情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2009-DBS-148 No.12
Vol.2009-FI-95 No.12
2009/7/28

転置ファイルによる大規模 n-gram データの検索システム

矢田 晋[†1,†2] 森田和宏[†1]
泓田正雄[†1] 青江順一[†1]

近年，大規模な単語 n-gram データを対象として，トライを索引とする検索システムが提案されている．しかし，トライ索引は強力な検索機能を有するものの，ディスク使用量や構築コストが非常に大きいという欠点を持つ．一方，文書検索においてよく用いられる転置ファイルは，検索時にディスクへのランダムアクセスが発生するという欠点があるものの，索引を容易に構築できるという利点を持つ．そこで，本稿では，大規模 n-gram データに対する転置ファイルの有効性について報告する．

A Large-scale N-gram Search System Using Inverted Files

Susumu Yata,[†1,†2] Kazuhiro Morita,[†1]
Masao Fuketa[†1] and Jun-ichi Aoe[†1]

Recently, a search system has been proposed for large-scale n-gram data. The system uses tries for indexing n-gram data because of its retrieval performance. However, the disk usage and the construction cost of the trie index are very large. In contrast, inverted files could be easily constructed but require random access to secondary storage in retrieval. Therefore, this paper reports the actual performance of inverted files for large-scale n-gram data.

1. はじめに

言語処理の研究において，大規模コーパスの利用は主流となりつつある[1)]．また，ウェブコーパスに対する期待の高まりとともに，利用されるコーパスの規模も拡大している．ただし，大規模コーパスの構築や管理・運用には大きなコストがかかるため，TSUBAKI[2)] のようなプロジェクトに関連する研究を除けば，検索エンジンを介して間接的にウェブコーパスを利用したり，新聞記事や Wikipedia などのコーパスを走査して統計情報を抽出したりする研究が多い．

研究用途で利用可能な大規模コーパスに，Google によって作成された n-gram データ[3),4)] が存在する．このコーパスは，ウェブコーパスに出現する単語 n-gram とその頻度を併せただけの単純なデータであるものの，抽出元のウェブコーパスが極めて大規模であることから，言語モデルとしての利用を中心に応用が進められている．応用例としては，統計的機械翻訳[5)–7)] や自動音声認識[8)]，スペル訂正[9)] などがあり，言語モデルとしての有効性が示されている．また，データマイニングにおける訓練データとしての利用[10)] や，ウェブコーパスにおける Document Frequency（DF）の推定[11)] なども試みられている．

しかし，コーパスの大規模化には，応用を困難にするという欠点がある．例えば，上述の大規模 n-gram データは，n-gram の異なり数が数十億，合計サイズが約 100 GB という規模のため，ディスクに格納されているコーパス全体を走査するだけで，少なくとも数分の時間を要する．したがって，入力に応じてコーパス全体から適切な n-gram を選択するという用途では，索引の構築が必要不可欠となり，1-gram をラベルとするトライの構築[6)] や MySQL による索引の構築[9)] などがおこなわれている．

さらに，大規模 n-gram データを効率的に利用する汎用的なツールとして，Minimal Prefix（MP）トライ[12)] を索引とする検索システムが提案されている[13),14)]．このシステムでは，任意の 1-gram に対応するワイルドカードを含むクエリを受け付けるため，n-gram の並びを変更したトライを構築する．なお，k-gram を検索対象とするとき，${}_kC_{k/2}$ パターンのトライを構築すれば，任意の位置に出現するワイルドカードに対応できる．実際に 5.7 億件の日本語 7-gram[4)] に対して索引を構築すると，3 個以下のワイルドカードを含むクエリに対する平均応答時間は 0.05 秒になることが報告されている．しかし，索引のサイズが約 490 GB と巨大になり，索引の構築には数ヶ月を要するなど，索引の構築コストが非常に大きいことも報告されている．

そこで，本稿では，構築の容易な転置索引[15)] を大規模 n-gram データに適用した結果を報告する．また，転置索引による検索システムの欠点となるディスクへのランダムアクセスについて，n-gram データの特徴を利用することにより，大幅に改善可能であることを示す．

以下，2 章で転置索引による検索システムの概要を述べ，3 章で大規模 n-gram データに対する効率化について説明する．4 章では，検索システムに対する評価を実験結果とともに

†1 徳島大学 University of Tokushima
†2 特別研究員 JSPS Research Fellow

Vol.2009-DBS-148 No.12
Vol.2009-FI-95 No.12
2009/7/28

表 1 文書集合 D に対する分かち書き
Table 1 Word segmentation of documents in D

ID	Document	Segmented words (separated by '/')
1	我輩は猫である	<S> / 我輩 / は / 猫 / で / ある / </S>
2	我輩は鼠である	<S> / 我輩 / は / 鼠 / で / ある / </S>
3	我輩は小僧である	<S> / 我輩 / は / 小僧 / で / ある / </S>

表 2 分かち書きの結果を用いた転置索引
Table 2 An inverted index using segmented words

ID	Term	DocID
1	</S>	1, 2, 3
2	<S>	1, 2, 3
3	ある	1, 2, 3

ID	Term	DocID
4	で	1, 2, 3
5	は	1, 2, 3
6	我輩	1, 2, 3

ID	Term	DocID
7	小僧	3
8	鼠	2
9	猫	1

示す．そして，5 章で本研究の成果と今後の課題についてまとめる．

2. 転置索引

2.1 文書検索における転置索引

一般的な文書検索システムでは，索引語から出現文書の一覧を取得するためのリスト（転置索引）を持つことで，大規模なデータに対する検索を実現する．索引語としては，文書中に出現する文字 n-gram や形態素解析器による分かち書きの結果を用いることが多い．文書集合 $D = \{d_1, d_2, d_3\}$ に対する分かち書きの例を**表 1** に示し，転置索引の例を**表 2** に示す．表 1 における ID は文書の ID（DocID）を表し，表 2 における ID は索引語の ID（TermID）を表している．

転置索引による検索では，クエリから索引語を取り出し，DocID のリストをマージすることによって，索引語を含む文書の一覧を得る．例えば，表 2 の転置索引に対して “我輩猫” というクエリが与えられた場合，“我輩” から DocID 1, 2, 3，“猫” から DocID 1 がそれぞれ得られるため，両方のリストに出現する d_1 が検索結果となる．

また，転置索引の構築においては，DocID のリストを差分のリストに変換し，さらに可変長のバイトコード（Variable Byte Code）へと符号化することで，固定長の整数で DocID を保存する場合と比べて，索引のサイズを大幅に削減できる[15]．

2.2 転置 n-gram 索引

本稿で扱う大規模 n-gram データ[4] は，分かち書きにより得られる単位語を 1-gram とし，

表 3 文書集合 D より得られる 2-gram データ
Table 3 2-gram data extracted from documents in D

ID	TermID	Term	Frequency
1	2, 6	<S> 我輩	3
2	3, 1	ある </S>	3
3	4, 3	で ある	3
4	6, 5	我輩 は	3
5	5, 7	は 小僧	1

ID	TermID	Term	Frequency
6	5, 8	は 鼠	1
7	5, 9	は 猫	1
8	7, 4	小僧 で	1
9	8, 4	鼠 で	1
10	9, 4	猫 で	1

ウェブコーパスに出現する n-gram を頻度とともに記録している．例として，文書集合 D より得られる 2-gram データを**表 3** に示す．このとき，1-gram を索引語，n-gram と頻度の組を文書とみなすことにより，従来の文書検索システムと同様の方法で，転置索引を容易に構築できる．また，頻度が降順になるように DocID を割り当てることで，頻度を基準とする検索結果のランキングを実現できる．

しかし，転置索引による検索システムは，ディスクへのランダムアクセスが検索時間のボトルネックになるという問題を抱えている．このボトルネックは，検索結果となる n-gram がディスク上に点在することを原因としており，クエリに一致する n-gram が多い状況において，検索時間の悪化を招く．そのため，与えられた単語の周辺情報を調べる場合や，索引のみでは実現できない絞り込みを後処理としておこなう場合など，大量の n-gram にアクセスする用途では，以下のような対策が必要となる．

- ランダムアクセスの回数を削減する．
- ランダムアクセスを高速化する．

前者については，n-gram データの圧縮と転置索引の展開を 3 章で説明する．一方，後者については，ランダムアクセスに強いとされる Solid State Disk（SSD）上に検索システムを構築した結果を 4 章で示す．

3. 転置 n-gram データベース

3.1 符号化による n-gram データの圧縮

ディスク上の大規模データが容易に圧縮・復号可能な場合，圧縮したデータをディスクに保存しておき，必要に応じてデータを復号することにより，ディスク使用量の削減とディスクアクセスの効率化を実現できる．そして，大規模 n-gram データは，語彙が限定的であり，さらに出現頻度の偏りが大きいという特徴を持つため，容易に圧縮・復号可能という条件にあてはまる．

Vol.2009-DBS-148 No.12
Vol.2009-FI-95 No.12
2009/7/28

本研究で用いた n-gram データについては，以下の手順で圧縮を施すことにより，ディスク使用量を 40% 程度にまで削減することができた．

(1) 頻度の高い 1-gram に短い符号を割り当てるため，頻度降順に TermID を割り当てる．
(2) 各 n-gram を TermID のリストに変換し，可変長のバイトコードへと符号化する．
(3) 各 n-gram の頻度情報を可変長のバイトコードへと符号化する．

また，上記の圧縮手法は，復号にほとんどコストのかからない圧縮データ構造を採用しているため，復号にかかる時間が検索時間に影響を与えにくく，ディスクアクセスの効率化が検索時間の短縮に反映されやすいという特徴を持つ．結果として，時間・空間の両面における検索システムの性能向上につながる．

符号化された n-gram データに対する検索システムは，**図 1** に示すように，以下の手順で n-gram を検索する．

(i) `Dictionary lookup`：クエリとして与えられた索引語を TermID に変換する．
(ii) `Index lookup`：索引語を含む n-gram の格納位置を転置索引より取得する．
(iii) `N-gram lookup`：転置索引より得た格納位置にアクセスして n-gram を取得する．
(iv) `Decode`：符号化された状態の n-gram を復号する．

なお，複数の索引語がクエリとして与えられた場合については，Index lookup の段階で格納位置のリストをマージすることにより，不要な n-gram へのアクセスを省略することができる．ただし，リストの長さが極端に異なる状況では，n-gram を取得した後でフィルタリングを適用する方が効率的である．以下，図 1 に登場するデータベースの構成を示す．

- `N-gram database`：大規模 n-gram データの本体であり，符号化された n-gram が頻度降順に格納されている．
- `Inverted index`：転置索引であり，N-gram database における n-gram の格納位置が，TermID 別にリストとして格納されている．また，TermID からリストの開始位置を得るためのテーブルが末尾に格納されている．
- `Term dictionary`：索引語の辞書であり，TermID との相互変換を実現するために，索引語から TermID への変換にはダブル配列[16]，TermID から索引語への変換には単純なテーブルを採用している．

3.2 転置 n-gram 索引の展開

転置索引による検索システムにおいて，ディスクへのランダムアクセスが発生する原因は，検索結果となる n-gram がディスク上に点在していることである．そのため，ランダムアクセスを削減するには，n-gram データの配置順序を変更する必要がある．しかし，n-gram の配置を変更するだけでは，一部のクエリに対する最適化が限界であり，さらにランキング用の後処理が必要になるという問題がある．

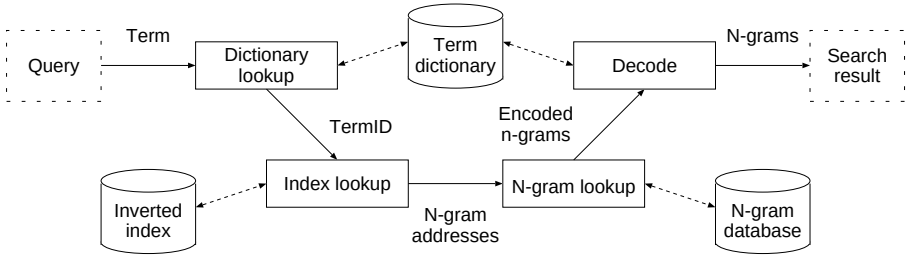

図 1 転置索引による n-gram 検索システム
Fig. 1 An n-gram search system using an inverted index

そこで，本研究では，n-gram データの特徴に着目し，転置索引の展開によるディスクアクセスの効率化を試みた．すなわち，通常の転置索引が DocID のリストを保存するのに対し，n-gram のリストを保存するように変更した．この変更により，特定の 1-gram を含む n-gram のリストに対するシーケンシャルアクセスが可能となり，ランダムアクセスの回数は大幅に削減される．以降，展開された転置索引のことを転置データベースと記述する．

転置データベースの自明な欠点はディスク使用量の増大であり，一般的な文書検索システムでは，各文書に出現する索引語が多いため，実現困難な手法となる．一方，わずかな索引語で構成される n-gram の場合，転置データベースのディスク使用量は，転置索引と比べて数倍程度に抑えられる．また，転置データベースは n-gram データを含んでいるため，検索システム全体のディスク使用量は 3 倍程度で収まる．結果として，実用的なディスク使用量で，シーケンシャルアクセスを主体とする n-gram 検索システムを構築できる．

転置データベースによる n-gram 検索システムの概要を**図 2** に示す．検索手順については，図 1 から Index lookup と N-gram lookup を取り除き，新しく Database lookup を加えた構造になっている．

- `Database lookup`：索引語を含む n-gram をデータベースから取得する．

また，図 2 における Inverted database は，図 1 における Inverted Index と N-gram database を統合したデータベースになっている．

- `Inverted database`：転置索引を展開して得られるデータベースであり，n-gram のリストが TermID 別に格納されている．また，TermID からリストの開始位置を得るた

Vol.2009-DBS-148 No.12
Vol.2009-FI-95 No.12
2009/7/28

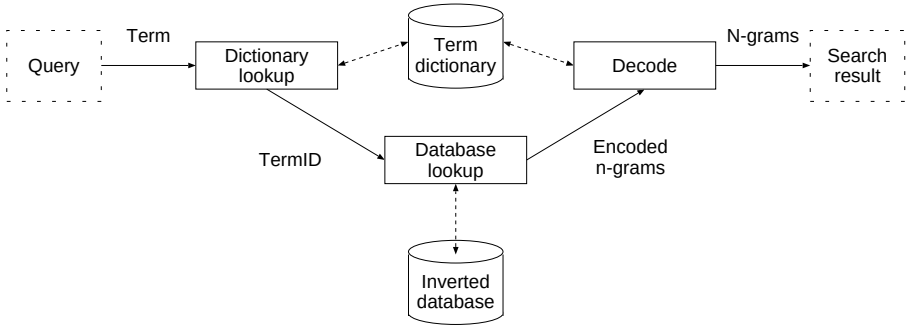

図 2 転置データベースによる n-gram 検索システム
Fig. 2 An n-gram search system using an inverted database

めのテーブルが末尾に格納されている．

そして，転置索引と転置データベースの重要な相違点が，複数の索引語により構成されるクエリの扱いである．転置索引による検索システムでは，ディスクへのランダムアクセスを削減するため，n-gram の格納位置をマージする方法を基本とする．一方，転置データベースによる検索システムでは，特定の索引語を含む n-gram に対するシーケンシャルアクセスが可能なため，低頻度の索引語を含む n-gram を取得し，高頻度の索引語でフィルタリングをかける方が，より少ないディスクアクセスで検索を実行できる．

4. 実験による評価

4.1 実験設定

大規模 n-gram データに対する転置索引および転置データベースの有効性を評価するため，64-bit Linux 環境に検索システムを構築し，検索時間を測定した．実験に用いたコーパスは，Google の作成した n-gram データ[4] であり，1-gram から 7-gram まで合計 32 億件の n-gram により構成されている．構築した索引構造は以下の 3 種類であり，それぞれのディスク使用量は**表 4** のようになる．

(a) `Uncompressed`：符号化していない n-gram データに対する転置索引
(b) `Compressed`　：符号化した n-gram データに対する転置索引
(c) `Extended`　　：符号化した n-gram データに対する転置データベース

検索時間の測定では，コーパスから 1-gram, 3-gram, 5-gram を無作為に 1,000 件ずつ抽出し，クエリとして用いた．

表 4 大規模 n-gram 検索システムのディスク使用量（kB）
Table 4 Disk usage of large-scale n-gram search systems (kB)

	Term Dic	Inverted Idx	N-gram DB	Inverted DB	Total
(a) Uncompressed	76,818	29,757,579	109,221,364	0	139,055,762
(b) Compressed	148,126	26,969,347	29,033,897	0	56,151,370
(c) Extended	148,126	0	0	150,720,315	150,720,463

表 5 検索時間の平均値・中央値・最大値（秒）
Table 5 Average, median and maximum search time (s)

	1-gram			3-gram			5-gram		
	Ave	Med	Max	Ave	Med	Max	Ave	Med	Max
(a) hdd	5.016	0.268	471.830	3.855	0.983	229.582	2.394	0.830	45.784
(b) hdd	2.467	0.179	128.178	3.272	0.868	132.045	2.156	0.761	36.814
(b′) ssd	0.547	0.012	142.863	0.699	0.148	31.212	0.577	0.162	14.262
(c) hdd	0.040	0.029	0.398	0.167	0.094	3.547	0.219	0.097	4.852

実験に用いた計算機の主要な構成は，Core 2 Quad 2.83 GHz, 8 GB RAM, 1.5 TB HDDx2 (RAID 0), 256 GB SSD であり，転置索引および転置データベースの構築はそれぞれ 24 時間以内で終了した．そして，すべてのデータを HDD に格納した状態で検索時間を計測した後，Compressed については SSD に格納した状態での検索時間も計測した．これは，ディスクへのランダムアクセスが検索時間のボトルネックになりやすい転置索引に対する SSD の有効性を確認することが目的である．

4.2 実験結果

実験において計測された検索時間の平均値・中央値・最大値を**表 5** に示す．符号化によりディスク使用量が抑えられている（b）については，同じデータを SSD 上に配置した（b′）による検索時間も併せて示している．実験結果より，n-gram データの符号化，SSD の利用，転置索引の展開によって検索時間を短縮できることがわかる．

符号化の有無が違いとなる（a）と（b）の比較では，n-gram データの符号化により，ディスク使用量を 40 %程度に削減しつつ，平均検索時間を 49–90 %程度に短縮できることがわかる．また，(b) と（b′）の比較では，HDD から SSD へと移行するだけで平均検索時間が 21–27 %にまで短縮されており，ランダムアクセスに強いという SSD の利点が表れている．そして，(b) と（c）の比較により，転置索引の展開によるランダムアクセスの削減は検索時間の短縮に大きく貢献することがわかる．ディスク使用量が（b）の 3 倍近くになるとい

Vol.2009-DBS-148 No.12
Vol.2009-FI-95 No.12
2009/7/28

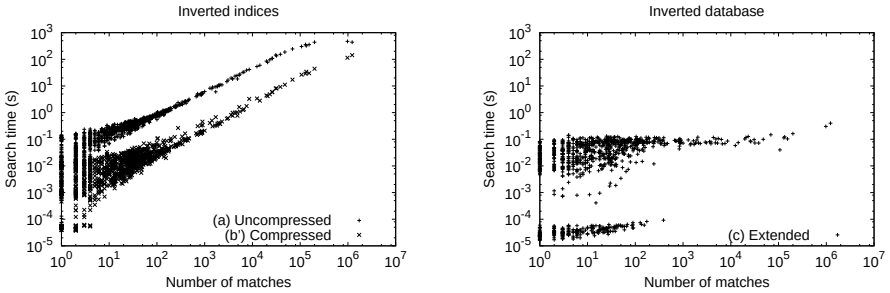

図 3 クエリ（1-gram）に一致する n-gram の数と検索時間の関係
Fig. 3 Dependencies between the number of results and the search time for unigram queries

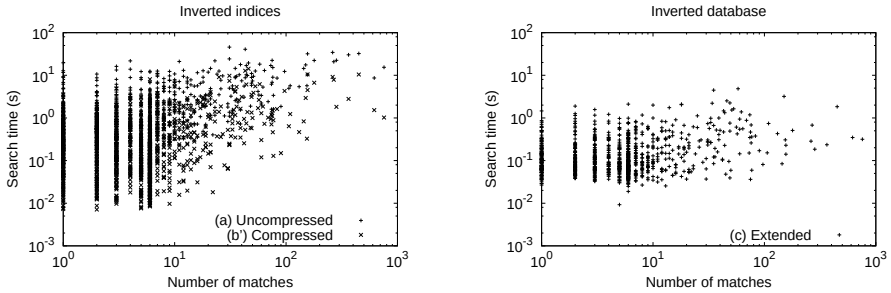

図 4 クエリ（5-gram）に一致する n-gram の数と検索時間の関係
Fig. 4 Dependencies between the number of results and the search time for 5-gram queries

う欠点はあるものの，(c) の平均検索時間は (b) と比べて約 1/10 以下であり，(b′) より高速な n-gram 検索システムとなっている．さらに，(c) は HDD 上で高い性能を出せるように設計されているため，SSD を利用する (b′) より容易に運用できるという利点を持つ．

次に，ディスクアクセスの効率に関する実験結果として，1-gram をクエリとする検索について，クエリに一致する n-gram の数を横軸，検索時間を縦軸にとったグラフを**図 3** に示す．転置索引 (a) と (b′) による実験結果を示す左側のグラフでは，クエリに一致する n-gram の増加にともなって検索時間が大きく悪化している．一方，転置データベース (c) による実験結果を示す右側のグラフでは，クエリに一致する n-gram の増加にともなう検索時間の悪化が小さく抑えられている．そして，ディスクアクセス以外に検索時間を大きく左右する要因が存在しないため，転置データベースにおけるディスクアクセスが効率的であることがわかる．なお，一部のクエリが極めて短時間で処理されているのは，必要なデータがディスクキャッシュとしてメモリ上にロードされていたことが理由と考えられる．

複雑なクエリ (5-gram) の検索に対する同様のグラフを**図 4** に示す．図 3 と比較した場合，(a) と (b′) の重なりが大きくなり，短時間で処理されるクエリがなくなるなどの違いはあるものの，同様の傾向が示されている．この結果は，格納位置のマージや n-gram に対するフィルタリングが検索時間を左右することを表すとともに，ディスクアクセスの効率が検索時間に与える影響の大きさを表している．

最後に，転置データベースとトライ索引[13] の比較を**表 6** に示す．ディスク使用量および索引構築時間がトライ索引の欠点であることは明らかであり，転置データベースはトライ索引より容易に構築・運用できることがわかる．一方，検索時間の面では，検索対象が異なることなどを考慮しても，ワイルドカードを含むクエリについて，トライ索引の方が転置データベースより高速に検索できている．しかし，トライ索引には，索引語の出現位置を問わない単純な AND 検索などにおいて，複数のトライを探索しなければならないという欠点がある．

表 6 転置データベースとトライ索引の比較
Table 6 Differences between an inverted database and a trie index

	転置データベース	トライ索引[13]
コーパス	1-gram から 7-gram まで約 32 億件	7-gram のみ約 5.7 億件
ディスク使用量	150 GB	490 GB
索引構築時間	1 GB のメモリで 24 時間以内	4 GB のメモリで 4–5 ヶ月以上
検索時間	5-gram の AND 検索で平均 0.22 秒（出現位置を制限しても同程度）	任意の 1-gram に一致するワイルドカードを 0–3 個含むクエリの検索で平均 0.05 秒

以上の実験結果をまとめると，転置索引による検索システムの性能向上に SSD が有効であることや，大規模 n-gram データの検索システムにおいて転置索引の展開が検索時間を大幅に短縮できることがわかる．また，転置データベースによる検索は高速であり，トライ索引と比べても平均検索時間の差は大きくない．さらに，短時間で検索システムを構築できるという利点を持つため，試験的なシステムの提供にも適しており，転置データベースは，単独の検索システムとして，あるいはトライ索引の補助を目的としても有効であるといえる．

5. おわりに

本稿では，転置索引を n-gram データに適用する方法を説明し，ディスクへのランダムア

Vol.2009-DBS-148 No.12
Vol.2009-FI-95 No.12
2009/7/28

クセスが問題になることを述べた．また，ディスクアクセスによるボトルネックを緩和する手法として，転置データベースが有効であることを示した．そして，トライ索引との比較により，転置データベースによる大規模 n-gram データの検索システムが実用的であることを示した．

今後の課題としては，トライ索引の研究[13] で述べられている分かち書きの問題を解決することに加えて，品詞や概念などのクラスにより検索できる枠組みを提供することが挙げられる．

参 考 文 献

1) Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, London, England, 6 edition, 2003.
2) Keiji Shinzato, Tomohide Shibata, Daisuke Kawahara, Chikara Hashimoto, and Sadao Kurohashi. TSUBAKI: An Open Search Engine Infrastructure for Developing New Information Access Methodology. In *IJCNLP2008*, pp. 189–196, Hyderabad, India, January 2008.
3) Thorsten Brants and Alex Franz. Web 1T 5-gram Version 1. `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13`, 2006.
4) 工藤拓, 賀沢秀人. Web 日本語 N グラム第 1 版. `http://www.gsk.or.jp/catalog/GSK2007-C/catalog.html`, 2007.
5) Andrew Finch, Etienne Denoual, Hideo Okuma, Michael Paul, Hirofumi Yamamoto, Keiji Yasuda, Ruiqiang Zhang, and Eiichiro Sumita. The NICT/ATR Speech Translation System for IWSLT 2007. In *IWSLT 2007*, pp. 103–110, Trento, Italy, October 2007.
6) Marcello Federico and Mauro Cettolo. Efficient Handling of N-gram Language Models for Statistical Machine Translation. In *the 2nd Workshop on Statistical Machine Translation*, pp. 88–95, Prague, Czech Republic, June 2007.
7) Jing Zheng, Wen Wang, and Necip Fazil Ayan. Development of SRI's Translation Systems for Broadcast News and Broadcast Conversations. In *Interspeech 2008*, Brisbane, Australia, September 2008.
8) Ghinwa F. Choueiter, Stephanie Seneff, and James R. Glass. New Word Acquisition Using Subword Modeling. In *Interspeech 2007*, pp. 1765–1768, Antwerp, Belgium, August 2007.
9) Andrew Carlson and Ian Fette. Memory-Based Context-Sensitive Spelling Correction at Web Scale. In *ICMLA 2007*, pp. 166–171, Cincinnati, Ohio, USA, December 2007.
10) Steven Bethard and James H. Martin. Learning Semantic Links from a Corpus of Parallel Temporal and Causal Relations. In *ACL-08: HLT, Short Papers (Companion Volume)*, pp. 177–180, Columbus, Ohio, USA, June 2008.
11) Martin Klein and Michael L. Nelson. Correlation of Term Count and Document Frequency for Google N-Grams. In *ECIR 2009*, pp. 620–627, Toulouse, France, April 2009.
12) John A. Dundas III. Implementing Dynamic Minimal Prefix Tries. *Software—Practice and Experience*, Vol.21, No.10, pp. 1024–1040, October 1991.
13) 関根聡. N グラム検索エンジン — Google 日本語 7 グラムを使って —. 言語処理学会第 14 回年次大会発表論文集, pp. 745–748, March 2008.
14) Satoshi Sekine. A Linguistic Knowledge Discovery Tool: Very Large Ngram Database Search with Arbitrary Wildcards. In *Coling 2008: Companion volume: Demonstrations*, pp. 181–184, Manchester, UK, August 2008.
15) Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
16) 青江順一．ダブル配列による高速ディジタル検索アルゴリズム．電子情報通信学会論文誌, Vol. J71–D, No.9, pp. 1592–1600, 1988.