Unordered N-gram Representation Based on Zero-suppressed BDDs for Text Mining and Classification

Ryutaro Kurai, Shin-ichi Minato, and Thomas Zeugmann

Division of Computer Science
Hokkaido University, N-14, W-9, Sapporo 060-0814, Japan
{ryu,minato,thomas}@mx-alg.ist.hokudai.ac.jp

Abstract. In this paper, we present a new method to analyze unordered n-grams by using ZBDDs (Zero-suppressed BDDs). n-grams have been used not only for text analysis but also for text indexing in some search engines. We newly use a variation of n-grams called unordered n-grams. Unordered n-grams abstract from the position of the characters in each n-gram, i.e., they just deal with the *range* of ordinary n-grams or, synonymously, they reduce ordinary n-grams to sets of symbols.
Furthermore, ZBDDs are known as a compact representation for combinatorial item sets. We newly apply ZBDD-based techniques to handle efficiently unordered n-grams. We conducted experiments for generating unordered n-gram statistical data for given real document files. The results obtained show the potentiality of our new methods.

1 Introduction

One of the important data models that has been used for text analysis are n-grams (cf., e.g., [2], [7], [8], [9]). Recently, n-grams have been used not only for text analysis but also for text indexing in some search engines [1]. If we can compactly represent n-gram data and efficiently manipulate them, it will greatly facilitate text database analysis and machine learning applications. In this paper, we propose a method of unordered n-gram computation by using an item set data structure based on Zero-suppressed BDDs. BDDs (Binary Decision Diagrams) are graph-based representations of Boolean functions, now widely used in system design and verification. Zero-suppressed BDDs (ZBDDs) are a special type of BDDs that are suitable for handling large-scale sets of combinations. Using ZBDDs, we can implicitly enumerate combinatorial item set data and efficiently compute set operations over the ZBDDs. For n-gram computations, we need to manipulate sets of "sequences," which is a more complicated data model than sets of "combinations." In the present paper, we present a method of manipulating sets of sequences using unordered n-grams. Unordered n-grams are a special type of n-grams which abstract from the position of the characters in each n-gram. Thus, they reduce ordinary n-grams to sets of symbols. We have implemented a prototype system and show experimental results to evaluate our new unordered n-gram analysis method.

2 ZBDDs and Item Set Manipulation

Within this section, we briefly describe the basic techniques of ZBDDs for representing efficiently sets of combinations.

2.1 Sets of Combinations and their Representation

A *set of combinations* consists of elements each of which is a combination of a number of items. There are 2^n combinations that can be chosen from n items, so we have 2^{2^n} variations of sets of combinations. For example, for a domain of five items a, b, c, d, e, we can express examples of sets of combinations as: $\{ab, e\}$, $\{abc, cde, bd, acde, e\}$, $\{1, cd\}$, $\emptyset$. Here "1" denotes a combination of null items, and "$\emptyset$" means the empty set. Sets of combinations are one of the basic data structures for handling combinatorial problems. They often appear in real-life problems, such as combinations of switching devices, sets of faults, paths in the networks, etc. Of course, sets of combinations can be used for representing frequent item set data, too.

A set of combinations over n items can be mapped into the Boolean space of n input variables. For example, Fig. 1 shows the truth table of the Boolean function $(ab + \overline{a}c)$, but it also represents the set

a	b	c	F	
0	0	0	0	
0	0	1	1	$\rightarrow c$
0	1	0	0	
0	1	1	1	$\rightarrow bc$
1	0	0	0	
1	0	1	0	
1	1	0	1	$\rightarrow ab$
1	1	1	1	$\rightarrow abc$

As a Boolean function:
$F = ab + \overline{a}c$

As a set of combinations:
$F = \{ab,\ abc,\ bc,\ c\}$

Fig. 1. Correspondence of Boolean functions and sets of combinations.

of combinations $\{ab, abc, bc, c\}$. Such Boolean functions are called *characteristic functions* for the sets of combinations. Using BDD manipulation for characteristic functions, we can implicitly represent and manipulate large-scale sets of combinations. In addition, we can enjoy more efficient manipulation using "Zero-suppressed BDDs" (ZBDD) (cf. [4]), which are a special type of BDDs optimized for handling sets of combinations.

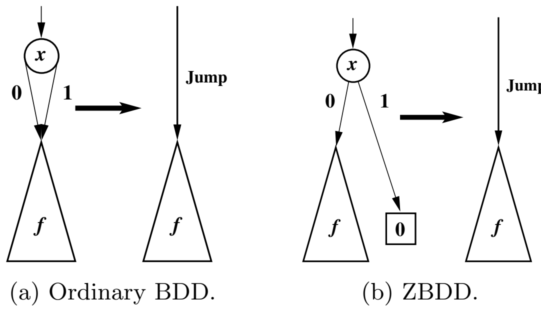

Fig. 2. Different reduction rules for BDD and ZBDDs.

ZBDDs are based on the reduction rule different from the one used in ordinary BDDs. As illustrated in Fig. 2(a), the ordinary reduction rule deletes the nodes whose two edges point to the same node. However, in ZBDDs, we do not delete such nodes but delete another type of nodes whose 1-edge directly points to the 0-terminal node, as shown in Fig. 2(b).

In ZBDDs, a 0-edge points to the subset (cofactor) of combinations not including the decision variable x, and a 1-edge points to the subset (cofactor) of combinations including x. If the 1-edge directly points to the 0-terminal node, it means that the item x never appears in the set of combinations. The Zero-suppressed reduction rule automatically deletes such a node with respect to the irrelevant item x, and thus ZBDDs more compactly represent sets of combinations than ordinary BDDs do.

The detailed techniques of ZBDD manipulation are described in the articles [4, 5]. A typical ZBDD package supports cofactoring operations to traverse 0-edge or 1-edge, and binary operations between two sets of combinations, such as union, intersection, and difference. The computation time for each operation is almost linear in the number of ZBDD nodes related to the operation.

2.2 Item Set Histograms and ZBDD Vectors

A *item set histogram* is the table for counting the number of appearances of each item combination in the given database. An example of an item set histogram is shown in Fig. 3. This is just a compressed table of the database to combine the tuples appearing more than once into one line accompanied with their frequency.

Our n-gram data structure is based on the item set histogram representation [6] using ZBDDs. Since ZBDDs are representation of sets of combinations, a simple ZBDD distinguishes only existence of each combination in the database. In order to represent the numbers of combination's appearances, we decompose the number into m-digits of ZBDD vector $\{F_0, F_1, \ldots, F_{m-1}\}$ to represent integers up to $(2^m - 1)$, as shown in Fig. 4. Namely, we encode the appearance numbers into binary digital code. Thus

Record ID	Comb.
1	abc
2	ab
3	abc
4	bc
5	ab
6	abc
7	c
8	abc
9	abc
10	ab
11	bc

Original Database

Comb.	Freq.
abc	5
ab	3
bc	2
c	1

Item set histogram

Fig. 3. Database example and item set histogram.

Comb.	frequency	F_2	F_1	F_0
abc	5 (101)	1	0	1
ab	3 (011)	0	1	1
bc	2 (010)	0	1	0
c	1 (001)	0	0	1

$F_0 = \{abc,\ ab,\ c\}$
$F_1 = \{ab,\ bc\}$, $F_2 = \{abc\}$

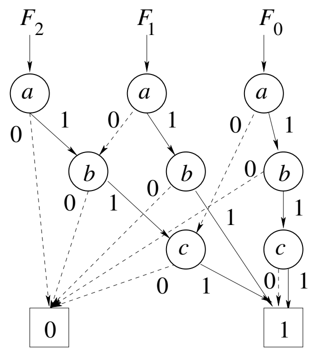

Fig. 4. ZBDD vector for item set histogram.

in Fig. 4, F_0 represents the set of tuples appearing an odd number of times (LSB = 1), F_1 represents the set of combinations whose appearance numbers have a 1 in the second lowest bit, and similarly we define the set of each digit up to F_{m-1}.

In the example of Fig. 4, the item set frequencies are decomposed as: $F_0 = \{abc, ab, c\}$, $F_1 = \{ab, bc\}$, $F_2 = \{abc\}$, and then each digit can be represented by a simple ZBDD. The three ZBDDs share their sub-graphs to one another.

When we construct a ZBDD vector of an item set histogram, the number of ZBDD nodes in each digit is bounded by the total appearance of items in all combinations. If there are many partially similar combinations in the database, the sub-graphs of ZBDDs are shared very well, and a compact representation is obtained. The bit-width of a ZBDD vector is bounded by $\log S_{max}$, where S_{max} is the number of appearances of most frequent items.

Once we have generated a ZBDD vector for the item set histogram, various operations can be executed efficiently. Here are the instances of operations used in our pattern mining algorithm.

– H.factor0(v): Extracts sub-histogram of combinations not including item v.
– H.factor1(v): Extracts sub-histogram of combinations including item v and then delete v from the combinations. (also considered as the quotient of H/v)
– $v \cdot H$: Attaches an item v on each combination in the histogram F.
– $H_1 + H_2$: Generates a new item set histogram with sum of the frequencies of corresponding tuples.
– H.tuplecount: The number of tuples appearing at least once.

These operations can be composed as a sequence of ZBDD operations. The result is also compactly represented by a ZBDD vector. The computation time is roughly linear in the total ZBDD size. The detailed techniques of ZBDD vector manipulation are described in the articles [6].

3 ZBDD-Based Representation for Unordered n-grams

Next we explain, in some more detail, our method of generating unordered n-gram data representations using ZBDDs.

3.1 Sets of Sequences and Sets of Combinations

As the method of the representation of ordinary n-gram data using ZBDDs, we have already proposed a way to extend the representation of sets of combinations to sets of sequences [11]. In our method, we define an item for each *symbol* used in each position of the corresponding sequence. The set of *Symbols* is determined by the set of all characters appearing in the preprocessed[1] text data. For example, in English then the set of symbols is {a, b, c, ..., z}. In general, we use Σ to denote the set of symbols and $|\Sigma|$ to denote the cardinality of Σ. Ordinary n-grams are sequences of symbols having length n, e.g., "aabc" and "cbaa" are 4-grams. For representing such sequences as combinations we define *items* to be ordered pairs of symbols and natural numbers. The symbol corresponds to the character and the number to the position of its appearance. So, when looking at ordinary n-grams, we have for each character precisely n items. Thus we get a total of $|\Sigma| \cdot n$ many items. For the sake of better readability, we denote the items by a_i, where $a \in \Sigma$ and $i \in \{1, \ldots, n\}$. Thus, any n-gram can be represented as a combination of items, e.g., "aabc" is represented as "$a_1a_2b_3c_4$" and "cbaa" as "$c_1b_2a_3a_4$." So, the item a_1 represents the symbol "a" at the first (leftmost) position. Similarly, c_4 express the symbol "c" at the 4th position. The resulting combinations of items can be treated by ZBDDs. Figure 5 shows an example of a ZBDD representing the set of sequences {aabc, cabc} as sets of combinations. These sets of combinations can be manipulated by using ZBDD-techniques.

Sequences	Combinations
aabc	$a_1a_2b_3c_4$
cabc	$c_1a_2b_3c_4$

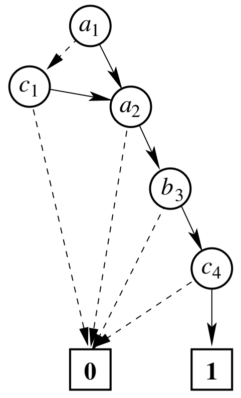

Fig. 5. ZBDDs example which expresses sets of sequences.

However, in this method, the number of items can quickly become too large. Since the number of items corresponds to the height of the ZBDDs, we may need too much time and space. In particular, when dealing with a 2 bytes language such as Japanese, the number of symbols is huge, and thus transforming symbols to items may be counterproductive.

Therefore, we propose to deal with unordered n-grams instead of ordinary n-grams to maintain the benefits of our method.

3.2 Unordered n-grams

An n-gram data is a histogram of all possible subsequences of length n included in a given sequence. An unordered n-gram data also has a histogram of all possible subsequences of the length n. But, the unordered n-grams abstract from the position of the characters or words, i.e., they just deal with the range of ordinary n-grams. Here, for any finite sequence $s_1, \ldots, s_n$ we define $range(s_1, \ldots, s_n) = \{s_1, \ldots, s_n\}$.

For example, if we consider the ordinary 4-grams of the string "abaabaa," we get as first and second 4-gram the sequence "abaa" and "baab," respectively. On the other hand, the unordered first and second

[1] Our preprocessing consists in replacing upper case characters by lower case characters, by removing all punctuation symbols and all spaces.

4-grams are just *range*(*abaa*) and *range*(*baab*), respectively, i.e., they coincide and equal to the set $\{a, b\}$. However, we record them simply as "ab."

Note that Nahnsen *et al.* [10] use a related though different approach. That is, they study *unsorted* n-grams, i.e., they ignore the order of elements in an n-gram but maintain the length n and the number of repetitions of the elements in an n-gram. As they have shown, unsorted n-grams can be quite useful in text analysis (cf. [10]).

3.3 ZBDDs Based Representation for Unordered *n*-grams

Next, we propose a method to represents unordered n-gram data by ZBDDs. As explained in Subsection 3.2, we abstract from the position of the characters in each unordered n-gram. So, now one character corresponds to precisely one item. For the sake of illustration, an example is shown in the table below. The first column shows ordinary 4-grams, the second one the corresponding unordered 4-grams, and the third one the combinations of items used for ZBDD representation. Fig. 6 displays the resulting ZBDD.

Sequences	Unordered 4-grams	Combinations
aabc	abc	*abc*
cabc	abc	*abc*
abcd	abcd	*abcd*
aacd	acd	*acd*

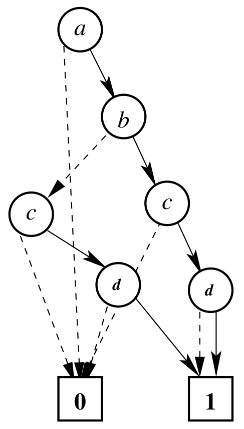

Fig. 6. ZBDDs example which expresses unordered sequences.

Consequently, dealing with unordered n-grams reduces the number of items at least by the factor $1/n$ compared to the method dealing with ordinary n-grams. Now, the hope is that this reduction will also imply a considerable reduction in the resulting ZBDD size.

This effect will be even more remarkable, if we apply our new method to "word based" n-grams, since the number of different words appearing in a text is usually much larger than the number of characters used.

On the other hand, when dealing with unordered n-grams, then the information concerning the position of items is lost. So, the problem arises how much such a loss is affecting classification or related text mining tasks.

In order to get a first impression, we performed experiments to see how much the ZBDDs size will decrease when switching from ordinary n-grams to unordered n-grams. Additionally, we analyzed to what extend the most frequent n-grams will change.

4 Experimental Results

4.1 Generating unordered *n*-gram Data Based on ZBDDs

For evaluating the feasibility of our method, we executed experiments for generating ZBDDs of the unordered n-grams for a given text data. We used the English plain text data "Alice in Wonderland."

Table 1. Experimental results

	Ordered n-gram				Unordered n-gram			
n	ZBDD nodes	#Sequences	Max frequency	Time(sec)	ZBDD nodes	#Sequences	Max frequency	Time(sec)
1	168	26	13,681	2.57	168	26	13,681	2.54
2	1,886	518	3,813	5.83	982	306	4,162	4.43
3	8,723	4,750	2,402	8.56	2,956	1,607	3,020	5.44
4	23,607	19,004	500	10.08	6,653	5,241	931	6.24
5	44,207	38,913	417	10.70	12,527	12,107	546	6.66
6	66,420	57,510	221	11.04	19,263	20,941	326	7.02
7	89,726	72,285	211	11.71	25,601	29,539	338	7.31
8	115,634	82,891	116	12.11	30,920	36,803	323	7.64
9	145,518	90,247	116	13.26	35,334	42,163	265	8.05
10	181,158	95,366	58	14.02	39,200	46,000	223	8.51

This text consists of over 3,000 lines and 26,000 words having a total of 138KB of plain text. In our experiments, we used a 3.00GHz Pentium 4 Linux PC with 1GB main memory.

The results of generating unordered n-grams for $n = 1, \ldots, 10$ are shown in Table 1. In this table, the column "ZBDD nodes" shows the total number of nodes in the ZBDD vector representing the n-gram. "#Sequences" gives the total number of different sequences contained in the n-gram. "Max frequency" shows the number of appearances of the most popular subsequences in the text. "Time" is the generating time of the data structure ZBDD. The ZBDD structure is generated by VSOP (see [6]), the Script Language which can manipulate large scale ZBDDs.

In this experiment, we used the methods described in Subsection 3.1 to store ordinary n-grams into ZBDDs. Furthermore, we used the methods explained in Subsection 3.2 to manipulate unordered n-grams.

From the result, we conjecture that the size of ZBDDs is almost proportional to the product of the length n and the number of sequences. The CPU time is also almost linear in the size of the ZBDD. Also, we clearly see that the resulting number of ZBDD nodes for unordered n-grams is considerably reduced when compared to the number of ZBDD nodes used for ordinary n-grams. In particular, in the case of large n, this advantage considerably improves the efficiency. It is very important that we could decrease the number of ZBDD nodes. This let us expect to be able to treat much larger texts in the future.

After generating the ZBDD vector of the ordinary and unordered n-grams, we can easily retrieve the frequent sequences by using algebraic operations over ZBDDs. The additional computation cost is relatively smaller than the cost for generating the n-gram data. Table 2 shows the top 10 frequent sequences in the 5-gram data of "Alice in Wonderland." The left columns show ordinary n-gram data. According to the data, the most frequent sequence is "alice," and it appears 417 times. The 2nd, 3rd, 4th, and 10th sequences seem to be part of the same sequence "he/she said the.". In contrast, the unordered n-gram data that are presented in left columns, just contain "aceil," i.e., the range of "alice" (see the 2nd row). And the 5th, 6th and 7th sequence seem to include the definite article "the." As described above, using the unordered n-gram method, we just get the range of words and so it may be difficult to find the corresponding words. On the other hand, in text clustering, sometimes unsorted n-gram data are effectively used (see [10]). So, we plan to perform experiments in the domain of text clustering, too, in order to see whether or not unordered n-gram data still as useful as unsorted n-gram data.

In this experiment, we removed spaces, special symbols such as question mark, and periods from the text. We only used case-independent alphabets and numbers to create n-grams.

Also note that there are some 4 letter sequences in the table below though Table 2 displays the 5-grams data. This is clearly caused by our representation of the range as (unordered) sequences.

If we use multisets to store the unordered n-grams, the 4 letter cases would have not appeared. So, using multisets allows us to handle unsorted n-grams, too.

But perhaps, there are no correlations between the easiness of human reading and the availability to extract useful features automatically. Future experiments should shed more light on this issue.

5 Conclusions

In the present paper, we have proposed a new method to represent unordered n-grams by using ZBDDs. In order to demonstrate the efficiency of this method, we have performed a preliminary experiment of

Table 2. Top 10 subsequences in the 5-gram of "Alice in Wonderland."

	Ordered n-gram		Unordered n-gram	
Rank	Sequence	Appearance	Characters	Appearance
1	"alice"	417	"ehrt"	546
2	"saidt"	265	"aceil"	534
3	"aidth"	223	"aeht"	489
4	"idthe"	220	"adist"	368
5	"thing"	170	"ehst"	361
6	"andth"	169	"ehot"	346
7	"dalic"	163	"ehnt"	318
8	"ofthe"	156	"ghint"	318
8	"ndthe"	156	"ahst"	286
10	"esaid"	133	"aehrt"	281

generating unordered n-gram data sets for a given text file. The experimental results show that we can construct the ZBDD-based unordered n-gram data structure by using a feasible amount of time and space.

Future work will compare the performance of text clustering between ordinary n-grams and unordered n-grams.

References

1. B. Hoffmeister and T. Zeugmann: Text Mining Using Markov Chains of Variable Length, In (K.P. Jantke, A. Lunzer, N. Spyratos, Y. Tanaka, eds), *Federation over the Web, International Workshop, Dagstuhl Castle, Germany, May 2005, Revised Selected Papers*, Lecture Notes in Artificial Intelligence Vol. 3847, pp. 1–24, Springer, Feb. 2006.
2. P. Jokinen and E. Ukkonen: Two algorithms for approximate string matching in static texts, In *Mathematical Foundations of Computer Science 1991, 16th International Symposium, MFCS'91, Kazimierz Dolny, Poland, September 9-13, 1991, Proceedings*, Lecture Notes in Computer Science Vol. 520, 240-248, Springer-Verlag 1991.
3. T. Kudo, K. Yamamoto, Y. Tsuboi, and Y. Matsumoto: Text mining using linguistic information, IPSJ SIG-NLP NL-148, (in Japanese), pp. 65–72, 2002.
4. S. Minato: Zero-suppressed BDDs for set manipulation in combinatorial problems, In Proc. 30th Design Automation Conference (DAC-93), pp. 272–277, ACM Press, June 1993.
5. S. Minato: Zero-suppressed BDDs and their applications, International Journal on Software Tools for Technology Transfer (STTT), 3(2):156–170, Springer, May 2001.
6. S. Minato: VSOP (Valued-Sum-of-Products) Calculator for Knowledge Processing Based on Zero-Suppressed BDDs, In (K.P. Jantke, A. Lunzer, N. Spyratos, Y. Tanaka, eds), *Federation over the Web, International Workshop, Dagstuhl Castle, Germany, May 2005, Revised Selected Papers*, Lecture Notes in Artificial Intelligence Vol. 3847, pp. 40-58, Springer, Feb. 2006.
7. M. Nagano and S. Mori: A new method of N-gram statistics for large number of n and automatic extraction of words and phrases from large text data of Japanese, In *Proc. 15th Conference on Computational Linguistics*, Vol. 1, pp. 611–615, Association for Computational Linguistics, Morristown, NJ, USA, 1994.
8. Y. Tsuboi: Mining frequent substrings, Technical Report of IEICE, NLC2003-47, 2003 (in Japanese).
9. E. Ukkonen: Approximate string-matching with q-grams and maximal matches, *Theoretical Computer Science*, 92(1):191–211, 1992.
10. T. Nahnsen, O. Uzuner, and B. Katz: Lexical Chains and Sliding Locality Windows in Content-based Text Similarity Detection, Computer Science and Artificial Intelligence Laboratory Technical Report, MIT-CSAIL-TR-2005-034,AIM-2005-017, May 19, 2005
11. R. Kurai, S. Minato, and T. Zeugmann: N-gram Analysis Based on Zero-suppressed BDDs, In Proc. of JSAI 4th Workshop on Learning with Logics and Logics for Learning (LLLL-2006), pp. 61-67, June 2006.