Lecture Séquentielle de Documents pour la Classification

Gabriel Dulac-Arnold — Ludovic Denoyer — Patrick Gallinari

University Pierre et Marie Curie - UPMC,
LIP6 - Case 169 - 4 Place Jussieu - 75005 PARIS - FRANCE
prénom.nom@lip6.fr

RÉSUMÉ. *Nous proposons un nouveau modèle de lecture séquentielle permettant la classification automatique de documents textuels. Il est basé sur la modélisation d'un agent qui lit un document phrases après phrases et qui peut à tout moment décider d'associer un document à une ou plusieurs catégories données. L'algorithme proposé se base sur une formalisation de la classification de texte en tant que Processus de Décision Markovien, et un apprentissage du modèle par des techniques de renforcement. Des experiences effectuées sur quatre corpus classiques du domaine montrent que l'approche proposée atteint des performances équivalentes à un SVM tout en lisant (en moyenne) que quelques phrases de chaque document.*

ABSTRACT. *We propose to model the text classification process as a sequential decision process. In this process, an agent learns to classify documents into topics while reading the document sentences sequentially and learns to stop as soon as enough information was read for deciding. The proposed algorithm is based on a modelisation of Text Classification as a Markov Decision Process and learns by using Reinforcement Learning. Experiments on four different classical corpora show that the proposed approach performs comparably to classical SVM approaches for large training sets, and better for small training sets. In addition, the model automatically adapts its reading process to the quantity of training information provided.*

MOTS-CLÉS : *Classification, Processus Séquentiels*

KEYWORDS: *Text Categorization, Markov Decision Process*

CORIA 2012, pp. 245–259, Bordeaux, 21-23 mars 2012

1. Introduction

La classification textuelle consiste a apprendre, à partir d'un corpus étiqueté de documents, une corrélation entre les contenus et des étiquettes thématiques, pour ensuite prédire l'étiquette de tout nouveau document rencontré. La classification textuelle a été beaucoup étudiée dans la littérature et est l'une des tâche les plus anciennes dans le domaine de la Recherche d'Informations. Les approches classiques sont basées sur des méthodes d'apprentissage bien connues telles que les modèles génératifs – Naive Bayes par exemple (Lewis *et al.,* 1994)(Lewis *et al.,* 1996) — ou bien des modèles discriminants comme les Machines à Vecteur de Support (SVM)(Joachims, 1998). Ces algorithmes considèrent le plus souvent une représentation en sac de mots des documents, dans laquelle l'ordre des mots et des phrases a été perdu, et tente de prédire l'étiquette d'un document en prenant en compte l'ensemble des termes présents dans ce document. Les SVM linéaires en particulier sont les modèles de référence dans le domaine et fonctionnent particulièrement bien (Dumais *et al.,* 1998). Cependant, quelques inconvénients majeurs ont été identifiés dans la littérature :

– Ces méthodes se basent sur le contenu complet d'un document afin de décider à quelle(s) catégorie(s) il appartient. L'hypothèse sous-jacente est que l'information thématique est une information homogène, distribuée dans l'ensemble du document. Cette hypothèse est adaptée au cas du traitement de documents courts, peu bruités, tels que la fréquence globale des mots peut être facilement associée à des catégories. Cependant, ces méthodes ne sont pas très adaptées au traitement de grands documents dans lesquels des informations thématiques multiples peuvent être présentes uniquement très localement, dans quelques phrases par exemple.

– De plus, les méthodes classiques ne peuvent être appliquées que si les documents à classer sont connus dans leur totalité. Dans le cas où il existe un coût associé à l'acquisition de l'information textuelle, les méthodes qui considèrent les documents en entier ne sont pas très efficaces car elles n'ont pas été conçues pour classer avec une information partielle uniquement.

En considérant ces inconvénients, des tentatives ont été faites afin d'utiliser la nature séquentielle des documents pour la classification textuelle, et aussi dans le cadre de la classification de passages de documents. Les premiers modèles développés pour le traitement séquentiel sont principalement des extensions du modèle Naive Bayes et de Modèles de Markov Cachés. Denoyer et al. (Denoyer *et al.,* 2001) par exemple propose un modèle original dont le but est de modéliser un document comme une séquence de passages non-pertinents et de passages pertinents pour une catégorie donnée. Dans (Wermter *et al.,* 1999), les auteurs proposent un modèle basé sur les Réseaux de Neurones Récurrents pour le routage de documents. D'autres approches ont proposé d'étendre les SVMs linéaires à des données séquentielles, notamment par l'utilisation de noyaux de chaînes de caractères (Lodhi *et al.,* 2002). Enfin, les modèles séquentiels ont été utilisés pour l'Extraction d'Informations (Leek, 1997, Amini *et al.,* 2000), la classification de passage de documents (Kaszkiel *et al.,* 1999, Jiang *et al.,* 2006) ou le développement de moteurs de recherche (Miller *et al.,* 1999, Bendersky *et al.,* 2008).

Nous proposons ici un nouveau modèle pour la classification textuelle qui est peu affecté par les éléments cités précédemment. Notre approche modélise un agent qui lit un document phrase par phrase, tout en décidant conjointement d'associer ou non le document a une ou plusieurs catégories. Elle est basée sur des processus de décision séquentiels dont le but est de classer un document en se concentrant sur les phrases pertinentes. Le modèle proposé apprend non seulement à classer un document dans une ou plusieurs catégories, mais aussi *quand* classifier, et quand arrêter la lecture du document. Ce dernier point est très important car il signifie que le système est capable de décider la classification *au plus tôt*, avant même d'avoir lu la totalité du document concerné.

Les contributions du papier sont :

1) Nous proposons un nouveau type de modèle séquentiel pour la classification textuelle basé sur l'idée de lire séquentiellement les phrases d'un document, tout en associant des catégories à ce document.

2) De plus, nous proposons un algorithme basé sur des méthodes d'Apprentissage par Renforcement qui apprend à se focaliser sur les passages pertinents du document. Cette algorithme apprend de plus quand s'arrêter de lire un document, de telle manière à ce que le document soit correctement classé *le plus vite possible*. Cette caractéristique peut être particulièrement utile pour des documents dont l'acquisition des phrases est coûteuse, comme des grands documents du Web ou des documents de type conversationnels.

3) Nous montrons sur des corpus classiques du domaine que notre approche est meilleure que les approches classiques quand la taille du corpus d'entrainement est petite, et obtient des résultats comparables dans les autres cas. Cependant, notre méthode obtient ces performances en ne lisant qu'une petite partie des documents à classer, sans considérer tout le contenu textuel. De plus, notre algorithme est capable d'adapter le nombre de phrases à lire en fonction de la difficulté de la tâche de classification, ne lisant que quelques phrases dans les problème faciles, et plus de phrases dans les problèmes complexes où par exemple le nombre d'exemples d'apprentissage est faible.

L'article est organisé comme suit : dans la Section 2 nous présentons une description générale de notre méthode. Nous formalisons notre algorithme à l'aide d'un Processus de Décision Markovien dans la Section 3 et nous détaillons l'approche dans le cas de la classification multi-label et dans celui de la classification mono-label. Nous présentons ensuite l'ensemble des expériences effectuées sur 4 corpus du domaine dans la Section 4.

2. Définition de la Tâche et Principes Généraux de l'Approche

Soit $\mathcal{D}$ l'ensemble des documents textuels possibles, et $\mathcal{Y}$ l'ensemble des C catégories numérotées de 1 à C. Chaque document d de $\mathcal{D}$ est associé avec une ou plusieurs

categories[1] de $\mathcal{C}$. Cette information d'étiquette est connue uniquement pour un sous-ensemble des documents $\mathcal{D}_{train} \subset \mathcal{D}$ appelé ensemble d'apprentissage, composé de N_{train} documents notés $\mathcal{D}_{train} = (d_1, ..., d_{N_{train}})$. Les étiquettes du document d_i sont stockées dans un vecteur de scores $y^i = (y^i_1, ..., y^i_C)$ tel que :

$$y^i_k i = \begin{cases} 1 \text{ si } d_i \text{ appartient à la catégorie } k \\ 0 \text{ sinon} \end{cases} .$$

Le but de la classification textuelle est de calculer, pour chaque document d de $\mathcal{D}$, le score correspondant à chaque catégorie. La fonction de classification f_θ de paramètres θ est alors définie comme :

$$f_\theta : \begin{cases} \mathcal{D} : \{0,1\}^C \\ d \rightarrow y^d \end{cases} .$$

Apprendre le classifieur consiste à trouver une paramétrisation optimale θ^* qui réduit le coût moyen tel quel

$$\theta^* = \underset{\theta}{\text{argmin}} \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} L(f_\theta(d_i), y^{d_i}), \quad (1)$$

où L est la fonction de coût qui permet de quantifier l'erreur de classification de $f_\theta(d_i)$.

2.1. *Vue Générale de l'Approche*

Cette section vise à fournir une description intuitive de l'approche proposée. Les idées présentées ici sont formellement décrites dans la Section 3, et sont ici abordées de manière plus informelle.

2.1.1. *Inférence*

Nous proposons de modéliser le processus de classification textuelle comme un processus séquentiel de décision. Dans ce processus, notre classifieur lit séquentiellement un document phrase après phrase et peut décider — à chaque étape du processus de lecture — si le document appartient à une ou plusieurs des catégories. Le classifieur peut aussi à tout moment décider d'arrêter de lire le document si il considère que les phrases déjà lues ont permis de bien décider de la classification de ce document.

Dans cet exemple décrit dans la Figure 1, la tâche consiste à classer un document composé de 4 phrases. Au début du processus, le document n'est pas classifié. Le

1. Dans cet article, nous considérons à la fois la tâche de classification mono-label, où un document est associé à exactement une catégorie, et la tâche de classification multi-label où un document est associé à une ou plusieurs catégories.

classifieur commence la lecture en lisant la première phrase du document. Parce qu'il considère que cette première phrase ne contient pas assez d'information pour décider d'une éventuelle assignation à une ou plusieurs catégories, le classifieur décide de lire la phrase suivante. Après la lecture des deux premières phrases, le classifieur décide alors de classer le document dans la classe *cocoa*, considérant que l'information déjà acquise est suffisante pour prendre une telle décision. Le classifieur décide alors de lire la troisième phrase et, en considérant la nouvelle information acquise, décide d'arrêter le processus de lecture. Le processus est alors terminé. Le document considéré a donc été classé dans la catégorie *cocoa*.

Si le document avait appartenu à plusieurs catégories, le classifieur aurait pu assigner d'autres catégories au document au fur et à fur mesure de la lecture de nouvelles phrases.

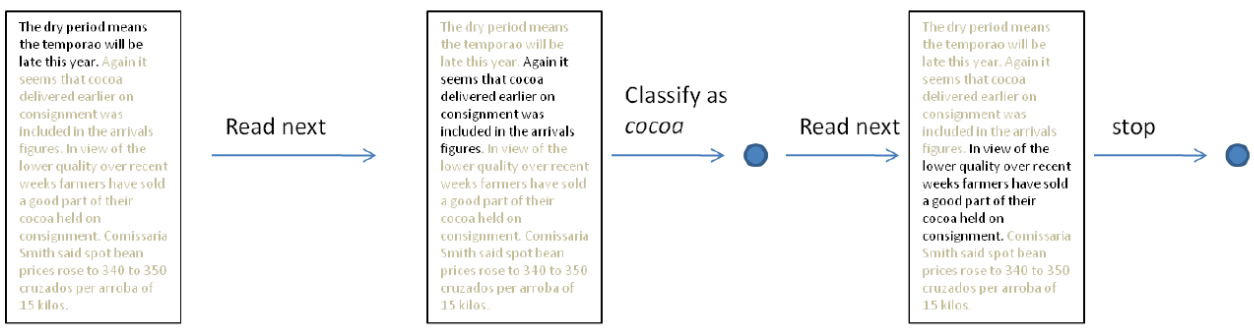

Figure 1. *Inférence sur un document d'exemple de la classe cocoa dans un cadre multi-label. On voit que le classifieur, aprés avoir lu deux phrases, décide de classer le document sous le label cocoa. En lisant une phrase de plus, le modèle décide que le document n'appartient à aucune autre classe et arrête la classification.*

Dans cet exemple, le modèle a pris quatre **actions** : *next*, *classify as cocoa*, *next* puis *stop*. Le choix de chaque action a été entièrement dépendant de l'état courant du processus de lecture. Le choix des actions en fonction de l'état courant est appelé une **politique** du classifieur. La politique — noté π — consiste à transformer des états en action en fonction d'un score calculé. Ce score est appelé Q-value — noté $Q(s, a)$ — et reflète l'intérêt à choisir l'action a dans l'état s du processus.

Par rapport à la Q-value, le processus d'inférence peut être vu comme un processus **gourmand** qui, à chaque pas de temps, choisit la meilleure action a^* définie comme l'action avec le score le plus élevé par rapport à $Q(s, a)$:

$$a^* = \underset{a}{\operatorname{argmax}}\, Q(s, a). \tag{2}$$

2.1.2. *Apprentissage*

Le processus d'apprentissage consiste à calculer la $Q\text{-}fonction$[2] qui minimise le coût de classification (comme présenté en équation (1)) des documents de l'ensemble

2. La *Q-fonction* est une approximation de $Q(s, a)$

d'apprentissage. La procédure d'apprentissage est basé sur une technique de Monte-Carlo permettant de trouver, pour chaque état du processus, les *bonnes* et les *mauvaises* actions. Les *bonnes* actions sont celles qui permettent d'obtenir un faible coût de classification. La discrimination entre les *bonnes* et les *mauvaises* actions est alors apprise à l'aide d'un classifieur classique du domaine comme un SVM ou un réseau de neurones.

Une illustration du processus d'apprentissage sur le même exemple que précédemment est donnée en figure 2. Pour commencer, un état possible du processus de classification est tiré aléatoirement. Ensuite, pour chaque action possible de cet état, la politique courante est appliquée jusqu'au bout, et le coût de classification final est ensuite calculé. L'algorithme d'apprentissage construit alors un ensemble de bonnes actions et de mauvaises actions — les bonnes actions étant celles qui ont obtenu le plus petit coût de classification. Ce processus est répété à partir de plusieurs états possibles issus des documents d'entrainement. À la fin, le système apprends un classifieur sur l'ensemble des *bonnes* et des *mauvaises* actions issues des étapes précédentes. Le classifieur appris correspond à la nouvelle politique pour laquelle ce processus d'apprentissage est répété jusqu'à obtention d'une politique optimale.

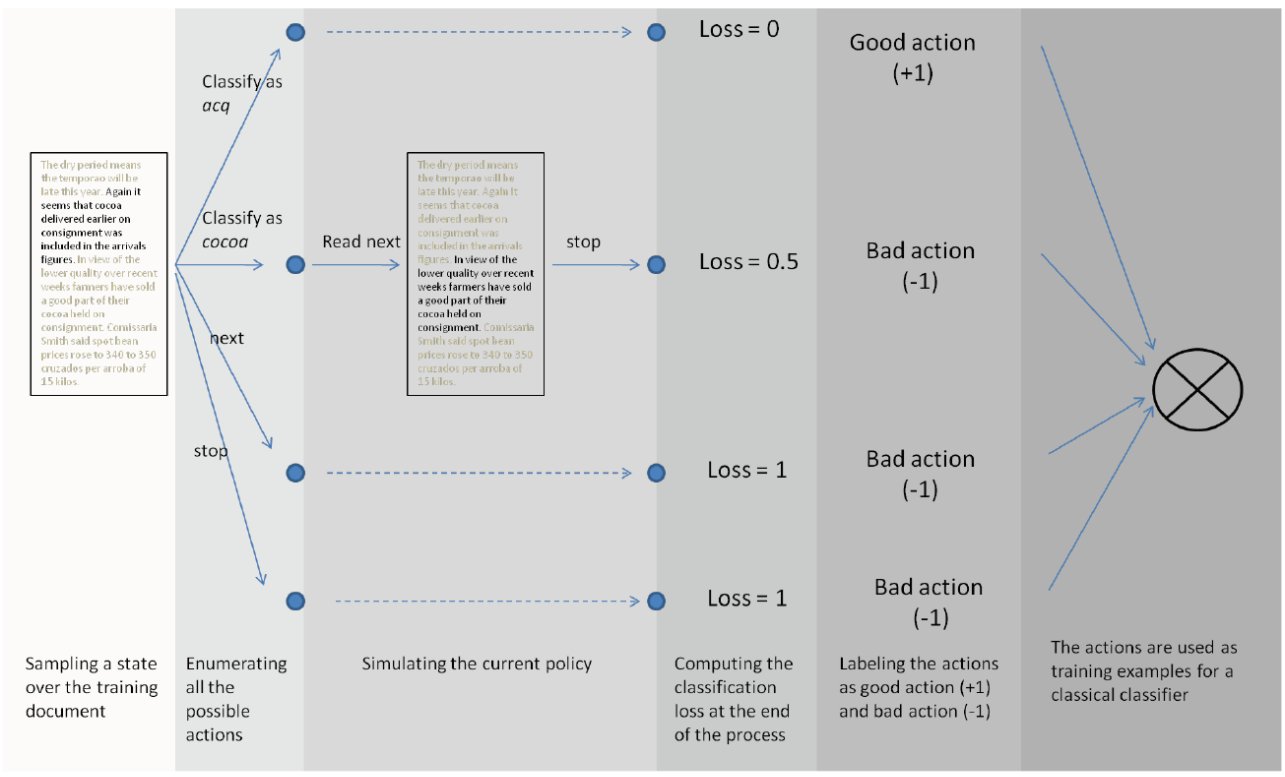

Figure 2. *Apprentissage du modèle séquentiel. Les différentes étapes pour une itération d'apprentissage sont illustrées de gauche à droite en considérant un unique document d'apprentissage.*

2.2. *Préliminaires*

Nous avons présenté les principes de notre approche et donné une description intuitive des processus d'inférence et d'apprentissage. Nous allons maintenant formaliser cet algorithme sous la forme d'un Processus de Décision Markovien (MDP) pour lequel la politique optimale sera trouvée grâce à l'utilisation d'algorithmes d'apprentissage par renforcement. Notez que nous n'allons ici utiliser que des notations pertinentes pour notre approche, et cette section ne décrira pas certaines "parties" du MDP non utiles dans notre cas.

2.2.1. *Processus de Décision Markoviens*

Un processus de décision markovien est un formalisme mathématique permettant de modéliser les processus de prise de décision séquentiel. Nous considérons ici des MDPs déterministes définis par un tuple $(\mathcal{S}, \mathcal{A}, T, r)$. Ici, $\mathcal{S}$ est l'ensemble des états possibles du système, $\mathcal{A}$ est l'ensemble des actions, et $T : \mathcal{S} \times \mathcal{A} \rightarrow S$ est la fonction de transition telle que $T(s, a) \rightarrow s'$ — cette fonction décrit le fait que le système bouge de l'état s à l'état s' quand l'action a est appliquée. La récompense $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ est une valeur qui reflète la qualité de l'action a dans l'état s par rapport au but fianl de l'agent qui prend les décisions. Nous noterons $\mathcal{A}(s) \subseteq \mathcal{A}$ pour référer à l'ensemble des actions possibles qu'un agent peut utiliser dans l'état s.

Un agent interagit avec le MDP en commençant dans un état initial $s \in \mathcal{S}$. L'agent choisit ensuite une action $a \in \mathcal{A}(s)$, l'applique au système et se retrouve ainsi dans l'état s' à travers la fonction de transition $T(s, a)$. En même temps, l'agent reçoit une récompense — éventuellement nulle — $r(s, a)$ et continue ensuite jusqu'à ce qu'il atteigne un état final du processus s_{final} pour lequel l'ensemble des actions possibles est vide : $\mathcal{A}(s_{final}) = \emptyset$.

2.2.2. *Apprentissage par Renforcement*

Soit $\pi : \mathcal{S} \rightarrow \mathcal{A}$ une politique stochastique telle que $\forall a \in \mathcal{A}(s), \pi(s) = a$ avec la probabilité $P(a|s)$. Le but de l'apprentissage par renforcement est de trouver une politique optimale qui maximise la récompense cumulée obtenue par l'agent, depuis l'état initial, jusqu'à l'état final. Nous considérons ici le cas horizon fini pour lequel la récompense cumulée est la somme des récompenses obtenues à chaque étape en utilisant la politique π. La politique optimale π^* qui maximise cette récompense cumulée est celle définie ainsi :

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{s \in \mathcal{S}} \mathbb{E}_\pi [\sum_{t=0}^{T} r(s_t, a_t)] \tag{3}$$

Plusieurs algorithmes d'apprentissage ont été développés afin de trouver une telle politique, selon la structure du MDP, la nature des états (discrets ou continus), etc. Dans la plupart des approches, une politique π est définie à l'aide d'une estimation de $Q(s, a)$ qui correspond à l'espérance de la récompense cumulée que l'agent peut

obtenir dans l'état s en appliquant l'action a. Dans ce cas, une politique π est alors définie comme :

$$\pi = \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}} Q(s,a) \tag{4}$$

et l'algorithme d'apprentissage vise à trouver une Q-fonction optimale Q^* qui correspond à la politique optimale π^*.

Étant donné le très grand nombre d'états auquel nous avons affaire dans notre cas, nous nous placerons dans le cadre de l'apprentissage par renforcement approché dans lequel la Q-fonction est approchée par une fonction paramétrée $Q_\theta(s,a)$ de paramètre θ tel que :

$$Q_\theta(s,a) = < \theta, \Phi(s,a) > \tag{5}$$

où $< \cdot, \cdot >$ est le produit scalaire entre deux vecteurs, et $\Phi(s,a)$ est un vecteur de description représentant la paire état-action (s,a). Le problème d'apprentissage correspond dans ce cas à trouver les paramètres optimaux θ^* tels que :

$$\pi^* = \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}} < \theta^*, \Phi(s,a) > . \tag{6}$$

3. Classification Textuelle et Processus Séquentiel de Décision

Formellement, nous considérons qu'un document d est composé d'une séquence de phrases tel que : $d = (\delta_1^d, \ldots, \delta_{n_d}^d)$, où δ_i^d correspond à la i-ème phrase du document et n_d est le nombre total de phrases qui le composent. À chaque phrase δ_i^d correspond un vecteur représentatif — un vecteur TF-IDF normalisé — qui décrit le contenu de cette phrase.

3.1. *MDP pour la Classification Multi-Label*

Notre processus séquentiel de classification multi-label est décrit à l'aide d'un MDP présenté ici. Il est important de noter que le MDP pour le processus de classification mono-label est une restriction du cas général que nous décrivons ici.

– Chaque état s est un triplet $(d, p, \hat{y})$ tel que :

- d est le document que l'agent est en train de lire ;

- $p \in [1, n_d]$ correspond à la phrase qui est en cours de lecture ; cela implique que δ_1^d à δ_{p-1}^d ont précédemment été lues ;

- $\hat{y}$ est l'ensemble des catégories qui sont déjà assignées au document, c'est-à-dire les catégories qui ont été choisies précédemment par l'agent au cours de sa lecture, tel que $\hat{y}_k = 1$ si et seulement si le document a été assigné à la catégorie k pendant le processus de lecture, et 0 sinon.

– L'ensemble des actions $\mathcal{A}(s)$ est composé de :

- Une ou plusieurs actions de **classification** notées *classify as* k pour chaque catégorie k telle que $\hat{y}_k = 0$. Ces actions correspondent au fait de décider d'assigner le document d à la catégorie k.

- Une action **next sentence** notée *next* qui correspond au choix de continuer la lecture du document — lecture de la phrase suivante.

- Une action **stop** notée *stop* qui correspond à la terminaison du processus de lecture.

– Un ensemble de transitions $T(s, a)$ telles que :

- $T(s, \textit{classify as } k)$ effectue la transformation de l'état suivante : $\hat{y}_k \leftarrow 1$.

- $T(s, \textit{next})$ effectue la transformation de l'état suivante : $p \leftarrow p + 1$.

- $T(s, \textit{stop})$ arrête le processus séquentiel — l'état d'arrivée est un état terminal du processus.

– La récompense $r(s, a)$ est définie telle que :

$$r(s,a) = \begin{cases} F_1(y, \hat{y}) & \text{si } a \text{ est une action } \textit{stop} \\ 0 & \text{sinon} \end{cases}, \tag{7}$$

où y est le vecteur des catégories réelles de d et $\hat{y}$ est le vecteur des catégories prédites pour d par le processus de classification à la fin du processus séquentiel. Le score F_1 d'un document est défini par :

$$F_1(y, \hat{y}) = 2 \cdot \frac{p(y, \hat{y}) \cdot r(y, \hat{y})}{p(y, \hat{y}) + r(y, \hat{y})} \tag{8}$$

avec

$$p(y, \hat{y}) = \sum_{k=0}^{C} \mathbb{1}(\hat{y_k} = y_k) / \sum_{k=0}^{C} \hat{y_k} \text{ et } r(y, \hat{y}) = \sum_{k=0}^{C} \mathbb{1}(\hat{y_k} = y_k) / \sum_{k=0}^{C} y_k$$

3.1.1. *MDP pour la Classification Mono-Label*

Dans le cadre de la classification mono-label, nous restreignons l'ensemble des actions possibles. Ainsi, l'action *classify as* k conduit à un état s du processus tel que $A(s) = \{\textit{stop}\}$, forçant ainsi le processus à s'arrêter dès qu'une catégorie a été assignée au document. Notez que, dans ce cas, la récompense correspond au taux d'erreur de classification classique : 1 si la catégorie choisie est la bonne, et 0 sinon.

3.2. *Vecteurs de Caractéristiques*

Nous devons aussi définir la fonction Φ permettant de représenter une paire état-action sous forme vectorielle. Le but de cette fonction est de représenter la paire (s, a) comme entrée d'un classifieur classique dont le but sera d'attribuer un score permettant de décider si l'action a dans l'état s est une *bonne* ou une *mauvaise* action.

Tandis que les méthodes de classification textuelle classiques ne représentent les documents que par des vecteurs globaux — typiquement des vecteurs TF-IDF calculés sur tous les termes du document — nous cherchons ici une représentation qui permette à la fois de représenter l'information déjà acquise par l'agent (l'ensemble des phrases déjà lues) ainsi que l'information additionnelle locale de la phrase suivante à lire. De plus, si dans l'état s le document a déjà été assigné à un ensemble de catégories, le vecteur $\Phi(s, a)$ doit contenir cette information. Nous commençons par définir une représentation vectorielle pour l'état s notée $\Phi(s)$ de la manière suivante :

$$\Phi(s) = \begin{pmatrix} \frac{\sum_{i=1}^{p} \delta_i^d}{p} \; \delta_p^d \; \hat{y_0} \dots \hat{y_C} \end{pmatrix} \tag{9}$$

$\Phi(s)$ est la concaténation de sous-vecteurs qui décrivent : la moyenne des caractéristiques des phrases précédemment lues, les caractéristiques de la dernière phrase lue, ainsi que l'ensemble des catégories déjà assignées.

À partir de $\Phi(s)$, le vecteur $\Phi(s, a)$ est obtenu en utilisant l'astuce du *block vector* décrit dans (Har-Peled *et al.,* 2002) qui consiste à projeter $\Phi(s)$ dans un espace de grande dimension tel que :

$$\Phi(s, a) = (0 \dots \phi(s) \dots 0)\,, \tag{10}$$

La position de $\Phi(s)$ dans ce grand vecteur $\Phi(s, a)$ dépend de l'action a. Le résultat est l'obtention d'un vecteur de très grande dimension facilement utilisable par un classifieur statistique linéaire.

3.3. *Apprendre à trouver la Politique Optimale de Classification*

Afin de trouver la meilleure politique de classification, nous avons utilisé un algorithme récent d'apprentissage par renforcement appelé *Approximate Policy Iteration with Rollouts*. Rapidement, cette méthode utilise une approche de monte-carlo pour évaluer la qualité de toutes les actions pour un ensemble d'états issus des données d'apprentissage échantillonné aléatoirement. Ensuite, le classifieur linéaire dont le but est de discriminer les *bonnes* actions des *mauvaises* actions est appris. Pour des raisons d'espace, nous n'entrons pas ici dans les détails de l'algorithme d'apprentissage et renvoyons au papier par Lagoudakis et al. (Lagoudakis *et al.,* 2003) pour plus de détails. Une description intuitive de cet algorithme est donnée en Section 2.1.

4. Expériences

Nous avons appliqué notre modèle sur quatre corpus différents : trois sont des corpus mono-label pour lesquels le but consiste à trouver l'unique catégorie associée

Corpus	Nb. de doc.	Nb. catègories	Nb. phrases / doc.	Tâche
R8	7678	8	8.19	Mono-label
R10	12 902	10	9.13	Multi-label
Newsgroup	18 846	20	22.34	Mono-label
WebKB	4 177	4	42.36	Mono-label

Tableau 1. *Caractéristiques des Corpus.*

à chaque document, et le dernier corpus est un corpus multi-label où le but est de trouver un ensemble de catégories pour chaque document. Les corpus sont décrits dans la table 1 :

– Le corpus Reuters-21578 nous a fourni deux jeux de données :

- Le corpus **Reuters8** [3] est un corpus mono-label composé de 8 catégories.

- Le corpus **Reuters10** est un corpus multi-label contenant 10 catégories possibles basé sur les 10 plus grandes catégories de Reuters-21578.

– Le corpus WebKB[4] est un corpus composé de pages Web réparties dans 4 catégories différentes.

– le corpus 20 Newsgroups[5] (20NG) est un corpus de news issues de 20 newsgroups différents correspondant aux 20 catégories.

Tous les jeux de données ont été préprocéssés de la même manière : la ponctuation a été enlevée, les stop-words SMART et les mots de moins de trois caractères ont été supprimés, et tous les mots ont été enracinés avec l'enrancineur de Porter. Les évaluations de référence ont été faites avec libSVM(Chang *et al.,* 2001) sur des vecteurs TF-IDF normalisés selon le protocole décrit dans (Joachims, 1998).

4.1. *Protocole d'Évaluation*

La plupart des algorithmes de classification issus de l'apprentissage statistique sont des méthodes de classification *molle* qui calculent un score ou une probabilité pour toutes les paires document-catégorie possibles. Dans notre cas, le modèle est un modèle de classification *dure* qui directement assigne une ou plusieurs catégorie à chaque document, sans passer par le calcul d'un score et un éventuel seuillage. Les mesures classiques comme le *breakeven point* par exemple ne sont pas adaptées à la classification dure et ne peuvent pas être utilisées pour comparer notre approche avec les méthodes classiques de type SVM par exemple. Nous avons fait le choix d'utiliser les mesures *micro-F1* et *macro-F1*. Ces mesures correspondent au score F_1 classique

3. http://web.ist.utl.pt/~acardoso/datasets/
4. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/
5. http://people.csail.mit.edu/jrennie/20Newsgroups/

calculé pour chacune des catégories et moyenné sur l'ensemble de catégories. La mesure *macro-F1* ne prend pas en compte la taille des catégories, tandis que la mesure *micro-F1* la prend en compte. Les performances obtenues ont été moyennées sur différents découpages entrainement/test qui ont été générés aléatoirement à partir des corpus originaux. Nous avons utilisé la même approche pour évaluer notre modèle et les modèles de référence. Pour chaque taille de corpus d'entrainement, les performances ont été moyennées sur 5 runs différents. Les hyper-paramètres des modèles ont été choisis manuellement de manière à maximiser la performance des systèmes. Concernant l'approche séquentielle, chaque politique a été apprise sur 10000 états du MDP échantillonés aléatoirement de manière uniforme, et nous avons effectué une unique simulation de monte-carlo par état et par action. La politique initiale est une politique purement aléatoire. Il est important de noter que, d'un point de vue pratique, la méthode séquentielle n'est pas plus dure à paramétrer qu'un SVM classique car ses performances dépendent en fait peu des hyper-paramètres choisis.

4.2. *Résultats Expérimentaux*

Dans les figures qui illustrent les résultats, SVM correspond à la machine à vecteur de support classique, tandis que STC (Sequential Text Classification) correspond à notre approche. Dans le cas des expériences mono-label (Figure 3 et 4-gauche), les performance des SVMs et de STC sont tout à fait comparable. Il est important de noter que, dans le cas des petits ensembles d'apprentissage (1%, 5%), STC est meilleure que SVM de 1% à 10% selon le corpus. Par exemple, sur le corpus R8 nous pouvons voir que pour les deux scores F_1, STC dépasse SVM de $\sim 5\%$ pour un ensemble d'apprentissage composé de 1% des documents du corpus. Le même effet peut être constaté sur le corpus Newsgroup, où STC fait $+10\%$ en comparaison de SVM pour un ensemble d'apprentissage de taille 1%. STC semble donc particulièrement intéressant quand le nombre d'exemples d'apprentissage est faible.

Le comportement du processus de lecture séquentiel est étudié dans la Figure 5. Dans cette figure, *Reading Size* correspond à la proportion moyenne de phrases lues par document. Nous pouvons constater que cette valeur décroit quand la taille de l'ensemble d'entrainement augmente. Cela est dû au fait que, quand le corpus d'apprentissage est petit, la tâche de classification est plus ambiguë et plus d'informations doivent être acquises par le système afin de discriminer les documents. Dans le cas où le corpus d'apprentissage est plus grand, la tâche de classification devient plus simple, et le modèle peut facilement choisir quelle(s) catégorie(s) assigner en ne lisant que quelques phrases. Sur la partie droite de la Figure 5, nous pouvons voir un histogramme du nombre de documents classés en fonction de la proportion de phrases lues. Nous pouvons noter que en moyenne, les documents sont classés en lisant 41% de leur contenu. Pour la plupart des documents, le système ne lit que très peu de contenu avant de les classer tandis que pour certains autres, le système décide de tout lire avant de prendre sa décision. Cette figure illustre le fait que notre système à bien appris à lire plus ou

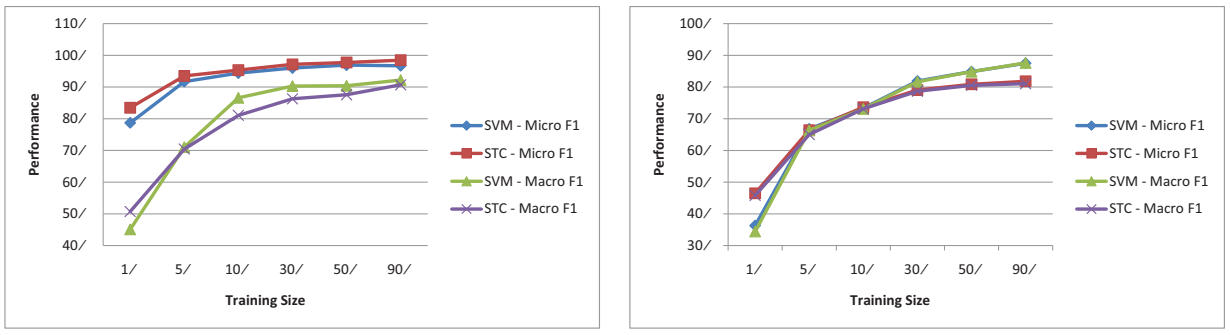

Figure 3. *Performances pour le corpus R8 (gauche) et le corpus NewsGroup (droite)*

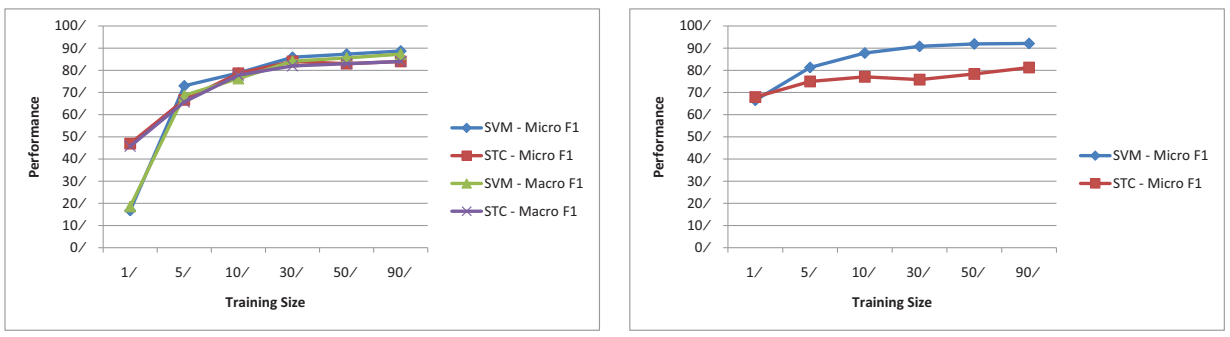

Figure 4. *Performances pour le corpus WebKB (gauche) et le corpus R10 (droite).*

moins les documents en fonction du contenu rencontré et est donc capable de s'adapter à chaque nouveau document.

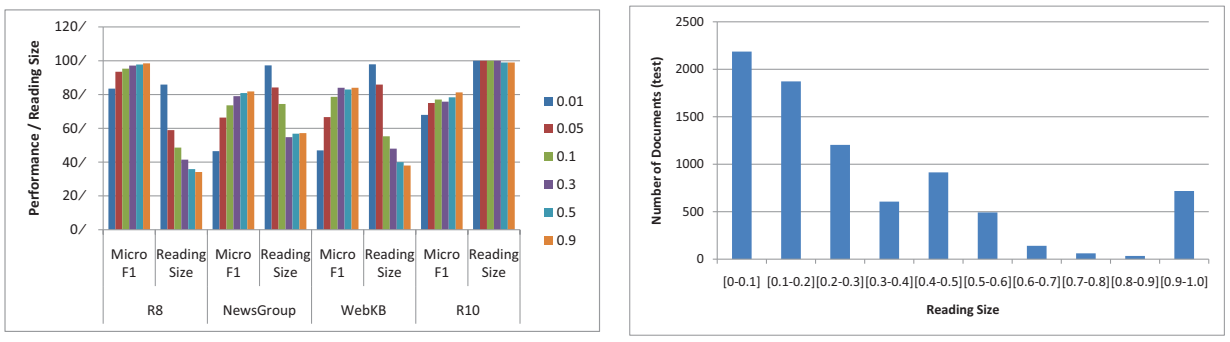

Figure 5. *Synthèse sur la proportion de phrases lues avant classification pour les différents corpus (gauche). Nombre de documents classés en fonction de la proportion de phrases lues pour le corpus R8 avec un ensemble d'apprentissage de taile 30% (droite).*

Dans le cas de la classification multi-label, les résultats sont un peu différents. Tout d'abord, nous pouvons voir que pour le corpus R10, la performance obtenue est moins bonne que la performance du modèle de base pour les grands corpus d'apprentissage. De plus, en moyenne, le système a tendance à lire toutes les phrases du document avant de le classer. Ce comportement semble cependant assez intuitif et normal : dans le cas de la classification multi-label, l'agent ne peut jamais être certain de l'appartenance d'un document à une catégorie avant d'avoir lu la dernière phrase de ce document. Il doit donc tout lire afin d'être certain que la dernière phrase ne contient pas un contenu associé à une catégorie non déjà assignée. Nous pensons de plus que la baisse de performance constatée est principalement due au fait que, contrairement à l'apprentissage de SVM où un modèle est appris par classe, nous apprenons ici un unique modèle de classification pour toutes les catégories, ce qui est une contrainte plus forte et tend à diminuer les performances du système.

5. Conclusions

Nous avons présenté un nouveau modèle qui apprend à classer des documents textuels en lisant les phrases de ce document séquentiellement, et qui étiquette le document dès que que possible, avant de l'avoir lu en entier. Cette méthode possède des propriétés intéressantes montrées sur plusieurs corpus. En particulier dans le cadre de la classification mono-label, le modèle est capable d'apprendre à ne lire qu'une petite partie du document quand l'ensemble d'apprentissage est de grande taille — c'est à dire quand la tâche est facile —, et à l'inverse à lire une grande partie des documents lorsque l'ensemble d'apprentissage est petit — c'est-à-dire quand la tâche de classification est plus ambiguë et plus difficile. Le système est donc capable d'adapter son comportement à la difficulté de la tâche, est permet d'obtenir à la fois un système plus rapide dont les performances sont très proches de celles des modèles classiques de l'état de l'art, voire meilleures quand les corpus d'apprentissage sont petits.

Ce travail ouvre beaucoup de perspectives de recherche intéressantes dans le domaine de la classification textuelle. En particulier, il est possible d'imaginer des actions additionnelles au MDP permettant par exemple à l'agent de classer un document de manière plus complexe. Par exemple, on pourrait considérer que l'agent possède la capacité de se déplacer dans un arbre XML, choisissant automatiquement les parties pertinentes d'un document semi-structuré pour bien le classer. De même, on peut aussi, de manière plus prospective, imaginer un agent capable, en présence de contenu complexe, d'aller consulter des ressources externes d'informations comme des dictionnaires, ou bien comme le Web.

6. Bibliographie

Amini M.-R., Zaragoza H., Gallinari P., « Learning for Sequence Extraction Tasks », *RIAO*, p. 476-490, 2000.

Bendersky M., Kurland O., « Utilizing passage-based language models for document retrieval », *ECIR'08*, p. 162-174, 2008.

Chang C.-C., Lin C.-J., « LIBSVM : a library for SVMs », 2001.

Denoyer L., Zaragoza H., Gallinari P., « HMM-based passage models for document classification and ranking », *Proceedings of ECIR-01*, p. 126-135, 2001.

Dumais S., Platt J., Heckerman D., M, « Inductive learning algorithms and representations for text categorization », *Proceedings of CIKM*, 1998.

Har-Peled S., Roth D., Zimak D., « Constraint classification : A new approach to multiclass classification », *Algorithmic Learning Theory*p. 1 - 11, 2002.

Jiang J., Zhai C., « Extraction of coherent relevant passages using hidden Markov models », *ACM Trans. Inf. Syst.*, vol. 24, n° 3, p. 295-319, 2006.

Joachims T., « Text categorization with support vector machines : Learning with many relevant features », *Machine Learning : ECML-98*, 1998.

Kaszkiel M., Zobel J., Sacks-Davis R., « Efficient passage ranking for document databases », *ACM Trans. Inf. Syst.*, vol. 17, n° 4, p. 406-439, 1999.

Lagoudakis M. G., Parr R., « Reinforcement learning as classification : Leveraging modern classifiers », *ICML*, 2003.

Leek T. R., « Information Extraction Using Hidden Markov Models », 1997.

Lewis D., Ringuette M., « A comparison of two learning algorithms for text categorization », *Third annual symposium on document analysis*p. 1-14, 1994.

Lewis D., Schapire R., Callan J., R, « Training algorithms for linear text classifiers », *ACM SIGIR*p. 120-123, 1996.

Lodhi H., Saunders C., Shawe-Taylor J., Cristianini N., Watkins C., « Text classification using string kernels », *J. Mach. Learn. Res.*, vol. 2, p. 419-444, 2002.

Miller D. R. H., Leek T., Schwartz R. M., « BBN at TREC7 : Using Hidden Markov Models for Information Retrieval », *In Proceedings of TREC-7*, p. 133-142, 1999.

Wermter S., Arevian G., Panchev C., « Recurrent neural network learning for text routing », vol. 2, p. 898 -903 vol.2, 1999.