F-013 ゼロサプレス型BDDを用いた無順序 n-gram 表現法の性能評価

Evaluation of Unordered N-gram Representation Based on Zero-suppressed BDDs

倉井 龍太郎 † **湊 真一** † **ツォイクマン トーマス** †
Ryutaro Kurai Shin-ichi Minato Thomas Zeugmann

1. はじめに

近年，Web の発達に伴う電子化されたドキュメントの増加により，n-gram データモデルを用いた検索エンジンやテキストの特徴抽出が広く行われている．我々は n-gram データモデルの亜種である，無順序 (Unordered)n-gram モデルを用いて n-gram のコンパクトな表現について考察を行ってきた．本稿では，VLSI CAD の分野で大規模論理関数データの表現方法として広く用いられている二分化決定グラフ（BDD：Binary Decision Diagrams），その中でも「ゼロサプレス型 BDD（ZBDD：Zero-suppressed BDD），と呼ばれるデータ構造を用いて，n-gram データモデルの生成と演算について実験と考察を行った．

2. ZBDD による n-gram 表現

2.1 BDD

BDD は，図 1 に示すような論理関数のグラフによる表現である．一般に，論理関数のそれぞれの変数について，0，1 の値を代入した結果を，二分岐の枝（0-枝/1-枝）で場合分けし得られる論理関数の値を，2 値の定数節点（0-終端節点/1-終端節点）で表現すると，図 1 のような二分木状のグラフになる．このとき，場合分けする変数の順序を固定し，冗長な節点を削除と等価な節点の共有という 2 つの縮約規則を可能な限り適用することにより,「規約」な形が得られ，論理関数をコンパクト且つ一意に表せることが知られている．複数の論理関数を表す BDD の間においても，変数順序を固定すればグラフを共有することが可能である．

BDD は，多くの実用的な論理関数を比較的少ない記憶量で一意に表現することができる．また，2 つの BDD を入力とし，それらの二項論理演算の結果を表す BDD を直接生成するアルゴリズムが考案されている．このアルゴリズムはハッシュテーブルを巧みに用いることで，データが計算機の主記憶に収まる限りは，その記憶量にほぼ比例する時間内で論理演算を効率よく実行できる．

2.2 ZBDD

BDD は元々は論理関数を表現するために考案されたものだが，これを用いて組み合わせ集合データを表現・操作することも出来る．組合せ集合とは,「n 個のアイテムから任意個を選ぶ組み合わせ」を要素とする集合である．これを BDD で表現するとき，類似する組合せが多ければ，部分的に共通する組合せがグラフ上で共有されて，記憶量や計算時間が大幅に削減される場合がある．さらに，組合せ集合に特化した「ゼロサプレス型 BDD（ZBDD）」を用いると，より簡潔な表現が得られ，一層効率よく扱うことが出来る．

†北海道大学大学院情報科学研究科

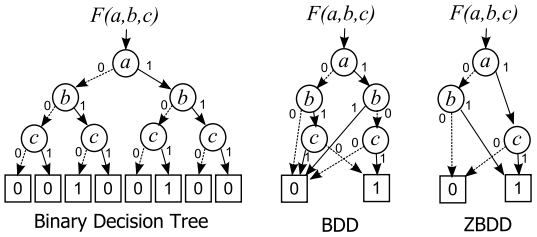

図 1: 二分決定木と BDD，ZBDD

文字列	アイテム組合せ
aabc	$a_1a_2b_3c_4$
cabc	$c_1a_2b_3c_4$

表 1: アイテム組合せによる n-gram 表現

ZBDD では，冗長な節点を削除する簡約化規則が通常の BDD と異なり，1-枝が 0-終端節点を直接指している節点を取り除く，という規則になっている．これにより ZBDD では図 1 のように，組合せ集合に一度も選ばれることのないアイテムに関する節点が自動的に削除されることになり，BDD よりも効率よく組合せ集合を表現・操作することが出来る．

2.3 n-gram の ZBDD 表現

n-gram は，与えられた文章に含まれる，長さ n のすべての部分文字列の出現頻度を数え上げるものである．一方で ZBDD はアイテムの組合せ集合を表現するのに適したデータ構造であるので，n-gram を ZBDD によって表現するためには，文字列をアイテムの組合せで表現しなければならない．我々はこの問題に対していくつかの手法を提案しており [2]，ここでは文字の種類と部分文字列中での出現場所を 1 つの組にしてアイテムとして表現する方法と無順序 (Unordered) n-gram を利用する方法を提示する．Unordered n-gram とは，n-gram に含まれる部分文字列の文字の出現位置の情報を破棄し，部分文字列を文字の集合ととらえるデータモデルである．

文字列	Unordered 4-grams	アイテム組合せ
aabc	abc	*abc*
cabc	abc	*abc*
abcd	abcd	*abcd*
aacd	acd	*acd*

表 2: アイテム組合せによる Unordered n-gram 表現

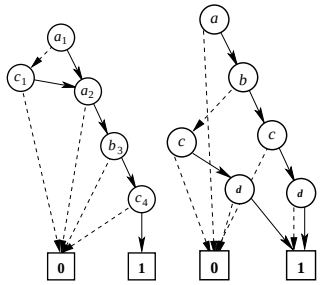

図 2: ZBDDD による ordered n-gram 表現 (左) と Unordered n-gram 表現 (右)

2.4 組合せ集合による文字列表現

表 1 の例はアイテム一つによって文字の種類とその出現位置を示す方法である．例えばアイテム a_1 は”a” という文字が部分文字列の 1 番目に現れることを示し，同様にアイテム a_2, b_3, c_4 はそれぞれ，2 番目の文字”a”，3 番目の “b”，4 番目の”c” を表す．そしてこの 4 つのアイテムの組合せである $a_1a_2b_3c_4$ が部分文字列”aabc” を表現している．同時にこのアイテム組合せは図 2(左) のような ZBDD で表現できる．表 2 の例は Unordered n-gram を利用した場合の表現である．Unordered n-gram では部分文字列を，文字の集合として扱い文字の出現位置については無視するので文字列”aabc” は単純にアイテムの組合せ abc で表現される．また重複アイテムの数え上げは行わないので，アイテム a は 2 回出現しない．また，出現位置を無視することから，部分文字列が”bca”，”cabb”，”ccab” などであってもこれらの文字列は皆，アイテムの組合せ abc で表現される．この手法では部分文字列”aabc” が図 2(右) のような ZBDD に対応する．

この Unordered n-gram に対し従来の文字の出現位置情報を保持し続ける n-gram を Ordered n-gram と呼ぶ．Unordered n-gram はテキストクラシフィケーションの分野で良い実績を上げている例がある [1].

3. 実験結果と考察

提案手法の性能評価のために，小説” 不思議の国のアリス” の英語原著を用いて Unordered n-gram データを ZBDD 上に生成した．簡単のために作品中のアルファベット 26 文字以外の文字と空白文字を削除する前処理を行った．本実験では Pentium 4, 3GHz，主記憶 1Gbyte を搭載した PC を使用し．OS に SUSE Linux 10.2，ZBDD の操作に ZBDD 処理系の一つである VSOP を利用した．

3.1 Ordered n-gram と Unordered n-gram の比較

Ordered n-gram と Unordered n-gram では表 3 のような，性能差があった．ZBDD ノード数は n-gram データを格納するのに必要であった，ZBDD の接点数，n-gram パターン数は部分文字列の種類がどれだけあったかを示している．表にあるように，Unordered n-gram の方が，2，3 倍ノード数を少なくすることに成功している．また一般的な Hash Table に Unordered n-gram を格納した場合と比較するために，同様の処理をプログラミング言語”Ruby” の Hash class を用いて実装した．表 4 のような結果となり，どちらの実装も速度の限界を突き詰めた実装ではないので厳密な比較とは言えないが，ZBDD を用いた手法でも従来手法と同等な速度が出ていることが確認できた．

また，ZBDD 上に生成した n-gram のデータは，ZBDD のもつアイテム集合に対しての，各種集合演算処理が容易に行えるので，その点で従来手法にはない利点があると考えられる．

	Ordered n-gram		Unordered n-gam	
n	ZBDD ノード数	n-gram パターン数	ZBDD ノード数	n-gram パターン数
2	1,886	518	982	306
4	23,607	19,004	6,653	5,241
6	66,420	57,510	19,263	20,941
8	115,634	82,891	30,920	36,803
10	181,158	95,366	39,200	46,000

表 3: Ordered n-gram と Unordered n-gram

n	ZBDD	Ruby Hash
2	4.43	1.29
4	6.24	1.94
6	7.02	2.80
8	7.64	3.61
10	8.51	4.81

表 4: Unordered n-gram 生成時間の比較 (秒)

4. おわりに

本稿では， n-gram データを ZBDD を用いて表現する方法について考察を行ってきた．標準的に使用される Ordered n-gram の他に， Unordered n-gram を用いた場合の処理についても比較検討を行った．残念ながら計算時間では，従来手法と比べて変化が認められなかったが今後，ZBDD のもつ集合演算処理を積極的に使った場合の処理について比較検討を進める予定である．

参考文献

[1] Thade Nahnsen, Ozlem Uzuner, Boris 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

[2] 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.