決定的有限状態変換器を使った日本語係り受け解析の実装

関根 聡
New York University, Computer Science Department
715 Broadway, 7th floor, New York, N.Y. 10003 USA
sekine@cs.nyu.edu

1 はじめに

決定的有限状態変換器 (Deterministic Finite State Transducer) は文字列を解析するための高速な装置と考えられている。長さがnの文字列を解析するための時間はO(n)であり、文長に比例する。最近、有限状態変換器を自然言語処理に応用した研究が見られる。

本論文では、この手法を応用した日本語の係り受け解析の実装を示す。決定的有限状態変換器による高速性を実現するために、若干の精度の劣化 (京都大学コーパスに基ずいて理論的に最大1.3%程度と推測される) を代償とした。システムは1文あたり0.24ミリ秒で解析し、データのサイズも約188KBと非常に小さなもので実現できた。

一般に、構文解析は句構造文法を利用したChart法やLR法などが知られているが、その解析時間は文長に対して3乗または、それ以上の時間がかかる。関根ら [関根ら 99] は、文長の2乗の時間で日本語の係り受け解析を行なう方法を提案したが、今回提案する方法は解析時間は理論的に文長に比例し、実験的にも文長に比例していることが確認できた。

係り受けの正解率は、語彙の情報をほとんど使用していないにかかわらず約81%である。この正解率は、デフォルト方式やベースラインの解析正解率に比較して、それぞれ17%,9%程高い。また、同様のタスクでの現在知られている最高の正解率までのギャップ (約7%) は語彙的または意味クラスの知識を導入することによって解決していくであろうと考えている。

2 背景

日本語の係り受けには、主に以下の特徴があるとされている。ここではこれらの特徴を仮定として採用し、解析手法を作成した

(1) 後方を修飾する。

(2) 係り受け関係は交差しない。(非交差条件)

(3) 係り要素は受け要素を1つだけ持つ。

表1に、ある文節を解析している時の後方にある係り可能な文節の数 (候補の数) と、実際にその内のどこの文節 (対象にしている文節から何番目の文節) に係るか (正解の位置) の関係を示す。データは京都大学コーパス [黒橋ら 97] の1月1日から1月8日まで (後に述べる実験のトレーニングデータと共通) を利用している。この表から、実際に候補となる文節の数は限られていることが分る。候補中の係り先の位置としては、左から4つ目までと最後の文節に係る割合が合計で98.7%と、非常に限られている。(表1では、この基準によって省かれる部分、つまり、5つ目以降最後の前までの文節が正解の係り先であるという部分をイタリックで示している。) 係り先の候補の範囲がかなり偏っているのであるから、係り先の候補の範囲を限定し、その範囲にある文節のなんらかのクラス (例えば主辞の品詞) のパターンをすべて記憶し、あらゆる種類のパターンに対してどの文節に係るべきか記憶してしまえば、定数時間での選択が可能になり、文長に比例した解析時間が実現できる。このアイデアを決定的有限状態変換器によって実装した。

候補の数	正解の位置							候補の数	正解の位置													
	1	2	3	4	5	6	7		1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	7918							8	340	110	83	47	*38*	*57*	*68*	121						
2	12824	4948						9	137	33	28	17	*19*	*21*	*19*	*24*	46					
3	10232	3292	3126					10	51	11	10	7	*3*	*10*	*3*	*6*	*8*	22				
4	6774	2244	1439	1968				11	17	5	2	0	*2*	*2*	*4*	*2*	*4*	*2*	10			
5	3692	1294	888	660	1114			12	5	1	1	2	*0*	*0*	*0*	*0*	*2*	*2*	*1*	3		
6	1843	614	398	331	*291*	551		13	3	0	1	0	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	0	
7	848	278	176	133	*133*	*125*	264	14	0	1	1	0	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	2

表 1: 候補の数と正解の位置

3 決定的有限状態変換器の実装

決定的有限状態変換器では、基本的には文節のパターンを状態とし、係り元の文節を遷移における入力に、どの文節に係るかを出力をした決定的有限状態変換器を作成した[1]。

係り先のクラスとしては予備のデータ (held-out data) で特徴のチューニングを行ない、JUMAN の品詞体系と読点があるかないかの情報を組み合わせた 18 のクラスを採用した。

遷移における入力は、現在解析しようとしている文節の後方部分の情報 (最後の助詞、または活用しているものがある場合にはその情報、そうでない場合には主辞) と読点があるかないかという情報に基づき、同様にチューニングの過程を経て 40 のクラスを採用した。

遷移における出力は、5 つの係り先候補の内、どれにかかるかという情報であり、1 から 5 までの数字で表わされる。トレーニングデータ中に、同じ状態における同じ入力でありながら違う出力にすべき場合が存在することが考えられる。これは、状態や入力のクラスが細分化されきっていないということであったり、係り先と係り元の情報だけでは曖昧で決定できないというような理由が考えられる。このような場合には、トレーニング中で最も頻度の高い出力を利用し、極力高い精度を求めた。

[1] 厳密には、出力に対する入力と、次の状態を決定するための入力に違うクラスを利用しており、説明よりもやや複雑な変換器となるが、概念的には、両者のクラスを掛け合せたクラスを想定し、それらに対して適切な出力と次の状態を設定することで変換器は作成できる。また、実装においては、より簡単に両者を分けることができるので問題はない。

データのサイズ

上記のような設計によると、状態の数は 18 の 5 乗で 1,889,568 個、そしてその各状態から出る遷移の数が 40 なので、合計 75,582,720 の遷移が存在することになる。これをそのまま実装すると最低でも数百メガのデータが必要になる。高速化を実現するためには、プロセスがメモリの領域に入っていることが必要であり、普通にある計算機で動作不可能となってしまっては、魅力的でない。また、これ以上の拡張が不可能になってしまうことが考えられる。

そこで、以下の 2 つの工夫を施した。まず、状態は後方の文節のクラスから一意に決求められるようにし、ある状態から別の状態に移る際には、新たに入ってくる文節の種類と状態を遷移する際の出力から自動的に計算できるようにした。このようにすれば、遷移先の状態を記録しておく必要はなく、データサイズの縮減に大きく役に立つ。また、これにより最初の脚注に述べたように、2 種類の入力を区別することが容易にできる。2 つ目の工夫としては、デフォルトの文節係り受け先を決定することによってデータサイズをさらに縮小した。日本語の係り受けでは、ある文節が次の文節に係る割合は 64%と非常に多い。したがって、この出力が行なわれるような遷移はデフォルトとしてデータには記録しない。つまり、ある状態からの遷移の際に、入力として受け取ったものが遷移の候補の中になければ、その出力は「次の文節」であるというようにした。この方法には別の意味での利点もある。未知の入力に対しては、次の文節に係るとするのが妥当であると考えられるが、上記

の方法を取れば、遷移の大部分 (90%以上) を占めるこの未知な遷移については何も記憶しないで良いということになる。未知の入力があった場合には、次の文節に係るという答を出し、次の状態を適切に計算し、新しい入力に備えればよいということになる。(未知の部分を減らそうという努力 (補間) については、後述する。)

上記の2つの工夫を組み合わせると、まったく情報のない状態というものも存在する。つまり、その状態に来た場合には、どのような入力であろうとも、次の文節を係り先として出力し、適切に計算される次の状態に移動すればよいという訳である。このような工夫により、実際にはほとんど訪れることのないような状態に関する情報は、まったく保持しないでよいことになる。

実際に、次に述べる補間を施す前では、1,006の状態について合計1,554の遷移を記録するだけで済み、補間を施した後でも10,457の状態について、合計31,316の遷移を記録するだけで済んだ。それぞれデータの量 (データファイルのサイズ) は、約15K byteと約188K byte程度である。

補間

トレーニングに使用する京大コーパスの量は、約8,000文と非常に少ない。平均文長は約10文節なので、72,000個のデータポイントがあることになる。これは、5文節の可能な組み合わせ (1,889,568個) より遥かに少なく、実際のシステムでデータスパースネスの問題に直面することは明らかであると思われる[2]。

そこで、未知の状態を補間するために、既存の構文解析器を利用した。つまり、大量の文を既存の構文解析器により解析し、その答を学習データに加えることにより、京大コーパスにはでてこなかった状態にも対処できるようにした。実際には毎日新聞の94年と、95年の内1月1日から10日を除いた部分 (ただし、効率化のために、文長は200までのものに限った。全部で2,536,229文) を内元らの係り受け解析システムで解析し、それをトレーニングデータに加えることで補間を行なった。

[2] コーパスにない状態に行った場合には次の文節に係るというデフォルトを採用することにより、プログラムの実行はできるし、それでも次の節で述べるようにかなりの精度が出る。

4 実験

この節では、決定的有限状態変換器を利用した係り受け解析の実験を報告する。実験に用いたコーパスは、京大コーパス (Version 2) の一般文の部分で、基本的にトレーニングには1月1日から8日までの7日分 (7960文)、試験には1月9日の1日分 (1246文) を用いた。パラメータなどのチューニングには1月10日の1日文 (1519文) を用いた (Heldout data)。解析システムへの入力文としては、正しい形態素、文節情報が付いている京大コーパスを利用した。

係り受け正解率

表2に係り受け正解率を示す。係り受け正解率とは、最後の文節を除いたすべての文節の係り受けが正しく解析されている率である。デフォルト方式は、つねにすべての文節は次の文節に係るとしたもので、ベースラインは人手により作成された簡単なルールベースの解析システムである。詳細は、[内元ら 98] にある。また、内元らのシステムは [内元ら 99] で報告されたシステムの正解率であり、トレーニングデータ、テストデータ共に同じものを使用している。本システムの係り受け正解率は、補間をしたもので81%、補間をせずにトレーニングデータのみを使用した場合には78%であった。補間をした結果は、デフォルト方式、ベースラインに比較して、それぞれ17%と9%ほど良く、限定された情報しか利用してないにも関わらず、比較的高い精度が得られた事を示している。内元らのシステムに比較すると、補間を行なったシステムでも7%程係り受け正解率が低くなっている。これは、内元らのシステムは、語彙の情報などの細かい情報や5つまでの素性の組み合わせといった総合的な情報も含め、約4万の素性を使用しているためであり、各文節あたり18の素性を使用し2つの素性の組み合わせまでしか使用していない本システムの精度がそれにおよばないのは理解できる[3]。今後、本システムにもなんらかの形で語彙情報などの細かな情報を加えていきたいと考えている。

[3] ただし、本システムでは5つの文節を同時に見ることによって、内元らのシステムでは得られない文脈的な情報を得ることができている。

システム	係り受け正解率
本システム (補間あり)	81.195 %
本システム (補間なし)	77.963 %
デフォルト方式	64.14 %
ベースライン	72.57 %
内元らのシステム	87.93 %

表 2: 係り受け正解率

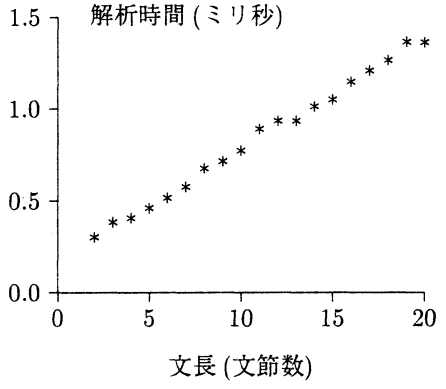

図 1: 解析速度

解析時間

次に、本システムの目標である解析時間について報告する。Pentium II, 500MHz PC 上の Linux で、"gcc -O" を使ってコンパイルしたシステムの解析時間は一文 (平均 10 文節) あたり平均で 0.24 ミリ秒であった。これは、KNP[Kurohashi 94] や内元ら [内元ら 99] のシステムに比べて約 100 倍高速である。また、文長と解析時間の関係を図 1 に示す。この実験では、上記に示した PC ではなく Ultra SPARC-I,170MHz を使用した。なるべく遅いマシンを利用して測定誤差を減らす目的である。それでも、大部分の文は測定時間の最小単位である 1 ミリ秒以下で解析できてしまうため、各文を 10 回解析する時間を用いて解析時間を求めた。図から、多少の定数時間が存在することと、解析時間は文長に比例することが分る。

トレーニング時間は、解析時間に比較してやや遅いという程度であり、京大コーパスの 8 日間からの学習は約 10 秒程度で終了する。

5 まとめ

決定的有限状態変換器を利用した日本語係り受け解析を提案した。1 文あたり 0.24 ミリ秒で解析を行なうこのシステムは、リアルタイムの情報検索、情報抽出、機械翻訳や音声認識など広い自然言語の応用分野において利用が可能と考える。例えば、Web の応用では、ページをダウンロードするとほぼ同時にそこにある文章の解析が終っているというシステムも考えられる。係り受け解析では、語彙情報なども取り込んだ精度向上の努力が行なわれている。本論文で提案する手法はこの方向に相反するものではない。今後は、これらの研究で得られた知見も含め語彙情報なども加えて、さらなる精度の向上を考えていきたい。また、本手法は 2 節で述べたような特徴を有する言語 (例えば、韓国語、トルコ語など) でも実現可能であると考えられる。

6 謝辞

本実験は京大コーパスに基づいて行なった。作成にあたった京都大学の黒橋氏を始め関係者に感謝と称賛の意を示したい。また、内元氏にはデータの整備や実験の手伝いをしてもらった。ここに記して感謝の印としたい。

参考文献

[黒橋ら 97] 黒橋、長尾:「京都大学テキストコーパスプロジェクト」第 3 回自然言語処理学会年次大会 (1997)

[Kurohashi 94] Kurohashi, Nagao: "KN Parser: Japanese Dependency/Case Structure Analyzer", Workshop on Sharable NL Resources (1994)

[関根ら 99] 関根、内元、井佐原:「文末から解析する統計的係り受け解析アルゴリズム」自然言語処理学会 論文誌 Vol.6 No.3 (1999)

[内元ら 98] 内元、関根、井佐原:「最大エントロピー法に基づくモデルを用いた日本語係り受け解析」情報処理学会論文誌 vol.40, No.9 (1999)

[内元ら 99] 内元、村田、関根、井佐原:「日本語係り受け解析に用いるＭＥモデルと解析精度」第 5 回自然言語処理学会年次大会併設ワークショップ (1999)