Classifieur probabiliste avec Support Vector Machine (SVM) et Okapi

Anh-Phuc TRINH (1), David BUFFONI (2), Patrick Gallinari (3)

(1) (2) (3) Laboratoire d'Informatique de Paris 6
Anh-Phuc.Trinh@lip6.fr, David.Buffoni@lip6.fr,
Patrick.Gallinari@lip6.fr
(Site Passy Kennedy, 104 av du Président Kennedy, 75016 Paris)

Résumé Ce papier présente le travail réalisé par l'équipe des jeunes chercheurs du LIP6 pour le 4ème DÉfi Fouille de Textes (DEFT'08). Cette année, le défi était de classifier les documents de 2 corpus différents en prenant en compte les variations en genre et en thème. Cet article présente un modèle de classification automatique sous la forme de SVMs estimant les probabilités *a posteriori* des classes pour chaque document.

Abstract This paper describes the participation of LIP6 at DEFT'08. We shall present a method based on SVM to realize this task. We propose a method which estimates posterior probabilities for each document.

Mots-clés : classification automatique de texte, apprentissage machine probabiliste, machine à vecteurs support (SVM), recherche d'information
Keywords: text classification, SVM, information retrieval, probability estimation

1 Introduction

Le défi DEFT 08 [DEF 08] consiste cette année, à classifier thématiquement un ensemble de documents provenant de deux corpus de genre différents (encyclopédique et journalistique). De cette manière, on cherche à trouver les améliorations possibles d'un système de classification thématique en prenant en compte le genre. Ceci a amené, les organisateurs à décomposer ce défi en deux tâches distinctes. La première consiste, à l'aide d'un classifieur automatique, à reconnaître le genre et la catégorie thématique de chaque document appartenant à l'un des deux corpus précédents. La deuxième quant à elle, à pour but de trouver la catégorie thématique de chaque document, indépendamment du corpus.

Nous avons mis en œuvre une méthode statistique essayant de résoudre le plus efficacement possible ces deux tâches. Pour cela, nous nous sommes inspirés du modèle Okapi, modèle de référence en Recherche d'Information, pour représenter les textes sous la forme de vecteurs de scores creux.
Ensuite, nous proposons une nouvelle méthode permettant d'estimer les sorties des classifieurs SVM sous la forme de probabilités, basée sur la maximisation de la log-vraisemblance conditionnelle.

Nous exposerons, tout d'abord, les prétraitements effectués sur les corpus (section 2), ce qui nous mènera, ensuite, à la présentation du modèle utilisé (section 3). Par la suite, nous ferons la synthèse des expériences réalisées avec leurs résultats, (section 4) ce qui nous permettra, enfin, de conclure notre travail.

2 Prétraitement

Pour mettre en œuvre notre modèle, nous avons dû faire certaines hypothèses. Tout d'abord, pour chaque tâche, nous considérons l'ensemble des documents comme des sacs de mots. Les tailles de ces deux ensembles sont données dans le tableau 1.

Tâche	Taille de sac de mots
1	167545
2	219117

Tableau 1 : Taille des sacs de mots par tâche

On peut, à partir de ces statistiques, émettre deux hypothèses distinctes afin d'appliquer notre modèle. La première, la moins restrictive, serait de prendre en compte tous les mots lors de la phase d'apprentissage. Quant à la seconde, afin de réduire les tailles des sacs de mots, nous avons décidé de retirer les mots les plus fréquents. Pour ce faire, nous avons téléchargé une liste de mots les plus fréquents de la langue française, disponible sur site [EDU].

Codages textuels

Nous transformons, à présent, chaque sac de mots en un espace vectoriel, avec en indice les mots associés à un score. Nous présentons ci-après, différentes façons de calculer ces scores :

- Binaire : le vecteur associé au sac de mots est sous la forme binaire ce qui correspond à l'apparition (score = 1) ou à l'absence (score = 0) du mot dans le texte. Cette vision, certes naïve, nous servira de témoin pour les expériences.

- Tf : le vecteur associé contient les fréquences d'apparitions des mots dans le texte.

- Tf-Idf : on remplit le vecteur par les scores Tf-Idf [SAL 88] des mots. Ce type de codage a donné de bons résultats par le passé et est généralement utilisé dans les méthodes dites statistiques.

- Okapi (k, b) : nous attribuons un score Okapi aux mots. Le modèle probabiliste Okapi BM25 [OKP 05] a fait ses preuves en Recherche d'Information en étant plus performant qu'un modèle Tf-Idf. Nous avons, donc, intégré Okapi afin de voir si les performances intrinsèques, meilleures que les autres codages, subsistaient après l'utilisation du classifieur.

$$Score(mot_i, texte) = \frac{(k+1)tf_i}{\left(k\left((1-b)+b\frac{dl}{avdl}\right)+tf_i\right)} \times \log\left(\frac{N-df_i+0.5}{df_i+0.5}\right)$$

où dl est la longueur du document, $avdl$ la longueur moyenne des documents dans tous les corpus, tf la fréquence d'apparition du mot_i dans le document, N le nombre total de documents dans les corpus et df_i le nombre de documents contenant le mot_i.

Notre travail est de fixer les deux paramètres k et b, servant à donner de l'importance à la fréquence d'un mot dans le texte (paramètre k) et à la taille du texte par rapport à la moyenne (paramètre b).

Espace d'étiquettes (8 pour la tâche 1, 5 pour la tâche 2)

Nous avons étiqueté les catégories thématiques suivant le genre, de la forme suivante :

Pour la tâche 1 :

	ART	**ECO**	**SPO**	**TEL**
W	1	2	3	4
LM	5	6	7	8

Pour tâche 2 :

FRA	**INT**	**LIV**	**SCI**	**SOC**
1	2	3	4	5

3 Classifieur probabiliste

Dans cette section, nous décrivons notre modèle de classification multi-classes à l'aide de SVMs retournant les probabilités d'appartenance aux classes.

Définition 1 : (Calcul de la probabilité *a posteriori* pour la classification multi-classes)

Soit E = {($\mathbf{x_1}$, y_1), ($\mathbf{x_2}$, y_2), …, ($\mathbf{x_m}$, y_m)} un ensemble d'apprentissage de m couples d'exemples. On suppose que chaque exemple $\boldsymbol{x_i} \in \Re^n$ et son étiquette associée, y_i, est un entier de l'ensemble Y = {1, 2, 3, …, k} où k correspond au nombre de classes. La probabilité *a posteriori* de classification multi-classes (k>2) est une probabilité conditionnelle, sachant l'exemple $\boldsymbol{x}$, d'étiquette y :

$$P(y/x) = p_i \ avec \sum_{i=1}^{k} p_i = 1 \qquad (1)$$

Un nouveau problème se pose à nous dans le cadre d'une classification multi-classes car les SVMs sont, en règle générale, des classifieurs binaires. Nous pouvons alors appliquer deux sortes de stratégies afin de faire de la classification multi-classes, soit, pour la première, «une classe contre une autre», soit, pour la deuxième, «une contre toutes» (cf. Figure 1). Selon ces deux stratégies, notre problème s'est décomposé en plusieurs sous-problèmes de classification binaire ce qui nous a amené à diviser l'ensemble E en k sous-ensembles différents. Pour la première stratégie, on compare $\binom{2}{k}$ fois, un sous-ensemble *Ei* par rapport à un autre sous-ensemble *Ej*, avec $i \neq j$. Quant à la deuxième stratégie, on compare k fois, un sous-ensemble *Ei* par rapport à $\bigcup_{j \in Y, j \neq i} E_j$. La solution adoptée dans notre travail, a été d'appliquer la première stratégie [HSU 02].

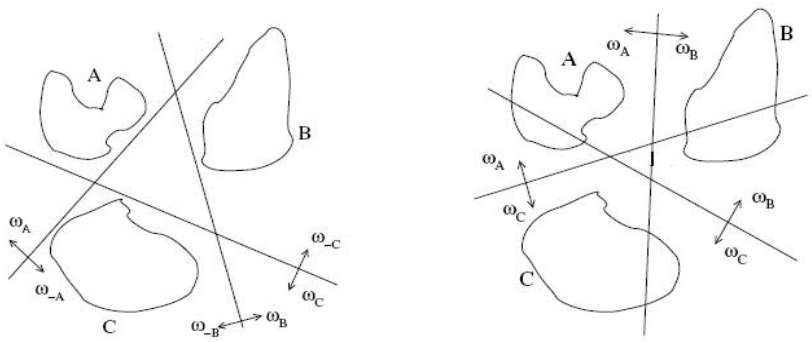

Figure 1 : Deux stratégies de classification multi-classes. A gauche, la stratégie « une contre toutes », à droite, la stratégie, « une contre une ».

Définition 2 : (Conversion de la valeur de sortie d'une classification binaire en une valeur de sortie pour une classification multi-classes)

Le classifieur SVM renvoie une valeur +1 ou -1 suivant qu'un exemple $\boldsymbol{x}$ appartienne à la première ou à la deuxième classe. Cependant, la sortie donnée par le SVM n'est applicable

qu'à un problème de classification binaire. Pour y remédier, nous avons défini une fonction $cl_{ij}(x)$ avec i≠j, renvoyant pour un exemple $\boldsymbol{x}$, le numéro de sa classe (et non +1 ou -1). En d'autres termes :

$$cl_{ij}(\mathbf{x}) = \begin{cases} i & si\ f_{ij}(\mathbf{x}) = -1 \\ j & si\ f_{ij}(\mathbf{x}) = +1 \end{cases} \qquad (2)$$

où *i* et *j* sont les indices des classes, et *f* la fonction de sortie du SVM.

3.1 Sortie probabiliste lors d'une classification binaire

D'après les définitions précédentes, les SVMs retournent des valeurs entières pour chaque exemple $\boldsymbol{x}$. Cependant, on souhaiterait avoir, à la place de ces valeurs, les probabilités d'appartenance de $\boldsymbol{x}$ aux différentes classes. Pour ce faire, dans le cadre de la classification binaire, [PLA 00] et [HER 07] proposent l'utilisation d'une fonction sigmoïde permettant de créer une sortie probabiliste :

$$P(y = i / y = i\ ou\ j, \mathbf{x}) = r_{ij} = \frac{1}{(1 + \exp(A \times f_{ij}(\mathbf{x}) + B))} \qquad (3)$$

où A et B sont estimés en maximisant la log-vraisemblance conditionnelle sur l'ensemble d'apprentissage $E_{i,j}$.

3.2 Sortie probabiliste lors d'une classification multi-classes

L'utilisation de l'équation (3) n'est pas adaptée pour de la classification multi-classes. Pour estimer les probabilités conditionnelles P(y = i | $\mathbf{x}$) = p_i, à partir des classifieurs locaux $cl_{ij}(\boldsymbol{x})$, plusieurs solutions existent, un bref aperçu, est donné dans l'état de l'art ci-dessous.

3.2.1 Etat de l'art

- L'idée la plus simple [KER 90] pour construire la probabilité *a posteriori* (1) à partir des classifieurs locaux $cl_{ij}(\boldsymbol{x})$ est d'utiliser la règle de vote. La probabilité P(y = i | $\boldsymbol{x}$) est égale au nombre moyen de votes pour la classe *i*. Soit la fonction prédicat V($\boldsymbol{x}$) = 1 si $\boldsymbol{x}$ est bien classé, et zéro sinon.

$$p_i = \frac{2}{k(k-1)} \sum_{j \in Y : j \neq i} V(cl_{ij}(\mathbf{x}) = i), \quad i = 1,2,...,k \qquad (4)$$

- Une idée proposée par [HAS 98] est de minimiser la distance de Kullback-Leiber (KL) entre les probabilités de sorties des classifieurs r_{ij} (équation (3)) et $\mu_{ij} = \frac{p_i}{(p_i + p_j)}$ ce qui donne le problème d'optimisation suivant :

$$\min_{\mathbf{p}} KL(\mathbf{p}) = \sum_{i \neq j} |E_{i,j}| \left(r_{ij} \log \frac{r_{ij}}{\mu_{ij}} + (1 - r_{ij}) \log \frac{1 - r_{ij}}{1 - \mu_{ij}} \right) \qquad (5)$$

$$s.c. \sum_{j\in Y: j\neq i} \left|E_{i,j}\right|\mu_{ij} = \sum_{j\in Y: j\neq i} \left|E_{i,j}\right| r_{ij}, \quad \sum_{i=1}^{k} p_i = 1,\ p_i > 0,\ i = 1,2,\ldots,k \qquad (6)$$

- [WU 04] ont suggéré, pour calculer la probabilité *a posteriori* p, de minimiser l'erreur quadratique provenant de l'égalité $r_{ij}.p_j = r_{ji}.p_i$. Pour résoudre ce problème d'optimisation, ils introduisent les multiplicateurs de Lagrange à (7) et à (8).

$$\min_{\mathbf{p}} \frac{1}{2} \sum_{i=1}^{k} \sum_{j\in Y: j\neq i} \left(r_{ji} p_i - r_{ij} p_j\right)^2 \qquad (7)$$

$$s.c. \sum_{i=1}^{k} p_i = 1,\ p_i \geq 0, \quad i = 1,2,3,\ldots k. \qquad (8)$$

3.2.2 Notre modèle

À partir du problème de minimisation de la distance de KL (5), nous proposons de créer un modèle avec une sortie probabiliste qui maximiserait la log-vraisemblance conditionnelle $l(y/\mathbf{x})$.

Par exemple, nous avons un problème de la classification multi-classes, avec $k = 3$. Supposons que l'on a la configuration des décisions $c(f_{ij}(\mathbf{x}))$, formée par l'ensemble de classifieurs locaux :

$$c(f_{ij}(\mathbf{x})) = \{ f_{12}(\mathbf{x}) = -1,\ f_{13}(\mathbf{x}) = +1,\ f_{23}(\mathbf{x}) = -1\}$$

L'étiquette associée à $\boldsymbol{x}$ est y = 2. Les fonctions caractéristiques actives Θ sont donc :

$$\Theta_2(\mathbf{x}, y) = \begin{cases} 1\ si\ f_{12}(\mathbf{x}) = -1\ et\ y = 2 \\ 0\ autrement \end{cases};\ \Theta_{11}(\mathbf{x}, y) = \begin{cases} 1\ si\ f_{13}(\mathbf{x}) = +1\ et\ y = 2 \\ 0\ autrement \end{cases};$$

$$\Theta_{14}(\mathbf{x}, y) = \begin{cases} 1\ si\ f_{23}(\mathbf{x}) = -1\ et\ y = 2 \\ 0\ autrement \end{cases}$$

On associe à chaque fonction caractéristique, un paramètre λ_i, ce qui nous donne un vecteur de paramètres $\boldsymbol{\lambda} \in \Re^d$. La probabilité conditionnelle $p(y/\mathbf{x}, \lambda)$ est alors définie sur l'ensemble de fonctions caractéristiques Θ :

$$p(y/\mathbf{x}, \lambda) = \frac{\exp\left(\sum_{l=1}^{d} \lambda_l \times \Theta_l(\mathbf{x}, y)\right)}{Z(\mathbf{x}, \lambda)} \qquad (9)$$

Avec la fonction auxiliaire $Z(\mathbf{x},\lambda)$:

$$Z(\mathbf{x},\lambda)=\sum_{y=1}^{k}\exp\left(\sum_{l=1}^{d}\lambda_l\times\Theta_l(\mathbf{x},y)\right) \qquad (10)$$

A présent, nous pouvons alors maximiser la log-vraisemblance, comme suit :

$$\min_{\lambda} l(\lambda/E)=-\sum_{j=1}^{m}\left[\sum_{l=1}^{d}\lambda_l\times\Theta_l(\mathbf{x}_j,y_j)+\log\frac{1}{Z(\mathbf{x}_j,\lambda)}\right] \qquad (11)$$

Nous initialisons les paramètres λ_l du modèle à 1.0 pour $l=1,2,...d$. De plus, nous considérons comme critère de convergence, la norme du gradient $||l(\lambda/E)|| < 0{,}008$.

4 Expériences réalisées

Pour toutes les expériences, nous avons décomposé les corpus mis à notre disposition, en deux sous-ensembles, 75% du corpus total pour l'apprentissage et les 25% restant pour le test. Concernant l'algorithme SVM, nous avons utilisé le logiciel libSVM [SVM 01].

Tout d'abord, nous avons comparé les différentes performances des solveurs SVMs suivant les différents prétraitements que l'on a vus en section 2. Dans un premier temps, nous avons testé l'utilité d'une liste générale de mots fréquents ajoutée à différents types de codage textuels. La synthèse des résultats peut être retrouvée dans le tableau 2. Par la suite, nous nous sommes intéressés au codage textuels (Binaire, Tf-Idf, Okapi…) afin de savoir celui qui discriminait le mieux les deux corpus. Nous nous sommes servis d'un modèle Okapi avec les paramètres par défaut, à savoir k = 1.2 et b = 0.75. En parallèle, nous avons cherché, pour les SVMs, le noyau offrant les meilleures performances entre les noyaux de type linéaires, polynomiaux et radial, les résultats sont regroupés dans le tableau 3.

Enfin comme dernière expérience, nous avons essayé, expérimentalement, de trouver la meilleure valeur de k et de b pour le modèle Okapi. Nous avons fait varier le paramètre k par pas de 0.5 allant de 1.0 à 10 et le paramètre b par pas de 0.1 de 0.1 à 1.0. On peut se référer au tableau 4, tableau résumé de l'original.

Expériences		Sans liste		Avec liste	
Code	Noyau	Tache 1	Tache 2	Tache 1	Tache 2
Binary	Linéaire	**89.9606**	85.4765	84.7832	**84.6102**
	Poly	**19.6583**	**27.7731**	18.3968	27.6372
	Radial	**19.6583**	**27.7731**	18.3968	27.6372
Tf	Linéaire	**89.3298**	84.5082	83.2589	**84.5932**
	Poly	**20.3154**	**28.2827**	19.3693	27.79
	Radial	**38.9488**	**41.2944**	27.0959	32.2066

Tf-Idf	Linéaire	**89.0145**	**86.5976**	83.1012	86.4957
	Poly	18.7648	27.807	**20.2628**	**28.2996**
	Radial	**48.7516**	**56.4294**	44.9671	53.151

Tableau 2 : les valeurs sont sous la forme d'accuracy donnée par le SVM avec voting rule. En gras, les meilleures performances, pour un type de codage, un noyau et une tâche. On cherche à savoir s'il faut ou non utiliser d'une liste de mots fréquents.

Expériences sans liste		Noyau		
Code	Tache	Linéaire	Poly	Radial
Binary	1	**90.199**	22.5448	68.4096
	2	**84.9682**	28.8769	70.7686
Tf	1	**89.2071**	26.9264	58.1817
	2	**84.9851**	27.9745	51.8259
Tf-Idf	1	**90.199**	29.7839	80.8382
	2	**86.9766**	32.5393	78.2972
Okapi k=1.2 b=0.75	1	**91.1895**	26.7162	81.8958
	2	**87.8514**	33.9618	81.0658

Tableau 3 : on cherche à déterminer le meilleur noyau et parallèlement le meilleur codage textuel sans utiliser de liste de mots fréquents.

Expériences sans liste et avec noyau linéaire		b		
Okapi		0.7	0.8	0.9
k	2.0	90.7254	91.488	90.276
	2.5	91.9619	**92.5138**	90.7773
	3.0	89.9106	91.724	90.4568

Tableau 4 : on cherche à trouver les valeurs de k et de b pour le codage Okapi donnant les meilleures performances.

Au vu de ces résultats préliminaires, nous avons décidé de réaliser un modèle SVM, utilisant un noyau linéaire et intégrant un codage textuel sous la forme d'un score Okapi avec comme paramètres k = 2.5 et b = 0.8.

Nous avons ensuite, soumis à DEFT 08 trois modèles différents. Le premier implémentant un algorithme simple utilisant la règle de vote (équation 4), le deuxième, notre modèle maximisant la log-vraisemblance (équation 11) et enfin le dernier, un classifieur proposé par Wu (équations (7) et (8)). Les résultats de nos trois soumissions sont à retrouver dans le tableau 5.

Tâche / Soumissions		Règle de vote (1)	Log-vraisemblance (2)	Wu (3)	Moyennes
Tâche 1 Genre	F-score Strict	0.9505	0.9734	**0.9755**	0.9596
	F-score Pondéré	0.6249	**0.9728**	0.9584	
Tâche 1 Catégorie	F-score Strict	0.8038	0.8796	**0.8942**	0.8258
	F-score Pondéré	0.3887	**0.8781**	0.8572	
Tâche 2 Catégorie	F-score Strict	0.8740	0.8738	**0.8758**	0.8105
	F-score Pondéré	0.3930	**0.8735**	0.8167	

Tableau 5 : des résultats pour DEFT 08

5 Conclusions

Nous avons présenté une nouvelle idée afin de convertir les sorties entières d'un solveur SVM en une sortie probabiliste pour un problème de classification multi-classes. Nous avons obtenu de bons résultats pour l'ensemble de ce défi. En effet, deux de nos soumissions sont meilleures que les performances moyennes lors de DEFT'08.

Nous constatons que le modèle de Wu est légèrement meilleur que le notre suivant la mesure F-score strict, mais la tendance s'inverse lors du choix d'un F-score pondéré. On ne peut pas donner d'explications précises car ces résultats dépendent fortement des données. Néanmoins, on peut supposer que le fait de créer r_{ij} par validation croisée (car valeur locale), permet à la méthode de Wu d'être plus précise que la notre. Cependant, elle est beaucoup plus coûteuse en temps.

En conclusion et en guise de perspective, nous pourrions appliquer cette idée pour un problème plus complexe comme c'est le cas avec des documents semi-structurés, où nous devons en plus, du texte, tenir compte de la structure.

Remerciements

- Nous remercions sincèrement le comité de DEFT'08 pour l'organisation et les explications qui nous ont été fournies au cours de ce défi.

Références

[DEF 08] http://deft08.limsi.fr/

[SAL 88] Salton, G., Buckley, C. (1988). Term-weighting approaches in automatic text retrieval, *Information Processing & Management 24*,513-523.

[EDU] http://eduscol.education.fr/D0102/liste-mots-frequents.htm

[OKP 05] Robertson S. (2005). How Okapi came to TREC, *Voorhees and Harman*

[KER 90] Knerr S., Personnaz L., Dreyfus G. (1990). Single-layer training revisited: a stepwise procedure for building and training a neural network, *Neurocomputing: Algorithm, Architectures and Applications*. J. Fogelman Springer-Verlag.

[HAS 98] Hastie, T., Tibshirani, R. (1998). Classification by Pairwise Coupling, *Advances in Neural Information Processing Systems 10*. M. I. Jordan, M. J. Kearns, S. A. Solla.

[PLA 00] Platt JC. (1999). Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods, *Advances in Large Margin Classifiers*, A. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans, 61-74.

[MUL 01] Müller K.-R., Mika S., Rätsch G., Tsuda K., Schölkopf B. (2001). An Introduction to Kernel-Based Learning Algorithms, *IEEE Transactions on Neural Networks 2*, 181-201

[LIN 02] Hsu CW, Lin CJ (2002). A comparison of methods for multi-class support vector machines, *IEEE Transactions on Neural Networks*, 13:415{425}

[WU 04] Wu TF, Lin CJ, Wenig RC (2004). Probability Estimates for Multi-class Classification by Pairwise Coupling, *Journal of Machine Learning Research*, 975-1005

[HER 07] Hérault R, Grandvalet Y, (2007). Sparse probabilistic classifiers. Actes d'*ICML 2007*, 337-344

[TRI 07] Trinh AP, (2007). Classification de texte et estimation probabiliste par Machine à Vecteurs de Support, Actes de *l'atelier du 3ème DEFT*, 71-85.

[SVM 01] Chang CC, Lin CJ (2001). LIBSVM : a library for support vector machines, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm