Classifying texts through natural language parsing and semantic filtering

Jacques Chauché, Violaine Prince

Classifying texts through natural language parsing and semantic filtering

Jacques Chauché*, **Violaine Prince***

* University of Montpellier 2 and LIRMM-CNRS
161 Ada Street, 34392 Montpellier, FRANCE
{chauché, prince}@lirmm.frl

Abstract

This paper presents a study in text classification through semantic and syntactic natural language processing. The authors have used a parser for French, SYGFRAN, and applied it to a real project of press articles classification. The results of this research on a corpus of $4,843$ texts containing more than $76,000$ sentences are described. Classification into 37 categories has been obtained through meaning discrimination by semantic filtering techniques, explained in the document.

1. Introduction

Automatic text classification is a domain where text mining and statistical techniques produce results relying on word occurrence frequency. A wide range of learning methods, including regression models (Y.Yang and Chute, 1992), *k-nn* based algorithms (Y.Yang and Liu, 1999), naive bayesian models (McCallum and alii, 1998) or associated with decision trees (Lewis and Ringuetee, 1994), have been applied to this field. All these techniques use a training set of text, which have been classified into a given number of pre-defined categories. If machine learning algorithms, either supervised or not, have led to interesting results, one of their main liabilities is that they are sensitive to their training data and/or to their test corpora. Whereas a content analysis environment, based on syntactic parsing and semantic interpretation, is able to produce the same type of result independently from its corpora characteristics, when provided with a semantic filtering technique that performs classification.

The goal of this paper is to study the capabilities of syntactic and semantic processing as a classification process. We have been using a parsing environment for the French language, SYGFRAN, very briefly described in next section. We have tackled a real application, through a contract with a company that offers bunches of press articles to its clients and must categorise more than $5,000$ texts per day. When we built up the common project, the company wanted to know if natural language techniques could supply it with a better model of automatic classification. This article presents the results of our study .The selected principle for classification is *semantic filtering*: Documents are filtered by centroids representing categories in a supervised classification method. The originality of the approach is that the parser, as a tool used by classification, is not modified by it, and classification algorithms rely on tuning and on the stability of centroids.

2. Syntactic and Semantic Parsing Environment

2.1. The Parser

The SYGFRAN parser is based on Markov's algorithms extended to trees (Chauché, 1984). It has been designed to analyse any language, the grammar of which could be written as a set of tree transducers. For French, a grammar of about $12,000$ rules has been written manually. Associated to SYGFRAN, the dictionary contains $60,000$ entries, represented by vectors, for semantic processing. A semantic representation is assigned to any segment of a text, as well as to a complete text, through a process that conjugates syntactic analysis and a vector calculus for semantics. As an output, SYGFRAN provides a syntactic tree of every sentence. Its nodes are composed of a set of tags (if not terminal), and a lexical entry and a set of tags (terminal). Tags determine the morphological and syntactic categories. Dependency functions labels (such as subject, object, complement) are also provided.

2.2. Semantic Vectors

The chosen representation is based on the formalism of space vectors, but does not use it in Salton's way (Salton, 1988), which is dependent on the documents collection that models the space. Our approach is a "Roget-based" representation for semantics, as it has been used in computational linguistics (Yarowsky, 1992; Wilks, 1998). It supposes the existence of a stable thesaurus (Roget, 1852), i.e. an ontology of basic concepts for language. Using a Roget-like generator family preserves the dimension of the vector space. For English, the lexical vector space dimension was originally 1043: Its most up-to-date version has shrank this number down to 1000. For French, the language with which we chiefly work, lexicologists have defined a family of 873 concepts, hierarchised in four levels (Larousse, 1992). This leads to a space which dimension is 873.

2.2.1. Indexing every term and defining the lexical vector space

All entries are indexed by the conceptual family. For instance, the word "today", has the indexes 196.1, 202.3, meaning that the adverb "today" is projected on concepts 196 (PRESENT) et 202 (RECENT). Values after the (".") are morphological indications. Formally, we represent this by the formula : $\Pi_i(\vec{t}) = 1$ if C_i indexes t, else $\Pi_i(\vec{t}) = 0$. The projection of the linguistic space into a vector space is thus formalised through the following process. Let the

vector $\vec{t}$ be associated to the lexical entry t. For this we define an application $\mathcal{V}$ from $\mathcal{D}$, space of the dictionary entries, and containing t terms, to $\mathcal{C}$ the conceptual family of Larousse concepts (dimension = 873). We have $\mathcal{C} \supset \mathcal{D}$ and $\mathcal{V}$ defined as following:
$\mathcal{V} : \mathcal{C} \times \mathcal{D} \rightarrow (0,1)^{873}$
(t,c$_i$) $\rightarrow \mathcal{V}(t, c_i) = 1$ if c_i indexes t else$\mathcal{V}(t, c_i) = 0$.
Let $\vec{\mathbf{V}}$ be the vector space generated by $\mathcal{C}$ and the application $\mathcal{V}$. We have : $\forall t \in \mathcal{D}, \exists \overrightarrow{t} \in \vec{\mathbf{V}}$, such as $\vec{t}$ is built by the application $\mathcal{V}$. In other words, every lexical entry has a vector representation.

2.2.2. Semantic Vector Space

Defining internal composition laws in $\vec{\mathbf{V}}$ such as the (normed) sum and the product by a scalar, allows to associate a vector to every result of these laws combination (J.Chauché, 1990). This vector, although representing semantics, could neither be associated to an elementary term nor to a dictionary entry. If we assume compositionality in linguistic semantics, according to which the meaning of a set of words is the result of a function taking this set as an argument, then $\vec{\mathbf{V}}$ defines a real semantic space, and not only a lexical semantic space. Therefore, for every set of words, $x = w_1, w_2, \ldots, w_n$ of $\mathcal{D}$ such as $\vec{w_1}, \vec{w_2}, \ldots, \vec{w_n}$ are their associated vectors in $\vec{\mathbf{V}}$, we have:
$\exists \vec{v_x} \in \vec{\mathbf{V}}, \forall n \in N, \exists f_n$ a function, such as :
$f_n : \vec{\mathbf{V}}^n \rightarrow \vec{\mathbf{V}}$ and $f_n(\vec{w_1}, \vec{w_2}, \ldots, \vec{w_n}) = \vec{v_x}$.
The problem being that any set of words is not necessarily "meaningful" in the linguistic semantic space, one has to define f_n functions as formal images of existing linguistic functions. The latter give rise to ordered and constrained sets of words that are sentences, and more largely, texts.

2.2.3. Texts Semantic Vectors

In natural language, it is the syntactic relationship that dictates association rules between words, and thus defines the semantics of the result. Computing a semantic vector for every text segment, seen as an ordered set of words, is based on a previous parsing that will allow to assign a weight to the vector of a word or a group of words, according to its syntactic role. Thus we need to provide equations to calculate the semantic vectors of a phrase and of a sentence.

Semantic Vector of a Phrase Let γ be a parsed phrase. It could be defined as an ordered set of words w_1, w_2, …, w_n. Its vector, $\vec{\gamma}$ is obtained as the recursive normed sum of : (1)words directly belonging to the phrase ; (2) subphrases composing the phrase. Thus, for every phrase of an i level in the parsing tree (root has got level 0 and terminal leafs the lowest level, n), we have the recursive formula for $\vec{\gamma} i$:

$$\vec{\gamma_i} = \frac{\sum_j \overrightarrow{(\lambda_j v_{j,i+1})}}{\| \sum_j \overrightarrow{(\lambda_j v_{j,i+1})} \|} \tag{1}$$

Where the $\vec{v_{j,i+1}}$ are vectors of either words, or subphrases of the phrase ϕ_i, their level being immediately inferior ($i+1$). j is the subscript that describes the span of the subtree whose root is γ_i. λ_j is the weight assigned to the vector $\vec{v_{j,i+1}}$, according to the syntactic role played by the word (or subphrase) whose vector it is. λ_j is such that if $\vec{v_{j,i+1}}$ is the vector of a governor , then λ_j equals $2 * \lambda_{j+1}$ which is the weight of the following vector $\vec{v_{j+1,i+1}}$.

Semantic Vector of a Sentence Let σ be a parsed sentence. Since σ is a phrase of level 0, let ϕ_j be the phrases of level 1 (directly under the root) composing σ. The formula computing σ is:

$$\vec{\sigma} = \frac{\sum_j \overrightarrow{(\lambda_j \phi_{j,1})_{nor}}}{\| \sum \overrightarrow{(\lambda_j \phi_{j,1})_{nor}} \|} \tag{2}$$

Texts Vectors The vector of a text is not calculated as the sentence vector, since a sentence represents a syntactic unit and is marked with a *syntactic closure*. A text is a stylistic unit, individually defined by an author, and has no particular syntactic properties. It appears as a set of sentences. Its image in space $\vec{\mathbf{V}}$ must thus be defined as the **centroid vector** of its sentences vectors. Centroids are used as categories representatives in classification literature (Eui-Hong and Karypis, 2000; Theeramunkong and Lertnattee, 2001). Here, we use the same concept for both texts and classification categories. Let T be a given text. $T = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ where the σ_i are sentences of T. These sentences are respectively represented by their vectors $\vec{\sigma_i}$ as shown in the preceding paragraphs. The centroid text vector $\vec{T}$ is defined as the **normed sum** of the σ_i. *Properties*: (1) Since all considered vectors are normed, then we remove from our notation the subscript *nor*. Any vector $\vec{v}$ is in fact a **normed vector**. (2) If $T = (\sigma)$ then $\vec{T} = \vec{\sigma}$.

Sets of documents Sets of documents (when documents are seen as texts) behave like sets of sets, from the point of view of their projection in $\vec{\mathbf{V}}$. Let $S = (T_1, T_2, \ldots, T_m)$ be a set of m documents, and $\vec{T_1}, \vec{T_2}, \ldots, \vec{T_m}$ their respective vectors calculated as indicated in the preceding paragraph. In compatibility with the centroid text vector, we define a **centroid set vector** $\vec{S}$ as the normed sum of $\vec{T_1}, \vec{T_2}, \ldots, \vec{T_m}$.

3. Application to Document Classification

Semantic projection of texts and sets of texts in a vector space could be used as a technique to classify texts according to a reference vector direction. A set of texts S may be considered by human experts as a 'thematic unit'. It is legitimate to consider $\vec{S}$, centroid vector of all elements of S and image of S in $\vec{\mathbf{V}}$, as the corresponding **thematic vector**. This is what underlies all propositions of centroid categories representants in classification literature using supervised methods. However, being so corpus dependent, two conditions are mandatory so that the assumption could be generalised:
- **Coverage** : The set S must be large enough in number, in order to represent every semantic direction relevant to the category
-**Stability** : S must be homogeneous enough so that *its vector $\vec{S}$ is stable*. In other words, if a new text T'_1 appears as a possible candidate for the category, the vector representing $T \cup (T'_1)$, i.e. $\overrightarrow{T \cup (T'_1)}$ is almost equal to $\vec{S}$. Thus, $\overrightarrow{T \cup (T'_1)} \approx \vec{S}$).

3.1. Classification Procedure: Building Categories Vectors

Since our system is not corpus sensitive, then using it for classification needs the following elements : (1) a set of categories defined by the final user (here the company's experts) ; (2) a set of texts representative of each category, enabling us to initialise the filter by computing the **categories vectors** . This set of texts is called the **tuning set** (or training, if we were in a learning context). (3) A stream of texts to be classified by the system, called the **test set** but with an existing human classification, to measure recall and precision. (4) A **filtering algorithm** able to assign one or more categories to a given text (experts have told us that a text could have a multiple classification). To classify texts, we did not choose to compare texts to each other, which is the most widespread technique. We have preferred to compare every input text T to every category vector and then to provide a **classification vector**: Its components are the set of categories vectors ordered from best fitting to worst.

3.1.1. Categories Vectors

Let K be the set of given categories. $K = (K_1, K_2, \ldots, K_n)$. For instance, *Education* is the name of a category, and we also have *International News*, *Companies services* or *Textile and Fashion*. The vector representing K_i is the centroid of all T_{ij} texts of the tuning set $\mathbf{N}$ assigned by experts to K_i. j belongs to $[1, 2, \ldots, n]$ where n is such that $\vec{K_i}$ is stable.

3.1.2. Classifying a text in a category : the Filtering Algorithm

Once the $\vec{K_i}$ are computed and stabilised, these vectors are used as references to classify texts belonging to $\mathbf{E}$, the test set. For this, several solutions are possible: (1)Using a cosine vector distance: In order to get a good discriminating power, categories vectors must be well differentiated in $\vec{\mathbf{V}}$; (2) Computing a **matching measure** and using this measure as a classification criterium: If the preceding solution is not appropriate, the category vector must be used as a **filter**, and the matching measure determines the level of filtering. (3) Using both a matching measure and a distance between matching vectors: It is a problem of relative categorisation, and category preference (over another), in particular when the matching measure matches a given text with many categories.

All three solutions have to be applied in order to get the best categorisation score. We indicate hereafter the formulas, and we show, in next subsection, how the application properties have driven us to undergo a fine tuning of some parameters.

3.1.3. Matching Measure

The category vector $\vec{K}$ is used here as a filter: We omit the subscript i in order not to introduce a confusion with indexes of vector components in the generator family $\mathcal{C}$. To get a reliable and discriminating matching, we proceed as follows: (1) $\vec{K}$ is sorted with a decreasing order of its components intensities. (2) $\vec{K}$ is reduced to its Nb highest values (we experimentally found $Nb = 250$ after many tests). This reduction helps not to be burdened with a space of dimension 873, and components with negligible values are discarded. The reduction to a space of Nb dimension not only accelerates computation, but also ensures a better discrimination. This sorted vector for K is called $\vec{K_{sort}}$. (3) Let T be a text, and $\vec{T}$ its vector. We also sort $\vec{T}$ according to a decreasing order of its components values, and we reduce it to its Nb first components. We name $\vec{T_{sort}}$ the result. (4)The **rank distance** between the Nb most intense components is first calculated. Rank distance denotes a difference between the rank of a given component (a C_t concept of the generator family) in $\vec{T_{sort}}$ and its rank in the reference category vector $\vec{K_{sort}}$. Let i be the rank of a given concept in $\vec{K_{sort}}$, and $\rho(i)$ its rank in $\vec{T_{sort}}$. The formula for rank distance is the following:

$$E(i, \rho(i)) = \frac{(i - \rho(i))^2}{Nb^2 + (1 + \frac{i}{2})} \tag{3}$$

When $\vec{K}$ and $\vec{T}$ activate the "same" concepts then, in their sorted vectors, $(i - \rho(i))^2$ tends towards 0 and so would be $E(i, \rho(i))$ for every i ranging from 1 to Nb. (5) The **intensity difference** needs also to be defined as a comparative measure between both vectors intensities for the same concept. Let a_i be the intensity of rank i (in $\overrightarrow{K_{sort}}$) and $b_\rho(i)$ the corresponding intensity in $\vec{T_{sort}}$. Both intensities address the same concept C_t in $\mathcal{C}$. The intensity difference is given by the formula :

$$I(i, \rho(i)) = \frac{\|a_i - b_{\rho(i)}\|}{Nb^2 + \frac{1+i}{2}} \tag{4}$$

When $\vec{K}$ and $\vec{T}$ activate concepts with a comparable intensity, then $\|a_i - b_{\rho(i)}\|$, in their sorted and reduced vectors, tends towards 0 and so does $I(i, \rho(i))$. (6) Both rank distance and intensity difference are then used in the formula of Matching measure P. The **Matching measure formula** is given by the following equation:

$$P(\overrightarrow{K_{sort}}, \overrightarrow{T_{sort}}) = \left(\frac{\sum_{i=0}^{Nb-1} \frac{1}{1+E(i,\rho(i))*I(i,\rho(i))}}{Nb}\right)^2 \tag{5}$$

If $\vec{K}$ and $\vec{T}$ tend to be very close in both concepts and intensity, then P being a inversely proportional function, of E and I, tends towards 1. Matching goes up with very similarly shaped vectors.

Note:P values belong to the interval $[0, 1]$. It not only measures the extend of matching between the text sorted vector components (the same profile bases), but also the matching of intensity between these profiles (the same peaks). P is not a classical similarity measure since P is not symmetrical.The Nb most intense components of $\vec{K}$ determine the restricted basis on which measures are calculated. Those of $\vec{T}$ might not be the same. *Convention*: As only sorted and reduced vectors are compared, we will neglect the $sort$ subscript for both sorted $\vec{K}$ and $\vec{T}$.

3.1.4. Matching Measure and Distance

Matching a text with a category implies more than considering this text vector as compatible with the category vector.The matching measure alone may assign a vector to many categories. We must then preferentially classify a

text in a category. Distance is here important again since it may offer a comparison between concording categories. For this, we use the cosine distance δ . A new distance between the sorted vectors $\vec{K}$ and $\vec{T}$ is defined as:

$$\triangle(\vec{K},\vec{T}) = \frac{P(\vec{K},\vec{T}) * \delta(\vec{K},\vec{T})}{\beta * P(\vec{K},\vec{T}) + (1-\beta)\delta(\vec{K},\vec{T})} \quad (6)$$

where β is a reinforcing scalar. $\triangle$ is an angular distance between sorted vectors, that is augmented or reduced according to the matching measure value.

3.1.5. Text Classification Vector

For every text T, that needs to be classified, let us consider that its vector, $\vec{T}$, has the following properties : (1) a centroid vector of all T sentences has been computed and normed ; (2) it has been sorted according to the decreasing intensity of its components (the original 873 concepts basis) ; (3) it has been reduced to its Nb most important components (reduction of the original space to a subspace of dimension Nb). This vector, for the sake of clarity, has received the name $\vec{T}$, although it is not the original centroid. But as we handle vectors for classification, only a sorted and reduced vector plays a role in our application. From this vector, we derive a **classification vector** of the text T, named $\overrightarrow{T_{class}}$. Two methods, related to Matching and distance between concordant vectors, are applicable for computing $\overrightarrow{T_{class}}$ components. If $K = (K_1, K_2, \ldots, K_p)$ is the set of categories in which the texts must be classified, and $\vec{K_1}, \vec{K_2}, \ldots, \vec{K_p}$ their respective sorted vectors of dimension Nb, the projection of $\overrightarrow{T_{class}}$ on its rank i components is computed as: (1) $P(\vec{K_i},\vec{T})$ where P is the matching measure: the i ranking component in the classification vector of T is the matching value between the vector of the category K_i, acting as a reference and $\vec{T}$; (2) $\triangle(\vec{K_i},\vec{T})$ where $\triangle$ is the distance calculated above. (3) A **Categories Vector of** T, named $\overrightarrow{cat(T)}$ is then computed. It is a vector of dimension p (p being the number of categories) and its projection on rank i is:

$$\Pi(\overrightarrow{cat(T)}, i) = K_j$$

where K_j is the category number j (in a set of p categories) and where the matching value P (respectively $\triangle$) between K_j and T appropriate vectors is such as it is the $i$eth in the list of concordant categories.

4. Experiment and Results

4.1. Data

The corpus provided by the company has been delivered in three steps. A first set of $2,122$ articles extracted from French newspapers and magazines and agencies, was first proposed. It was "indexed" with 301 categories and no ontology (hierarchy between categories) was provided. These categories were press topics, and transmitted to us as a flat list. Moreover, several categories overlapped, in no predictible way. Researchers such as (Theeramunkong and Lertnattee, 2002), when confronted with the same situation, highlighted a problem in defining precision in classification. Texts size varied from some sentences to a few pages. This set has been classified by human experts (journalists) and, as some articles could belong to more than one category, the $2,122$ articles generated $3,568$ links. A *link* is the relationship between an article and a category. We considered this first set as a *tuning set*. When applying our computing method (detailed in the previous subsection), we noted very quickly that a great majority of category vectors did not stabilise. When looking for coverage and stability, we noticed that some categories contained too few articles. 301 categories gave an average of only 10 links per category. Therefore, we asked for a more compact tuning set and received a second one. The second set was composed of: (1) The tuning set $\mathbf{N}$ of $2,400$ articles, representing $2,555$ links and 37 categories (average number of links per category = 69). According to our first experiment where many categories failed to stabilise, we noticed that under 30 links per category, the behaviour was chaotic. Over 30, most category vectors stabilised. In this set, every article contained from 10 to 400 sentences. Every sentence ranged from 8 to 50 words. (2) A test corpus, $\mathbf{E}$, composed of $2,443$ articles, with close characteristics in number of words and size to the tuning test ; (3) The $2,469$ links of the test corpus, to measure recall and precision.

4.2. Recall and Precision

Since we always calculate a classification vector for a text, we generate p links for every text, among which, the one to three expert links are necessarily present. When considering a complete classification vector, every text is classified into every category, thus creating an important noise. Recall is 100% (all texts are linked to all categories) but precision is then meaningless. What is overlooked in this raw approach to evaluation, is that a classification vector provides a rank of reliability when linking a text to a category. Therefore one needs to define which part of the classification vector should be considered as a reasonable translation of a 'correct classification'. The crucial question we were asked was the following: 'Given a text T, not yet classified, what is the probability to have it correctly classified in one (or more) of our categories? Since categories constitution was not at stake, but the point of view of the text to be classified was crucial, we have been led to define the notion of **correct system link**, as a global measure, to feed a relevant evaluation frame. The important clue is the rank of a category c, given as a link in the pair (t, c) for the text t, in the category vector of t, $\overrightarrow{cat(t)}$. If this rank is further than a given acceptable value m, then the link provided by the system is to be neglected. If it is less or equal to m then the link is considered as sensible. This considerably reduces the number of eligible system links. Since many articles were classed in several categories, considering links only was not a good evaluation. So, we defined the concepts of **scope** and **sequence**. The scope, denoted by m, corresponds to the number of categories among which we try to meet the human expert's classification. The sequence is the m-uplet of fitting categories provided by the system or the expert. If the system sequence contains the human expert sequence, then it is considered as correct and counts as 1. If the sequence is

totally distinct from the human expert sequence then it is wrong and counts for 0. Last, if the sequence intersects the human expert sequence but does not contain it, it counts as partially correct and scores 0.5. The **scoped overall recall** is : $\rho_m = \frac{rc+pc}{ec}$ where rc is the amount of correct system sequences for all texts, pc the partially correct ones, and ec is the number of expert sequences for all texts in a scope of m categories. We also describe an **scoped overall precision** as : $\pi_m = \frac{rc+pc}{rc+pc\omega c}$ where $rc + pc$ is the number of correct and partially correct system sequences for all texts, and ωc is the number of wrong classifications for all texts in a scope of m categories.

4.3. Obtained Results

We run tests with scopes equal to 1, 2, 3 and 10. We did them for both the tuning and the test set. The values are provided in the following table. Notice that there is almost no difference between the tuning and test sets for scoped recall and precision: This confirms that the system is not sensitive to training. Both recall and precision evolve the same way: This is caused by their broad definition(inclusion instead of equality, and intersection scoring as half). A strict recall and precision (with only rc) can never reach 100%.

Recall Scope	Tuning Set	Test Set
1	0.47	0.47
2	0.61	0.62
3	0.94	0.93
10	1	1
Precision Scope	**Tuning Set**	**Test Set**
1	0.47	0.47
2	0.69	0.72
3	0.82	0.85
10	1	1

Table 1: The Scoped Recall and Precision Values

5. Conclusion

The results obtained with this semantic filtering technique with a non strict definition of recall (R) and precision (P) for evaluation have been provided afterwards to the human experts who came up with the following conclusions. (1) Most of their own sequences, done with haste, were reconsidered as unsufficient. Many categories "discovered" by the system with a scope superior to 2 were qualified as interesting and sensible. This tended to suggest a mandatory multiple classification for the articles. (2) The experts found that a scope of "3" was the best trade-off with a good reliability measure (the ratio P over R is a good one) and discarded higher scopes, because the increase in P and R was a side effect of the broad definition, and not a real good measure. (3) When considering partially false positives, i.e. categories associated by the system to texts within partially correct sequences, experts saw that, most of the time, these categories corresponded to minor topics in the articles, something that a thorough analysis was able to show, but that a quick overview has naturally overlooked. (3) Last, for completely wrong sequences, experts noticed that it was mostly related to categories that were either vague, or needed a real contextual and pragmatic knowledge, deriving from their personal expertise, something that an automatic analysis could not achieve with pure algorithmic or calculation methods.

So in conclusion, the provided method has been considered as reliable as a **topical classification method**, with the following requirements: (1) Texts must be totally parsed and transformed into vectors. (2) Categories must be first tuned with coverage and stability conditions. (3) Multiple classification is to be encouraged since texts might possibly appear as multi-topical.

6. References

Chauché, J., 1984. Un outil d' analyse multi-dimensionnelle du discours. In *Proceedings of COLING-84*.

Eui-Hong, H. and G. Karypis, 2000. Centroid-based document classification: Analysis and experimental results. In *Proceedings of PKDD*.

J.Chauché, 1990. Détermination sémantique en analyse structurelle : une expérience basée sur une définition de distance. *TA Information*, 1/1:17–24.

Larousse, 1992. *Thésaurus Larousse - des idées aux mots, des mots aux idées*. Ed.Larousse, Paris.

Lewis, D.D and M. Ringuetee, 1994. A comparison of two learning algorithms for text categorisation. In *Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval*.

McCallum, A. and alii, 1998. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the 15th International Conference on Machine Learning.*.

Roget, P., 1852. *Thesaurus of English Words and Phrases*. Longman, London.

Salton, G., 1988. *Term-Weighting Approaches in Automatic Text Retrieval*. McGraw-Hill computer science series.

Theeramunkong, T. and V. Lertnattee, 2001. Improving centroid-based text classification using term-distribution based weighting system and clustering. In *Proceedings of ISCIT01*.

Theeramunkong, T. and V. Lertnattee, 2002. Multi-dimensional text classification. In *Proceedings of COLING2002*.

Wilks, Y., 1998. Language processing and the thesaurus. In *Proceedings of the National Language Research Institute*.

Yarowsky, D., 1992. Word-sense disambiguation using statistical models of roget's categories trained on large corpora. In *Proceedings of COLING92*.

Y.Yang and C.G. Chute, 1992. A linear least square fit mapping method for information retrieval from natural language texts. In *Proceedings of COLING92*.

Y.Yang and X. Liu, 1999. A re-examination of text categorisation methods. In *Proceedings of the 22nd ACM SIGIR Conference*.