N グラム検索エンジン
－ Google 日本語７グラムを使って －

関根聡
ニューヨーク大学

1. はじめに

現在、自然言語処理の研究は、いくつかのクラスへの分類問題に帰着できるような構文的解析から、教師付学習では解決が非常に難しい意味的な問題へとその視点は移ってきている。文章における意味的な問題を解決するためには、その文章には書かれていないバックグラウンドの知識が必要になり、その解決方法は、非常に狭くドメインを絞り人手で知識を書き出すと言った方法以外に良い手段は見つかっていない。この問題は 1980 年代の人工知能の研究で"Knowledge Bottleneck"と呼ばれ、当時の人工知能研究が頓挫した一つの原因と考えられている。しかし、当時と比較して現在の我々は、テラ単位の非常に大きなコーパスを持ち、それを解析できるような計算機リソースを手に入れることができる。現在の自然言語の研究では、このような幸運な資源を利用して、コーパスからの意味的知識の獲得が盛んに行なわれている。例えば (Hearst 92) (Collins and Singer 99) (Brin 99) (Hasegawa et al. 04)がある。これらの多くの研究は、単語やフレーズの近傍のコンテキストを手がかりにして意味的知識の抽出を行なっている。例えば(Hearst 92)は、"NP such as NP"といった Lexico Syntactic Pattern (LSP)と呼ばれる表現から上位下位関係を抽出している。また、(Hasegwa et al. 04)では、固有表現の間に現れる 5 単語までのコンテキストを用いて、固有表現間に存在しうる意味的な関係の抽出を行なっている。

このような意味的知識の獲得は、コーパスのサイズが大きいほど、広く深い知識を獲得することができる。しかしながら、コーパスの規模が大きくなると、処理スピード等の面で問題が生じる。例えば、我々の所で 80 年分の英語の記事をスキャンしてある LSP にマッチする表現を抽出したところ、数十時間もの時間がかかった。もし、この数時間が数秒（またはそれ以下の単位）になれば、インターラクティブな操作が可能になったり、数百万規模のコンテキストの抽出が可能になったりと、この分野の研究において様々な質的変化を引き出すことが期待できる。

この問題を解決するため、いくつかの方法が試みられている。一つは Google などの WEB 上の商用検索エンジンを使う方法である (Chklovski and Pantel 04)。しかし、それには以下のような問題が考えられる。

- 得られる検索結果の数に限界がある
- アクセス数に限界がある
- 解読不可能なランク付けに依存する
- 時期、場所によって結果が異なる
- インターネットを介しているために遅い

また、別のアイデアとして、自前の検索エンジンを開発するというアイデアも存在する。(Cafarella and Etzioni 05) (Shinzato et al. 08) (Pantel 07)。自らの検索エンジンを開発することは、検索表現に POS や係り受けを含めるなど任意のデザインが可能になるという大きな利点が存在するが、その開発は非常に大規模になってしまう。多大なコストも必要であるため誰でもがそれを開発できるわけではなく、また、より重要なことに、誰もがシステムを自由に使えるわけでもない。

本研究の目的は、上記とは異なる方法で知識獲得の研究ができるようなプラットフォームを、普通の計算機を持つユーザーにも提供できるツールとして開発することである。その要求仕様は以下の通りである。

要求仕様

1. 任意のワイルドカードを含む N グラムを１～10 億のデータ中から検索。例えば「＊などの＊」「する＊さん」「＊で＊から＊まで」「＊大統領の＊政策＊＊」
2. 頻度とワイルドカードの中身を出力
3. 普通の検索要求に対し１秒以下で出力
4. １台の PC（１CPU）上で動作
5. ４GB 程度のメモリ
6. インデックスのため 750~400GB 程度のディスク容量

2. アルゴリズム

基本的検索アルゴリズムとして、転置インデックスとトライを検討した。その結果、いくつかの理由から転置インデックスでは要求仕様を満たさないと判断し、トライを採用した。この詳細については紙面の都合上割愛する。

トライによる検索は、検索のためのインデックスを木構造で表現することに特徴がある。トライ構造は知られているように、要素の順序に深く依存している。つまり、例えば 7 グラムに対してその単語順にトライを作ったとすると、「* * * など の 本 を」といった形でワイルドカードがリテラルの前にある検索要求に対する検索では、ほぼ全ての木を走査しなければ答えが得られない。この問題に対する解決策としては、すべてのワイルドカードとリテラルの位置的な組合せに対してトライを用意するという方法が考えられる。例えば、上記の検索要求例に対しては、4，5，6，7 番目の単語の順になったトライを作るという方法である。しかしながら、容易に気がつくように、この方法では膨大な組合せができ、７グラムでは２^{7}＝１２８個のトライが必要になり、元々のNグラムのデータ量を考えると、検索エンジンの実現が非常に難しくなると言わざるを得ない。次のセクションで説明するようにトライ自身の縮小のために 2 つの方法を導入しているが、要求仕様の 6 を実現するために、以下の方法で組合せの数を縮小した。まず一つ目は、ワイルドカードがサフィックスにある場合には、そこにリテラルがある場合のトライで代用することと、二つ目は、常に前方から後方の順に単語を並べてトライを作るのではなく、適切な位置からトライをスタートさせるという方法である。例えば、5 グラムにおいて、ABCDEという並びに対するトライは、ABCD*、ABC**、AB***、A****という検索パターン（アルファベットはリテラル、＊はワイルドカード）に対しての検索でも使用する。また、BCDEAというトライパターンはABCDEというパターンと重複せずに効率的に多くの検索パターンに対応することが可能であるため、効率的に組み合わせられる。

試行錯誤の結果、Nグラム全ての検索パターンをカバーするための最低限必要なトライパターンの数は、$_{N/2}C_N$であることがわかった。詳細な証明は割愛するが、基本的な考え方は以下の通りである。ある特定の数（５グラムの時には０から５個のいずれか）のワイルドカードを含むすべての検索パターンを考える。その検索パターンの数の中では、ワイルドカードの数がN/2 個（5 グラムの際には２か３個）である検索パターンの数が最大であり、そのすべての検索パターンを１つずつカバーするトライパターンを作れば、トライパターンの数は$_{N/2}C_N$となる。同じ数のワイルドカードをもつ複数の検索パターンをカバーできるトライパターンは存在しないため、$_{N/2}C_N$が全ての検索パターンをカバーできる最小の数のトライパターンと言うことができる。表１にはNが 3 から 9 の時の、全検索パターンの数と最小のトライパターンの数を示す。また、図１には、5 グラムに対する最小パターンセットの例を示す。ちなみに、我々はNが 9 までの最小トライパターンを作成することに成功している（10 以上については、まだ試していない)。

表１　N と全パターン、最小パターンの数

N	全パターンの組合せ(2^N)	最小パターン数($_{N/2}C_N$)
3	8	3
4	16	6
5	32	10
6	64	20
7	128	35
8	256	70
9	512	126

トライパターン:　検索パターン

ABCDE:　ABCD*, ABC**, AB***, A****
BCDEA:　*BCDE, *BCD*, *BC**, *B****
CDEAB:　A*CDE, **CDE, **CD*, **C**
DEABC:　AB*DE, A**DE, ***DE, ***D*
EABCD:　ABC*E, AB**E, A***E, ****E
ACDEB:　A*CD*, A*C**
BDEAC:　*B*DE, *B*D*
CEABD:　A*C*E, **C*E
DABCE:　AB*D*, A**D*
EBCDA:　*BC*E, *B**E

ABCDE と*****はすべてのパターンがカバー

図１　５グラムのトライパターンセット

システムは検索要求を受け取った後、ワイルドカードの有無やその位置によって、その検索パターンを作りだし、その検索パターンをカバーするトライパターンを特定した後に、検索要求をそのトライパターンの順に並べ替え、実際の検索を対応するトライに対して行なう。

3. システムの実現

データサイズの縮小

まず、トライのサイズを縮小する二つの方法について述べる。一つは、トライの実現でよく行われるように、トライのある段階でそのノードに達するデータがひとつしかない場合には、それ以降のデータを木構造ではなく、そのままのデータの列挙として表現することでデータを縮小する。ただ、今回の場合には残りのデータもトライ内部には格納せずに、あるノードにおけるデータが一つになった時点で、トライ内部には Ngram の ID のみを記録する。これは、数多くのトライが存在するため、重複した往訪の格納を省く効果があり、インデックスの量の削減に大きく寄与している。二つ目の方法は、一般にトライは更新することを前提としてデータ構造が設計されるが, 今回は一度 N グラムのセットが与えられた後は更新することはない。したがって、一般的なトライのデータ構造の内、親ノードへのポインタと兄弟ノードへのポインタを省くことができる。各ノードには、最初の子ノードへのリンクと、子供の数だけが記録してある。この二つの方法によって、約 70%ものインデックスサイズの削減が実現できる。

セグメント

各トライのサイズは、まだ、要求仕様の 5 を満たすほどには小さくなっていない。したがって、それぞれのトライを細かく切断しなければならない。切断されたセグメントは文字順になっており、検索要求に対応するセグメントを探すことは容易に実現できる。ただし、頻繁に使用される単語が最初にある場合などには対応するセグメントは1つ以上あることに注意しなければならない。この方法はデータサイズの縮小には役立っていないが、インデックスをメモリのサイズ内に押さえるために必要なことである。また、プログラムでは mmap を使用してオンデマンドに必要な部分だけをメモリに持ってくるようになっており、高速な検索を実現している。また、セグメントへの分割はインデックスを作成する際にも非常に役に立つ。

システムの動作

システムの動作概要を図 2 に示し、以下に順を追って説明する。

1） 入力された検索要求の処理をする。単語を検索し、すべてを ID に直す。未知語があった場合には、対応する N グラムは存在しない。ワイルドカードの位置により検索パターンを特定する。検索要求が「＊＊で＊から＊まで」であった場合には、検索パターンは、”**C*E*G”になる。そして、これに対するトライパターンを図1のようなテーブルから特定する。

2） 該当トライをアルファベット順に並べて切断したセグメントの中から、今回の検索要求に対応するセグメントを特定する。セグメントは 1 つ以上の場合もある。

3） トライパターンの順に単語を置き換えた入力によりトライの検索を行なう。もし、検索がトライの中間ノードで終わった場合には複数の N グラムがマッチしたことになる。終端ノードで終わった場合には1つの結果がマッチしている。もし、検索の途中で終端ノードになった場合には、マスタ DB にある対応する N グラムと入力を比較し、マッチするかどうかを判断する。

4） マッチした 1 つまたは複数の N グラムを出力する。現在のシステムでは出力は、頻度のみ、N グラム ID のみ、N グラムの 3 種類を用意してある。

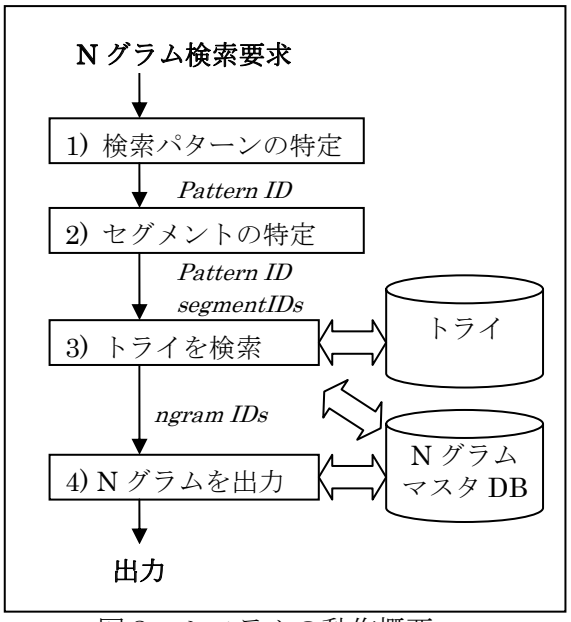

図 2　システムの動作概要

4. 実験

今回の実験では、GSK から配布されている「Web 日本語 N グラム第 1 版」の 7 グラムを使用した。このデータは Google によって約 200 億文の日本語データから作成した 5.7 億異なりの 7 グラムデータである。

検索結果の具体例

本ツールはもちろん「はじめに」で挙げたような、上位下位関係、2 項関係、固有表現抽出、固有表現間の関係の抽出などの意味的な知識獲得に非常に役に立つと考えられる。また、それ以外でも「どこかの大統領の具体的政策への態度」のような個別の知識を「＊大統領の＊政策＊＊」というパターンを用いて以下のように抽出することもできる。ツールの大きな利点の一つにはこのようなパターンをインターラクティブに調整できることにもある。

ブッシュ	大統領の	イラク	政策	を批判
サルジコ	大統領の	新	政策	に不安
プーチン	大統領の	チェチェン	政策	を批判
ブッシュ	大統領の	エネルギー	政策	に反対
米	大統領の	北朝鮮	政策	は失敗

データサイズと実行時間

今回の 7 グラムのインデックスは、トライノード数が 1 トライパターンにつき約 9 億個、インデックスファイルは合計で 490G 程度であった。検索の実行時間を測定するために、7 グラムをランダムに 1000 個選び、その中の 0 個から 3 個の単語をワイルドカードに置き換えた検索要求を作成し、検索エンジンを走らせたところ、4 GB のメモリを載せた Xeon 2GHz の PC—Linux 上で平均 0.05 秒で結果が得られた。

5. 今後の発展

本システムは単なるツールである。しかしながら、コーパスからの知識獲得において非常に強力なツールになると確信している。当然、次のステップは知識獲得のために本ツールを使うことであるが、それとは関係なく問題点や改良のアイデアについて述べる。

任意数のワイルドカードの指定

現在のシステムでは、ワイルドカードは 1 単語としてしか指定できない。本来は「する＊さん」といった場合に、任意の数のワイルドカードとのマッチングができるようになっていると便利である。このためには、7 以下の N グラムも同時に検索できることが必要になるが、頻度のスレッシュホルドの影響で、7 グラムのインデックスのみでそれ以下の N グラムをも対応することはできない。1～6 グラムのインデックスも作成する以外に方法はないが、その場合にはデータサイズの問題が生じる。

柔軟な検索要求

現在の検索要求では OR や NOT は使用できない。これらが可能になると、より便利な検索を実現させることができる。これは、現在の検索結果に対して後処理を行なうことによって実現できるのではないかと考えている。

形態素解析の問題

日本語の場合には、単語の単位は形態素解析に頼らざるを得ない。しかし、ユーザーが思った単語と違った形でシステムが単語分割をした場合には想定した N グラムが得られないことが起きる。

インデックスの作成

現在の我々が持つ環境ではインデックスの作成は延べ 4～5 ヶ月と非常に時間がかかっている。現在 64GB のメモリを持つ計算機を購入して、その上で高速にインデックスを作成する方法を検討中である。

参考文献

Web 日本語 N グラム第 1 版 by Google. GSK カタログ GSK2007-C

M. J. Cafarella and O. Etzioni. “A Search Engine for Natural Language Applications”. WWW-05.

Sergey Brin. “Extracting Patterns and Relations from the World Wide Web”. Workshop on Web and DataBase-98.

Timothy Chklovski and Patrick Pantel. “VerbOcean: Mining the Web for Fine-Grained Semantic Verb Re-lations”. EMNLP-04.

Michael Collins and Yoram Singer. “Unsupervised Models for Named Entity Classification”. EMNLP-98.

Marti A. Hearst. “Automatic Acquisition of Hyponyms from Large Text Corpora”. COLING-92.

Takaaki Hasegawa; Satoshi Sekine; Ralph Grishman “Discovering Relations among Named Entities from Large Corpora”. ACL-04.

Patrick Pantel. “Data Catalysis: Facilitating Large-Scale Natural Language Data Processing”. 2007. ISUC-07.

Keiji Shinzato, Tomohide Shibata, Daisuke Kawahara, Chikara Hashimoto and Sadao Kurohashi. “TSU-BAKI: An Open Search Engine Infrastructure for Developing New Information Access Methodology”, IJCNLP-08