CPT+: Decreasing the time/space complexity of the Compact Prediction Tree

Ted Gueniche[1], Philippe Fournier-Viger[1],
Rajeev Raman[2], and Vincent S. Tseng[3]

[1] Dept. of computer science, University of Moncton, Canada
[2] Department of Computer Science, University of Leicester, United Kingdom
[3] Dept. of computer science and inf. eng., National Cheng Kung University, Taiwan
{etg8697, philippe.fournier-viger}@umoncton.ca, r.raman@leicester.ac.uk,
tsengsm@mail.ncku.edu.tw

Abstract. Predicting next items of sequences of symbols has many applications in a wide range of domains. Several sequence prediction models have been proposed such as DG, All-k-order markov and PPM. Recently, a model named Compact Prediction Tree (CPT) has been proposed. It relies on a tree structure and a more complex prediction algorithm to offer considerably more accurate predictions than many state-of-the-art prediction models. However, an important limitation of CPT is its high time and space complexity. In this article, we address this issue by proposing three novel strategies to reduce CPT's size and prediction time, and increase its accuracy. Experimental results on seven real life datasets show that the resulting model (CPT+) is up to 98 times more compact and 4.5 times faster than CPT, and has the best overall accuracy when compared to six state-of-the-art models from the literature: All-K-order Markov, CPT, DG, Lz78, PPM and TDAG.

Keywords: sequence prediction, next item prediction, accuracy, compression

1 Introduction

Sequence prediction is an important task with many applications [1, 11]. Let be an alphabet of items (symbols) $Z = \{e_1, e_2, ..., e_m\}$. A sequence s is an ordered list of items $s = \langle i_1, i_2, ...i_n \rangle$, where $i_k \in Z$ $(1 \leq k \leq n)$. A prediction model is trained with a set of training sequences. Once trained, the model is used to perform sequence predictions. A prediction consists in predicting the next items of a sequence. This task has numerous applications such as web page prefetching, consumer product recommendation, weather forecasting and stock market prediction [1, 3, 8, 11].

Many sequence predictions models have been proposed. One of the most popular is Prediction by Partial Matching (PPM)[2]. It is based on the Markov property and has inspired a multitude of other models such as Dependancy Graph (DG)[8], All-K-Order-Markov (AKOM)[10], Transition Directed Acyclic

Graph (TDAG)[7], Probabilistic Suffix Tree (PST)[1] and Context Tree Weighting (CTW)[1]. Although, much work has been done to reduce the temporal and spatial complexity of these models (e.g. [1, 3]), few work attempted to increase their accuracy. Besides, several compression algorithms have been adapted for sequence predictions such as LZ78 [12] and Active Lezi [4]. Moreover, machine learning algorithms such as neural networks and sequential rule mining have been applied to perform sequence prediction [6, 11].

However, these models suffer from some important limitations [5]. First, most of them assume the Markovian hypothesis that each event solely depends on the previous events. If this hypothesis does not hold, prediction accuracy using these models can severely decrease [5, 3]. Second, all these models are built using only part of the information contained in training sequences. Thus, these models do not use all the information contained in training sequences to perform predictions, and this can severely reduce their accuracy. For instance, Markov models typically considers only the last k items of training sequences to perform a prediction, where k is the order of the model. One may think that a solution to this problem is to increase the order of Markov models. However, increasing the order of Markov models often induces a very high state complexity, thus making them impractical for many real-life applications [3].

CPT[5] is a recently proposed prediction model which compress training sequences without information loss by exploiting similarities between subsequences. It has been reported as more accurate than state-of-the-art models PPM, DG, AKOM on various real datasets. However, a drawback of CPT is that it has an important spatial complexity and a higher prediction time than these models. Therefore, an important research problem is to propose strategies to reduce the size and prediction time of CPT. Reducing the spatial complexity is a very challenging task. An effective compression strategy should provide a huge spatial gain while providing a minimum overhead in terms of training time and prediction time. Furthermore, it should also preserve CPT's lossless property to avoid a decrease in accuracy. Reducing prediction time complexity is also very challenging. An effective strategy to reduce prediction time should access as little information as possible for making predictions to increase speed but at the same time it should carefully select this information to avoid decreasing accuracy.

In this paper, we address these challenges by proposing three strategies named FSC (Frequent Subsequence Compression), SBC (Simple Branches Compression) and PNR (Prediction with improved Noise Reduction). The two first strategies are compression strategies that reduce CPT size by up to two orders of magnitude while not affecting accuracy. The third strategy reduces the prediction time by up to 4.5 times and increases accuracy by up to 5%. This paper is organized as follows. Section 2 introduces CPT. Sections 3 and 4 respectively describes the two compression strategies (FSC and SBC) and the prediction time reduction strategy (FNR). Section 5 presents an experimental evaluation of each strategy on seven real datasets against five state-of-the-art prediction models. Finally, Section 6 draws conclusion.

2 Compact Prediction Tree

The Compact Prediction Tree (CPT) is a recently proposed prediction model [5]. Its main distinctive characteristics w.r.t to other prediction models are that (1) CPT stores a compressed representation of training sequences with no loss or a small loss and (2) CPT measures the similarity of a sequence to the training sequences to perform a prediction. The similarity measure is noise tolerant and thus allows CPT to predict the next items of subsequences that have not been previously seen in training sequences, whereas other proposed models such as PPM and All-K-order-markov cannot perform prediction in such case.

The training process of CPT takes as input a set of training sequences and generates three distinct structures: (1) a Prediction Tree (PT), (2) a Lookup Table (LT) and (3) an Inverted Index. During training, sequences are considered one by one to incrementally build these three structures. For instance, Fig. 1 illustrates the creation of the three structures by the successive insertions of sequences $s_1 = \langle A, B, C \rangle$, $s_2 = \langle A, B \rangle$, $s_3 = \langle A, B, D, C \rangle$, $s_4 = \langle B, C \rangle$ and $s_5 = \langle E, A, B, A \rangle$, where the alphabet $Z = \{A, B, C, D, E\}$ is used. The *Prediction*

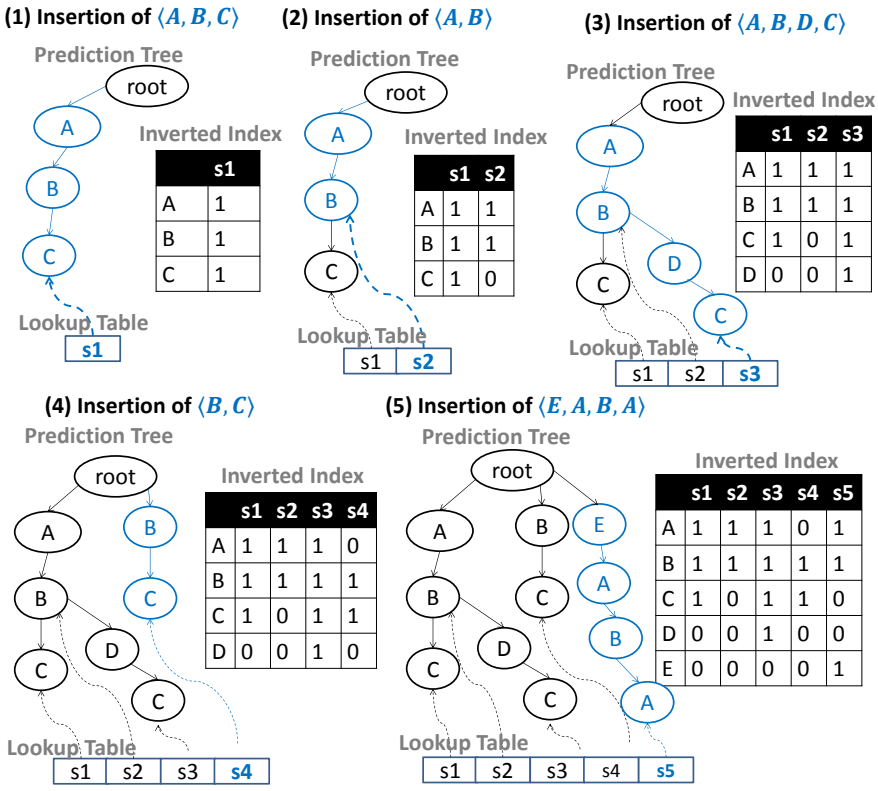

Fig. 1. Building CPT structures

Tree is a type of *prefix tree* (aka *trie*). It contains all training sequences. Each tree node represents an item and each training sequence is represented by a path starting from the tree root and ending by an inner node or a leaf. Just like a prefix tree, the prediction tree is a compact representation of the training sequences. Sequences sharing a common prefix share a common path in the tree. The *Lookup Table* is an associative array which allows to locate any training sequences in the prediction tree with a constant access time. Finally the *Inverted Index* is a set of bit vectors that indicates for each item i from the alphabet Z, the set of sequences containing i.

CPT's prediction process relies on the three aforementioned data structures. For a sequence $s = \langle i_1, i_2, ...i_n \rangle$ of n elements, the suffix of s of size y with $1 \leq y \leq n$ is defined as $P_y(s) = \langle i_{n-y+1}, i_{n-y+2} \ ...i_n \rangle$. Predicting the next items of s is performed by identifying the sequences similar to $P_y(s)$, that is the sequences containing all items in $P_y(s)$ in any order. The suffix length is a parameter similar to the model's order of All-k-order Markov and DG. Identifying the optimal value is done empirically by starting with a length of 1. CPT uses the *consequent* of each sequence similar to s to perform the prediction. Let $u = \langle j_1, j_2, ...j_m \rangle$ be a sequence similar to s. The *consequent* of u w.r.t to s is the longest subsequence $\langle j_v, j_{v+1}, ...j_m \rangle$ of u such that $\bigcup_{k=1}^{v-1} \{j_k\} \subseteq P_y(s)$ and $1 \leq v \leq m$. Each item found in the consequent of a similar sequence of s is stored in a data structure called *Count Table* (CT). The count table stores the support (frequency) of each of these items, which is an estimation of $P(e|P_y(s))$. CPT returns the most supported item(s) in the CT as its prediction(s).

The similarity measure in CPT is initially strict for each prediction task but is dynamically loosened to ignore noise. Identifying similar sequences, and more particularly the noise avoidance strategy of CPT, is very time consuming and account for most of CPT's prediction time [5]. For a given sequence, if CPT cannot find enough similar sequences to generate a prediction, it will implicitly assume the sequence contains some noise. The prediction process is then repeated but with one or more items omitted from the given sequence. CPT's definition of noise is implicit and has for sole purpose to ensure that a prediction can be made every time.

3 Compression Strategies

CPT has been presented as one of the most accurate sequence prediction model [5] but its high spatial complexity makes CPT unsuitable for applications where the number of sequences is very large. CPT's size is smaller than All-k-Order Markov and TDAG but a few orders of magnitude larger than popular models such as DG and PPM. CPT's prediction tree is the largest data structure and account for most of its spatial complexity. In this section, we focus on strategies to reduce the prediction tree's size.

Strategy 1 Frequent subsequence compression (FSC). In a set of training sequences, frequently occurring subsequences of items can be found. For some datasets, these subsequences can be highly frequent. The FSC strategy

identifies these frequent subsequences and replace each of them with a single item. Let be a sequence $s = \langle i_1, i_2, ..., i_n \rangle$. A sequence $c = \langle i_{m+1}, i_{m+2}, ..., i_{m+k} \rangle$ is a subsequence of s, denoted as $c \sqsubseteq s$, iff $1 \leq m \leq m + k \leq n$. For a set of training sequences S, a subsequence d is considered a *frequent subsequence* iff $|\{t|t \in S \wedge d \sqsubseteq t\}| \geq minsup$ for a minimum support threshold $minsup$ defined per dataset.

Frequent subsequences compression is done during the training phase and is performed in three steps: (1) identification of frequent subsequences in the training sequences, (2) generation of a new item in the alphabet Z of items for each frequent subsequence, and (3) replacement of each frequent subsequence by the corresponding new item when inserting training sequences in the prediction tree. Identifying frequent subsequence in a set of sequences is a known problem in data mining for which numerous approaches have been proposed. In FSC, we use the well known PrefixSpan [9] algorithm. PrefixSpan is one of the most efficient sequential pattern mining algorithm. It has been adapted by incorporating additional constraints to fit the problem of sequence prediction. Subsequences have to be contiguous, larger than a minimum length $minSize$ and shorter than a maximum length $maxSize$. Both $minSize$ and $maxSize$ are parameters of this compression strategy that are defined per application.

A new data structure, *Subsequence Dictionary* (DCF), is introduced to store the frequent subsequences. This dictionary associates each frequent subsequence with its corresponding item. The DCF offers a fast way to translate each subsequence into its respective item and vice-versa, O(1). When inserting training sequences into the prediction tree, the DCF is used to replace known frequent subsequences with single items. For example, figure 2 illustrates the resulting prediction tree after applying FSC to the tree shown in Fig. 1. The frequent subsequence $\langle A, B \rangle$ has been replaced by a new symbol x, thus reducing the number of nodes in the prediction tree. The FSC compression strategy influences the shape of the prediction tree by reducing its height and number of nodes. With respect to the prediction process, FSC only influences execution time. The additional cost is the on-the-fly decompression of the prediction tree, which is fast and non intrusive because of the DCF structure.

Strategy 2: Simple Branches Compression (SBC). Simple Branches Compression is an intuitive compression strategy that reduces the size of the prediction tree. A *simple branch* is a branch leading to a single leaf. Thus, each node of a simple branch has between 0 and 1 children. The SBC strategy consists of replacing each simple branch by a single node representing the whole branch. For instance, part (2) of Fig. 2 illustrates the prediction tree obtained by applying the DCF and SBC strategies for the running example. The SBC strategy has respectively replaced the simple branches D, C, B, C and E, x, A by single nodes DC, BC and ExA. Identifying and replacing simple branches is done by traversing the prediction tree from the leafs using the inverted index. Only the nodes with a single child are visited. Since the Inverted Index and Lookup Table are not affected by this strategy, the only change that needs to be

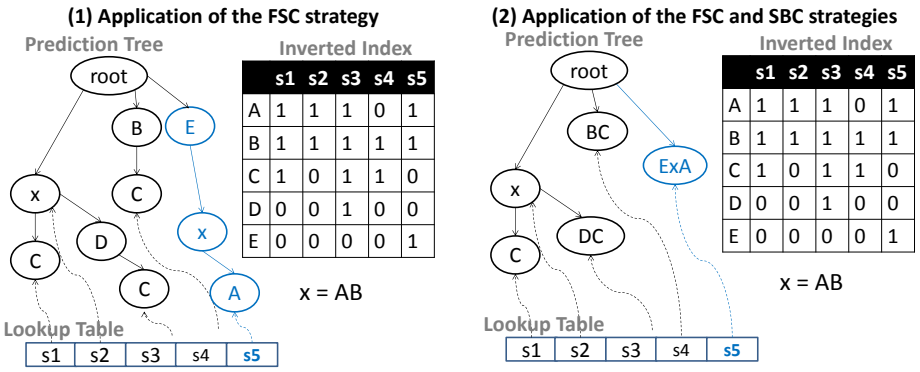

Fig. 2. Application of the FSC and SBC compression strategies

done to the prediction process is to dynamically uncompress nodes representing simple branches when needed.

4 Time Reduction Strategy

Strategy 3: Prediction with improved Noise Reduction (PNR). As previously explained, to predict the next item i_{n+1} of a sequence $s = \langle i_1, i_2, ..., i_n \rangle$, CPT uses the suffix of size y of s denoted as $P_y(s)$ (the last y items of s), where y is a parameter that need to be set for each dataset. CPT predicts the next item of s by traversing the sequences that are similar to its suffix $P_y(s)$. Searching for similar sequences is very fast ($O(y)$). However, the noise reduction mechanism used for prediction (described in Section 2) is not. The reason is that it considers not only $P_y(s)$ to perform a prediction, but also all subsequences of $P_y(s)$ having a size $t > k$, where k is a parameter. The more y and k are large, the more subsequences need to be considered, and thus the more the prediction time increases.

For a prediction task, items in a training sequence may be considered as noise if their sole presence negatively impact a prediction's outcome. The PNR strategy is based on the hypothesis that noise in training sequences consists of items having a low frequency, where an item's frequency is defined as the number of training sequences containing the item. For this reason, PNR removes only items having a low frequency during the prediction process. Because the definition of noise used in CPT+ is more restrictive than that of CPT, a smaller number of subsequences are considered. This reduction has a positive and measurable impact on the execution time, as it will be demonstrated in the experimental evaluation (see Section 5).

The PNR strategy (Algorithm 1) takes as parameter the prefix $P_y(s)$ of a sequence to be predicted s, CPT's structures and the noise ratio TB and a minimum number of updates, MBR, to be performed on the count table (CT) to perform a prediction. The noise ratio TB is defined as the percentage of items in

a sequence that should be considered as noise. For example, a noise ratio of 0 indicates that sequences do not contain noise, while a ratio of 0.2 means that 20% of items in a sequence are considered as noise. PNR is a recursive procedure. To perform a prediction, we require that PNR consider a minimum number of subsequences derived from $P_y(s)$. PNR first removes noise from each subsequence. Then, the CT is updated using these subsequences. When the mimimum number of updates is reached, a prediction is performed as in CPT using the CT. The PNR strategy is a generalization of the noise reduction strategy used by CPT. Depending on how the parameters are set, PNR can reproduce the behavior of CPT's noise reduction strategy. The three main contributions brought by PNR are to require a minimum number of updates on the CT to perform a prediction, and to define noise based on the frequency of items, and to define noise proportionally to a sequence length. Finding the appropriate values for both TB and MBR can be achieved empirically.

Algorithm 1: The prediction algorithm using PNR

```
input  : P_y(s): a sequence suffix, CPT: CPT's structures, TB: a noise ratio,
         MBR: minimum number of CT updates
output: x: the predicted item(s)
queue.add(P_y(s));
while updateCount < MBR ∧ queue.notEmpty() do
    suffix = queue.next();
    noisyItems = selectLeastFrequentItems(TB);
    foreach noisyItem ∈ noisyItems do
        suffixWithoutNoise = removeItemFromSuffix(suffix, noisyItem);
        if suffixWithoutNoise.length > 1 then
            queue.add(suffixWithoutNoise);
        end
        updateCountTable(CPT.CT, suffixWithoutNoise);
        updateCount++;
    end
    return performPrediction(CPT.CT);
end
```

5 Experimental Evaluation

We have performed several experiments to compare the performance of CPT+ against five state-of-the-art sequence prediction models: All-K-order Markov, DG, Lz78, PPM and TDAG. We picked these models to match the models used in the original paper describing CPT [5] and added both Lz78 and TDAG. To implement CPT+, we have obtained and modified the original source code of CPT [5]. To allow reproducing the experiments, the source code of the prediction

models and datasets are provided at `http://goo.gl/mL4Klx`. All models are implemented using Java 8. Experiments have been performed on a computer with a dual core 4th generation Intel i5 with 8 GB RAM and a SSD drive connected with SATA 600. For all prediction models, we have empirically attempted to set their parameters to optimal values. PPM and LZ78 do not have parameters. DG and AKOM have respectively a window of four and an order of five. To avoid consuming an excessive amount of memory, TDAG has a maximum depth of 7. CPT has two parameters, *splitLength* and *maxLevel* and CPT+ has six parameters; three for the FSC strategy, two for the PNR strategy and *splitLength* from CPT. The values of these parameters have also been empirically found and are provided in the project source code. Experiment specific parameters are the minimum and maximum length of sequences used, the number of items to be considered to perform a prediction (the suffix length) and the number of items used to verify the prediction (called the consequent length). Let be a sequence $s = \langle i_1, i_2, ...i_n \rangle$ having a suffix $S(s)$ and a consequent $C(s)$. Each model takes the suffix as input and outputs a predicted item p. A prediction is deemed successful if p is the first item of $C(s)$. Datasets having various characteristics have

Table 1. Datasets

Name	**Sequence count**	**Distinct item count**	**Average length**	**Type of data**
BMS	15,806	495	6.01	webpages
KOSARAK	638,811	39,998	11.64	webpages
FIFA	573,060	13,749	45.32	webpages
MSNBC	250,697	17	3.28	webpages
SIGN	730	267	93.00	language
BIBLE Word	42,436	76	18.93	sentences
BIBLE Char	32,502	75	128.35	characters

been used (see Table 1) such as short/long sequences, sparse/dense sequences, small/large alphabets and various types of data. The BMS, Kosarak, MSNB and FIFA datasets consist of sequences of webpages visited by users on a website. In this scenario, prediction models are applied to predict the next webpage that a user will visit. The SIGN dataset is a set of sentences in sign language transcribed from videos. Bible Word and Bible Char are two datasets originating from the Bible. The former is the set of sentences divided into words. The latter is the set of sentences divided into characters. In both datasets, a sentence represents a sequence.

To evaluate prediction models, a prediction can be either a *success* if the prediction is accurate, a *failure* if the prediction is inaccurate or a *no match* if the prediction model is unable to perform a prediction. Four performance measures are used in experiments: *Coverage* is the ratio of sequences without prediction against the total number of test sequences. *Accuracy* is the number of

successful predictions against the total number of test sequences. *Training time* is the time taken to build a model using the training sequences *Testing time* is the time taken to make a prediction for all test sequences using a model.

Experiment 1: Optimizations comparison. We have first evaluated strategies that aims to reduce the space complexity of CPT (cf. Section 3) by measuring the compression rate and the amount of time spent for training. Other measures such as prediction time, coverage and accuracy are not influenced by the compression. For a prediction tree A having s nodes before compression and $s2$ nodes after compression, the compression rate tc_a of A is defined as $tc = 1 - (s2/s)$, a real number in the [0,1[interval. A larger value means a higher compression. The two strategies are first evaluated separately (denoted as FSC and SBC) and then together (denoted as CPT+). Compression provides a spatial gain but increases execution time. Figure 3 illustrates this relationship for each compression strategy.

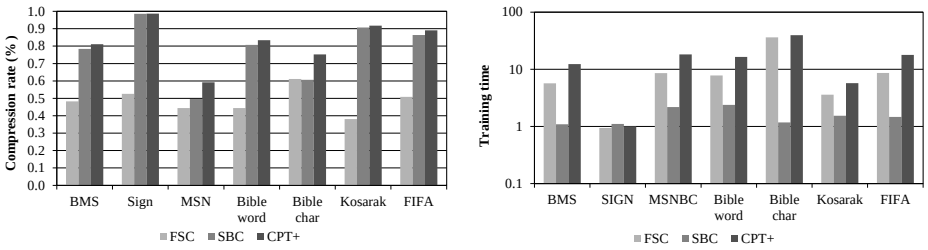

Fig. 3. Compression rate and training time of the compression strategies

It can also be observed in Fig. 3 (left) that the compression rate varies depending on the dataset from 58.90% to 98.65%. FSC provides an average compression rate of 48.55% with a small standard deviation (6.7%) and SBC offers an average compression rate of 77.87% with a much larger standard deviation (15.9%). For each tested dataset, SBS has a compression rate very similar to CPT+ with the exception of MSNBC and Bible Char. It accounts for most of CPT+ compression rate and thus making FSC relevant only in applications requiring a smaller model. MSNBC is the least affected by the compression strategies. The reason is that MSNBC has a very small alphabet and thus that the tree is naturally compressed because its branches are highly overlapping. In fact, MSNBC has only 17 distinct items, although the length of its sequences is similar to the other datasets.The dataset where the SBC and SFC strategies provide the highest compression rate is SIGN. Even though SIGN contains a small number of sequences, each sequence is very long (an average of 93 items). It causes the branches of SIGN's prediction tree to rarely overlap, and a large amount of its nodes only have a single child. SIGN is a very good candidate for applying the SBC strategy. Using only SBC, a compression rate of 98.60 % is attained for SIGN.

Figure 3 also illustrates the training time for the FSC and SBC strategies. It is measured as a multiplicative factor of CPT's training time. For example, a factor x for SBC means that the training time using SBC was x times longer than that of CPT without SBC. A small factor, close to one, means the additional training time was small. It is interesting to observe how the training time when both strategies are combined is less than the sum of their respective training time. Althought SBC and FSC are independently applied to CPT, SBC reduces the execution time of FSC because less branches have to be compressed. Overall, SBC has a small impact on the training time while providing most of the compression gain, this makes SBS the most profitable strategy. While SFC has a higher impact on the training time, SFC enhances the compression rate for each dataset.

We also evaluated the performance improvement in terms of execution time and accuracy obtained by applying the PNR strategy. Fig. 4 (left) compares the prediction time of CPT+ (with PNR) with that of CPT. The execution time is reduced for most datasets and is up to 4.5 times smaller for SIGN and MSNBC. For Bible Word and FIFA, prediction times have increased but a higher accuracy is obtained, as shown in Fig. 4 (right). The gain in prediction time for CPT+ is dependant on both PNR and CPT parameters. This gain is thus dataset specific and non linear because of the difference in complexity of CPT and CPT+. The influence of PNR on accuracy is positive for all datasets except MSNBC. For Bible Word, the improvement is as high as 5.47%. PNR is thus a very effective strategy to reduce the prediction time while providing an increase in prediction accuracy.

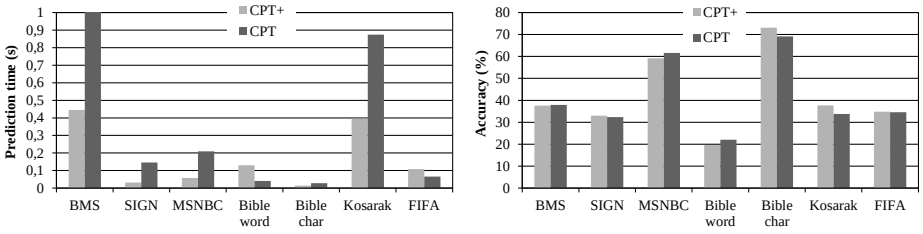

Fig. 4. Execution time and accuracy gains provided by the PNR strategy

Experiment 2: Scalability. In this experiment we compared the spatial complexity of CPT+ (with both compression strategies) against CPT, All-K-order Markov, DG, Lz78, PPM and TDAG. Only the FIFA and Kosarak datasets were used in this experiment because of their large number of sequences. In Fig. 5, the spatial size of each model is evaluated against a quadratically growing set of training sequences - up to 128,000 sequences. Both PPM and DG have a sub linear growth which makes them suitable for large datasets. CPT+'s growth is only an order of magnitude larger than PPM and DG and a few orders less than CPT, TDAG and LZ78. The compression rate of CPT+ tends to slightly dimin-

ish as the number of sequences grows. This is due to more branches overlapping in the prediction tree, a phenomenon that can generally be observed in tries.

An interesting observation is that for both datasets, when the number of training sequences is smaller than the number of items, CPT+ has a smaller footprint than the other prediction models. For the FIFA dataset, when 1000 sequences are used, CPT+'s node count is 901 compared to the 1847 unique items in the alphabet. Results are similar for Kosarak. Models such as PPM and DG can't achieve such a small footprint in these use cases because they have a least one node per unique item.

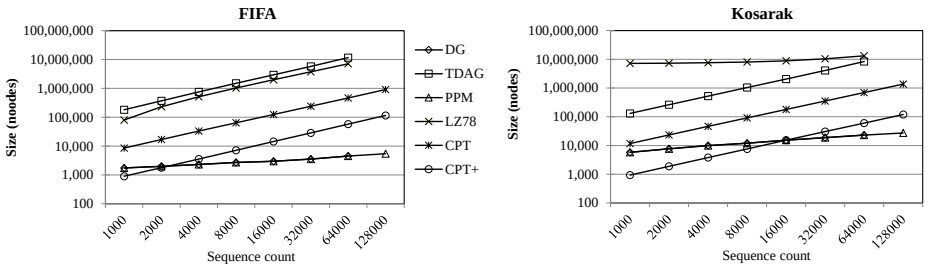

Fig. 5. Scalability of the prediction models.

Experiment 3: Comparison with other prediction models. In experiment 1, we have compared the prediction accuracy of CPT+ and CPT to assess the improvement obtained by applying the PNR strategy. In this experiment, we compare the accuracy of CPT+ with five state-of-the-art prediction models commonly used in the literature, All-K-order Markov, DG, Lz78, PPM et TDAG, on the same datasets. Each prediction model has been trained and tested using k-fold cross validation with $k = 14$ to obtain a low variance for each run. Table 2 shows the prediction accuracy obtained by each model. Results indicates that CPT+ offers a generally higher accuracy than the compared models from the literature while also being more consistent across the various datasets.

Table 2. Predictions models and their accuracy

Datasets	**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	23.33
Bible char	**73.52**	69.14	7.96	0.00	3.02	0.10	9.90
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

6 Conclusion

In this paper we presented CPT+, a variation of CPT that includes two novel compression strategies (FSC and SBC) to reduce its size and a strategy to improve prediction time and accuracy (PNR). Experimental results on seven real datasets shows that CPT+ is up to 98 times smaller than CPT, performs predictions up to 4.5 times faster, and is up to 5% more accurate. A comparison with six state-of-the art sequence prediction models (CPT, All-K-Order Markov, DG, Lz78, PPM and TDAG) shows that CPT+ is on overall the most accurate model. To allow reproducing the experiments, the source code of the prediction models and datasets are provided at `http://goo.gl/mL4Klx`.

Acknowledgement This project was supported by a NSERC grant from the Government of Canada.

References

1. Begleiter, R., El-yaniv, R., Yona., G.: On prediction using variable order markov models. Journal of Artificial Intelligence Research, vol. 22, pp. 385-421 (2004)
2. Cleary, J., Witten, I.: Data compression using adaptive coding and partial string matching. IEEE Trans. on Inform. Theory, vol. 24, no. 4, pp. 413–421 (1984)
3. Deshpande, M., Karypis G.: Selective markov models for predicting web page accesses. ACM Transactions on Internet Technology, vol. 4, no. 2, pp. 163–184 (2004) `https://developers.google.com/prediction`, Accessed: 2014-02-15
4. Gopalratnam, K., Cook, D.J.: Online sequential prediction via incremental parsing: The active lezi algorithm. IEEE Intelligent Systems, vol. 22, no. 1, pp. 52-58 (2007)d
5. Gueniche, T., Fournier-Viger, P., Tseng, V.-S.: Compact Prediction Tree: A Lossless Model for Accurate Sequence Prediction. In: Proc. 9th Intern. Conf. Advanced Data Mining and Application, Springer LNAI 8347, pp. 177–188 (2013)
6. Fournier-Viger, P., Gueniche, T., Tseng, V.S.: Using partially-ordered sequential rules to generate more accurate sequence prediction. In Proc. 8th Intern. Conf. Advanced Data Mining and Applications, Springer LNAI 7713, pp. 431442. (2012)
7. Laird, P., Saul, R.: Discrete sequence prediction and its applications. Machine learning, vol. 15, no. 1, 43-68 (1994)
8. Padmanabhan, V.N., Mogul, J.C.: Using Prefetching to Improve World Wide Web Latency, Computer Communications, vol. 16, pp. 358–368 (1998)
9. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Mining sequential patterns by pattern-growth: the PrefixSpan approach. IEEE Trans. Known. Data Engin. 16(11), 1424–1440 (2004)
10. Pitkow, J., Pirolli, P.: Mining longest repeating subsequence to predict world wide web surng. In: Proc. 2nd USENIX Symposium on Internet Technologies and Systems, Boulder, CO, pp. 13–25 (1999)
11. Sun, R., Giles, C. L.: Sequence Learning: From Recognition and Prediction to Sequential Decision Making. IEEE Intelligent Systems, vol. 16 no. 4, pp. 67–70 (2001)
12. Ziv, J., Lempel, A.: Compression of individual sequences via variable-rate coding. Information Theory, IEEE Transactions on 24(5), 530-536 (1978)