用例主導型機械翻訳の超並列連想プロセッサIXM2による高速化

5B−5

大井 耕三　隅田 英一郎　飯田 仁　樋口 哲也†　北野 宏明‡
ATR自動翻訳電話研究所　†電子技術総合研究所　‡カーネギー・メロン大学

1　はじめに

用例主導型機械翻訳(EBMT)[1]では入力に類似した用例を検索しそれを利用して翻訳を行なう。その際、各用例ごとに入力との距離を計算するため、用例数の増大に伴って翻訳処理時間が増大するという問題がある。そこで、EBMTの高速化を実現するために、EBMTの処理全体の中で最も処理時間がかかる用例検索処理を、超並列連想プロセッサIXM2[2]を用いて実現した。

なお、実験は、従来の規則主導の機械翻訳では訳語選択が難しい「AのB」(A,Bは名詞)という形の日本語の名詞句を入力として行なった。

2　EBMTの用例検索処理

EBMTは用例とシソーラスの2つのデータベース、及び、解析、用例検索、生成の3つの処理からなる。

用例は「国際会議の登録に関する対話」のデータベースから抽出し、シソーラスは大野、浜西の体系[3]に準拠している。シソーラスの類語コードは3桁の数字で表されており、百の位が大分類を、十の位が中分類を、一の位が小分類を意味している(図1)。

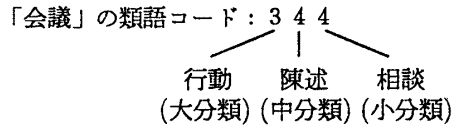


図1: 類語コードの例

本稿ではEBMTの中心となる用例検索処理を扱う。図2に名詞句「京都での会議」を使って処理の概要を示す。用例検索処理での類似検索処理(3)では入力と全用例との距離計算を行なう。距離計算では名詞の類語コード、名詞に付属している接尾語、助詞のそれぞれの距離と重みから全体の距離を求める(詳細は[1])。1用例あたりの計算時間は短いが用例数に比例した処理時間を要する。類語コード間の距離の求め方を図3に示す。

3　システム構成

今回の実験システムの構成を図4に示す。IXM2の連想プロセッサ1台が、トランスビュータボードを通してSPARCstation2と接続された構成である。IXM2は、電総研で開発された連想メモリマシンで、64台の連想

Implementation of Example-Based Machine Translation on a Massively Parallel Associative Processor, IXM2
Kozo Oi, Eiichiro Sumita, Hitoshi Iida, Tetsuya Higuchi† and Hiroaki Kitano‡
ATR Interpreting Telephony Research Laboratories
†Electrotechnical Laboratory
‡Carnegie Mellon University

名詞句「京都での会議」の解析結果
AM=「京都」, AC=004, AS=「」, NO=「での」,
BM=「会議」, BC=344, BS=「」
注) 名詞句「AのB」に対して各記号の意味は次の通り。
AM: Aの名詞の見出し,　BM: Bの名詞の見出し,
AC: Aの名詞の類語コード, BC: Bの名詞の類語コード,
AS: Aの接尾語の見出し,　BS: Bの接尾語の見出し,
NO: 助詞「〜の」の見出し

用例検索処理　↓↓ 入力

(1) 見出し完全一致検索
(2) 類語コード完全一致検索((1)で検索されなかった場合)
(3) 類似検索((1)(2)で検索されなかった場合)
(3-1) AC,BC,NO,AS,BSそれぞれに対する、入力と用例の距離(NO,AS,BSの場合、等しい時0、異なる時1、AC,BCの場合は図3参照)を求める。
(3-2) (3-1)のそれぞれの距離に重みを乗じ、その合計を全体の距離とする。
(3-3) (3-2)の距離の値が最小の用例を出力する。

↓↓ 出力

距離	AM	AC	NO	BM	BC	変換タイプ
0.4	東京	004	での	滞在	319	B in A
0.4	香港	004	での	滞在	319	B in A

図2: 用例検索処理の概要

条件	例	距離
$IC_1IC_2IC_3 = EC_1EC_2EC_3$	347 = 347	0
$IC_1IC_2 = EC_1EC_2, IC_3 \neq EC_3$	347 = 345	1/3
$IC_1 = EC_1, IC_2 \neq EC_2$	347 = 355	2/3
$IC_1 \neq EC_1$	347 = 855	1

注) IC, EC はそれぞれ、入力,用例の類語コードを、1,2,3はそれぞれ、百,十,一の位を表す。

図3: 類語コード間の距離の求め方

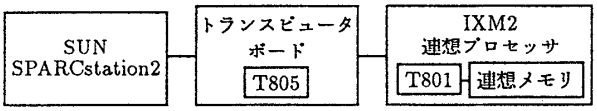


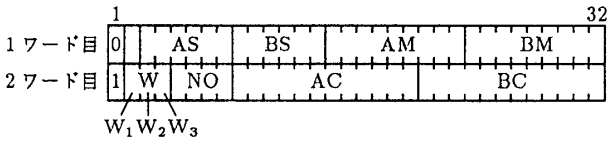
図4: 実験システムの構成

プロセッサと9個の通信プロセッサからなる[2]。各々の連想プロセッサはインモス社のトランスビュータ[1]に連想メモリを装備したものである。連想メモリ上のデータに対する検索・書き込みなどが並列に行なえる。連想メモリにEBMTの用例を格納し、用例検索の並列処理により高速化が実現できる。

[1]並列処理用言語Occam2用に設計された並列処理用32bitsマイクロプロセッサ。

4 連想メモリ上の用例のデータ構造

IXM2 の連想メモリは、1 ワード 40 ビットからなる。実験では 1 用例に対して 2 ワード (1 ワードの中では 32 ビットのみ使用) を用いた。その 2 ワードのデータ構造 (ビットフォーマット) を図 5に示す。AS, BS, AM, BM, NO の見出しデータは、文字列ではなくコード化したデータを格納する。W は、類似検索処理時に使用する書き込み用のエリアである。1 ビット目の 0, 1 は、1 ワード目と 2 ワード目を区別するためのものである。



注) W は書込み用のエリアで、W_1, W_2, W_3 の 3 ビットからなる。
その他の記号の意味は図 2参照。
AS, BS, AM, BM, NO にはコード化されたデータが入る。

図 5: 連想メモリ上の用例のデータ構造

5 IXM2 での用例検索アルゴリズム

IXM2 では、連想メモリに対して並列検索・並列書き込み命令が実行できる。連想メモリに対する検索・書き込み命令時には、命令の種類、検索あるいは書き込み用のワードデータ、マスク用のワードデータ[2]を与えるのが基本となる。図 2の用例検索処理は連想メモリ上で以下のように実現した。

【見出し完全一致検索】 AM, BM, AS, BS, NO が入力と一致する用例データの検索命令を実行し、検索された用例のアドレスを得る。[3]

【類語コード完全一致検索】 AC, BC, AS, BS, NO が入力と一致する用例データの検索命令を実行し、検索された用例のアドレスを得る。

【類似検索】

(1) 全用例の距離を 0 に[4]、用例データの W_1, W_2, W_3 の 3 ビット (図 5参照) を 0 に初期化する。

(2) 入力の AS, BS, NO 各々に対して、一致する用例データの検索命令を実行し、検索されなかった用例の距離に (1 × 重み) を加算する。

(3) 入力の類語コード AC, BC 各々に対して、(4) 〜 (9) の処理を行なう。

(4) 類語コードの百の位が一致する用例データの検索命令を実行し、検索された用例に対して、W_1 に 1 を書き込む命令を実行する。

(5) 類語コードの十の位が一致する用例データの検索命令を実行し、検索された用例に対して、W_2 に 1 を書き込む命令を実行する。

(6) 類語コードの一の位が一致する用例データの検索命令を実行し、検索された用例に対して、W_3 に 1 を書き込む命令を実行する。

(7) W_1 が 1, W_2 が 1, W_3 が 0 である用例データの検索命令を実行し、検索された用例の距離に (1/3 × 重み) を加算する。(図 3の 2 番目の条件に相当)

(8) W_1 が 1, W_2 が 0 である用例データの検索命令を実行し、検索された用例の距離に (2/3×重み) を加算する。(図 3の 3 番目の条件に相当)

(9) W_1 が 0 である用例データの検索命令を実行し、検索された用例の距離に (1 × 重み) を加算する。(図 3の 4 番目の条件に相当)

(10) 全用例の中から、距離が最小の用例を見つけ、その用例を出力する。(トランスビュータのみの処理)

[2]検索あるいは書き込みを無効にしたいビットをマスクするためのワードデータ。

[3]IXM2 のトランスビュータのメモリにこのアドレスと用例との対応表を持ち、この対応表を参照して検索した用例を出力する。

[4]距離はトランスビュータのメモリに格納する。

6 実験結果 (IXM2 vs. SPARC)

実験は、IXM2 と SPARC 上で EBMT の用例検索の処理時間を測定した。IXM2 は Occam2 言語で、SPARC は LISP 言語でプログラミングした。実験結果を図 6に示す。図は用例数を 100 から 1000 まで変化させた時の処理時間を示している。用例数 1000 でみると、IXM2 は SPARC に比べて約 12 倍高速であることが分かる。

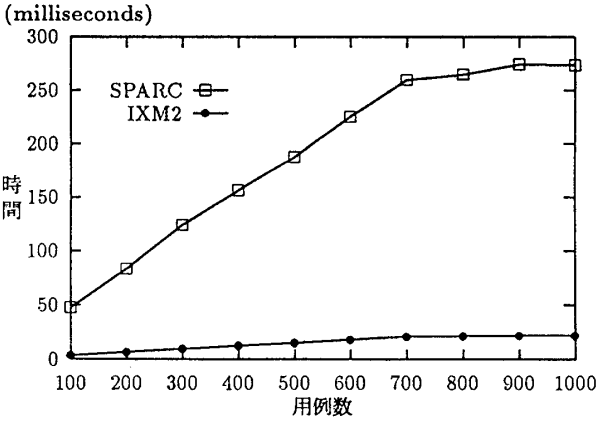


図 6: 用例検索の処理時間の比較

7 まとめ

用例主導型機械翻訳における用例検索処理を超並列連想プロセッサ IXM2 上に実現した。実験の結果、IXM2 を使った用例検索処理は、逐次計算機 SPARCstation2 の約 12 倍高速化できた。これにより自動翻訳電話のためのリアルタイムな処理を実現する見通しを得たと考える。今後は、より大規模な用例による実験、IXM2 以外の超並列コンピュータによる実験が必要と考える。

参考文献

[1] Sumita, E. and Iida, H. : "Example-Based Transfer of Japanese Adnominal Particles into English", IEICE TRANS. INF. & SYST., Vol.E75-D, No.4 (1992.7)

[2] Higuchi, T., Kitano, H., et al.: "IXM2: A Parallel Associative Processor for Knowledge Processing", Proc. of AAAI-91 (1991)

[3] 大野, 浜西 : "類語新辞典", 角川書店 (1984)