Quand simplicité rime avec efficacité : analyse d'un catégoriseur de textes

Romain Vinot — François Yvon

ENST, Ecole Nationale Supérieure des Télécommunications
46, rue Barrault - 75634 Paris Cedex
{romain.vinot,francois.yvon}@enst.fr

RÉSUMÉ. *Les mesures de performances habituellement utilisées pour les outils de catégorisation de textes sont insuffisantes pour comprendre l'adéquation entre algorithmes et types de corpus. Cet article apporte des éléments de comparaison complémentaires à propos d'un séparateur linéaire très simple originellement proposé par J. Rocchio. Nous montrons que ce classifieur est particulièrement robuste au bruit et que la présence de sous-classes naturelles dans les données n'affecte que marginalement sa capacité de généralisation, deux propriétés importantes pour les applications réelles.*

ABSTRACT. *Evaluation mesures used for text categorization algorithms are too coarse to help understand why some algorithms are better suited (ie. achieve better performance) for some tasks or corpora than others. This paper presents a detailled analysis of the behavior of a simple linear text classifier originally proposed by J. Rocchio. Our experiments show that this classifier is particularly robust to noise and to the existence of "natural" sub-classes in the data; these are two important features for real applications.*

MOTS-CLÉS : *catégorisation de textes, Rocchio, k plus proches voisins, SVM*

KEYWORDS: *text categorization, Rocchio, k nearest neighbors, SVM*

1. Introduction

Les travaux autour de la catégorisation de textes constituent un sous-domaine actif du traitement automatique de l'information. La catégorisation de textes se prête à de multiples applications pratiques [Seb99], parmi lesquelles le *filtrage*, consistant à déterminer si un document est pertinent ou non (décision binaire), et le *routage*, consistant à affecter un document à une catégorie parmi n. S'il existe de multiples algorithmes permettant d'inférer, à partir de données étiquetées, des outils de catégorisation, le choix opérationnel du classifieur à utiliser pour traiter un type un corpus donné, spécifique à une application, reste une question difficile.

Les évaluations de classifieurs de textes ([Yan97, RS01]) ne fournissent, en effet, que des réponses partielles, car elles agrègent les performances sur des corpus standards (Reuters, bases utilisées dans TREC...), ne permettant d'apprécier que des comportements moyens. Par ailleurs, les catégories de documents que ces corpus distingent étant fondées sur des oppositions thématiques, elles sont bien apprises par des classifieurs opérant sur les profils lexicaux. Ceci est moins vrai pour des applications émergentes telles que SwiftFile (classement de courrier électronique) [SK00], Topic Detection and Tracking (TDT) [TDT01], détection de sentiments [PLV02] ou Mail Center [BSA00, VY01] (réponse automatique à des requêtes de clients), pour lesquelles la définition des classes implique des propriétés extérieures au document[1], rendant ces distinctions plus difficiles à apprendre. Enfin, certaines applications exigent une précision que ne peuvent offrir des systèmes automatiques, la décision finale engageant toujours un opérateur humain (cas du Mail Center). Dans ce contexte, la qualité d'un système ne dépend pas tant de la précision de la classe proposée que de sa capacité à restreindre les choix de l'opérateur à un petit nombre de classes.

Les mesures de performances sur les corpus standards sont donc insuffisantes pour apprécier le potentiel des divers classifieurs. Elles sont aussi parfois contradictoires, sans que les raisons de ces contradictions soient clairement identifiées. Nous proposons, dans cet article, d'apporter des éléments de comparaison complémentaires en étudiant qualitativement le comportement d'un classifieur linéaire très simple [Roc71], qui, suivant les articles, est crédité de performances très médiocres ou bien tout à fait compétitives. Notre objectif est ici double : (i) analyser les performances réelles de ce classifieur sur plusieurs tâches de routage, en le comparant avec deux classifieurs de l'état de l'art et (ii) identifier, dans les corpus de textes, des caractéristiques qui permettent d'expliquer le succès de ce classifieur. Cet article est organisé comme suit : nous présentons d'abord, dans la section 2, l'algorithme Rocchio, ainsi que diverses extensions récentes. La section 3 décrit une série d'expériences visant à évaluer l'influence sur les performances de deux facteurs liés au corpus : le bruit engendré par des exemples mal étiquetés, ou la complexité de la surface de séparation des classes, caractéristiques susceptibles d'apparaître dans les applications opérationnelles de la catégorisation de textes. La section 4 tire les enseignements de ces expériences.

2. L'algorithme Rocchio

Rocchio est un classifieur proposé dans [Roc71] pour améliorer les systèmes de recherche documentaires. Ce classifieur s'appuie sur une représentation vectorielle [SWY75] des documents : chaque texte d est représenté par un vecteur $[d]$ de R^n, chaque coordonnée d_w se déduisant de la fréquence $Occ(w,d)$ du terme w dans d, par :

$$d_w = TFIDF(w,d) = log(1 + Occ(w,d)) * log(\frac{N}{N(w)})$$

1. Par exemple dans le Mail Center pour une question portant sur l'état d'une commande, la réponse va dépendre de l'information concernant cette commande.

avec N le nombre de documents du corpus et $N(w)$ le nombre de documents dans lequel w apparaît au moins une fois. Un terme se voit donc attribuer un poids d'autant plus fort qu'il apparaît souvent dans le document et rarement dans le corpus complet. Chaque vecteur $[d]$ est ensuite normalisé[2] en $[\underline{d}]$ afin de ne pas favoriser les documents les plus longs. Un profil prototypique $[c]$ s'en déduit, pour chaque classe c, selon :

$$c_w = \frac{t}{N_c} \sum_{d \in c} \underline{d_w} - \frac{1-t}{N_{\overline{c}}} \sum_{d \notin c} \underline{d_w} \qquad [1]$$

avec N_c le nombre de documents dans c, $N_{\overline{c}}$ le nombre de documents n'appartenant pas à C, t un paramètre du modèle compris entre 0 et 1. Ces profils correspondent donc au barycentre des exemples (avec un coefficient positif pour les exemples de la classe et négatif pour les autres). Ces vecteurs sont également normalisés. Le classement de nouveaux documents s'opère en calculant la distance euclidienne entre la représentation vectorielle du document et celle de chacune des classes ; le document est assigné à la classe la plus proche.

Diverses améliorations récentes de ce modèle se sont avérées fructueuses : de nouvelles méthodes de calcul de $[d]$; un choix plus raisonné des exemples négatifs intervenant dans l'équation (1) (*Query Zoning* et feedback dynamique [SMB97, BS95]) ont ainsi permis d'améliorer sensiblement les performances de cet algorithme, le hissant, dans certaines conditions expérimentales, au niveau des meilleurs algorithmes.

Dans le contexte de cette discussion, Rocchio présente deux caractéristiques importantes :

– il implémente une règle de décision dessinant des séparations linéaires (hyperplans) dans l'espace de représentation des textes. Ceci lui confère une expressivité identique à celle d'un perceptron ; comme le montre [LSCP96] sur des tâches de filtrage, les performances de Rocchio avec feedback dynamique sont comparables à celles d'un réseau de neurone entraîné par descente de gradient (Widrow-Hoff). Rocchio devrait donc être peu adapté aux tâches pour lesquelles la séparation des classes n'est pas linéaire ;

– chaque exemple contribue identiquement à la construction du centroïde de sa classe. Rocchio s'oppose ainsi d'une part aux algorithmes "dirigés par les erreurs" (réseaux de neurones, SVM (Support Vector Machine [Vap95, Joa98])), qui donnent plus d'importance aux exemples mal classés ; d'autre part aux algorithmes "locaux", qui n'utilisent qu'une faible partie des exemples à chaque classification (par exemple les k-PPV (k plus proches voisins)) ; deux stratégies induisant une plus grande sensibilité au bruit. Dans les applications réelles, ce bruit peut résulter d'assignations d'étiquettes de classes incohérentes, erronnées, ou non directement corrélées aux profils lexicaux ; ou bien encore de bruit dans les documents eux-mêmes, fautes d'orthographe ou de syntaxe par exemple lorsque les textes sont des courriers électroniques.

2. Chaque coordonnée d_w est divisé par la norme euclidienne du vecteur : $\underline{d_w} = \frac{d_w}{\sqrt{\sum_w d_w^2}}$

Algorithme	Newsgroups		Mail Center	
	Perf1	**Perf5**	**Perf1**	**Perf5**
Rocchio	0.81	0.98	0.50	0.88
Rocchio QZ	0.81	0.98	0.53	0.87
1-ppv	0.81	0.97	0.43	0.52
50-ppv	0.85	0.98	0.54	0.84
200-ppv	0.83	0.98	0.51	0.85
SVM	0.87	0.97	0.58	0.81

Newsgroups et Mail Center

Algorithme	Reuters	
	"Scut"	**Perf1**
Naïve Bayes	0.71	0.72
C4.5	0.79	0.79
Rocchio	0.75	0.80
kPPV	0.85	0.82
SVM	NA	0.86

Reuters (d'après [Yan97] et [Joa98])

Tableau 1: Résultats avec la meilleure performance de chaque algorithme

Nous nous proposons de vérifier ces deux hypothèses (résistance au bruit, trop faible expressivité) sur diverses tâches de routage dans la section suivante.

3. Expérimentations

La base *Newsgroups* comprend 20000 messages, répartis de façon uniforme dans 20 newsgroups. La liste des groupes est indiqué dans le tableau 2. Nous avons utilisé 2/3 de la base pour l'apprentissage, 1/3 pour les tests. Le deuxième corpus concerne l'application du Mail Center. Il est constitué de 2393 courriers électroniques reçus par le service clientèle d'une entreprise classés dans 40 catégories (voir [VY01]). Etant donné le faible nombre de documents, les tests ont été réalisés par validation croisée avec corpus divisé en 10 groupes. Les mesures de performances utilisés pour le routage sont : *Perf1*, le pourcentage d'exemples bien classés ; *Perf5*, le pourcentage d'exemples dont la classe correcte est l'une des cinq premières propositions du classifieur. Nous avons utilisé le programme SVM-light[Joa99] pour tester l'algorithme SVM et réimplémenté les autres.

3.1. *Evaluation sur les corpus Newsgroups et Mail Center*

Nous avons effectué des tests pour la tâche de routage sur ces deux corpus. La Table 1 résume ces expériences et les compare à d'autres résultats obtenus en filtrage sur la base Reuters. On peut mettre en évidence que :

– Nos tests recoupent les évaluations de [Yan97, Joa98] sur le corpus Reuters avec Perf1 : Rocchio est globalement moins performant que k-PPV, lui-même moins performant que SVM.

– Rocchio est systématiquement inférieur aux autres algorithmes avec Perf1, et supérieur avec Perf5. Avec les k-PPV, il est impossible de choisir k pour optimiser à la fois Perf1 et Perf5. L'algorithme SVM étant optimisé pour les classifications binaires, il traite (dans l'implémentation utilisée) les problèmes multi-classes de manière ad-hoc ; il semble avoir encore plus de difficulté avec la mesure Perf5.

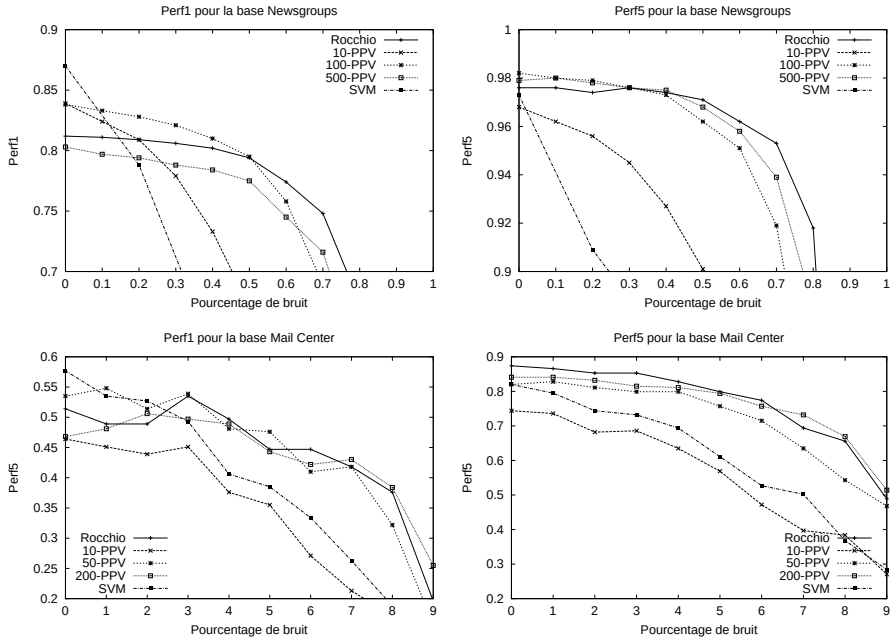

Figure 1: Performance suivant la quantité de bruit du corpus.

– L'algorithme de *Query Zoning* est moins utile en routage qu'en filtrage. Les évaluations effectués dans [SMB97] en filtrage présentent une amélioration de 9 à 12 % par rapport à la version "de base" de Rocchio. Dans nos tests, les améliorations sont plus faibles (nulles pour Newsgroups et 6% pour Mail Center). Une explication possible est que Query Zoning vise à améliorer la qualité du profil de chaque classe en focalisant sur les exemples importants. En routage, ces exemples sont déjà pris en compte par les profils des autres classes et n'apportent en fait que peu d'informations nouvelles pour la discrimination.

3.2. *Effet du bruit*

Pour valider l'hypothèse que Rocchio est peu sensible au bruit, nous avons réalisé des expériences en bruitant peu à peu les étiquettes des classes : on change aléatoirement l'étiquette d'une certaine partie des exemples. Les résultats sont synthétisés dans la Figure 1.

Avec le corpus Newsgroups, les performances de SVM se dégradent très rapidement avec le bruit. Comme on peut s'y attendre, les performances de k-PPV sont de moins en moins sensible au bruit lorsque l'on augmente k. Mais au delà de $k = 50$, les performances commencent à décroitre pour le corpus non bruité. Il n'y a donc pas

de valeur optimale pour k quelque soit le niveau de bruit du corpus. Rocchio, quant à lui, est exceptionnellement peu sensible au bruit et même avec 50% des exemples bruités, ses performances sont presque inchangés.

Avec le corpus Mail Center, les performances de tous les algorithmes restent inchangées jusqu'à 40% de bruit, puis le même comportement qu'avec la base Newsgroups apparaît. Ceci suggère que la base Mail Center contient un important bruit intrinsèque sur les étiquettes de classe : de faibles perturbations aléatoires des étiquettes n'affectent pas les performances, qui toutefois finissent par se dégrader au fur et à mesure que des perturbations plus fortes sont appliquées.

3.3. *Effet des sous-classes*

Cette troisième expérience vise à évaluer le comportement de Rocchio dans la configuration où les classes à discriminer sont dotées d'une structure interne, par exemple lorsqu'il existe différents sous-groupes thématiquement homogènes au sein d'une même classe. Cette situation se recontre dans les corpus réels : ainsi pour une tâche de filtrage de courrier électronique, les catégories "courrier valide" et "courrier non-sollicité" recoupent en fait des sous-groupes thématiquement très disparates.

Dans une telle configuration, nous aimerions tester l'intérêt d'intégrer explicitement dans un classifieur de type Rochhio ces sous-catégories, sachant k-PPV et SVM sélectionnant un nombre restreint d'exemples du corpus, ils utilisent donc cette séparation implicitement dans leur choix des exemples. Rocchio est par contre incapable de saisir cette information. Dans ce but, nous avons constitué, à partir du corpus Newsgroup, deux nouvelles bases de documents, qui correspondent à deux manières différentes de fusionner les classes d'origine.

Les vingt groupes utilisés par le corpus Newsgroups et présentés dans le Tableau 2 se placent naturellement dans la hiérarchie de Usenet. Dans une première expérience, nous avons créé un nouveau corpus avec quatre super-classes constituées des newsgroups présentant le même préfixe Usenet[3]. Nous avons aussi créé un deuxième corpus avec quatre super-classes constitués de newsgroups les plus hétérogènes possibles.

Dans ces nouvelles expériences, la tâche consiste à discriminer les quatre superclasses (sauf pour la "base normale" où il s'agit toujours de catégoriser les vingt newsgroups). Nous nous intéressons essentiellement à mesurer l'écart relatif de Rocchio par rapport aux autres algorithmes pour diverses configurations des corpus. Ces résultats sont résumés dans la Table 3.

Pour la base 1, les performances de Rocchio sont équivalentes à celles de la base de départ (-4% par rapport à 30-PPV et -7% par rapport à SVM dans les deux cas). Avec la base 2, Rocchio obtient des performances médiocres (on passe de -4% à -22% pour 30-PPV et de -7% à -25% pour SVM). Ceci s'explique par le fait que dans le cas de la base 1, s'il existe bien des sous-classes plus homogènes que la classe

3. Nous avons placé les newsgroups *soc.religion.christian*, *alt.atheism* et *misc.forsale* dans les super-classes les plus proches pour avoir des super-classes de taille homogène.

1. comp.graphics	11. alt.atheism
2. comp.windows.x	12. sci.electronics
3. comp-os.ms-windows.misc	13. sci.crypt
4. comp.sys.mac.hardware	14. sci.space
5. comp.sys.ibm.pc.hardware	15. sci.med
6. talk.politics.guns	16. misc.forsale
7. talk.politics.mideast	17. rec.sport.baseball
8. talk.politics.misc	18. rec.sport.hockey
9. talk.religion.misc	19. rec.autos
10. soc.religion.christian	20. rec.motorcycles

Base 1	**Base 2**
classes homogènes	classes hétérogènes
C1 : 1,2,3,4,5	C1 : 1,2,6,12,16
C2 : 6,7,8,9,10,11	C2 : 3,7,8,13,17
C3 : 12,13,14,15	C3 : 4,9,10,14,18
C4 : 16,17,18,19,20	C4 : 5,11,15,19,20

Tableau 2: Liste des groupes du corpus Newsgroups et les fusions effectuées

Algorithme	**Base normale**	**Base 1**	**Base 2**	**Base Spam**
Rocchio	0.814	0.895	0.710	0.778
1-PPV	+0%	+1.8%	+17.7%	-27.9%
30-PPV	+4.3%	+4.2%	+21.8%	+19.6%
SVM	+6.8%	+7.2%	+25.4%	+25.2%
Rocc (20 gp)	-	+4.2%	+19.5%	-
Rocc (clust)	+0%	+2.5%	+17.3%	+24.3%

Tableau 3: Tests sur le corpus Newsgroups avec sous-classes
Les résultats sont donnés en variation relative de la mesure Perf1 par rapport à Rocchio

de base, mais toutes ces sous-classes sont dans la même zone de l'espace relativement à l'écart des autres, et les performances de Rocchio ne sont donc pas affectées. Dans le cas de la base 2, les sous-classes sont réparties dans tout l'espace et rapprochées de sous-classes d'autres classes et compliquent donc l'apprentissage.

Pour compléter ces résultats, nous avons réalisé deux nouvelles expériences. Dans la première, nous avons entraîné Rocchio avec les 20 classes mais testé sur les quatre super-classes (lors de la classification, la super-classe choisie est celle qui contient le newsgroup le plus proche du document). Dans la deuxième expérience, nous avons utilisé les k-means, un algorithme de classification non supervisée, pour créer des

clusters à l'intérieur de chaque super-classe (méthode inspirée de [dKMK96]). On entraine ensuite Rocchio sur l'ensemble des clusters et on teste sur les super-classes par la même méthode que précédemment. Les résultats sont indiqués sur les lignes *Rocc (20 gp)* et *Rocc (clust)*. Dans les deux cas, les résultats sont conformes aux expériences précédentes. Ceci confirme que pour que Rocchio soit performant, il faut que le corpus ne contienne pas de sous-classes imbriquées. En revanche, l'existence de sous-classes homogènes mais peu mélangées ne gène pas l'apprentissage de Rocchio.

Dans la mesure enfin où le clustering améliore sensiblement les performances obtenues avec les bases 1 et 2, nous avons utilisé la même méthode avec la base d'origine. Les performances sont inchangées par rapport au Rocchio "de base", suggérant que les classes originales sont thématiquement trop homogènes pour que cette méthode apporte ici une amélioration. L'utilisation d'un corpus différent, contenant des e-mails qu'il s'agit de catégoriser entre les mails non sollicités (communément appelés "Spam") et les mails légitimes, nous a toutefois permis de mieux mettre en évidence l'intérêt de cette approche. Ce corpus contient 2193 messages dont 1460 spams. Les messages en anglais et en français sont mélangés dans les deux classes. Sur ce corpus, Rocchio est beaucoup moins performant que k-PPV ou SVM. En revanche, si on utilise le clustering pour créer de manière non-supervisée des sous-classes, il redevient tout à fait compétitif et surpasse les k-PPV en termes de performances. Ceci montre qu'il existe bien des corpus pour lesquels les catégories à discriminer sont structurées de manière interne en sous groupes cohérents qu'il est rentable d'identifier. Du point de vue de l'apprentissage, celà revient à introduire une couche de traitement permettant d'adapter automatiquement le nombre de paramètres utilisés par le classifieur (le nombre de prototypes) à la complexité des données à discriminer.

4. Conclusion

Cet article a permis d'éclairer les résultats contradictoires concernant les performances de l'algorithme Rocchio en précisant sur quels types de tâche et qur quels types de corpus il est plus ou moins adapté. Il a été montré que Rocchio est particulièrement performant sur les corpus très bruités et sur les tâches de routage où l'algorithme peut proposer plusieurs classes. Sa simplicité, et la faiblesse d'expressivité qui en découle, ne semble pas nuire aux performances même en présence de sous-classes thématiquement homogènes, sauf si ces thèmes sont trop éparpillés dans les différentes classes.

Nous avons également pu mettre en évidence que face à une telle configuration, il était possible d'améliorer les performances de l'algorithme Rocchio par utilisation préalable d'une classification non supervisée. Nous travaillons actuellement à la mise au point d'un algorithme de clustering spécifique au problème de détection de sous-thèmes éparpillés dans plusieurs classes.

5. Bibliographie

[BS95] Chris Buckley and Gerard Salton. Optimization of relevance weights. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference of Research and Develop-*

ment in Information Retrieval, pages 351–357, 1995.

[BSA00] Stephan Busemann, Sven Schmeier, and Roman G. Arens. Message classification in the call center. In *Proc. 6th Conference on Applied Natural Language Processing*, pages 159–165, Seattle, WA, 2000.

[dKMK96] H. de Kroon, T. Mitchell, and E. Kerckhoffs. Improving learning accuracy in information filtering. In *International Conference on Machine Learning - Workshop on Machine Learning Meets HCI*, 1996.

[Joa98] Thorsten Joachims. Text categorization with support vector machines : Learning with many relevant features. In *ECML-98, Tenth European Conference on Machine Learning*, pages 137–142, 1998.

[Joa99] Thorsten Joachims. *Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning*. B. Schölkopf and C. Burges and A. Smola, MIT-Press, 1999.

[LSCP96] David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *Proceedings of SIGIR-96*, pages 298–306, Zürich, CH, 1996.

[PLV02] Bo Pang, Lillian Lee, and Shivakumar Vaithyannathan. Thumbs up ? sentiment classification using Machine Learning techniques. In *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2002*, 2002.

[Roc71] Joseph John Rocchio. *The SMART Retrieval System : Experiments in Automatic Document Processing*, chapter 14, Relevance Feedback in Information Retrieval, pages 313–323. Gerard Salton (editor), Prentice-Hall Inc., New Jersey, 1971.

[RS01] Stephen Robertson and Ian Soboroff. The trec2001 filtering track report. In *Tenth Text REtrieval Conference*, Gaithersburg, Maryland, 2001. NIST, DARPA.

[Seb99] Fabrizio Sebastiani. Machine learning in automated text categorisation. Technical report, Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, 1999.

[SK00] Richard B. Segal and Jeffrey O. Kephart. Incremental learning in swiftFile. In *Proc. 17th International Conf. on Machine Learning*, pages 863–870. Morgan Kaufmann, San Francisco, CA, 2000.

[SMB97] Amit Singhal, Mandar Mitra, and Christopher Buckley. Learning routing queries in a query zone. In *Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval*, pages 25–32, Philadelphia, US, 1997.

[SWY75] Gerard Salton, A. Wong, and C.S. Yang. A vector space model for information retrieval. *Communications of the ACM*, 18(11) :613–620, November 1975.

[TDT01] The Topic Detection and Tracking 2001 (tdt-2001) evaluation project, 2001.

[Vap95] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[VY01] Romain Vinot and François Yvon. Semi-automatic response in a Mail Center. In *ASMDA 2001, 10th International Symposium on Applied Stochastic Models and Data Analysis*, pages 992–997. Université de Technologie de Compiègne, 2001.

[Yan97] Yiming Yang. An evaluation of statistical approach to text categorization. Technical Report CMU-CS-97-127, Carnegie Mellon University, 1997.