Pré-traitements classiques ou par analyse distributionnelle : application aux méthodes de classification automatique déployées pour DEFT08

Eric Charton[1] Nathalie Camelin[1] Rodrigo Acuna-Agost[1] Pierre Gotab[1]
Remi Lavalley[1] Remy Kessler[1] Silvia Fernandez[1]
(1) LIA - Université d'Avignon, BP1228 84911 Avignon cedex 09 France
{eric.charton, nathalie.camelin, rodrigo.acuna-agost, remy.kessler, silvia.fernandez}@univ-avignon.fr
{pierre.gotab, remi.lavalley}@etd.univ-avignon.fr

Résumé. Nous décrivons dans cet article un ensemble de méthodes de classification automatique mises en œuvre dans le cadre de la campagne DEFT08. Notre approche a d'abord consisté à évaluer les performances des systèmes état de l'art de l'apprentissage automatique (classifieurs SVM, AdaBoost, probabilistes et cosine). Nous avons ensuite cherché à améliorer les performances de ces classifieurs en élaborant une méthode de construction de classes discriminantes par analyse distributionnelle. Cette méthode de normalisation des classes, appliquée aux classifieurs SVM et probabilistes, a amélioré significativement nos résultats. Deux méthodes de fusion des divers résultats obtenus par les classifieurs ont également été testées.

Abstract. In this paper we describe a set of automatic classification methods applied to the DEFT08 campaign. First, we evaluated and compared some of state-of-the-art classifiers like SVM, AdaBoost, probabilistic-based classifiers, and cosine-based classifiers. Subsequently, we developed a method to normalize classes using a distributional analysis of the text with the aim of improving the performance. Lastly, some additional results were obtained by two merging methods that showed to increase the scores of the individual classifiers

Mots-clés : Méthodes de Classification Automatique, SVM, AdaBoost, Classifieur Bayésien Naïf, Analyse distributionnelle .

Keywords: Automatic Classification Methods, SVM, AdaBoost, Naïve Bayesian Classifier, Distributionnal analysis.

1 Introduction

La campagne DEFT 2008 (DÉfi de Fouilles de Texte) a pour sujet la classification en genre et en thème de textes. Au-delà de la reconnaissance de la catégorie du document, la reconnaissance de son genre est utile pour guider l'utilisation ultérieure qui peut en être faite (*e.g.* orientation de courriels, veille scientifique, ...). Mais reconnaître à la fois le thème et le genre d'un document relève-t-il d'une problématique particulière ?

Le défi propose deux tâches distinctes qui permettent de réfléchir à ce problème. L'une confronte classification en genre et en thème tandis que l'autre se focalise uniquement sur la classification en thème.

2 Analyse des corpus

2.1 Corpus relatif à la tâche 1 : Classification en genre et en thème

Ce corpus, noté *CORPUS1*, est étiqueté en thème selon quatre classes : économie (*ECO*), art (*ART*), télévision (*TEL*) et sport (*SPO*). Chaque document est également annoté selon qu'il provient du journal Le Monde (*LM*) ou de Wikipedia (*W*). La répartition des volumes de documents pour chaque classe dans le corpus d'apprentissage (*APP1*) et d'évaluation (*EVAL1*) est indiquée dans le tableau 1.

CORPUS1	*ECO*	*TEL*	*ART*	*SPO*	*LM*	*W*	Échantillons
APP1	30.41%	8.88%	37.88%	22.82%	57.97%	42.02%	15223
EVAL1	29.11%	12,75%	36.27%	21.84%	53.63%	46.36%	10596

TABLE 1 – Répartition des volumes de documents de *CORPUS1* pour chaque classe

On observe sur le corpus d'apprentissage de cette tâche, l'existence de classes dont la répartition est très disparate. Ce déséquilibre est conjugué à des imbrications inter-classes importantes [1]. L'imbrication est particulièrement marquée dans le cas des classes *TEL* et *ART*. La confusion entre ces deux classes est conjuguée à une faible représentation de *TEL* par rapport à *ART* (respectivement 8.88% et 37.88% des documents).

2.2 Corpus relatif à la tâche 2 : Classification en thème

Le corpus de la tâche 2, noté *CORPUS2*, est composé de documents également issus des deux sources Le Monde et Wikipédia mais est annoté uniquement en thèmes selon cinq classes : société (*SOC*), actualité ou information française (*FRA*), ou internationale (*INT*), sciences (*SCI*) et littérature (classe *LIV*). La répartition des classes dans *CORPUS2* est donnée dans le tableau 2 pour les corpus d'apprentissage (*APP2*) et d'évaluation (*EVAL2*).

CORPUS2	*SOC*	*FRA*	*INT*	*LIV*	*SCI*	Échantillons
APP2	16.04%	14.12%	22.52%	19.43%	27.87%	23550
EVAL2	16.03%	14.12%	22.53%	19,42%	27.87%	15693

TABLE 2 – Répartition des volumes de documents de *CORPUS2* pour chaque classe

Il apparaît que contrairement à *CORPUS1*, *CORPUS2* repose sur une répartition des classes plus homogène. On note, à nouveau, une légère sous-représentation d'une classe (*FRA*) associée à une sur-représentation d'une autre (*SCI*). Les trois classes *FRA*, *SCI* et *SOC* ont par ailleurs des intersections fortes, pouvant conduire à une faiblesse de rappel sur les classes *FRA* et *SOC* et un manque de précision très marqué sur *SOC*.

1. Les distances entre classes ont été mesurées par similarité cosine

3 Présentation théorique des différentes méthodes de classification automatique appliquées à DEFT08

Les Machines à supports vectoriels Les machines à supports vectoriels (Support Vector Machine - *SVM*) sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de discrimination et de régression (Vapnik, 1995). La première idée clé est la notion de marge maximale, *i.e* la distance entre la frontière de séparation et les échantillons les plus proches est maximisée. L'autre idée maîtresse des SVM est de transformer, grâce à une fonction noyau, l'espace de représentation des données d'entrées en un espace de plus grande dimension, dans lequel il est probable qu'il existe un séparateur linéaire. Les deux systèmes implémentés ont été développés à partir de l'outil LIBSVM (Fan et al., 2005).

Le boosting Le principe du boosting réside dans la combinaison de plusieurs hypothèses dites *faibles* afin d'améliorer la précision des règles de classification. Chaque hypothèse faible est obtenue par itérations successives de l'algorithme de boosting. À chacune des exécutions de l'algorithme, une nouvelle distribution de probabilité *a priori* sur les exemples d'apprentissage est calculée en fonction des résultats de l'algorithme à l'exécution précédente. Introduit par (Freund & Schapire, 1995), l'algorithme standard s'appelle AdaBoost (adaptive boosting). Deux implémentations particulières de l'algorithme Adaboost ont été utilisées : BoosTexter et icsiboost.

Classification Bayésienne Naïve Le classifieur Bayésien Naïf est un classifieur probabiliste simple appliquant le théorème de Bayes. La formule générique de ce classifieur est la suivante : la probabilité qu'un document D contenant des mots m appartienne à une classe k est égale à :

$$P(k|D) = \frac{p(D|k)\, p(k)}{p(D)} \tag{1}$$

En règle générale, les classifieurs Bayésiens Naïfs appliquent la méthode de l'estimateur de maximum de vraisemblance pour décider de l'attribution d'une classe, comme suit $P(k|D) = p(m_1, m_2, ..., m_n|k) = \prod_{i=1}^{n} p(m_i|k)$. Un des avantages du classifieur Bayésien Naïf réside dans sa capacité d'estimation des paramètres avec peu de données d'apprentissage.

Mesure de similarité Cosine La mesure de similarité par la formule Cosine permet de calculer le cosinus de l'angle entre un article et un vecteur représentant de chacune des classes (Salton & McGill, 1983). Plus celui-ci est élevé, plus l'article est proche de la classe correspondante. La formule classique a été adaptée à la tâche de classification en se basant sur le critère d'impureté de Gini $Gin(i) = \sum (P_k)^2$ afin d'en augmenter le pouvoir discriminant :

$$s(n,k) = \frac{\sum (W_{in} \times Wik \times Gin(i))}{\sqrt{\sum W_{ik}^2}} \tag{2}$$

où W_{ik} représente le poids du mot *i* dans le modèle d'apprentissage de la classe *k* et W_{in} le poids du mot *i* dans l'article *n*.

Énergie textuelle L'énergie textuelle correspond au modèle magnétique d'Ising déjà utilisé avec succès dans des tâches de traitement de la langue naturelle (Fernandez et al., 2008). Dans ce modèle, k sous-ensembles correspondant aux documents d'apprentissage pour chaque classe sont considérés comme k matériaux différents. Chaque matériau est composé de n atomes (des mots différents) qui interagissent différemment pour chaque classe. Ces interactions entre mots sont données par la matrice $J_k = \{j_{i,j}\}$ où chaque élément $j_{i,j} = \sum_k tf_i^k tf_j^k$ correspond à la co-occurrence de ces mots dans le sous-ensemble k. Les k matrices $J_1, J_2, ..., J_k$ seront utilisées pour classer les documents de test Dt. Pour chaque Dt, l'énergie des mots est calculée selon :

$$E_k = S \times J_k \times S^T \tag{3}$$

où S est la représentation vectorielle du document à classer. Plus la valeur de E_k est élevée, plus il est probable que ce document soit un échantillon du matériau k.

4 Implémentation des différents systèmes de classification

4.1 Protocole expérimental

Pour chaque tâche, la méthode de validation des classifieurs est une validation croisée classique sur N partitions du corpus d'apprentissage. Chacun des corpus d'apprentissage de *APP1* et *APP2* a été divisé en N parties égales qui constituent N sous-corpus. Nous utilisons $N-1$ sous-corpus en tant que données d'apprentissage et le Nième corpus est réservé au test. N jeux de tests *tournants* sont ainsi créés, le score final d'une classification étant ensuite calculé par la moyenne de tous les scores obtenus sur chacun des N corpus de test.

4.2 Méthodes de pré-traitements de texte

Les méthodes de classification (classifieurs) ont été appliquées avec des classes de mots issues des corpus d'apprentissage, au besoin normalisés selon plusieurs techniques dont l'efficacité est attestée par l'état de l'art (*e.g.* : étiquetage morpho-syntaxique, lemmatisation, filtrage des textes, n-grammes). Nous avons complété ces techniques par une méthode de filtrage des mots d'après leurs distributions dans les classes.

4.2.1 Pré-traitements classiques

Dans un premier temps, le texte à classer a été filtré par plusieurs opérations : suppression de toutes marques typographiques, minusculisation.

Dans un second temps, plusieurs représentations de la phrase ont été proposées :
– ensemble des mots ;
– représentation morpho-syntaxique : chaque mot de la phrase est représenté par sa forme morpho-syntaxique (Part Of Speech - *POS*) ;
– représentation canonique : chaque mot de la phrase est représenté par son lemme.

Ces deux dernières représentations différentes pour un mot sont obtenues grâce à l'utilitaire LIA-TAGG [2]. Chaque participant de l'équipe a fait ses propres choix quant à la représentation

2. http://www.lia.univ-avignon.fr/chercheurs/bechet/download_fred.html

du texte qui sont indiqués dans chacune des descriptions des systèmes.

Diverses techniques d'agglutination de groupes de mots par n-grammes ont également été proposées. Nous avons retenu pour la majorité des classifieurs des classes de mots composées de tri-grammes qui permettent une amélioration des performances.

Un antidico [3] contenant notamment : un ensemble de mots fonctionnels (*e.g.* : être, avoir, pouvoir, falloir) ; des expressions courantes (*e.g.* : c'est-à-dire, chacun de) ; des chiffres (numériques et/ou textuels) ; des symboles comme <$>, <#>, <*>, a également été utilisé dans certains systèmes pour filtrer le texte.

4.2.2 L'hypothèse distributionnelle et les distributions de Zipf-Mandelbrot

Nous introduisons dans cette tâche une méthode de normalisation de textes originale, exploitant les propriétés de la distribution des mots dans un corpus pour élaborer des classes de détection. Des auteurs ont déjà souligné les performances obtenues dans une tâche de classification de données textuelles, sur des classifieurs SVM associés à des classes de mots extraites d'après des distributions de Zipf-Mandelbrot (Leopold & Kindermann, 2002).

La loi de Zipf-Mandelbrot est une distribution de probabilité discrète. Elle est connue également sous le nom de loi de Pareto-Zipf (Zipf, 1949), la loi de Pareto étant la forme continue de la loi de Zipf. La loi de Zipf prédit que si dans un texte de longueur N on range les mots dans l'ordre de leur fréquence d'apparition, alors la fréquence $f(r)$ du mot de rang r est approximativement de forme : $f(r) = \frac{K}{r}$

Considérons des objets textuels O, mot ou un groupe de mots (n-grammes) représentant les occurrences rencontrées dans le corpus. Si nous postulons que les fréquences des O dans un corpus suivent approximativement des distributions de Zipf-Mandelbrot, nous pouvons établir une table de fréquences des O pour un corpus de textes tels que ceux présentés dans ce défi. Dans cette table, nous trouverons en premier lieu dans les plus hauts rangs, des mots outils (ou des objets O contenant des mots outils). Dans le cadre applicatif de DEFT08, les mots de plus haute fréquence dans *CORPUS1* et *CORPUS2* sont sans surprise les mots outils : *de, la, le et, des, les, en, du, un, est, dans* .... On observe à la suite de cette liste de mots outils, l'apparition de premiers mots plus caractéristiques d'une classe (*e.g.* sur *APP1* : "france" en 32 ème rang dans la classe *TEL*, "équipe" au rang 39 de *SPO*, "film" au 45 ème rang de *ART*).

Il est déduit de ces observations qu'il est possible, en exploitant plusieurs distributions de Zipf-Mandelbrot modélisant la distribution de mots de l'intégralité du corpus et celle de chaque sous corpus, de supprimer automatiquement les mots de probabilité proche (qui correspondent aux mots outils ou peu discriminants) et de ne conserver que les mots discriminants pour une classe donnée.

Cette méthode est schématisée par la figure 1. Sur la gauche, un exemple de mot outils dont la probabilité d'apparition dans chaque sous corpus est très proche de celle observée dans la totalité du corpus : ces mots sont supprimés dans toutes les classes de détection. Au centre, un exemple de mot discriminant ("équipe" de la classe SPO) dont la probabilité d'apparition est très supérieure à celle observée par la distribution de Zipf modélisée d'après l'ensemble du corpus. On choisira ici de conserver ce mot uniquement dans la classe de détection SPO. On utilisera pour procéder à la sélection un intervalle de confiance ou le critère d'impureté de Gini.

3. http://www.up.univ-mrs.fr/~veronis/data/antidico.txt

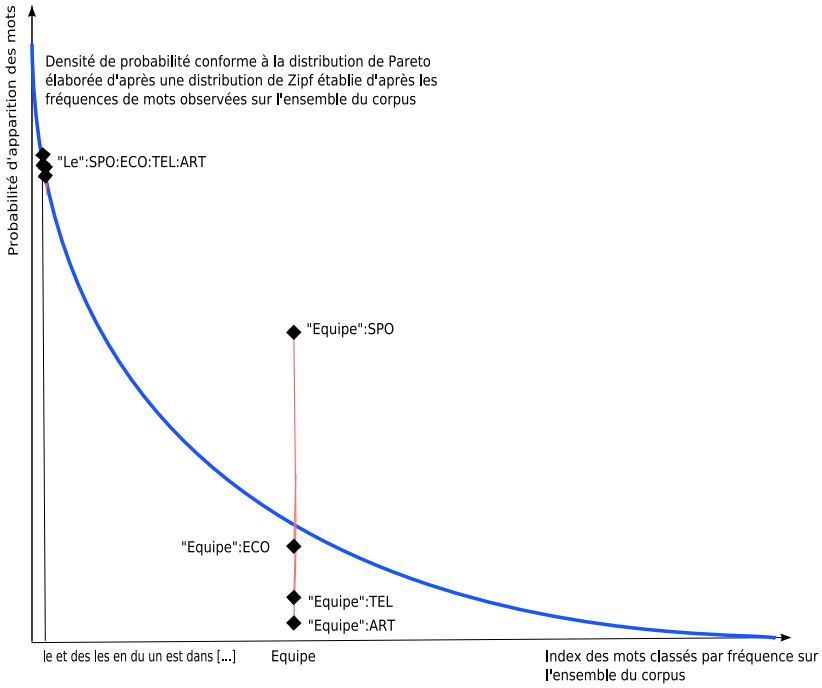

FIGURE 1 – Représentation schématique d'une distribution de Zipf et des variations des probabilités d'apparition des mots selon les classes

4.3 Présentation des systèmes

Sept systèmes ont été implémentés dans le cadre de ce défi. Nous les décrivons ci-dessous.

SVM_Baseline Le classifieur SVM_baseline est conçu comme suit : chaque document est transformé en vecteur de mots préalablement filtrés selon l'antidico (*c.f.* sous-section 4.2.1). Les verbes fléchis sont ramenés à leur racine, ainsi que les mots pluriels et/ou féminins au masculin singulier. Les vecteurs sont ensuite soumis au classifieur pour la phase d'apprentissage des SVM. Le noyau choisi a été le noyau linéaire, celui-ci s'étant avéré le plus performant, ceci s'expliquant par la taille importante de notre lexique.

SVM-Extended Ce système SVM utilise les distributions de mots préalablement filtrés. Le filtrage consiste à ne conserver que les mots sous une forme majuscule. En premier lieu, deux séries d'index des n-grammes et de leur fréquence sont générées : un premier index pour l'ensemble du corpus et une seconde série d'index correspondant à chaque classe. Dans un second temps, tous les n-grammes apparaissant dans plusieurs catégories avec une probabilité proche (l'intervalle de variabilité est réglable) sont supprimés des index. Cette phase permet de supprimer automatiquement les mots outils et non discriminants. Selon le même procédé, ne sont conservés dans les index de chaque catégorie que les n-grammes caractéristiques de cette catégorie. Il est ensuite procédé à la construction des vecteurs caractéristiques de chaque classe. Dans le cadre applicatif décrit ici, les mots d'occurrence 1 ont été supprimés, ces derniers augmentant considérablement le temps de calcul du SVM, sans améliorer notablement ses performances.

Naïve_Bayes_extended Le classifieur Bayésien Naïf utilise k classes de n-grammes, correspondant aux k thèmes. Le contenu des classes est normalisé selon ce qui a été décrit dans la sous-section 4.2.2. Les n-grammes dont la probabilité est identique dans les k classes sont supprimés (mots outils ou non discriminants). Un intervalle de variabilité α est utilisé pour choisir les n-grammes considérés comme discriminants pour une classe donnée. Le n-gramme qui sort de l'intervalle de variabilité est versé dans la classe de détection où il est le plus prégnant et supprimé dans les autres classes. Aucune limite sur le nombre d'occurrences d'un n-gramme justifiant de le maintenir dans une classe de détection n'a été définie. Les classes sont donc de grande taille. Les documents sont ensuite attribués à une classe par un test de maximum de vraisemblance entre les n-grammes contenus dans un document et ceux contenus dans les k classes.

BoosTexter Dans le cadre de cette implémentation de *BoosTexter*[4], le classifieur faible est un arbre de décision binaire à un niveau qui teste la présence/l'absence d'un n-gramme dans la phrase et en déduit sa classification, l'hypothèse faible. Les éléments choisis par les classifieurs faibles sont des n-grammes avec $1 \leq n \leq N$ et $N = 3$ déterminé par l'utilisateur. Le nombre de tours d'itération de l'algorithme est de $T = 1500$, empiriquement un bon compromis entre performances et temps de calcul. La phrase est représentée par sa suite de lemmes (*c.f.* sous-section 4.2.1) qui permet une légère augmentation des performances par rapport à l'utilisation des mots.

icsiboost *icsiboost* est une version open-source de BoosTexter développée par le laboratoire ICSI[5]. Le paramètre $N = 3$ a été choisi. Concernant le paramètre T définissant le nombre de tours de l'algorithme, les expériences ont montré qu'un optimum était atteint aux alentours des 2500 tours. Les composants choisis pour représenter la phrase sont à la fois les lemmes et les POS (*c.f.* sous-section 4.2.1). L'utilisation de cette représentation par rapport à une représentation en mots permet une augmentation des performances.

Cosine_Discriminant L'équation 2 a été implémentée. Les unigrammes de lemmes sont considérés mais sans minusculisation préalable. De plus, plusieurs filtres sont appliqués : les signes de ponctuations et les déterminants sont ôtés ainsi que l'ensemble des mots de l'antidico (*c.f.* sous-section 4.2.1).

Enertex Enertex est le classifieur basé sur le concept d'énergie textuelle présenté dans la section 3. Il a été appliqué sur le texte après le même pré-traitement que celui indiqué dans le système SVM-Extended.

4. `http://www.research.att.com/sw/download/`
5. `http://www.icsi.berkeley.edu/`

Fscores APP	tâche1-catégorie	tâche1-genre	tâche2-catégorie
SVM_baseline	0.8391	0.93	0.78
SVM_extended	**0.9150**	0.9594	**0.8445**
N_Bayes_extended	0.8629	0.9353	0.8271
BoosTexter	0.8958	**0.9869**	0.8316
icsiboost	0.9051	**0.9858**	**0.8409**
Cosine_Discriminant	0.8508	0.9222	0.8244
Enertex	0.8328	0.8390	0.7561

TABLE 5 – Performances des différents systèmes lors de la phase d'apprentissage

5 Expériences et résultats

5.1 Phase d'apprentissage

L'ensemble des Fscores obtenus par chaque système sur les trois tâches, lors de la validation croisée de la phase d'apprentissage, est présenté dans le tableau 5.

5.2 Mode évaluation

Exécution 1 : Fusion par vote ternaire Les résultats par fusion par vote ternaire (TAB.6) sont les meilleurs obtenus sur les trois exécutions. On observe un maintien des performances sur tâche1-genre et tâche2-catégorie et une baisse de performance assez marquée sur la tâche1-catégorie, explicable par la différence de répartition des classes entre les corpus APP1 et EVAL1.

Exécution 1	Précision	Rappel	Fscore
tâche1-genre	0.9795	0.9800	0.9798
tâche1-catégorie	0.9082	0.8448	0.8754
tâche2-catégorie	0.8814	0.8759	0.8786

TABLE 6 – Évaluation : Performances de l'exécution1

Exécution 2 : Fusion probabiliste Cette soumission a étrangement obtenu une mauvaise performance sur la tâche2-catégorie (TAB.7). Nous réaliserons prochainement un dépouillement de ces résultats pour en éclaircir les causes.

Exécution 2	Précision	Rappel	Fscore
tâche1-genre	0.9584	0.9600	0.9592
tâche1-catégorie	0.7919	0.8267	0.8089
tâche2-catégorie	0.8124	0.558393	0.6618

TABLE 7 – Évaluation : Performances de l'exécution2

Exécution 3	Précision	Rappel	Fscore
tâche1-genre	0.9795	0.9800	0.9798
tâche1-catégorie	0.8972	0.8006	0.8442
tâche2-catégorie	0.8553	0.8497	0.8525

TABLE 8 – Évaluation : Performances de l'exécution3

Exécution 3 : les meilleurs systèmes La tâche1-catégorie est réalisée par SVM_Extended, les deux autres tâches tâche1-genre et tâche2-catégorie sont réalisées avec icsiboost. Les résultats obtenus (TAB.8) confirment les bonnes performances des classifieurs SVM_Extended et icsiboost obtenues lors de la phase d'apprentissage. On note néanmoins une baisse de performances sur tâche1-catégorie.

6 Conclusions et perspectives

Nous avons appliqué plusieurs méthodes diversifiées de classification automatique à la problématique posée par DEFT08. Ces méthodes, prises séparément, présentent des performances proches. Nous avons enrichi ces méthodes par une fusion de leurs résultats. La plus performante étant celle par vote ternaire. Notre système complet est simple à régler, performant et exploite une méthode de préparation de classe originale à base d'analyse distributionnelle. Notre postulat de départ consistait à envisager la classification en genre et thème comme deux tâches indépendantes. Cette idée initiale s'est trouvée confortée par les résultats obtenus lors des expériences. Nous déduisons de ces résultats que les grandes différences de forme et de méthode de rédaction, utilisées pour le journal Le Monde et l'encyclopédie Wikipédia, sont suffisantes pour entraîner des classifieurs *état de l'art* et donner aux classes qu'ils utilisent un caractère suffisamment discriminant. Nous considérons que les méthodes présentées ici possèdent encore un bon potentiel d'amélioration que nous allons nous attacher à étudier dans les mois qui vont suivre.

Références

FAN R.-E., CHEN P.-H. & LIN C.-J. (2005). Towards a Hybrid Abstract Generation System, Working set selection using the second order information for training SVM. In NIPS 2005.

FERNANDEZ S., SANJUAN E. & TORRES-MORENO J. M. (2008). Enertex : un système basé sur l'énergie textuelle. In TALN2008.

FREUND Y. & SCHAPIRE R. E. (1995). A Desicion-Theoretic Generalization of On-Line Learning and an Application to Boosting. In EuroCOLT : Springer.

LEOPOLD E. & KINDERMANN J. (2002). Text categorization with support vector machines. how to represent texts in input space. Machine Learning.

SALTON G. & MCGILL M. (1983). Introduction to modern information retrieval. Computer Science Series McGraw Hill Publishing Company.

VAPNIK V. N. (1995). The nature of statistical learning theory. Springer-Verlag New York, Inc.

ZIPF G. K. (1949). Human behavior and the principle of least-effort. Addison-Wesley.