4M−06

EDR 日本語コーパスにおけるキーワード検索のためのデータ構造の検討*

伴 祐介　鈴木 しとな　上原 徹三　荒井 秀一†
武蔵工業大学‡

1 はじめに

EDR 日本語コーパス [1] は,20 万余の日本語の例文について, 構成する形態素, 構文構造, 意味構造などの情報を与える. これを文法規則の学習や検証, 解析結果の評価等に用いる利用者は, 複雑な階層構造を理解し, 目的に応じた情報を取出して処理するプログラムを作成する必要がある. しかしこの作成は容易でなく, 利用者の本来の目的以前の段階で手間がかかる. そこで, コーパスの各情報の取得, 操作を容易にすることを目的とした, EDR 日本語コーパス参照支援ツール [2] (以下, 本ツール) を試作している. 本ツールは, コーパス中の各情報を取得する関数, 係り先情報を取出す構文情報利用のための関数などを含む. 利用者はこれらの関数を利用することにより, 本来の目的に応じたプログラム機能の作成に専念できる. 本稿では, 本ツールのコーパス検索効率の向上のために, インデックスを利用したキーワード検索のデータ構造について, 探索速度及び主メモリ使用量の観点から検討を行う.

2 インデックスを利用した検索方式

2.1 インデックスの利用法

本ツールでは, 次の 3 つのインデックスを用意することで参照の高速化を図る.

- レコード番号インデックス: レコード番号をキーとし,EDR 日本語コーパスファイル中の該当レコードのオフセット (file-offset) を保持する.
- 品詞インデックス: 15 種の品詞ごとに該当品詞の出現するレコード番号を保持する.
- 表記インデックス: 表記をキーに, 該当表記が出現するレコード番号を保持する.

品詞・表記インデックスは, レコード番号インデックスを一次インデックスとする二次インデックスとして利用する. 以下, 表記インデックスに関する検討を行う.

2.2 表記インデックスのデータ構造

各表記ごとに, その表記が出現するレコード番号のリスト (Inverted List) を保持する Inverted File をディスク上に置く. また各表記とそれに対応する Inverted List のオフセットの対を保持する構造体の配列である Key List を主メモリ上に置く. Key List は,EDR 日本語コーパス中の 203,687 文 (係り受けの交差を含む文を除いた) に出現する単語の全異なり表記 120,400 語を保持する. 助詞「が」など高頻度の語の Inverted List を保持すると, Inverted File のサイズが膨大になるので, 高頻度の語については, Key List に表記を置いてその語の存在を示すが, Inverted List を省略してオフセット部は NULL とする. 高頻度語の検索時はこのオフセット部の NULL 表示を見てコーパスの全件検索に移ることになるが, 該当レコード数が多いので無駄は少ない. この結果, それぞれのデータサイズは, Key List が 1.63MB, Inverted File が 34.5MB となった.

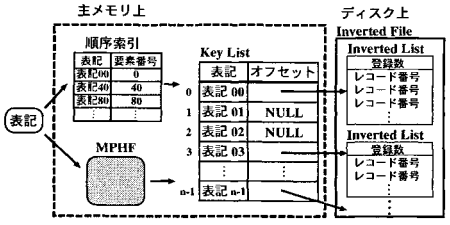

図 1: 表記インデックスのデータ構造

高速検索の実現には Key List の探索方法が重要である. 検索対象のコーパスは固定データであることに留意し, 次の 2 つの Key List 探索法を検討する.

- 順序索引を用いた方法
- 完全最小ハッシュ法 (MPHF) を用いた方法

3 順序索引について

Key List を表記順に整列し, その上位のインデックスを昇順に並べて順序索引と呼び, 主メモリ上に置く. Key List を順序索引の登録インデックス数で割った値の組に分け, 各組の先頭の表記とその表記の Key List 中の配列要素番号を, 順序索引に保持する. 探索の際は, 順序索引を 2 分探索し,Key List 中の該当組の要素番号を得て,Key List の該当組を 2 分探索する.

4 MPHF について

最小完全ハッシュ関数 MPHF(Minimal Perfect Hash Function) は, 全てのキーに対して, 一意にハッシュ値が決定し, ハッシュ表のサイズがキー数と同数のハッシュ関数である. また, 衝突が発生しないため, 探索に必要な計算量はキー数に依らず一定である. 大規模のキーセットでも固定であれば MPHF が導出できるため静的辞書などの検索手段として注目されている. ここでは, 文献 [3] の導出法を参考にして, 上記の 120,400 個の異なり表記より成るキーセットに対する MPHF を導出した.

---

*Data structure for retrieving by keyword in EDR Japanese Corpus

†Yusuke BAN, Shitona SUZUKI, Tetsuzou UEHARA, Shuichi ARAI
‡Musashi Institute of Technology

4.1 MPHF の導出

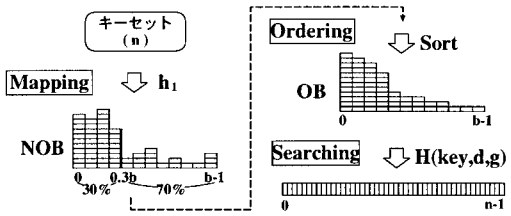

図 2: MPHF 導出の流れ

図 2に MPHF の導出の流れを示す. MPHF の導出は 3 ステップから成る. まず Mapping において,n 個のキーを関数 h_1に適用し, b 個の Non Ordered Bucket(NOB) に配置する. 関数 h_1は, キーの 60%をバケツ番号 {0,…,0.3b} の範囲に, 40%をバケツ番号 {0.3b+1,…,b-1} の範囲に配置する関数とする. 次に Ordering において,NOB をキー登録数の降順に整列し, その結果を Ordered Bucket(OB) と呼ぶ. 最後に Searching において,OB の各バケツごとに H(key,d,g)={h_2(key,d)+g} (mod n) なる関数 H を適用する. d(0 から 255 までの整数),g(0 から n-1 までの整数) はそれぞれ, キーの間の衝突を回避するためにバケツごとに設定できるパラメータである. d は散らばらせ方を変えるパラメータであり, バケツ内のキー同士の衝突回避に用いる. g はバケツ内の各キーを g 分だけずらし, 他のバケツのキーとの衝突回避に用いる.g による衝突回避が不可能な場合は,d で散らばらせ方を変更する. 全バケツの全キーについて衝突の起こらない d,g を求めることができれば MPHF の導出成功となる.

導出成功時には,NOB と OB のバケツ番号の対応を記録した NOtoO 配列と, OB のバケツごとの d,g の値を保持する DG Table を出力する. これらは探索時に MPHF の関数定義の一部として必要になる. 小さな b で MPHF を導出できれば,DG Table のサイズは減少する. なお, いかなる d,g でも衝突回避が不可能な場合は導出失敗となり, バケツ数 b, 関数 h_1, 関数 h_2のいずれかの変更を要する.

4.2 MPHF による探索

探索キーを関数 h_1に適用し, 配置される NOB 番号を求め, NOtoO 配列により移動先の OB 番号を求め,DG Table から d,g の値を得る. これを関数 H に適用し, ハッシュ値を得る. この値で Key List を引くと, 存在するキーであれば表記欄の表記と一致する.MPHF の関数定義の一部である NOtoO 配列と DG Table は主メモリ上に置く.

5 探索方法の比較評価

表記インデックスにおいて, 異なり表記 120,400 語を登録させた順序索引と MPHF の 2 つの探索方法について, 表記キーを Key List から探索するまでの時間と, 探索時の主メモリ使用量の比較を行った. 順序索引は 500 個のインデックスを登録した. このとき 1 個のインデックスあたり 241 キー, 順序索引サイズは 7KB となった. また MPHF はバケツ数 b=9,428 で導出に成功したものを使用した. このとき関数定義のサイズは 90KB となった. 表 1に比較結果を示す. なお, 実行環境は,CPU:Celeron 433MHz, 主メモリ容量:256MB,OS:FreeBSD4.1 である.

表 1: 性能比較の結果

探索方法	順序索引	MPHF
1,000 キーの探索時間	7 ms	7 ms
1,000,000 キーの探索時間	3990 ms	5943 ms
探索時の主メモリ使用量	1.63 MB	1.72 MB

両方法とも, 高速な探索が可能で, 探索時の主メモリ使用量も, 現在の計算機の主メモリ容量から考えて実用的範囲にある. 順序索引法が MPHF 法より高速なのは, 今回導出に成功した MPHF の関数 H が, 衝突回避のための複雑な計算を含むことに依る. 計算量の上では, 順序索引法では, 順序索引および Key List の該当部分を 2 分探索するのでキー数に依存し,MPHF 法では O(1) で一定だが, 120,400 規模の登録キー数では, 順序索引が高速となった. また探索時の主メモリ使用量も順序索引の方が少ない. さらに,MPHF は導出が困難であり, 順序索引は実現が容易である. 以上より,EDR 日本語コーパスの表記インデックスへの適用において, 順序索引を用いることとした. MPHF でも, 関数 H の単純化による高速化の可能性はある. また, より小さなバケツ数 b で MPHF を導出できれば, 関数定義のデータサイズは減少する可能性がある. しかし, 関数 H の単純化, および b=9,428 より小さな b での導出を試みているが現在までのところでは成功していない.

6 おわりに

EDR 日本語コーパスに対するキーワード検索のためのデータ構造として, 表記インデックスを例に, 順序索引および MPHF を検討した. 探索速度, 探索時の主メモリ使用量の比較の結果, 12 万の規模の固定キーセットに対しては, 両方法とも実用的範囲にあったが, 順序索引の方が, より少ない主メモリ使用量で高速な探索が可能であった. また, 順序索引の方が, 実現が簡単でキーセットの変更にも応じられる. そこで順序索引を採用することとした. MPHF が有効となる状況の検討は今後の課題である.

なお, 本稿の一部は文部省科学研究費補助金 (基盤研究 C2 No.13680492) によって実施したものである.

参考文献

[1] 日本電子化辞書研究所: EDR 電子化辞書仕様説明書, 日本電子化辞書研究所 (1995).

[2] 鈴木 しとな, 上原 徹三, 石川知雄: 構文木を含むタグ付きコーパス参照支援ツール, 言語処理学会 第 7 回年次大会発表論文集 C6-6 (2001).

[3] Fox E.A., Chen Q.F. and Heath L.S.: "A Faster Algorithm for Constructing Minimal Perfect Hash Functions" , Proceedings ACM SIGIR'92 Conference, pp.266-273, Copenhagen (1992).