Réduction de la complexité spatiale et temporelle du Compact Prediction Tree pour la prédiction de séquences

Ted Gueniche*, Philippe Fournier-Viger*

*Département d'informatique, Université de Moncton
18 Antonine-Maillet, Moncton, NB E1A 3E9
ted.gueniche@gmail.com, philippe-fournier-viger@umoncton.ca

Résumé. La prédiction de séquences de symboles est une tâche ayant de multiples applications. Plusieurs modèles de prédiction ont été proposés tels que DG, All-k-order markov et PPM. Récemment, il a été montré qu'un nouveau modèle nommé Compact Prediction Tree (CPT) utilisant une structure en arbre et un algorithme de prédiction plus complexe, offre des prédictions plus exactes que plusieurs approches de la littérature. Néanmoins, une limite importante de CPT est sa complexité temporelle et spatiale élevée. Dans cet article, nous pallions ce problème en proposant trois stratégies pour réduire la taille et le temps de prédiction de CPT. Les résultats expérimentaux sur 7 jeux de données réels montrent que le modèle résultant nommé CPT+ est jusqu'à 98 fois plus compact et est 4.5 fois plus rapide que CPT, tout en conservant une exactitude très élevée par rapport à All-K-order Markov, DG, Lz78, PPM et TDAG.

1 Introduction

Le problème de prédiction de séquences est un problème important en fouille de données, défini de la façon suivante. Soit un alphabet $Z = \{e_1, e_2, ..., e_m\}$ contenant un ensemble d'éléments (symboles). Une séquence est une suite d'éléments totalement ordonnée $s = \langle i_1, i_2, ...i_n \rangle$, où $i_k \in Z$ $(1 \leq k \leq n)$. Un modèle de prédiction M est un modèle entraîné avec un ensemble de séquences d'entraînement. Une fois entraîné, le modèle peut être utilisé pour effectuer des prédictions. Une prédiction consiste, à prédire le prochain élément i_{n+1} d'une séquence $\langle i_1, i_2, ...i_n \rangle$ en utilisant le modèle M. La prédiction de séquences a des applications importantes dans une multitude de domaines tels que le préchargement de pages Web (Deshpande et Karypis, 2004; Padmanabhan et Mogul, 1996), la recommandation de produits de consommation, la prévision météorologique et la prédiction des tendances du marché boursier.

Un grand nombre de modèles de prédictions ont été proposés pour la prédiction de séquences. Un des modèle les plus connus est PPM (Prediction by Partial Matching)(Cleary et Witten, 1984). Ce modèle, basé sur la propriété de Markov, a engendré une multitude d'approches dérivées telles que Dependancy Graph (DG)(Padmanabhan et Mogul, 1996), All-k-order-Markov (Pitkow et Pirolli, 1999) et Transition Directed Acyclic Graph (TDAG) (Laird et Saul, 1994). Bien que des propositions ont été faites pour réduire la complexité temporelle

et spatiale de ces modèles (Begleiter et al., 2004), l'exactitude de leurs prédictions a subi peu d'amélioration. D'autre part, un certain nombre d'algorithmes de compression ont été adaptés pour la prédiction de séquences tels que LZ78 (Ziv et Lempel, 1978) et Active Lezi (Gopalratnam et Cook, 2007). De plus, des algorithmes d'apprentissage machine comme les réseaux de neurones et la découverte de règles d'association séquentielles ont été employés pour faire de la prédiction de séquences (Fournier-Viger et al., 2012; Sun et Giles, 2001). Néanmoins, ces modèles souffrent de limites importantes. Premièrement, la plupart d'entre eux partent de l'hypothèse Markovienne qu'un événement ne dépend que de son prédécesseur. Or, ce n'est pas le cas pour de nombreuses applications, ce qui nuit à l'exactitude des prédictions. Deuxièmement, tous ces modèles sont construits avec perte d'information par rapport aux séquences d'entraînement. Donc, ils n'utilisent pas toute l'information disponible dans les séquences d'entraînement pour effectuer les prédictions.

Pour pallier ces limites, un modèle nommé Compact Prediction Tree (CPT) (Gueniche et al., 2013) a été récemment proposé. Il utilise une structure en arbre pour compresser les séquences d'entraînement sans perte ou avec une perte minime d'information. De plus, il emploie un algorithme de prédiction conçu pour tenir compte du bruit et de plusieurs événements antérieurs lors d'une prédiction plutôt que seulement le dernier. Il a été montré que ce modèle peut obtenir des prédictions jusqu'à 12 % plus exactes que PPM, DG et All-K-order-markov sur des jeux de données provenant de divers domaines, ce qui constitue un gain important. Néanmoins, une limite de CPT est sa complexité temporelle et spatiale élevée. Dans cet article, nous pallions ces problèmes en proposant trois stratégies pour réduire la taille et le temps de prédiction de CPT. De plus, nous présentons une comparaison expérimentale avec davantage de modèles de prédiction de la littérature : All-K-order Markov, DG, Lz78, PPM et TDAG. Les résultats expérimentaux sur 7 jeux de données réels montrent que le modèle résultant nommé CPT+ est jusqu'à 98 fois plus compact et est jusqu'à 4.5 fois plus rapide que CPT. De plus, CPT+ conserve une exactitude très élevée par rapport aux autres approches de la littérature.

Le reste de cet article est organisé de la façon suivante. La section 2 décrit brièvement le modèle CPT. Les sections 3 et 4 proposent respectivement de nouvelles stratégies pour réduire la taille du modèle CPT et ses temps de prédiction. La section 5 présente l'évaluation expérimentale avec plusieurs jeux de données et les principaux modèles de prédictions de la littérature. Finalement, la section 6 est dédiée à la conclusion et aux travaux futurs.

2 Le Compact Prediction Tree

Le CPT (Compact Prediction Tree) est un modèle de prédiction récemment proposé (Gueniche et al., 2013). Ses principales caractéristiques distinctives sont (1) qu'il stocke sous forme compressée l'ensemble des séquences d'entraînement sans perte ou avec une perte minime d'information, et (2) qu'il utilise une mesure de similarité pour identifier les séquences similaires à une séquence à prédire pour faire une prédiction. Cette mesure est tolérante au bruit, ce qui permet à CPT de prédire les prochains éléments de sous-séquences qui n'ont pas été vues dans les séquences d'entraînement, alors que d'autres approches plus strictes telles que PPM et All-K-order-markov ne peuvent pas faire de prédiction dans ce cas. Le modèle CPT est défini par deux processus : un processus d'entraînement et un processus de prédiction.

2.1 Le processus d'entraînement

Le processus d'entraînement génère trois structures distinctes à partir des séquences d'entraînement : (1) un Arbre de Prédiction (AP), (2) un Dictionnaire de Séquences (DS) et (3) un Index Inversé (II). Pendant l'entraînement, les séquences sont considérées les unes après les autres pour construire incrémentalement ces trois structures. À titre d'exemple, la figure 1 illustre la création des structures de CPT par insertion successive des séquences $s_1 = \langle A, B, C \rangle$, $s_2 = \langle A, B \rangle$, $s_3 = \langle A, B, D, C \rangle$, $s_4 = \langle B, C \rangle$ et $s_5 = \langle E, A, B, A \rangle$, où l'alphabet $Z = \{A, B, C, D, E\}$ est utilisé.

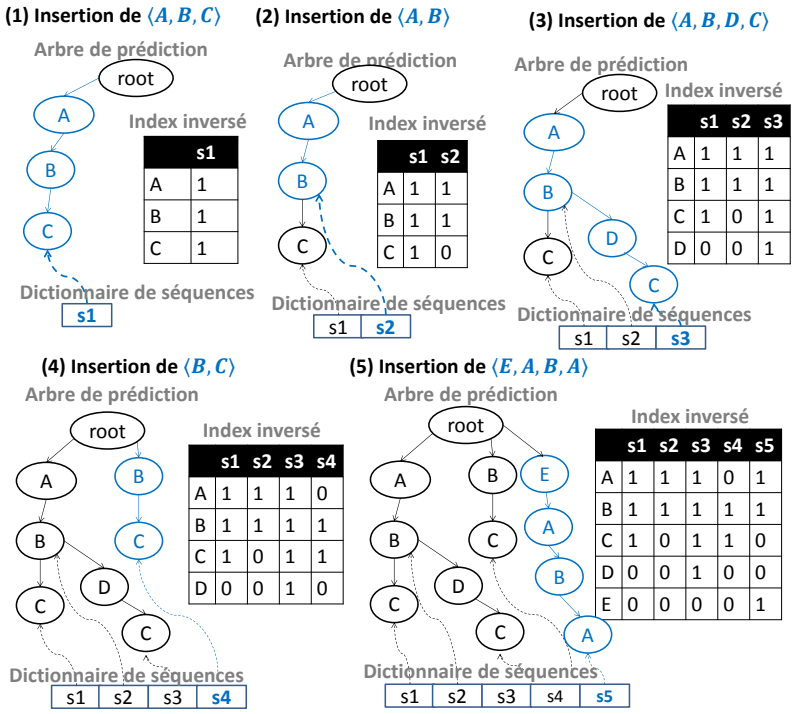


FIG. 1 – *Construction des structures de CPT*

L'*Arbre de Prédiction* est une forme d'*arbre préfixe* (alias *trie*). Chacun des noeuds de l'arbre représente un élément et chacune des séquences d'entraînement est représentée par un chemin partant de la racine de l'arbre et se terminant par un noeud interne de l'arbre ou une feuille. La construction de cet arbre a une basse complexité. Insérer une séquence de m éléments demande de parcourir/créer au plus m noeuds. La construction complète de l'arbre est $O(n)$ où n est le nombre de séquences à insérer. Tout comme un arbre préfixe, cet arbre est une représentation compacte des séquences d'entraînement, car les séquences partageant un préfixe commun partagent un chemin dans l'arbre. Par exemple, à la figure 1, les séquences s_1, s_2 et s_3 partagent le même chemin correspondant au préfixe $\langle A, B \rangle$. Dans le pire cas, le gain spatial offert par cette compression est nul, mais en pratique, tout dépendant de la densité et de

la similarité des séquences du jeu de données utilisé, l'arbre peut offrir une réduction spatiale très importante allant jusqu'à 98% (Gueniche et al., 2013).

Le *Dictionnaire de Séquences* est une structure qui permet d'extraire chacune des séquences d'entraînement de l'arbre de prédiction. Lors de la construction du modèle CPT, un identifiant unique est assigné à chaque séquence. Il est égal à 1 pour la première séquence insérée (dénoté s_1) et est incrémenté d'un pour chaque séquence subséquente (s_2, s_3, ...). Le dictionnaire de séquences associe chaque identifiant de séquence s_a à un pointeur vers un noeud de l'arbre. Ce noeud représente le dernier élément de la séquence s_a dans l'arbre. Grâce à cette structure, il est possible de parcourir chaque séquence d'entraînement dans l'arbre de prédiction du dernier au premier élément.

L'*Index Inversé* permet d'identifier rapidement dans quelles séquences apparaît un ensemble d'éléments d'une séquence à prédire. L'index inversé contient un vecteur de bits v_e pour chaque élément e de l'alphabet Z présent dans les séquences d'entraînement. Le k-ième bit d'un vecteur de bit v_e prend la valeur 1 si l'élément e apparaît dans la séquence s_k, sinon il prend la valeur 0. Par exemple, à la figure 1, le vecteur de bit de l'élément C après l'insertion des séquences s_1, s_2, s_3, s_4 et s_5 est 10110, car C apparaît dans les séquences s_1, s_3 et s_4. L'index inversé est utilisé pour déterminer rapidement les séquences d'entraînement contenant un ensemble d'éléments d'une séquence à prédire. Cela est réalisé en faisant l'intersection des vecteurs de bits des éléments. Par exemple, déterminer l'ensemble des séquences contenant les éléments A et C est réalisé par l'opération $11101 \wedge 10110$, donnant le résultat 10000, autrement dit $\{s_1\}$. Grâce à l'index inversé, cette tâche est très rapide ; $O(i)$ où i est le nombre d'éléments dans l'ensemble.

2.2 Le processus de prédiction

Le processus de prédiction de CPT utilise les trois structures décrites précédemment. Soit une séquence $s = \langle i_1, i_2, ...i_n \rangle$ de n éléments et y, un nombre entier représentant le nombre d'éléments de s à considérer pour faire une prédiction. Le *suffixe de taille* y de s dénoté $P_y(s)$ est défini comme étant $P_y(s) = \langle i_{n-x+1}, i_{n-x+2} ...i_n \rangle$. La prédiction du prochain élément de s est effectuée de la façon suivante : CPT identifie tout d'abord les séquences similaires à $P_y(s)$, c.à.d. qui contiennent les derniers y éléments de $P_y(s)$ dans n'importe quel ordre et positions. Puis, pour chaque séquence similaire, CPT considère son *conséquent*. Le *conséquent* d'une séquence u est la sous-séquence débutant après le dernier élément en commun avec $P_y(s)$ jusqu'à la fin de u. Chaque élément e dans un de ces conséquents est ensuite stocké dans une structure nommé *Table de Compte* (TC) avec son nombre d'occurrences (ce nombre est une estimation de la probabilité $P(e|P_y(s))$). L'élément ayant le plus grand nombre d'occurrences est l'élément prédit par CPT. La mesure de similarité utilisée pour déterminer les séquences similaires est de nature stricte, mais est relâchée dynamiquement par le processus de prédiction, pour deux raisons. Premièrement, avec une mesure de similarité trop stricte, une séquence à prédire peut n'être similaire à aucune séquence d'entraînement, et donc aucune prédiction n'est possible. Deuxièmement, une mesure de similarité trop stricte ne permet pas de considérer qu'une séquence peut-être partiellement similaire à une autre. Or, dans les applications réelles, il y a souvent des éléments présents dans les séquences qui sont du bruit. Pour relâcher la mesure de similarité, CPT suppose qu'un ou plusieurs éléments présents dans le suffixe de la séquence à prédire sont du bruit et qu'ils peuvent être ignorés lors du calcul de similarité. Le calcul de similarité pour un suffixe $P_y(s)$ est fait par niveau, où à chaque

niveau k = $1, 2, ..., |P_y(s)| - 1$ toutes les sous-séquences de taille $|P_y(s)| - k$ de $P_y(s)$ sont générées. Chacune des sous-séquence u est utilisée pour trouver les séquences similaires dans l'ensemble de séquences d'entraînement et pour mettre à jour la TC. Ce relâchement de la mesure de similarité se poursuit pour la séquence à prédire d'un niveau à l'autre tant que TC n'a pas été mise à jour un nombre minimum de fois.

3 Stratégies de compression de l'arbre de prédiction

Bien que CPT offre des prédictions plus exactes que les principaux modèles de prédiction de la littérature selon une étude antérieure (Gueniche et al., 2013), une limite importante de CPT est sa complexité spatiale. Il a été montré que la taille des structures de CPT est inférieure à All-k-order Markov, mais demeure nettement supérieures à d'autres modèles comme DG et PPM. L'arbre de prédiction étant la structure la plus imposante de CPT, nous proposons ci-après deux stratégies pour réduire sa taille.

Stratégie 1 : Compressions des Chaînes Fréquentes (CCF). Certaines répétitions peuvent être identifiées dans les séquences d'entraînement. Dépendamment du jeu de données, ces répétitions peuvent être nombreuses et fréquentes. La *compression des chaînes fréquentes* consiste à identifier les sous-chaînes fréquentes d'éléments apparaissant dans les séquences d'entraînement, puis à remplacer les sous-chaînes fréquentes par des éléments individuels.

Soit une séquence $s = \langle i_1, i_2, ..., i_n \rangle$. Une séquence $c = \langle j_{m+1}, j_{m+2}, ..., j_{m+k} \rangle$ est une *sous-chaine de* s, dénoté $c \sqsubseteq s$, si et seulement si $1 \leq m \leq m + k \leq n$. Pour un ensemble de séquences d'entraînement S, une sous-chaîne d est fréquente si $|\{t | t \in S \wedge d \sqsubseteq t\}| > minsup$ pour un seuil $minsup$ fixé par l'utilisateur.

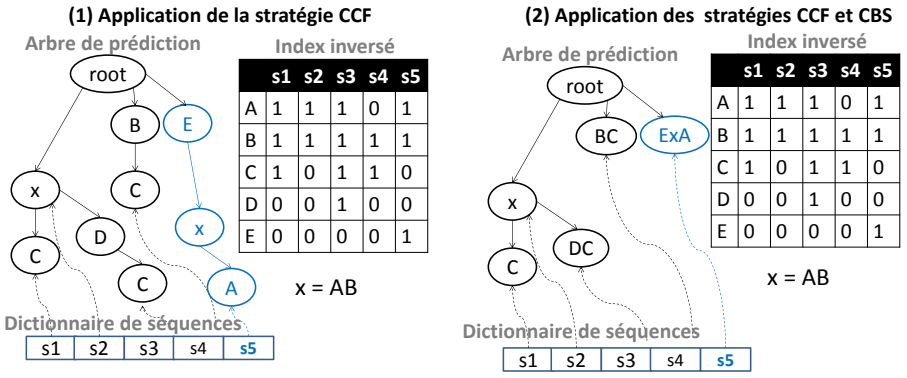
La compression des chaînes fréquentes est effectuée pendant la phase d'entraînement de CPT en trois étapes : (1) identifier les chaînes fréquentes dans l'ensemble des séquences d'entraînement, (2) créer un nouvel élément dans l'alphabet Z pour représenter chaque sous-chaîne fréquente et (3) remplacer les sous-chaînes fréquentes par l'élément correspondant lors de la construction de l'arbre de prédiction de CPT.

L'identification de séquences fréquentes dans un ensemble de séquences est un problème populaire en fouille de données, pour lequel un grand nombre d'algorithmes ont été proposés. Pour cette tâche, nous avons adapté un des algorithmes les plus performants nommé PrefixSpan (Pei et al., 2001), afin de ne découvrir que les séquences fréquentes d'éléments consécutifs (sous-chaînes). De plus, nous avons ajouté la contrainte que les sous-chaînes fréquentes doivent respecter des contraintes de longueur minimale $minSize$ et maximale $maxSize$ (deux paramètres).

Les sous-chaînes fréquentes identifiées sont stockées dans une nouvelle structure nommée *Dictionnaire des chaînes fréquentes* (DCF). Cette structure associe un nouvel élément non présent dans l'alphabet Z (dans les séquences d'entraînement) à chaque sous-chaîne fréquente. Le DCF permet de rapidement convertir une sous-chaîne en son élément correspondant et vice-versa. Lors de l'insertion des séquences d'entraînement dans l'arbre de prédiction, le DCF est utilisé pour remplacer chaque sous-chaîne par son élément correspondant.

À titre d'exemple, l'illustration (1) de la figure 2 affiche la compression de l'arbre de prédiction de l'illustration (5) de la figure 1 par la stratégie CCF. La sous chaîne fréquence $\langle A, B \rangle$ a été remplacée par un nouveau symbole x, réduisant le nombre de noeuds de l'arbre de prédiction.

La stratégie de compression de séquences CCF a un effet seulement sur l'arbre de prédiction où son nombre de noeuds et sa hauteur tendent à diminuer grandement. La stratégie CCF est transparente pour le processus de prédiction de CPT. En effet, lors de l'extraction de séquences similaires, les branches de l'arbre de prédiction sélectionnées sont décompressées à la volée par DCF.



	s1	s2	s3	s4	s5
A	1	1	1	0	1
B	1	1	1	1	1
C	1	0	1	1	0
D	0	0	1	0	0
E	0	0	0	0	1

	s1	s2	s3	s4	s5
A	1	1	1	0	1
B	1	1	1	1	1
C	1	0	1	1	0
D	0	0	1	0	0
E	0	0	0	0	1

FIG. 2 – *Application des stratégies de compression de l'arbre CCF et CBS*

Stratégie 2 : Compressions des Branches Simples (CBS). La compression des branches simples est une stratégie de compression intuitive et efficiente pour réduire la taille de l'arbre de prédiction. Une *branche simple* est une branche qui mène à une seule feuille. Chaque noeud d'une branche simple a donc entre 0 et 1 noeud fils. La stratégie CBS consiste à remplacer chaque branche simple par un seul noeud représentant son chemin vers la feuille du début à la fin. Par exemple, l'illustration (2) de la figure 2 illustre l'arbre de prédiction de l'exemple après l'application des stratégies CCF et CBS. La stratégie CBS a respectivement remplacé les branches simples $D - C$, $B - C$ et $E - x - A$ par des noeuds uniques DC, BC et ExA.

L'identification et le remplacement de branches simples sont faits en un seul parcours de l'arbre de prédiction. L'index inversé et le dictionnaire de séquences n'étant pas influencés par cette approche, le seul changement au processus de prédiction est la décompression dynamique des branches simples lorsque nécessaire. La complexité de ce remplacement est de $O(n * (1 - t))$ où s est le nombre de séquence et t le taux de recouvrement de l'arbre, ce dernier est défini comme le "ratio" de noeuds qui partagent plusieurs séquences par le nombre total de noeuds dans l'arbre.

4 Stratégie de réduction des temps de prédiction

Stratégie 3 : Prédiction avec réduction du Bruit Amélioré (PBA). Tel qu'expliqué précédemment, pour prédire le prochain élément $s_n + 1$ d'une séquence $s = \langle i_1, i_2, ..., i_n \rangle$, CPT utilise le suffixe de taille y de s dénoté $P_y(s)$ (les y derniers éléments de s), où y est un paramètre propre à chaque jeu de donnée. CPT prédit le prochain élément de s en parcourant les séquences similaires à son suffixe $P_y(s)$. La recherche de séquences similaires est rapide ($O(y)$). Toutefois, le mécanisme de réduction du bruit lors des prédictions (décrit à la section 2)

ne l'est pas, car il requiert de considérer non seulement $P_y(s)$ pour une prédiction, mais aussi toutes les sous-séquences de $P_y(s)$ de taille $t > k$. Plus y et k sont grands, plus le nombre de sous-séquences à considérer l'est aussi, et donc le temps de prédiction. Lors d'une tâche de prédiction, certains éléments dans une séquence à prédire peuvent être considérés comme du bruit si leur simple présence affecte de façon négative le résultat de la prédiction. La stratégie PBA se base sur l'hypothèse que le bruit observé dans une séquence est constitué des éléments ayant une faible fréquence, où la fréquence d'un élément est le nombre de séquences d'entraînement contenant l'élément. Pour cette raison, PBA enlève seulement les éléments qui ont une faible fréquence pendant la phase de prédiction. Puisque la définition du bruit de CPT+ est plus restrictive que celle de CPT, un moins grand nombre de sous-séquences sont considérées. Cette réduction à un impact positif et tangible sur les temps de calculs tel que présentés dans notre évaluation expérimentale (section 5). Le pseudo-code illustrant la stratégie PBA est présenté ci-après (Algorithme 1). L'algorithme prend en paramètres le préfixe $P_y(s)$ à prédire, les autres structures de CPT, un taux de bruit et un nombre minimum de mise à jour à faire à la TC pour faire une prédiction. Le taux de bruit représente le pourcentage d'éléments dans une séquence qui doivent être considérés comme du bruit ; un taux de bruit de 0 indique que les séquence n'ont pas de bruit alors qu'un taux de bruit de 0.4 signifie que 40% des éléments d'une séquence pourrait être du bruit. PBA est récursive de nature et considère un nombre minimal de sous-séquences dérivées de $P_y(s)$ pour faire une prédiction. Le bruit est d'abord retiré de chaque sous-séquence. Puis la TC est mise à jour. Lorsque le nombre minimal de mise à jour est atteint, une prédiction est faite comme dans CPT en utilisant la TC. La stratégie PBA est une généralisation de la stratégie de réduction du bruit utilisée par CPT. En effet, selon les paramètres utilisés, il est possible de reproduire le fonctionnement original de CPT. Les trois contributions principales apportées par PBA sont l'imposition d'un nombre minimal de mise à jour de la TC pour faire une prédiction, la définition du bruit basé sur la fréquence d'un élément et la réduction relative du bruit par rapport à la longueur de la séquence.

Algorithme 1 : L'algorithme de prédiction avec PBA

```
input  : PS : le suffixe Ps, CPT : les structures de CPT, TB : le taux de bruit
output : Seq : un ou plusieurs éléments prédits

file.ajouter(PS);
while nombreMiseAjour < minNombreMiseAJourTC ∧ file.nonVide() do
    suffixe = file.prochain();
    elementsBruit = selectionnerElementsMoinsFrequents(TB);
    foreach elementBruit ∈ elementsBruit do
        suffixeSansBruit = copierSuffixeSansBruit(suffixe, elementBruit);
        if suffixeSansBruit.length > 1 then
            file.ajouter(suffixeSansBruit);
        end
        mettreAJourCountTable(CPT.countTable, suffixeSansBruit);
        nombreMiseAjour++;
    end
    retourne faireUnePrediction(CPT.countTable);
end
```

5 Évaluation expérimentale

Nous avons effectué une série d'expériences pour comparer la performance de CPT+, CPT et les principaux modèles de prédiction de la littérature All-K-order Markov, DG, Lz78, PPM et TDAG. Pour implémenter CPT+, nous avons obtenu et modifié le code source proposé dans l'article original de CPT (Gueniche et al., 2013). Pour permettre la reproduction des expériences, le code source des modèles et jeux de données sont fournis à l'adresse `http://goo.gl/LE4uYO`. Tous les modèles sont implémentés en Java 8. Les expériences ont été réalisées sur une machine dotée d'un processeur deux coeurs Intel i5 de 4ème génération avec 8 Go de mémoire vive et un SSD en SATA 600. Tous les modèles de prédiction utilisés ont été configurés empiriquement pour tenter de donner des valeurs optimales à chacun de leurs paramètres. PPM et LZ78 n'ont pas de paramètres, DG et AKOM ont respectivement une fenêtre de 4 et un ordre de 5, finalement, par soucis d'espace, TDAG à une hauteur maximale de 6. CPT à 4 paramètres et CPT+ en a 8, leurs valeurs sont elles aussi déterminées via une exploration expérimentale de l'espace de valeurs possibles. Ces valeurs sont accessible dans les fichiers sources du projet. Les paramètres propres à l'expérience se limitent à la longueur minimale et maximale des séquences utilisées, la taille du suffixe à considérer pour une séquence à prédire et la quantité d'éléments à prédire pour chacune des séquences.

Des jeux de données ayant des caractéristiques variées ont été utilisés (cf. Table 1) : séquences courtes/longues, séquences denses/éparses, petit/grands alphabets et divers types de données. Les jeux de données BMS, Kosarak, MSNBC et FIFA consistent en des séquences de pages Web visitées par des utilisateurs sur un site Web. Dans ce scénario, les modèles de prédiction sont appliqués pour prédire la prochaine page Web que visitera chaque utilisateur. Le jeu de données SIGN est un ensemble de phrases exprimées en langage des signes, transcrites à partir de vidéos. Bible Word et Bible Char sont deux jeux de données qui proviennent de la Bible, livre religieux, le premier est l'ensemble des phrases découpées en mots et le second est l'ensemble des phrases découpées en caractères.

Pour l'évaluation des prédictions des modèles, une prédiction est soit un *succès*, un *échec*, ou une *abstention* (si un modèle ne peut effectuer une prédiction). Deux mesures sont utilisées. La *couverture* est le nombre d'abstentions divisé par le nombre de séquences à prédire. *L'exactitude* (alias précision) le nombre de succès divisé par le nombre de séquences à prédire.

Nom	**Nombre de séquence**	**Éléments uniques**	**Longueur moyenne**	**Type de données**
BMS	15,806	495	6.01	Pages Web
KOSARAK	638,811	39,998	11.64	Pages Web
FIFA	573,060	13,749	45.32	Pages Web
MSNBC	250,697	17	3.28	Pages Web
SIGN	730	267	93.00	Langage
BIBLE Word	42,436	76	18.93	Phrases
BIBLE Char	32,502	75	128.35	Caractères

TAB. 1 – *Jeux de données*

Expérience 1 : comparaisons des optimisations. Dans cette première expérience, nous avons tout d'abord évalué les améliorations spatiales présentées à la section 3 en terme de taux de compression et de temps de calcul à l'entraînement. Les autres mesures de performance telles que le temps de prédiction, la couverture et l'exactitude ne sont pas affectées par la compression de l'arbre de prédiction. Pour un arbre de prédiction A avec s noeuds avant compression et $s2$ noeuds après compression, le taux de compression tc_a de A est défini comme $tc = 1-(s2/s)$, et est compris entre 0.0 et 1.0 non inclusivement. Plus la valeur est haute, plus la compression est importante. Les deux stratégies de compression sont évaluées d'abord individuellement (dénotées CCF et CBS) puis en conjonction (dénoté CPT+). Toute compression permet d'obtenir un gain spatial au prix d'un coût temporel. La figure 3 présente cette relation pour chacune des stratégies de compression.

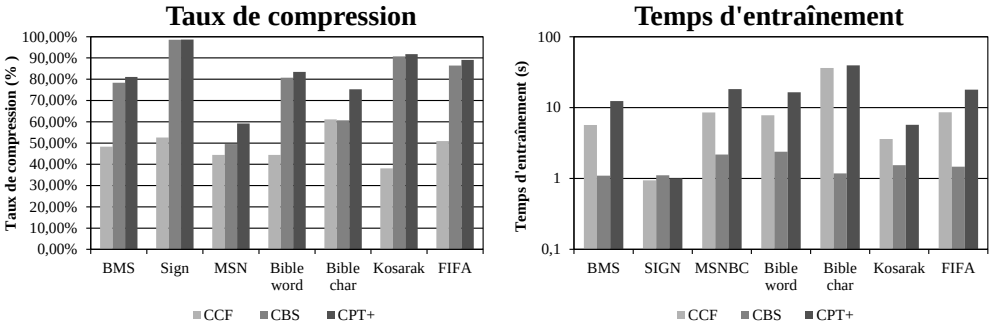


FIG. 3 – *Taux de compression et temps d'entraînement des stratégies de compression.*

Les résultats présentés à la figure 3 montrent que le taux de compression de l'arbre varie selon le jeu de données de 58.90% à 98.65%. CCF offre un taux de compression moyen de 48.55% avec un faible écart type de 6.7%. alors que CBS à un taux de compression moyen de 77.87% avec un écart type beaucoup plus prononcé de 15.9%. L'efficacité de CBS est dépendante au jeu de données ; dans le cas de MSNBC, qui est le jeu de données le moins affecté par les stratégies de compression, la faible cardinalité de son alphabet permet à MSNBC d'être naturellement compressé grâce au fort recouvrement des branches de son arbre de prédiction. En effet, MSNBC ne possède que 17 éléments uniques et même si la taille moyenne des séquences ressemble à celle des autres jeux de données, la taille de son arbre avant compression est très petite. Le jeu de données où les stratégies de compression CCF et CBS sont les plus effectives est SIGN. SIGN a un très faible nombre de séquences, mais chacune d'elle est très longue (en moyenne 93 éléments). Ces caractéristiques font en sorte que son arbre de prédiction a un faible taux de recouvrement et donc une importante partie de ses noeuds n'ont qu'un seul fils ; ce qui rend ce jeu de données un candidat idéal pour la stratégie CBS. CBS offre un taux de compression de 98.60 % pour SIGN.

La figure 3 présente également les temps d'entraînement engendrés par les deux stratégies de compression de CPT, CBS et CCF. La mesure utilisée est un facteur multiplicatif du temps d'entraînement. Par exemple, un facteur de x pour CBS signifie que CBS a eu une phase d'entraînement x fois plus longue. Pour tous les jeux de données à l'exception de SIGN, CBS est plus rapide que CCF. Il est intéressant d'observer que le temps pris par la combinaison des deux stratégies de compression n'est pas simplement une addition de leur coût d'entraînement.

CBS et CCF sont appliqués indépendamment à CPT et pourtant l'utilisation de CBS réduit les temps de calcul de CCF grâce à une diminution du nombre de branches qui ont besoin d'être compressées.

Nous avons également évalué le gain en temps de prédiction et l'exactitude (précision) obtenue en appliquant la stratégie PBA. La figure 4 (gauche) illustre les temps de prédiction de CPT+ (avec CBA), et ceux de CPT. Les gains temporels sont importants pour la plupart des jeux de données notamment pour SIGN et MSNBC où les temps d'entraînement sont jusqu'à 4.5 fois moindres. Pour les jeux de données Bible Word et FIFA, les temps de prédiction sont plus élevés pour obtenir un gain en exactitude comme le montre la figure 4 (droite). L'effet de CBA sur l'exactitude des prédictions est positif pour tous les jeux de données sauf MSNBC. Cette amélioration s'élève jusqu'à 5.47% dans le cas de Bible Word. CBA se montre donc une stratégie effective pour à la fois réduire les temps de prédiction et augmenter l'exactitude des prédictions.

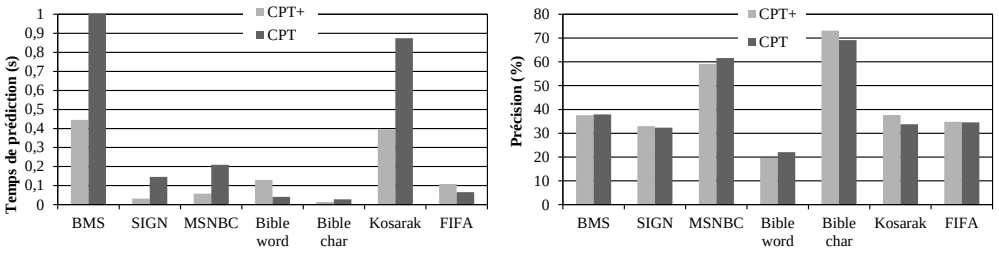


FIG. 4 – *Gains en temps de prédiction et exactitude avec l'ajout de PBA.*

Expérience 2 : Mise à l'échelle. Nous avons également comparé la complexité spatiale de CPT+ (avec ses deux stratégies de compression) avec celle de CPT et All-K-order Markov, DG, Lz78, PPM et TDAG, en termes de mise à l'échelle par rapport au nombre de séquences. Les deux seuls jeux de données utilisés sont FIFA et Kosarak à cause de leur grand nombre de séquences (573,060 et 638,811 respectivement). L'accroissement du nombre de séquences dans cette expérience est quadratique et s'arrête à 128,000 séquences dû aux énormes temps de calcul requis pour réaliser chaque expérience. La figure 5 présente les résultats. Le taux de compression de CPT+ tend à baisser très légèrement avec l'accroissement du nombre de séquences, ce phénomène est causé par un recouvrement de plus en plus important des branches dans l'arbre de prédiction ; car la taille de l'alphabet étant constante, plus de séquences sont utilisées et plus de branches s'unifient. Les modèles DG et PPM ont une croissance linéaire, car ils sont basés sur la taille de l'alphabet et indirectement sur le nombre de séquences d'entraînement. Les autres modèles ont tous une croissance beaucoup plus importante que DG et PPM, notamment TDAG et LZ78.

Expérience 3 : Comparaison avec les autres modèles de prédiction

Dans l'expérience 1, nous avons comparé l'exactitude des prédictions de CPT+ avec celle de CPT afin d'évaluer la contribution de la stratégie PBA. Dans cette expérience, nous effectuons une comparaison de l'exactitude celle des autres principaux modèles de prédiction de la littérature All-K-order Markov, DG, Lz78, PPM et TDAG, sur les mêmes jeux de données. Il est à noter que nous ajoutons dans cette comparaison deux modèles de prédictions (Lz78 et TDAG) qui n'ont pas été utilisés dans l'article original proposant CPT. Chaque modèle de

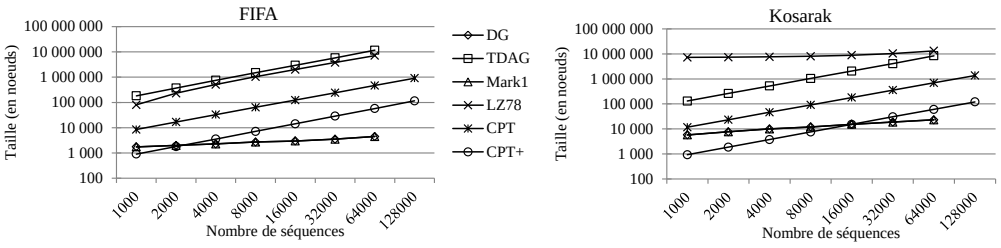


FIG. 5 – *Mise à l'échelle des modèles de prédiction.*

prédiction est entraîné et testé en validation croisée à k plis avec $k = 14$ pour assurer une faible variance inter-expériences. L'exactitude des prédictions obtenues par les différents modèles est présentée dans la table 2. Les résultats montre que CPT+ continue, comme CPT, d'offrir une exactitude généralement nettement supérieure aux autres modèles populaires de la littérature.

Jeu de données	**CPT+**	**CPT**	**AKOM**	**DG**	**LZ78**	**PPM**	**TDAG**
BMS	**38.25**	37.90	31.26	36.46	33.46	31.06	6.95
SIGN	**33.01**	32.33	8.63	3.01	4.79	4.25	0.00
MSNBC	61.50	**61.64**	47.88	55.68	43.64	38.06	31.14
Bible word	27.52	22.05	**38.68**	24.92	27.39	27.06	11.17
Bible char	**73.52**	69.14	7.96	0.00	3.02	0.10	0.00
Kosarak	**37.64**	33.82	20.52	30.82	20.50	23.86	1.06
FIFA	**35.94**	34.56	25.88	24.78	24.64	22.84	7.14

TAB. 2 – *Les modèles de prédiction évalués sur leurs exactitude*

6 Conclusion

Dans cet article, nous avons présenté trois stratégies pour réduire la taille et le temps de prédiction de CPT, nommées CCF (Compression des Chaînes Fréquences), CBS (Compression des Branches Simples) et PBA (Préduction avec réduction du Bruit Améliorée). Les résultats expérimentaux sur 7 jeux de données réels ont montré que le modèle résultant nommé CPT+ est jusqu'à 98 fois plus compact que CPT, et que cette compression demeure lorsque le nombre de séquence augmente. En termes de temps d'exécution, CPT+ s'est montré jusqu'à 4.5 fois plus rapide que CPT. Finalement, CPT+ s'est montré comme étant le modèle offrant les prédictions généralement les plus exactes dans une comparaison avec les principaux modèles de la littérature CPT, All-K-order Markov, DG, Lz78, PPM et TDAG.

Comme travaux futurs, nous adapterons CPT+ pour la prédiction de séquences dans le contexte d'un flux infini de séquences. CPT+, de par sa nature incrémentale, pourrait être adapté à ce problème.

Références

Begleiter, R., R. El-yaniv, et G. Yona (2004). On prediction using variable order markov models. *Journal of Artificial Intelligence Research 22*, 385–421.

Cleary, J. G. et I. Witten (1984). Data compression using adaptive coding and partial string matching. *Communications, IEEE Transactions on 32*(4), 396–402.

Deshpande, M. et G. Karypis (2004). Selective markov models for predicting web page accesses. *ACM Transactions on Internet Technology (TOIT) 4*(2), 163–184.

Fournier-Viger, P., T. Gueniche, et V. S. Tseng (2012). Using partially-ordered sequential rules to generate more accurate sequence prediction. In *Advanced Data Mining and Applications*, pp. 431–442. Springer.

Gopalratnam, K. et D. J. Cook (2007). Online sequential prediction via incremental parsing : The active lezi algorithm. *Intelligent Systems, IEEE 22*(1), 52–58.

Gueniche, T., P. Fournier-Viger, et V. S. Tseng (2013). Compact prediction tree : A lossless model for accurate sequence prediction. In *Advanced Data Mining and Applications*, pp. 177–188. Springer.

Laird, P. et R. Saul (1994). Discrete sequence prediction and its applications. *Machine learning 15*(1), 43–68.

Padmanabhan, V. N. et J. C. Mogul (1996). Using predictive prefetching to improve world wide web latency. *ACM SIGCOMM Computer Communication Review 26*(3), 22–36.

Pei, J., J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, et M.-C. Hsu (2001). Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pp. 0215–0215. IEEE Computer Society.

Pitkow, J. et P. Pirolli (1999). Mining longest repeating subsequence to predict world wide web surfing. In *Proc. USENIX Symp. On Internet Technologies and Systems*, pp. 1.

Sun, R. et C. L. Giles (2001). Sequence learning : from recognition and prediction to sequential decision making. *IEEE Intelligent Systems 16*(4), 67–70.

Ziv, J. et A. Lempel (1978). Compression of individual sequences via variable-rate coding. *Information Theory, IEEE Transactions on 24*(5), 530–536.

Summary

Predicting the next symbol of a sequence of symbols is a task with wide applications. The Compact Prediction Tree (CPT) is a recently proposed prediction model that provide more accurate predictions than several state-of-the-art prediction models. In this paper, we introduce new strategies to reduce the size of CPT and its prediction time. Experimental results on seven datasets shows that the resulting model is up to 98% more compact than CPT and 4.5 times faster, and remains on overall much more accurate than state of the art predictions models All-K-order Markov, DG, Lz78, PPM and TDAG.