# ZIP の圧縮率を超えた単語エントロピー符合化と意味構造符合化の一体化による言語解析圧縮データの再利用

大倉 清司¶　片岡正弘¶¶　出内将夫¶¶

¶（株）富士通研究所　¶¶ 富士通（株）

## 1. はじめに

従来、PC や電子辞書などの端末上で実行されていた情報検索においては、計算量に制約があるため、見出し語検索を主流としたエコな検索技術が求められていた。しかし近年、端末にネットワーク接続機能が付与されたことから、インターネットサービスを利用する形態が主流となっている。ネットワーク上の検索サービスでは、計算機資源が豊富にあるクラウド環境を利用して、Hadoop[1]に代表される並列分散処理などを活用し、あいまい検索やランキング検索といった高度な検索が実現されている。

ネットワーク上に存在するデータ量は BigData という言葉が示すように膨大であるため、Hadoop や RDB などのデータ管理機構の内部で Disk I/O を高速化するための圧縮技術が活用されている[2]。Hadoop における圧縮方式では、Disk I/O の高速化だけでなく、圧縮状態のままでレコードを識別可能とすることで、分散処理の効率を上げる取り組みがされている[3]。

ただし、高度な検索の実現において、クラウドでもインデックスによる容量の増加は課題として残っている。また端末でも、伸長と照合を 1 パス化し高速に検索する技術は、依然として求められている。圧縮形式のまま高速に照合する技術が求められている。

本稿では、言語処理に特化した圧縮手法を提案する。圧縮時に併せて形態素解析や意味解析を行い、伸長せずに解析結果を再利用できる。例えば、高速な意味構造検索を実現できる。原テキストの圧縮率では約 28%と ZIP の約 33～35%を上回り、意味解析結果の圧縮率でも 12.5%と ZIP の 17.3%を上回った。さらに、本圧縮形式では圧縮したまま解析結果を再利用できる利点がある。今後は、様々なアプリケーションに対し、この圧縮技術の適用や実装検討を推進する。

## 2. 従来技術

### 2.1. 文字の符合化技術とその課題

1948 年にシャノンが発表した情報エントロピーの理論[4]をもとに、1952 年にハフマンが英数字など 8bit 文字コードのエントロピー符号化の圧縮技術を確立した[5]。一方、1977 以降、最長一致文字列に着目したジブ－レンペル符号化の圧縮技術が確立された[6]。

1997 年、富士通により JIS かな漢字コード(16bit)のエントロピー符号化が考案され、広辞苑などの辞書データの圧縮技術として実装され JIS-X4081[7]に反映された。さらに、エントロピー符号化の副産物として、かな漢字のビットマップ型インデックスや、1 パス照合伸長技術が考案され、富士通により 2008 年に圧縮型全文検索技術として製品化された[8]。

**(1) JIS かな漢字コードのエントロピー符号化**

ハフマン圧縮は 8bit 文字コードに対し 256 個の葉を持つ。16bit 対応は葉が 64K に増加し、ハフマン木の肥大と生成時間が課題であった。

辞書の文字種と頻度を調査したところ、約 8K 種の文字に対し、高頻度な文字は 1K 種であることが判明した。そこで低頻度な文字は 8bit の合成と見なし、1K+256 の葉で JIS かな漢字コードのエントロピー符号化を行う圧縮技術を確立した。

**(2) ビットマップ型インデックスと全文検索**

符号化の過程で、出現文字のビットを ON することで図 1 のようなビットマップ型インデックスを生成することができる。例えば、「森鴎外」[1]を検索する場合、各文字のビットマップを AND 演算すると、「森鴎外」が含まれるファイルを絞込むことができ、全文検索の高速化

[1] 紙の広辞苑では「森鷗外」だが、デジタルでは第 3 水準・第 4 水準の漢字を表示できない端末があるため、この表記が割り当てられている。



Copyright(C) 2015 The Association for Natural Language Processing.
All Rights Reserved.

を実現した。(1 漢字の出現率は平均 1/13 であり、3 漢字で約 1/2K に減少。)

**(3) 1 パス照合伸長技術**

圧縮データを照合する場合、2 パスで伸長と照合が行う必要がある。図 2 のように、ハフマン木の葉の構造体を拡張することで、伸長と照合が同時に実行でき、1 パス化が図れる。

例えば、「森」の葉に「森鴎外」と照合を行うように設定する。伸長にて「森」の符号が発見されると、「森」に伸長すると同時に「森鴎外」と照合することができる。

複数の文字列やタグを組合せ、照合条件付きの部分伸長を実現することができる。

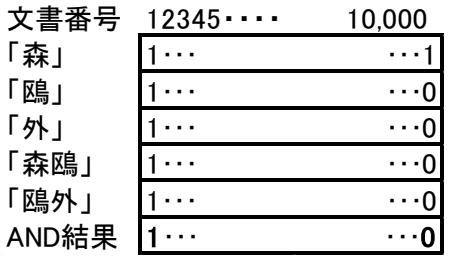


図 1. ビットマップ型インデックス

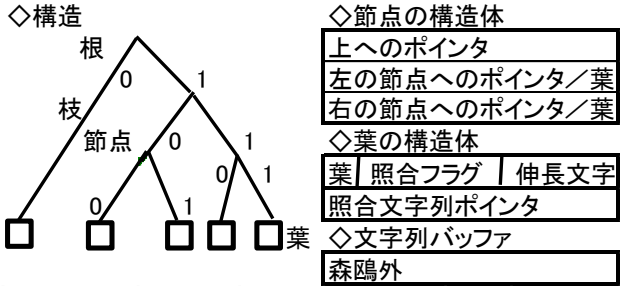


図 2. ハフマン木とその構造

このような文字符合化技術を実現した一方で、1999 年頃より単語のエントロピー符号化の方式が考案されているものの、以下のような課題が残されていて、実用化が困難であった。

- 高速復号化
- 可変長の符号割当てとその最適化
- 英単語と空白などの記号の符号割当て

## 2.2. 意味構造を対象とする検索技術とその課題

意味構造を、2 つの直接つながるノードとその関係を表す 3 つ組である意味最小単位にばらして検索を行うことにより、意味構造を検索する技術を開発した[9,10]。しかし、この技術には大規模データに対応するには課題があった。解析をしてから検索のためのインデックスを作成するが、解析結果は原文テキストの 40 倍もの情報量になり、大量のデータを保存するために圧縮を余儀なくされる。そのため圧縮したデータを伸張してからインデックス作成を行わなければならないが、そのとき全データを読み込まなければならず、解凍処理に時間がかかってしまう。さらに、意味解析した結果の意味記号に付与する符合が、原文の符合と一体化されていなかったため、意味最小単位が検索されても、それに含まれる意味記号と原文の単語との対応がつかなかった。対応する原テキストを併せて参照するためには、検索された意味記号にそれぞれ付与された原文との対応情報を元に、原文に当たる必要があった。このとき、原文の圧縮ファイルを伸張する必要があった。

またインデックス作成時の計算量が膨大であることも問題であった。意味構造を検索するのに、任意のノードの組み合わせがどの文書にマッチするかを格納する必要があるが、意味記号数が大きいと、その組み合わせ数が意味記号数の 2 乗のオーダーになる。例えば、意味記号数が 5M ならば、組み合わせのオーダーは 5M*5M＝25T になってしまう。

意味構造検索技術により、キーワードベース検索に比べて検索の精度は向上できるものの、計算量の問題があり、大規模文書を対象に検索するには、実用化が困難であった。

## 3. 単語符合化と意味構造符合化の融合

(1)高速な複号化をするための単語を単位とした符号化技術開発 (2)意味構造を圧縮したまま再利用する技術開発を目的とした。単語単位で圧縮するため単語単位で検索できる。また意味記号と単語を関連付けられるため、意味構造との親和性が高くなる。

しかし、従来技術そのままでは、問題があった。形態素と意味記号の符号が別個であるため、そのままでは対応付けができない。また意味構造検索のために再利用するには、速度の面から意味最小単位を固定長で持つことが望ましいが、検索対象全体の意味記号数が増えると意味記号を格納するビット数が増えてしまう。例えば、意味記号数が 256 以下ならば情報を格納するために必要なビット数は 8 で済むが、意味記号数が 500M の場合は 23 ビット必要になる。そのため、検索速度と圧縮率を両立させた、意味記号格納技術が求められていた。


Copyright(C) 2015 The Association for Natural Language Processing.
All Rights Reserved.

### 3.1. 単語符合化による圧縮率向上

単語単位のエントロピー符号化に関し、高速復号化として、2008 年に無節点な復号化木を考案し、それに基づき符号割当ての最適化、空白付き英単語の符号化を考案した。なお、符合化については、2 分木探索で辞書に登録されている単語とテキストの文字列を照合すると低速になるため、N グラム文字などの単語フィルタを設け、辞書に登録されている単語を無節点でポイントすることで、ハフマン圧縮と同程度まで高速化した。

#### 1. 無節点な復号化木による高速復号化

2 分木に対し、節点が無い復号化木（ケヤキ木と呼ぶ）の構造を図 3 に示す。従来の 2 分木によるハフマン圧縮は、可変長符号を各節点の 1bit 毎に判定するため、低速であった。ケヤキ木は節点が無く、最大符号長(16bit)に対し、64K 本の枝を持ち、1 回の判定で葉をポイントすることができ、高速な復号化を実現できた。(復号する単語の文字列や実符号長は葉の構造体に格納されている。)

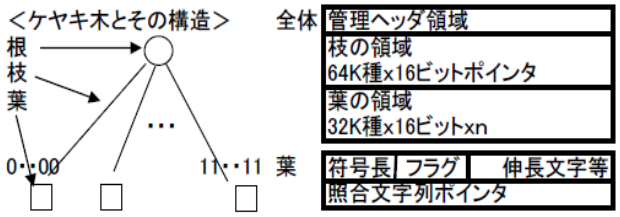


図 3. ケヤキ木とその構造

#### 2. 頻度補正による最大符号長の割当て

ハフマン圧縮では、低頻度な記号に 17bit 以上の符号割当てが行われる場合がある。単語の出現頻度を補正し、ケヤキ木の構造体のサイズの肥大化を防止する方法が考案された。例えば、the の頻度が $1/2^5$～$1/2^4$ であり、5bit の符号が割当てられるが量子化損が生じる。各単語の頻度の総和は 1 以下であれば良く、頻度 $1/2^{17}$ 以下を $1/2^{16}$ に補正し、最大符号長を 16bit に抑え、ケヤキ木のサイズ縮小を図ることができた。

#### 3. 空白付き英単語への符号割当て

英文テキストでは、空白の頻度が高く、相対的に単語の頻度が低下し、割当て符号長が長くなり、圧縮率が低下する原因になっている。

そこで、空白付き英単語に符号割当てを行う方法が考案され、圧縮率の向上が図られた。

例えば、英文では、the が最頻出単語であリ、3+1 文字の 32bit に対し 5bit が割当てられ 5/32 の圧縮効果がある。しかし、単語と空白を分離すると the に 6～7bit が、空白に数 bit が割当てられ、圧縮率が低下する。なお、文末の単語に続くピリオドについては、［BS＋.］の仮想的な単語に置換される。

### 3.2 意味構造の符合化による圧縮率向上

上記の単語を符合化する技術をベースに、意味構造を符合化することにより形態素解析結果と関連づけて意味解析結果を符合化し、圧縮したまま検索することが可能になった。そのためのポイントは以下の 3 つである。

#### 1. 字句解析結果と意味解析結果の符号化の一体化

原テキストを単語ベースで圧縮することが可能になったため、図 4 に示すように、形態素と意味記号の符号を 1 つの辞書エントリーとして一体化して登録することが可能になった。その結果、意味構造検索の際には単語ビットマップにより検索対象を大幅に絞り込んでから意味構造を検索することが可能になった。また、単語の文字列だけでなく、形態素情報（品詞や属性情報）なども符号に組み込むことができ、形態素解析の結果も含めての符合化を実現できる。

| 単語コード | 形態素＋品詞＋属性＋意味記号 |
|---|---|
| 0 | は＋助詞＋+// |
| 1 | が＋助詞＋+// |
| 2 | を＋助詞＋+// |
| : | : |
| 5023 | 翻訳＋名詞＋抽象物＋/TRANSLATION/ |
| : | : |
| 7025 | 作業＋サ変名詞＋動作＋/WORK/ |
| : | : |
| 8653 | 翻訳者＋名詞＋人間＋/TRANSLATOR/ |
| : | : |

図 4. 形態素と意味記号の符合一体化

#### 2. 文書内の意味記号の符号化

検索対象文書全ての形態素数は膨大になる可能性があるが、1 文書に含まれる形態素数はそれほどではない。この性質を利用して、単語コード辞書の符合と 1 文書内の意味記号の符号(意味コードと呼ぶ)とをマッピングする情報を格納することにより、1 文書当たりに含まれる


Copyright(C) 2015 The Association for Natural Language Processing.
All Rights Reserved.

単語数を大幅に減らすことができ、圧縮時に 1 単語当たりのビット数を低減できる。

## 3. 形態素と意味記号の対応情報の埋め込み

意味構造を表す意味最小単位の符号の中に、形態素解析結果の符合との対応情報を埋め込む（図 5, 図 6）。意味最小単位の以下 3 つのパターンそれぞれに対応した。

(ア) パターン 1：（ノード 1,ノード 2,アーク）
(イ) パターン 2：（ノード 1,NIL,アーク）
(ウ) パターン 3：（NIL,ノード 1,アーク）

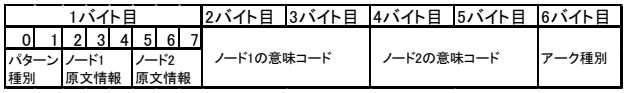


図 5. 意味最小単位の符合化（パターン 1）

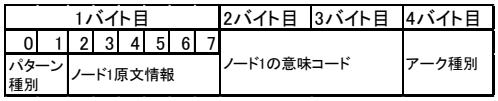


図 6. 意味最小単位の符合化（パターン 2,3）

これにより、意味解析結果を大幅に圧縮することが可能になった。

## 4. 評価実験

本技術の有効性について評価した。単語符合化に関しては速度と圧縮率を、意味構造の符合化に関しては圧縮率を評価した。圧縮率の評価は ZIP と本技術を比較することにより行った。

### 4.1. 単語符合化の高速化に関する評価

2 分木で符合化した場合と本技術を使って符合化した場合で実装したところ、ループにおける実行ステップ数が本技術のほうが約半分になり、実行速度も 2 倍になることを確認した。

### 4.2. 単語符合化による圧縮率の評価

グーテンベルグプロジェクト[11]より約 160 種の電子書籍をダウンロードした。ZIP の圧縮率は 35.6%であった。約 19 万語の単語の頻度を集計すると、最頻出単語 the+スペース(32bit)は約 $1/2^5$ で 5bit が割当てられ、5/32 に圧縮される。順に高頻出単語を累積すると、約 28%に圧縮することができた。

なお、ビットマップ型全文インデックスを文字から単語に拡張することで、n グラム文字による検索ノイズを減少することもできる。

この技術は単語に区切られていれば言語に依存しない。日本語における圧縮率を青空文庫[12]で評価した結果、ZIP による圧縮率は 33.4％なのに対して本技術による圧縮率は 28.1%であった。

### 4.3. 意味解析結果の圧縮率の評価

我々の意味構造検索技術は特許を検索対象としているため、特許明細書約 10000 文書の本文を本技術を使って意味解析結果をどのくらい圧縮できるか検証した。その結果、意味解析結果を ZIP の 17.3%に対して 12.5%に圧縮することができた。同時に、ZIP では不可能だった、圧縮したままのデータの再利用を実現した。この評価では意味構造だけで評価したが、実際に使う場合は、単語符合化と組み合わせて使う。

## 5. 今後の展望

本技術の計算量（メモリ消費量など）や ZIP との速度比較を定量的に評価していく。意味構造検索のインデックス作成にかかる計算量および検索時間の評価もしていく。また、様々なアプリケーションに対し、この圧縮技術の適用や実装検討を推進する。

## 参考文献

[1]Apache Hadoop. http:// hadoop.apache.org/
[2]若林拓馬, 山室健,寺本純司,西村剛. カラム構造と圧縮による Hadoop から PostgreSQL へのロード高速化に関 する実験と考察. 第 12 回日本データベース学会年次大会. 2014.
[3]Hadoop LZO 圧縮機能の検証. 日立ソリューションズホワイトペーパー. 2012.
[4]C.E.Shannon,”A Mathematical Teory of Communication”, Bell Syst. Tech. Journal, vol.27, no.3, pp379-423, Jun.1948
[5]D.A.Huffman, “A Method for the Construction of Minimum Redundancy Codes”, Proc.IRE, vol.40, no.9, pp.1098-1101, Sep.1952
[6]J.Ziv, A.Lempel, “A Universal Algorithm for Sequential Data Compression”, IEEE Trans.on Inform. Theory, vol.IT-23, no3, pp.337-349, May 1977
[7]http://www.jisc.go.jp/app/pager?%23jps.JPSH0090D:JPSO0020:/JPS/JPSO0090.jsp=&RKKNP_vJISJISNO=X4081
[8]http://pr.fujitsu.com/jp/news/2008/05/9.html
[9]大倉清司,潮田明. 意味検索のプロトタイプシステムの構築. 言語処理学会第 18 回年次大会. 2012.
[10]大倉清司,潮田明. 複雑な文に対応した意味構造検索システムの開発. 言語処理学会第 20 回年次大会. 2014.
[11]Project Gutenberg. https://www.gutenberg.org/
[12]青空文庫. http://www.aozora.gr.jp/


Copyright(C) 2015 The Association for Natural Language Processing.
All Rights Reserved.