboración de varios etiquetadores. *Procesamiento del Lenguaje Natural*, 37:11–18, 2006.

[36] F. Enriquez, J. A. Troyano, F. Cruz, and F. J. Ortega. Bootstrapping applied to a corpus generation task. *Computer Aided Systems Theory (Eurocast 2007)*, pages 130–131, 2007.

[37] F. Enriquez, J. A. Troyano, F. Cruz, and F. J. Ortega. Generación semiautomática de recursos. *Procesamiento del Lenguaje Natural*, 39:173–180, 2007.

[38] X. Fang and H. Sheng. A hybrid approach for chinese named entity recognition. *Discovery Science 2002, LNCS*, 2534:297–301, 2002.

[39] R. Feldman, B. Rosenfeld, and M. Fresko. Teg: a hybrid approach to information extraction. *Knowledge and Information Systems*, 9(1):1–18, 2006.

[40] J.G. Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU-97)*, pages 347–354, 1997.

[41] R. Florian. Named entity recognition as a house of cards: classifier stacking. *COLING-02: proceedings of the 6th conference on Natural language learning*, pages 1–4, 2002.

[42] R. Florian, S. Cucerzan, C. Schafer, and D. Yarowsky. Combining classifiers for word sense disambiguation. *Natural Language Engineering*, 4(8):327–341, 2002.

[43] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 168–171, 2003.

[44] R. Florian and D. Yarowsky. Modeling consensus: classifier combination for word sense disambiguation. *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 25–32, 2002.

[45] R. Frederking and S.Ñirenburg. Three heads are better than one. *In Proceedings of the fourth Conference on Applied Natural Language Processing (ANLP-94)*, pages 95–100, 1994.

[46] S. French. Group consensus probability distributions: a critical survey. *Bayesian Statistics*, 2:183–202, 1985.

[47] Y. Freund, Y. Mansour, and R. Schapire. Why averaging classifiers can protect against overfitting. *In Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, 2001.

[48] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[49] J. Fürnkranz. Hyperlink ensembles: a case study in hypertext classification. *Information Fusion*, 3(4):299–312, 2002.

[50] J. Gama. Combining classiers by constructive induction. *Ninth European Conference on Machine Learning, Springer*, 1997.

[51] K. Glass and S. Bangay. Evaluating parts-of-speech taggers for use in a text-to-scene conversion system. *SAICSIT '05: Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, pages 20–28, 2005.

[52] H.V. Halteren, W. Daelemans, and J. Zavrel. Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics*, 27(2):199–229, 2001.

[53] H.V. Halteren, J. Zavrel, and W. Daelemans. Improving data driven wordclass tagging by system combination. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, 1:491–497, 1998.

[54] L. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[55] J.C. Henderson and E. Brill. Exploiting diversity in natural language processing: Combining parsers. *In 1999 Joint Sigdat Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (ACL)*, pages 187–194, 1999.

[56] J.C. Henderson and E. Brill. Bagging and boosting a treebank parser. *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 34–41, 2000.

[57] J. Huang, O. Madani, and C.L. Giles. Error-driven generalist+experts (edge): a multi-stage ensemble framework for text categorization. *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 83–92, 2008.

[58] Y. S. Huang and C. Y. Suen. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:90–93, 1995.

[59] D.A. Hull, J.O. Pedersen, and H. Schutze. Method combination for document filtering. *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*, pages 279–288, 1996.

[60] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[61] H. Jing, R. Florian, X. Luo, T. Zhang, and A. Ittycheriah. Howtogetachinesename(entity): segmentation and combination issues. *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 200–207, 2003.

[62] T. Joachims. *Making large-Scale SVM Learning Practical*, chapter 11. MIT Press, 1999.

[63] A. Kilgarriff and J. Rosenzweig. Framework and results for english senseval. *Computers and the Humanities*, 34(1–2):15–48, 2000.

[64] D. Klein, K. Toutanova, H.T. Ilhan, S.D. Kamvar, and C.D. Manning. Combining heterogeneous classifiers for word-sense disambiguation. *Proceedings of the ACL-02 workshop on Word sense disambiguation*, pages 74–80, 2002.

[65] R. Kohavi. The power of decision tables. *In Proceedings of the European Conference on Machine Learning. LNCS 914*, pages 174–189, 1995.

[66] Z. Kozareva, O. Ferrández, A. Montoyo, R. Muñoz, A. Suárez, and J. Gómez. Combining data-driven systems for improving named entity recognition. *Data & Knowledge Engineering*, 61(3):449–466, 2007.

[67] S. Kripke. *Naming and Necessity*. Boston: Harvard University Press, 1982.

[68] L.I. Kuncheva and C.J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51:181–207, 2003.

[69] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[70] M. Kuta, M. Wrzeszcz, P. Chrząszcz, and J. Kitowski. Accuracy of baseline and complex methods applied to morphosyntactic tagging of polish. *ICCS '08: Proceedings of the 8th international conference on Computational Science, Part I*, pages 903–912, 2008.

[71] W. Lam and K.-Y. Lai. A meta-learning approach for text categorization. *Proceedings of the 24th ACM International Conference on Research and Development in Information Retrieval (SIGIR01)*, pages 303–309, 2001.

[72] L.S. Larkey and W.B. Croft. Combining classifiers in text categorization. *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*, pages 289–297, 1996.

[73] A.-C. Le, V.-N. Huynh, A. Shimazu, and H.-C. Dam. Weighted combination of classifiers for word sense disambiguation based on dempster-shafer theory. *Proceedings of the 4th IEEE International Conference on Research, Innovation and Vision for the Future (RIVF'06)*, pages 133–138, 2006.

[74] S. Li, C. Zong, and X. Wang. Sentiment classification through combining classifiers with multiple feature sets. *IEEE NLP-KE 2007 - Proceedings of International Conference on Natural Language Processing and Knowledge Engineering, art. no. 4368024*, pages 135–140, 2007.

[75] R. P. Lippmann. A critical overview of neural network pattern classifiers. *In Proc. IEEE Workshop on Neural Networks for Signal Processing*, pages 266–275, 1991.

[76] R. Liu and B. Yuan. Multiple classifier combination by clustering and selection. *Information Fusion*, 2:163–168, 2001.

[77] H. Loftsson. Tagging icelandic text: An experiment with integrations and combinations of taggers. *Language Resources and Evaluation*, 40(2):175–181, 2006.

[78] G. Lucarelli, X. Vasilakos, and I. Androutsopoulos. Named entity recognition in greek texts with an ensemble of svms and active learning. *International Journal on Artificial Intelligence Tools*, 16(6):1015–1045, 2007.

[79] T.R. Lynam, C. Buckley, C.L.A. Clarke, and G.V. Cormack. A multisystem analysis of document and term selection for blind feedback. *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 261–269, 2004.

[80] T.R. Lynam and G.V. Cormack. On-line spam filter fusion. *29th ACM SIGIR Conference on Research and Development on Information Retrieval*, 2006.

[81] L. Marquez, L. Padr, and H. Rodriguez. Improving tagging accuracy by voting taggers. *Proceedings of the 2nd Conference on Natural Language Processing & Industrial Applications (NLP+IA/TAL+AI)*, pages 149–155, 1998.

[82] L. Marquez, H. Rodriguez, J. Carmona, and J. Montolio. Improving pos tagging using machine-learning techniques. *Proceedings of the 1999 Faint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 53–62, 1999.

[83] J. Maudes, J.J. Rodríguez, and C. García-Osorio. Cascading for nominal data. *Multiple Classifier Systems, Lecture Notes in Computer Science*, 4472:231–240, 2007.

[84] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, features induction and web-enhanced lexicons. *In Proc. Conference on Computational Natural Language Learning*, 2003.

[85] R. Mihalcea. Bootstrapping large sense tagged corpora. *In Proceedings of the 3rd International Conference on Languages Resources and Evaluations (LREC 2002)*, 2002.

[86] R. Mihalcea. Co-training and self-training for word sense disambiguation. *In Proceedings of the Conference on Natural Language Learning (CoNLL 2004)*, 2004.

[87] R. Mihalcea and T. Chklovski. Open mind word expert: Creating large annotated data collections with web users' help. *In Proceedings of the EACL 2003 Workshop on Linguistically Annotated Corpora (LINC 2003)*, 2003.

[88] G. Miller. Wordnet: A lexical database for english. *Communication of the ACM*, 38(11):39–41, 1995.

[89] G. Miller, C. Leacock, T. Randee, and R. Bunker. A semantic concordance. *In Proceedings of the 3rd DARPA Workshop on Human Language Technology*, pages 303–308, 1993.

[90] M. Montague. *Metasearch: Data fusion for document retrieval*. PhD thesis, Dartmouth College, 2002.

[91] M. Montague and J.A. Aslam. Condorcet fusion for improved retrieval. *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 538–548, 2002.

[92] D.Ñadeau, P. Turney, and S. Matwin. Unsupervised named entity recognition: Generating gazetteers and resolving ambiguity. *In Proc. Canadian Conference on Artificial Intelligence*, 2006.

[93] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae investigationes: Revue internationale de linguistique française et de linguistique générale*, 30(1):3–26, 2007.

[94] H.T. Ng. Getting serious about word sense disambiguation. *In Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, pages 1–7, 1997.

[95] V. Ng and C. Cardie. Weakly supervised natural language learning without redundant views. *In Human Language Technology/Conference of the North American Chapter of the Association for Computational Linguistics (HLTNAACL)*, 2003.

[96] N. Oza and K. Tumer. Input decimation ensembles: Decorrelation through dimensionality reduction. *In Proc. 12nd International Workshop on Multiple Classifier Systems - MCS'01 - Lecture Notes in Computer Science, Cambridge, UK*, 2096:238–247, 2001.

[97] Sankar K. Pal. *Pattern recognition: from classical to modern approaches*. World Scientific, 2001.

[98] T. Pedersen. A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation. *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 63–69, 2000.

[99] X. Qi and B.D. Davison. Knowing a web page by the company it keeps. *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 228–237, 2006.

[100] X. Qi and B.D. Davison. Web page classification: Features and algorithms. *ACM Computing Surveys (CSUR)*, 41(2):1–31, 2009.

[101] Lisa F. Rau. Extracting company names from text. *In Proc. Conference on Artificial Intelligence Applications of IEEE*, 1991.

[102] G. Rogova. Combining the results of several neural network classifiers. *Neural Networks*, 7:777–781, 1994.

[103] B. Rosenfeld and R. Feldman. Ures: an unsupervised web relation extraction system. *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, 9(1):667–674, 2006.

[104] M. Rössler. Using markov models for named entity recognition in german newspapers. *In Proceedings of the Workshop on Machine Learning Approaches in Computational Linguistics*, pages 29–37, 2002.

[105] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C.D. Spyropoulos, and P. Stamatopoulos. Stacking classifiers for anti-spam filtering of e-mail. *Proceedings of EMNLP01, 6th Conference on Empirical Methods in Natural Language Processing*, pages 44–50, 2001.

[106] E. F. Tjong Kim Sang. Text chunking by system combination. *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning*, pages 151–153, 2000.

[107] E. F. Tjong Kim Sang, W. Daelemans, H. Dejean, R. Koeling, Y. Krymolowsky, V. Punyakanok, and D. Roth. Applying system combination to base noun phrase identification. *In Proceedings of COLING00, Germany*, pages 857–863, 2000.

[108] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *Proceedings of CoNLL-2003*, pages 142–147, 2003.

[109] H. Schmid. Probabilistic part-of-speech tagging using decision trees. *In Proceedings of the Conference on New Methods in Language Processing*, 1994.

[110] J.A. Shaw and E.A. Fox. Combination of multiple searches. *Proceedings of the 3rd Text Retrieval Conference (TREC-3)*, pages 105–108, 1994.

[111] Lei Shi and Rada Mihalcea. Putting pieces together: Combining framenet, verbnet and wordnet for robust semantic parsing. *In Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics, Mexico*, 2005.

[112] S. Siersdorfer, S. Sizov, and G. Weikum. Goal-oriented methods and meta methods for document classification and their parameter tuning. *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 59–68, 2004.

[113] G. Sigletos, G. Paliouras, C.D. Spyropoulos, and M. Hatzopoulos. Combining information extraction systems using voting and stacked generalization. *Journal of Machine Learning Research*, 6:1751–1782, 2005.

[114] C. Silva and B. Ribeiro. Rvm ensemble for text classification. *International Journal of Computational Intelligence Research*, 3(1):31–35, 2007.

[115] J. Sjöbergh. Combining pos-taggers for improved accuracy on swedish text. *Proceedings of No DaLiDa 2003*, 2003.

[116] M. Stevenson and Y. Wilks. The interaction of knowledge sources in word sense disambiguation. *Computational Linguistics*, 27(3):321–349, 2001.

[117] M. Surdeanu, L. Márquez, X. Carreras, and P.R. Comas. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29(1):105–151, 2007.

[118] I. Takashi and I. Kentaro. Committee-based decision making in probabilistic partial parsing. *Proceedings of the 18th conference on Computational linguistics*, pages 348–354, 2000.

[119] M. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research I*, pages 211–214, 2001.

[120] J. A. Troyano, Vicente Carrillo, F. Enríquez, and Francisco J. Galán. Named entity recognition through corpus transformation and system combination. *Lecture Notes in Artificial Intelligence (Estal 2004)*, 3230:255–266, 2004.

[121] J. A. Troyano, Víctor J. Díaz, F. Enríquez, J. Barroso, and V. Carrillo. Identificación de entidades con nombre basada en modelos de markov y Árboles de decisión. *Procesamiento del Lenguaje Natural*, 31:235–242, 2003.

[122] J. A. Troyano, Víctor J. Díaz, F. Enríquez, and Luisa Romero. Improving the performance of a named entity extractor by applying a stacking scheme. *Lecture Notes in Artificial Intelligence (Iberamia 2004)*, 3315:295–304, 2004.

[123] J.A. Troyano, Víctor J. Díaz, F. Enríquez, and Vicente Carrillo. Applying stacking and corpus transformation to a chuncking task. *Computer Aided Systems Theory (Eurocast 2005). LNCS*, 3643:150–158, 2005.

[124] José A. Troyano, Fernando Enríquez, Fermín Cruz, José M. Cañete, and F. Javier Ortega. Improving the performance of a tagger generator in an information extraction application. *Journal of Universal Computer Science*, 13(9):1287–1299, 2007.

[125] K. Tsukamoto, Y. Mitsuishi, and M. Sassano. Learning with multiple stacking for named entity recognition. *In Proceedings of CoNLL-2002*, pages 191–194, 2002.

[126] K. Tsutsumi, K. Shimada, and T. Endo. Movie review classification based on a multiple classifier. *In: The 21th Pacific Asia Conference on Language, Information and Computation (PACLIC)*, 2007.

[127] T. Utsuro, M. Sassano, and K. Uchimoto. Combining outputs of multiple japanese named entity chunkers by stacking. *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 281–288, 2002.

[128] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[129] V.Ñ. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[130] H. Wang and T. Zhao. Identifying named entities in biomedical text based on stacked generalization. *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, pages 160–164, 2008.

[131] X. Wang and Y. Matsumoto. Trajectory based word sense disambiguation. *COLING '04: Proceedings of the 20th international conference on Computational Linguistics (Art.903)*, 2004.

[132] Ian H. Witten and Eibe Frank. Generating accurate rule sets without global optimization. *In Proceedings of the 15th International Conference on Machine Learning*, pages 144–151, 1998.

[133] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2000.

[134] D.H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[135] D. Wu, G.Ñgai, and M. Carpuat. A stacked, voted, stacked model for named entity recognition. *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 200–203, 2003.

[136] W.-L. Wu, R.-Z. Lu, F. Gao, and Y. Yuan. Combining multiple statistical classifiers to improve the accuracy of task classification. *Lecture Notes in Computer Science*, 3406:452–462, 2005.

[137] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their application to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22:418–435, 1992.

[138] J. Zavrel and W. Daelemans. Bootstrapping a tagged corpus through combination of existing heterogeneous taggers. *In Proceedings of the 2nd International Conference on Language Resources and Evaluation*, pages 17–20, 2000.

[139] D. Zeman and Z. Žabokrtský. Improving parsing accuracy by combining diverse dependency parsers. *Proceedings of IWPT-2005*, pages 171–178, 2005.

[140] Y. Zhang. Using bayesian priors to combine classifiers for adaptive filtering. *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 345–352, 2004.

[141] G. Zhou, D. Shen, J. Zhang, J. Su, and S. Tan. Recognition of protein/gene names from text using an ensemble of classifiers. *BMC Bioinformatics*, 6(Suppl 1):S7, 2005.

[142] C. Zong and M. Seligman. Toward practical spoken language translation. *Machine Translation*, 19(2):113–137, 2005.

Índice alfabético

30 de junio de 2011