Categorización de Textos Multilingües basada en Redes Neuronales

Manuel García Vega
M. Teresa Martín Valdivia
L. Alfonso Ureña López
Departamento de Informática. Universidad de Jaén. Spain
{ mgarcia, maite, laurena}@ujaen.es

Resumen: Los métodos de acceso a la información, hoy en día, deben mejorarse para superar la sobrecarga de información existente. Las tareas de clasificación de textos como la categorización de documentos puede ayudar a los usuarios a acceder a gran cantidad de información (texto) disponible en Internet y en sus organizaciones. En este trabajo presentamos un sistema de categorización multilingüe basado en corpus paralelos, concretamente la Biblia Políglota, en español e inglés. El objetivo es categorizar textos en estas lenguas usando un entrenamiento de textos multilingües. Para ello, empleamos Redes Neuronales en CT, que se comportan mucho mejor que el ampliamente utilizado algoritmo de Rocchio. El algoritmo de Widrow–Hoff y el basado en el Gradiente Exponenciado de Kivinen–Warmuth han sido usados con éxito en PLN y en particular en CT. Proponemos el uso de un método, novedoso en PLN, de aprendizaje competitivo, concretamente el algoritmo de aprendizaje por cuantificación vectorial (LVQ). Los resultados que presentamos muestran que el LVQ mejora significativamente a los otros algoritmos de aprendizaje.

Palabras Clave Categorización de Textos (CT), Modelo del Espacio Vectorial (MEV), Recuperación de Información (RI), Redes Neuronales (RN), LVQ, Recuperación de Información Multilingüe.

1. Introducción

La categorización de texto es una tarea particular dentro de la clasificación de texto que consiste en la asignación de una o más categorías preexistentes a cada documento [Lewis92].

La mayoría de sistemas de categorización usan colecciones de documentos de entrenamiento para predecir las categorías de nuevos documentos [Yang99, Saham98, Yang99b]. Esto justifica la existencia de recursos, como por ejemplo: Reuters–21.578 [Lewis92], Ohsumed [Hersh94].

Habitualmente, son tres los algoritmos relacionados con la categorización de texto: Rocchio [Rocch71], Widrow–Hoff (WH) [Widro85] y Kivinen–Warmuth (KW) [Kivin97]. Los dos últimos utilizan reglas de aprendizaje basadas en RN. La característica más importante de las RN es su capacidad de aprender a partir de ejemplos, que les permite generalizar sin tener que formalizar el conocimiento adquirido. Con las RN se intenta expresar la solución de problemas complejos, no como una secuencia de pasos, sino como la combinación de una gran cantidad de elementos simples de proceso interconectados que operan en paralelo.

El algoritmo de WH utiliza la conocida regla delta y el algoritmo KW es una modificación del algoritmo de WH. En [Lewis96] se demuestra experimentalmente que ambos algoritmos de WH y KW son más efectivos que el ampliamente usado algoritmo de Rocchio en muchas tareas de categorización y enrutamiento de textos.

En esta comunicación proponemos el uso de un algoritmo de aprendizaje competitivo para entrenar una colección de documentos para categorización de textos. Los algoritmos de aprendizaje competitivo en RN utilizados con más efectividad son los basados en el modelo de Kohonen [Kohon88]. Este modelo presenta dos variantes: Mapa auto-organizativo (*Self-Organizing Map*) o SOM y Aprendizaje por Cuantificación Vectorial (*Learning Vector Quantization*) o LVQ. Aunque ambos utilizan un aprendizaje competitivo, la diferencia radica en que el SOM utiliza un método de aprendizaje no supervisado, mientras que en el LVQ es supervisado.

Aunque la versatilidad de este tipo de red es muy amplia, lo que le permite clasificar todo tipo de información, desde la literaria [Honke95] hasta la económica [Kaski95], el modelo de Kohonen posee dos limitaciones. Por un lado, el proceso de aprendizaje suele ser largo y arduo, y en segundo lugar, para aprender nuevos datos es necesario repetir el proceso de aprendizaje por completo.

Puesto que se trata de un aprendizaje supervisado, hemos elegido para este trabajo el algoritmo LVQ de Kohonen. Para comprobar la efectividad del algoritmo LVQ, hemos desarrollado una serie de experimentos con un recurso, ampliamente difundido, de libre disposición y traducido a todas las lenguas: la Biblia. Hemos comparado los algoritmos de Rocchio, WH, KW y LVQ. Los resultados obtenidos muestran que los algoritmos basados en RN son más efectivos que el algoritmo de Rocchio, siendo el algoritmo LVQ, que usa una regla de aprendizaje competitivo, el que mejor precisión obtiene.

La investigación en RI se desarrolla con muchos modelos, en particular y de manera notable, el modelo de espacio vectorial, el modelo probabilístico y el modelo booleano. En este artículo usaremos el modelo de espacio vectorial, que en el campo de la recuperación de información está considerado como un modelo efectivo [Salto83].

El resto de la comunicación se organiza como sigue. En primer lugar, presentamos la tarea de CT, con un breve resumen introductorio al MEV. A continuación, describimos la Biblia Políglota como recurso lingüístico para CT. Seguidamente exponemos los cuatro algoritmos utilizados en los experimentos. En la sección 5, detallamos los experimentos realizados para evaluar el comportamiento de los algoritmos. Los resultados de esta evaluación se describen en el apartado 6, para terminar con las conclusiones y líneas de trabajos futuros.

2. *Categorización de textos con MEV*

El MEV fue originalmente desarrollado para RI, aunque puede ser usado en otras tareas de CT [Buen97]. El fundamento del MEV para RI es representar una expresión del lenguaje natural como un vector de pesos de términos, donde cada peso mide la importancia del término en la expresión en lenguaje natural, la cual, puede ser un documento o una consulta. La proximidad semántica entre documentos y consultas se computa con el coseno del ángulo que forman sus vectores.

De manera análoga, para el caso que nos ocupa, en la CT podemos considerar que un documento pertenece a una categoría en particular, si la similitud entre el documento y la categoría es mayor que un cierto umbral o, simplemente es la mejor puntuada.

Así, dados tres conjuntos de N términos, M documentos y L consultas, el vector de pesos para el documento j es (w_{1j}, w_{2j}, …, w_{Nj}) y el vector de pesos para la consulta k (q_{1k}, q_{2k}, …, q_{Nk}). La similitud entre el documento j y la consulta k se obtiene con arreglo a la fórmula:

$$sim(w_j, q_k) = \frac{\sum_{i=1}^{m} w_{ij} \cdot q_{ik}}{\sqrt{\sum_{i=1}^{m} w_{ij}^{\,2} \cdot \sum_{i=1}^{m} q_{ik}^{\,2}}}$$

Los pesos de los términos para los vectores de los documentos se calculan haciendo uso de la conocida fórmula basada en la frecuencia de los términos [Salto89]:

$$w_{ij} = tf_{ij} \cdot \log_2\left(\frac{M}{df_i}\right)$$

donde tf_{ij} es la frecuencia del término i en el documento j, y df_i es el número de documentos en que aparece.

3. *Corpus paralelos como recurso multilingüe para la Categorización de Textos*

Los corpus paralelos son un recurso multilingüe, cada vez más disponible en Internet [Grefe98], que se está usando en distintas tareas de PLN como desambiguación [Davis98], RI [Nie99], traducción automática [McCar98], etc.

La Biblia Políglota [Resni99] es uno de ellos con unas características especialmente atractivas: es de libre disposición, está traducido a todas las lenguas, las traducciones son prácticamente perfectas y está alineado con respecto a versículos [Grefe98].

En esta comunicación proponemos un categorizador de textos en español e inglés basado en un corpus paralelo multilingüe. Para ello hemos usado dos traducciones de la Biblia que son la edición *Reina Valera*, para el caso del español, y la *American Standard Version*, para el inglés. Para ello, hemos creado una

Biblia bilingüe, mezclando ambas, versículo a versículo (ver Figura 1).

Si tomamos como unidad semántica el versículo, esta *Biblia*, contiene información en ambas lenguas de muy alta calidad, puesto que en un mismo *versículo* está la traducción sin error del correspondiente en los dos idiomas. Este nuevo documento sirve de base para la construcción de un categorizador multilingüe, en nuestro caso bilingüe, de forma que puede interrogarse con documentos tanto en español como en inglés para obtener la categoría adecuada independientemente del lenguaje usado.

```
01O 1 1 In the beginning God created
the heavens and the earth. EN el
principio crió Dios los cielos y la
tierra.
01O 1 2 And the earth was waste and
void; and darkness was upon the face of
the deep: and the Spirit of God moved
upon the face of the waters Y la tierra
estaba desordenada y vacía, y las
tinieblas estaban sobre la haz del
abismo, y el Espíritu de Dios se movía
sobre la haz de las aguas.
01O 1 3 And God said, Let there be
light: and there was light. Y dijo
Dios: Sea la luz: y fué la luz.
01O 1 4 And God saw the light, that it
was good: and God divided the light
from the darkness. Y vió Dios que la
luz era buena: y apartó Dios la luz de
las tinieblas.
01O 1 5 And God called the light Day,
and the darkness he called Night. And
there was evening and there was
morning, one day. Y llamó Dios á la luz
Día, y á las tinieblas llamó Noche: y
fué la tarde y la mañana un día.
```

Figura 1: La Biblia *bilingüe* generada

Para nuestros experimentos hemos dividido la Biblia bilingüe en dos partes (75% y 25%) dejando la primera como entrenamiento y la segunda para la evaluación de nuestro categorizador. Además, hemos propuesto 66 categorías diferentes, coincidiendo con los 66 libros de la Biblia. Se trata pues de averiguar de qué libro es un versículo determinado de la partición de evaluación.

El versículo de la consulta puede estar escrito en cualquiera de los dos idiomas, incluso en una mezcla de ambos y, dado que ambas lenguas son prácticamente disjuntas, los resultados serán similares.

Dado que cada libro de la Biblia está dividido en capítulos y estos a su vez en versículos, al procesarla aparecen suficientes ejemplares de cada categoría, como para justificar el uso de un algoritmo de entrenamiento adecuado.

4. Algoritmos de entrenamiento para la categorización de textos

Hemos seleccionado los algoritmos de Rocchio, WH y KW como algoritmos de entrenamiento, ya que proporcionan una manera de calcular los vectores de pesos para las categorías, habiendo sido usados en otros trabajos [Buena97, Ureña98], para, de esta forma, poder contrastar su eficacia con el algoritmo LVQ.

4.1 El Algoritmo de Rocchio

El algoritmo de Rocchio produce un nuevo vector de pesos **w** a partir de uno existente $\mathbf{w_0}$ y una colección de documentos de entrenamiento. La componente *i* del vector **w** es calculado por la fórmula:

$$w_i = \alpha w_{0,i} + \beta \frac{\sum_{j \in C_k} x_{j,i}}{n_{C_k}} + \gamma \frac{\sum_{j \notin C_k} x_{j,i}}{n - n_{C_k}}$$

donde C_k es el conjunto de documentos que forman la *k*-ésima categoría, $w_{0,i}$ es el peso inicial de la palabra *i*-ésima de la categoría *k*, $x_{j,i}$ es el peso de la palabra *i*-ésima del documento *j*, y n_k el número de documentos etiquetados con la *k*-ésima categoría. Los parámetros α, β y γ controlan el relativo impacto de los pesos iniciales, de los vectores bien categorizados y de los que no lo están, respectivamente en el nuevo vector.

Como en [Lewis96], hemos usado los valores α=1, β=16 y γ=4. Restringimos el categorizador para no hacer uso de pesos negativos, así al final el peso w_{ki} será positivo, o retornará a 0 si es negativo.

El vector inicial w_0 es frecuentemente tomado como vector nulo, pero esto puede ser instanciado con un conjunto de pesos iniciales calculados por la utilización de un recurso externo.

4.2 El Algoritmo de Widrow-Hoff

El algoritmo de WH comienza con un conjunto de vectores de pesos existente $\mathbf{w_k}(t)$, con $k \in [1,66]$, que será actualizado con el procesamiento de todos los documentos del entrenamiento. El peso de la componente *j*-ésima del vector $\mathbf{w_k}(t+1)$ para una categoría dada, *k*, es obtenido a partir del *i*-ésimo

documento del entrenamiento y del vector de pesos actual según la fórmula:

$$w_{kj}(t+1) = w_{kj}(t) - 2\eta[\mathbf{w_k}(t)\cdot\mathbf{x_i} - y_i]x_{ij}$$

donde $\mathbf{x_i}$ es el i-ésimo documento de entrenamiento, w_{kj} es el peso actual del término j-ésimo de la categoría k, y_i es 1 si el i-ésimo documento es asignado a la categoría correcta y 0 en otro caso. La constante η es la tasa de aprendizaje, la cual controla cuánto de rápido le está permitido cambiar al vector de pesos y cuánto influye cada nuevo ejemplar sobre éste. Un valor típicamente usado para η es $1/4X^2$, siendo X la máxima norma de los vectores que representan los documentos de entrenamiento.

4.3 El algoritmo de Kivinen–Warmuth

El algoritmo del gradiente exponenciado de Kivinen–Warmuth es similar al WH, ya que mantiene un vector de pesos $\mathbf{w_k}$ e introduce los vectores de entrenamiento una vez cada uno. Sin embargo, con KW, las componentes del vector $\mathbf{w_k}$ no pueden ser negativas. La regla con la que se actualizan los pesos de $\mathbf{w_k}$, análoga al algoritmo de WH es:

$$w_{kj}(t+1) = \frac{w_{kj}(t)\exp[-2\eta(\mathbf{w_k}\cdot\mathbf{x_i} - y_i)x_{i,j}]}{\sum_{j=1}^{N} w_{kj}(t)\exp[-2\eta(\mathbf{w_k}\cdot\mathbf{x_i} - y_i)x_{i,j}]}$$

Como antes, la tasa de aprendizaje $\eta>0$ controla el impacto de cada nuevo ejemplar de entrenamiento.

4.4 El algoritmo LVQ

El LVQ destaca por la sencillez de las heurísticas que usa y se adapta directamente a la tarea de CT. El LVQ es un método de clasificación basado en los conceptos competitivos de RN que permite definir un conjunto de categorías sobre el espacio de los datos de entrada mediante un aprendizaje reforzado tanto positivo (premio) como negativo (castigo).

Dada una secuencia de documentos de entrada, se selecciona un conjunto inicial de vectores de referencia $\mathbf{w_k}$. Iterativamente, se selecciona un documento $\mathbf{x_i}$ y se actualiza el conjunto $\mathbf{w}$ de forma que se adapte mejor a $\mathbf{x_i}$.

El algoritmo LVQ funciona de la siguiente manera:

Para cada clase, k, se asocia un vector de pesos $\mathbf{w_k}$. En cada iteración, el algoritmo selecciona un documento de entrada $\mathbf{x_i}$ y lo compara con cada uno de los vectores de pesos $\mathbf{w_k}$ utilizando la distancia euclídea $\|\mathbf{x_i}-\mathbf{w_k}\|$, de manera que declara como ganador al vector de pesos $\mathbf{w_k}$ más cercano a $\mathbf{x_i}$, siendo c el índice de ese vector de pesos:

$$\|\mathbf{x_i} - \mathbf{w_c}\| = \min_k\{\|\mathbf{x_i} - \mathbf{w_k}\|\}$$

Las clases compiten entre sí para ver cuál es la que más se parece al vector de entrada, de manera que se elige como ganadora aquella que tiene la menor distancia euclídea con respecto al documento de entrada. Únicamente la clase ganadora modificará sus pesos utilizando un algoritmo de aprendizaje reforzado positivo o negativo dependiendo de si la clasificación ha sido o no correcta. Así, si la clase ganadora pertenece a la misma clase que el vector de entrada (la clasificación ha sido correcta), incrementa los pesos acercándose ligeramente al vector de entrada (premio). Por el contrario, si la clase ganadora es diferente de la clase del vector de entrada (la clasificación ha sido errónea), decrementa los pesos alejándose ligeramente del vector de entrada (castigo).

Sea $\mathbf{x_i}(t)$ el documento de entrada en el instante t, y $\mathbf{w_k}(t)$ representa al vector de pesos para la clase j en el instante t. Las siguiente ecuaciones definen el proceso básico de aprendizaje para el algoritmo LVQ.

$$\mathbf{w_c}(t+1) = \mathbf{w_c}(t) + s\cdot\alpha(t)[\mathbf{x_i}(t) - \mathbf{w_c}(t)]$$

donde $s = 0$, si $k \neq c$; $s = 1$, si $\mathbf{x_i}(t)$ y $\mathbf{w_c}(t)$ son de la misma clase; y $s = -1$, si no lo son y donde $\alpha(t)$ es la tasa de aprendizaje, siendo $0<\alpha(t)<1$, una función monótona decreciente. En la mayoría de los casos $\alpha(t)$ se inicializa con un valor bastante pequeño, por ejemplo, menor que 0,1 [Kohon88] y va decreciendo con el tiempo hasta un cierto umbral, u, muy próximo a 0. En nuestros experimentos $\alpha(t)$ se ha inicializado a 0,05 y decrementa linealmente hasta u=0,001 de acuerdo con la siguiente ecuación

$$\alpha(t+1) = \alpha(t) - \frac{\alpha(0) - u}{K}$$

donde K es el número de clases.

5. Experimentos

5.1 Generación del corpus

Para comparar los cuatro algoritmos de categorización, hemos utilizado un conjunto de ficheros generados a partir de la Biblia bilingüe. Dado que la Biblia está estructurada en libros, capítulos y versículos, hemos podido crear una partición muy ajustada a nuestros experimentos. Hemos generado un total de 1.189 documentos de entrenamiento y 1.189 documentos de evaluación, cada uno de ellos perteneciente a

una de las 66 clases posibles, es decir, cada libro de la Biblia define a una clase distinta.

Nuestro sistema de generación de documentos ha sido el siguiente:

Para cada capítulo de cada libro hemos creado dos archivos *li_cj_tr.txt* (fichero para el entrenamiento) y *li_cj_ev.txt* (fichero para la evaluación) donde *i* toma valores entre 1 y 66 y *j* entre1 y el número total de capítulos que tenga el libro *i*. La partición de cada capítulo se realiza considerando unidades de cuatro versículos, distribuyendo tres al archivo de entrenamiento y uno al de evaluación. Cada documento es de una clase determinada, concretamente la correspondiente al libro al que pertenece.

Para minimizar la dimensión del espacio vectorial, hemos eliminado las palabras vacías, pasando la lista de parada [Frake92] de SMART [Buckl85], y extraído las raíces [Stemm01]. Después de este procesamiento se obtiene un total de 22,054 palabras que constituyen el dominio de la aplicación.

5.2 Inicializaciones

Los vectores de entrada **x** se generan aplicando el MEV a los 1.189 ficheros de entrenamiento tomados como un solo corpus. Los vectores de consulta **q** para la evaluación se obtienen aplicando la fórmula basada en la frecuencia de términos [Salto89] como se muestra en el apartado 2.

Para inicializar los vectores de pesos **w** se pueden utilizar varios métodos: inicializarlos a cero (como en el algoritmo de Rocchio), de manera aleatoria o a un valor específico dado. Concretamente, para el algoritmo LVQ el vector de pesos iniciales puede incorporar cierto conocimiento sobre el problema a resolver, mejorando de esta manera los resultados [Kohon98]. Precisamente, por esto, hemos optado por inicializar los vectores de pesos de entrenamiento de los tres algoritmos basados en RN con esta información adicional, fusionando todos los ficheros de entrenamiento pertenecientes a la misma clase *j* en un único fichero *clasej.txt*. Así se obtienen 66 ficheros para generar los pesos iniciales de los vectores de entrenamiento aplicándoles el MEV.

Cada vector de pesos $\mathbf{w_j}$ tiene la misma dimensión que los vectores de entrada **x** y se tienen tantos vectores de pesos como categorías, de manera que el vector de pesos $\mathbf{w_1}$ se corresponde con la clase 1, el vector $\mathbf{w_2}$ con la categoría 2 y así sucesivamente hasta el vector $\mathbf{w_{66}}$ que representa a la clase 66.

5.3 Entrenamiento y evaluación

El entrenamiento con el algoritmo de Rocchio se lleva a cabo aplicando la regla de cálculo del apartado 4.1 para cada una de las categorías iterativamente.

Para los algoritmos basados en RN todos los vectores de entrada **x** son procesados durante el entrenamiento tantas veces como categorías haya, en nuestro caso 66 veces (t=1,..,K).

Para cada uno de los vectores de entrada se aplica la regla de aprendizaje correspondiente a cada uno de los algoritmos, quedando finalmente los vectores de pesos **w** con los valores definitivos para la evaluación

Tras el entrenamiento, la categorización se realiza calculando la similitud de cada vector de evaluación **q** con los vectores de pesos **w**, seleccionando, a continuación, la categoría con mayor similitud como solución.

Estos vectores **q**, son de tres clases. En primer lugar, hemos recogido los ficheros generados como evaluación en el proceso de generación del corpus bilingüe, obteniendo el primer conjunto de evaluación, que hemos etiquetado como *español-inglés*. En segundo lugar, hemos procesado estos vectores **q**, separando en dos ficheros las palabras de cada idioma que contiene. De esta forma obtenemos dos conjuntos de documentos de evaluación, etiquetados como español e inglés, respectivamente.

6. Resultados

La Tabla 1 muestra los resultados obtenidos después del entrenamiento de cada uno de los algoritmos para el total de ficheros de evaluación. La precisión *macroaveraging* que presentamos es la media de las precisiones *microaveraging* de todas las categorías.

No se presentan valores de *recall*, ya que este enfoque de categorización siempre toma una decisión por algún significado, con lo cual el *recall* es igual a 1 (cobertura del 100%).

Como se puede observar el algoritmo LVQ supera al resto. Además los tres algoritmos basados en un enfoque neuronal dan mejores resultados que el ampliamente utilizado algoritmo de Rocchio. Los algoritmos de RN funcionan mejor porque realizan un entrenamiento iterativo. Además el LVQ durante el entrenamiento tiene en cuenta como va evolucionando el sistema, premiándolo o

castigándolo según se comporte a medida que avanza el tiempo. Se trata de un aprendizaje supervisado que dirige la modificación de pesos según el comportamiento del sistema en cada momento, haciendo que las clases compitan entre sí.

	P microavg	***P macroavg***
	Español	
Rocchio	56,27	63,34
WH	64,42	59,12
KW	66,44	61,12
LVQ	73,76	66,42
	Inglés	
Rocchio	61,40	75,69
WH	70,56	77,62
KW	73,00	76,51
LVQ	76,87	80,18
	Español-Inglés	
Rocchio	58,03	61,65
WH	67,37	61,24
KW	69,64	60,58
LVQ	75,11	66,23

Tabla 1

Los resultados de las consultas bilingües promedian los valores obtenidos por los experimentos de consultas monolingües, como era de esperar, ya que ambas lenguas son casi disjuntas.

En la Tabla 2 se muestra el porcentaje de mejora del algoritmo LVQ con respecto al resto de algoritmos evaluados entendiendo como porcentaje de mejora, *M*, lo siguiente:

$$M = \frac{E - E_{LVQ}}{E} \times 100$$

donde E_{LVQ} es el número de errores obtenidos con el algoritmo LVQ y E es el número de errores en cada uno de los otros algoritmos.

	Rocchio	**WH**	**KW**
E	40,00%	26,24%	21,80%
I	43,76%	21,43%	14,33%
E-I	40,68%	23,71%	18,01%
Prom.	41,48%	23,79%	18.05%

Tabla 2

El mejor *microaveraging* se obtiene cuando la evaluación se hace con consultas exclusivamente en inglés, mientras que el peor resulta con consultas exclusivamente en español. Una posible causa podría ser la elección de un *stemmer* español menos adecuado que para el inglés. Por otra parte, las diferencias en los resultados obtenidos con LVQ para los tres tipos de consulta no difieren mucho entre sí (E-73,76, E-I-75,11 e I-76,87) con lo que se puede deducir que al algoritmo LVQ le afecta menos la elección del *stemmer* que a los otros algoritmos.

7. Conclusiones y futuros trabajos

En este trabajo hemos presentado un sistema de categorización multilingüe empleando un corpus paralelo, previamente generado.

Se ha realizado una evaluación directa de nuestro método de categorización basado en el aprendizaje competitivo, obteniéndose resultados muy significativos y superiores a los obtenidos por otros algoritmos usados con éxito en categorización. De hecho se consigue una mejora de nuestro método del orden de 27,77% frente a otros.

También hemos expuesto un método de evaluación sistemático de la categorización que nos permite comparar la efectividad de varios enfoques.

Consideramos como principal línea de trabajo futuro el estudio del LVQ en otras tareas, así como la integración de recursos lingüísticos. Asimismo, estamos interesados en evaluar la aportación que puede realizar la desambiguación en esta tarea multilingüe.

8. Referencias

[Buckl85] Buckley, C.; Implementation of the Smart Information Retrieval System. Technical Report 85-686, Cornell University. 1985.

[Buena97] Buenaga, M.; Gómez, J.M.; Díaz, B.; Using WordNet to Complement Training Information in Text Categorization. Proceedings of Second International Conference on Recent Advances in Natural Language Processing (RANLP), 1997.

[Davis98] Davis, M.W.; On the effective use of large parallel corpora in cross language text retrieval. In [Grefe98].

[Davies99] Davies, M. The Polyglot Bible. En http://mdavies.for.ilstu.edu/polyglot/ , 1999.

[Frake92] Frakes, W.; Baeza, R. Information Retrieval: Data Structures and Algorithms. Prentice-Hall. 1992.

[Grefe98] Grefenstette, G.; Cross-Lingual Information Retrieval. Edit. Kluwer Academic Publisher. 1998.

[Hersh94] Hersh. W.; Buckley, C.; Leone, T.J.; Hickman, D.; Oshumed: an interactive retrieval evaluation a new large text

collection for research. Proceedings of ACM SIGIR, 1994.

[Honke95] Honkela, T.; Pulkki, V.; Kohonen, T. Contextual relations of words in Grimm tales, analysed by self-organizing map. Proceedings of International Conference on Artificial Neural Networks, ICANN-95. París 1995. EC2 et Cie. P. 3-7.

[Kaski95] Kaski, S.; Kohonen, T. Exploratory data analisis by the self-organizing map: structures of welfare and poverty in the world. En: REFENES, A. y otros (eds.) Neural Networks in Financial Engineering: Proceedings of the Third International Conference on Neural Networks in the Capital Markets. London, 11-13 October 1995. pp. 498-507.

[Kivin97] Kivinen, J.; Warmuth, M.K.; Exponentiated gradient versus gradient descent for linear predictors. Information-and-Computation. vol.132, no.1; 10 Jan. 1997; p.1-63.

[Kohon88] Kohonen, T.; Self-organization and associative memory. 2ª Edición, Springer.Verlag, Berlín, 1988.

[Kohon98] Kohonen, T.(1998). The self-organizing map. Neurocomputing vol. 21, pp. 1-6, 1998

[Lewis92] Lewis, D. D.; Representation and learning in information retrieval. PhD thesis, Department of Computer and Information Science, University of Massachusetts, 1992.

[Lewis96] Lewis, D. D.; Schapire, R. E.; Callan, J. P.; Papka, R.; Training algorithms for linear text classifiers. In SIGIR'96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1996.

[McCar98] McCarley, J.S.; Roukos, S.; Fast documents traslations for cross languaje information retrieval. Proceedings of AMTA98. Springer-Verlag, 1998.

[Nie99] Nie, J.Y.; Isabelle, P.; Simard, M.; Durand, R.; Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the Web, *ACM-SIGIR conference*, Berkeley, CA, pp. 74-81. 1999.

[Resni99] Resnik, P; Olsen, M.D.; Diab, M.; The Bible as parallel corpus: annotating the "Book of 2000 Tongues". Computers and the Humanities

[Rocch71] J.J Rocchio Jr. Relevance feedback in information retrieval. In G. Salton, editor, The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall, 1971.

[Saham98] Sahami, M. Editor of Proceedings of the AAAI'98/ICML'98 Workshop on Learning for Text Categorization, 1998.

[Salto83] Salton, G; McGill, M.J.; Introduction to modern information retrieval. Mcgraw Hill, 1983.

[Salto89] Salton, G.; Automating text processing: the transformation, analysis and retrieval of information by computer. Addison-Wesley, 1989.

[Stemm01] http://www.smartlogik.com/.

[Ureña98] Ureña, L.A.; Buenaga, L.A.; García , M.; Gómez, J.M.; Integrating and evaluating WSD in the adaptation of a lexical database in text categorization task. Proceedings of the First Workshop on Text, Speech, Dialogue, 1998.

[Widro85] B. Widrow and S. Sterns. Adaptative Signal Processing. Prentice-Hall, 1985.

[Yang99] Yang, Y.; An Evaluation of Statistical Approaches to Text Categorization. Information Retrieval Journal, Vol 1, N ½, 1999.

[Yang99b] Yang, Y; Liu, X.; A re-examination of text categorization methods. Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 1999.