TR-I-0318

超並列連想プロセッサ IXM2 を使った 用例主導型機械翻訳の超並列化手法

Example-Based Machine Translation on
a Massively Parallel Associative Processor, IXM2

大井 耕三　　隅田 英一郎　　飯田 仁
Kozo OI　　Eiichiro SUMITA　　Hitoshi IIDA

1993.3

概要

用例主導型機械翻訳 (EBMT : Example-Based Machine Translation) では、入力に類似した用例 (原文と訳文の対) を検索しそれを利用して翻訳を行なう。全用例と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 を使って実現した。本稿では、IXM2 上での超並列化手法について述べる。

IXM2 に搭載されている連想メモリは、データに対する検索・書き込みなどが並列に行なえるため、EBMT の用例データを連想メモリに格納することにより、EBMT の用例検索処理の超並列化が可能となる。実験の結果、IXM2 を使った EBMT の用例検索処理は、逐次計算機 SPARCstation2 に比べ約 12 倍高速化できた。これにより自動翻訳電話に必須なリアルタイム処理を実現する見通しを得た。

ATR 自動翻訳電話研究所
ATR Interpreting Telephony Research Laboratories

目次

図目次

第 1 章

はじめに

用例主導型機械翻訳 (EBMT : Example-Based Machine Translation) [1] では、入力に類似した用例 (原文と訳文の対) を検索しそれを利用して翻訳を行なう。全用例と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。自動翻訳電話の実現にあたっては、翻訳処理にリアルタイム性が要求されるため、用例数の増大に伴う翻訳時間の増大が 1 つの問題となる。

一方、近年になって、超並列人工知能という新しい研究分野が Waltz[2] や Kitano 他 [3] による一連の活動によって注目されるようになってきた。そして現在は、米国を中心に超並列マシンの開発・商用化が積極的に行なわれつつある。日本での超並列マシンの開発・商用化はまだ数少ないが、その中で電総研で開発された超並列連想プロセッサ IXM2[4] がある。IXM2 は、Memory-Based Translation [5], や Memory-Based Parsing [6] で超並列化によりリアルタイムな処理を実現している。そこで、IXM2 を使って EBMT の用例検索処理を超並列化することを試みた [7]。

以下、まず第 2章で、EBMT の概要を述べ、第 3章で IXM2 を紹介する。そして、第 4章で IXM2 を使って EBMT の用例検索処理を超並列化する手法について述べる。第 5章では、IXM2 と逐次計算機 SPARCstation2 上で行なった実験結果を示す。最後に第 6章で、まとめと今後の課題などを述べる。

なお、本稿での実験 (第 5章参照) 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対象として行なった。

第 2 章

EBMT の概要

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：** 原文と訳文の対。本稿の第 5章実験では、「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た用例を用いている。
- **シソーラス：** 単語を共通の意味概念に従って体系化した辞書で、大野、浜西の体系 [8] に準拠している。シソーラスの各単語に付与されている類語コードは 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 2.1)。

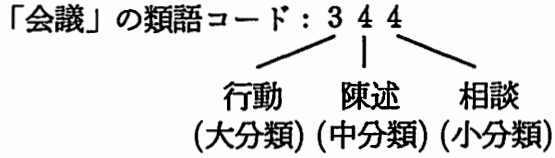


図 2.1: 類語コードの例

- **解析処理：** 入力文の形態素解析を行なう処理。
- **用例検索処理：** 全用例の中から入力文に最も類似した用例を検索する処理 (詳細は 2.1 で述べる)。
- **生成処理：** 検索した用例を基に訳文を生成する処理。

2.1 EBMT の用例検索処理

用例検索処理は、EBMT で中心的な処理で、EBMT 全体の処理時間の大半をこの処理が占めている。図 2.2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 2.3に示す。

「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」

↓↓ 入力

用例検索処理

(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 2.3参照) 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 2.2: 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 2.3: 類語コード間の距離の求め方

第 3 章

超並列連想プロセッサ IXM2

超並列連想プロセッサ IXM2 は、電総研で開発された連想メモリマシンで、意味ネットワークをハードウェアレベルでサポートする目的で開発されたものである。64 台の連想プロセッサと 9 個の通信プロセッサからなり、ホストプロセッサである SUN-4/260 の制御下で動作する [4]。各々の連想プロセッサはインモス社のトランスピュータ[1]に連想メモリ (4K ワード) を装備したものである。この連想メモリでは、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ため、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。

連想メモリ　連想メモリとは、記憶内容によるデータアクセスを基本機能とするメモリであり、CAM (Content Addressable Memory : 内容アクセスメモリ) と呼ばれることが多い。通常のメモリ (RAM) では、アドレスを用いてデータアクセスを行なうのに対して連想メモリでは検索データを入力し、これと各ワードの記憶データの内容を照合、検索し、該当する内容をもつワードに対してアクセスを行なう。連想メモリの基本機能は、検索データと記憶データとの一致照合 (一致検索機能)、一致したワードのアドレス生成、アドレスによる読み出し・書き込み動作である。IXM2 に装備されている連想メモリには、この基本機能の他に、複数のワードにまたがる検索機能、ビット単位の書き込み機能、検索結果に基づく書き込み機能などの機能を備えている。

[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。

第 4 章

IXM2 を使った EBMT の超並列化

4.1 システム構成

今回行なった超並列化実験のシステム構成を図 4.1に示す。IXM2 の連想プロセッサ 1 台を使い、トランスピュータボード (VOLVOX-1/S) を通して SPARCstation2 のホストマシンと接続した構成である。IXM2 及びトランスピュータボード上のプログラム言語は Occam2 を使用した。プログラミング及びコンパイル等はホストマシン上で行ない、コマンドによって、実行プログラムをトランスピュータボード及び IXM2 上のトランスピュータにロードし実行を行なう。

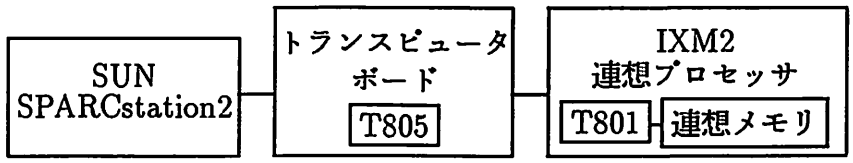
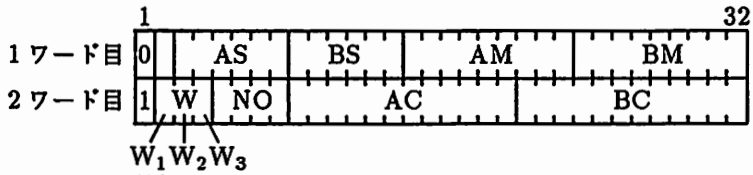


図 4.1: 実験システムの構成

4.2 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 4.2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している。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。EBMT における 1 つの用例中には、AC, BC の類語コードが複数存在するデータがあるが、このような用例を連想メモリ上に格納する際には、AC

とBCの組合せの数だけ用例を展開している。例えば、ACの数が2でBCの数が4である1つの用例の場合、ACとBCの組合せの数は8となるので、この用例は連想メモリ上の16(=2×8)ワードに格納している。



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2.2参照。
AS, BS, AM, BM, NOにはコード化されたデータが入る。

図 4.2: 連想メモリ上の用例のデータ構造

4.3 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[1]を与えるのが基本となる。図2.2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。[2]

【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[3]、用例データのW_1, W_2, W_3の3ビット(図4.2参照)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。

---

[1]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。

[2]IXM2のトランスピ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。

[3]距離はトランスピュータのメモリに格納する。

(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 2.3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 2.3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 2.3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。(トランスピュータのみの処理)

第 5 章

実験

5.1 実験の概要

実験は、IXM2 と SPARCstation2 上で EBMT の用例検索の処理時間を測定した。用例検索処理の検索タイプに応じて 4 種類 (図 5.1)、用例数は 100 から 1000 まで 100 ずつ変化させた時の時間測定を行なった。

実験の種類	内容
検索タイプ ：見出し完全一致検索	見出し完全一致検索でヒットする 入力 100 件のトータルの処理時間を測定し、 入力 1 件当りの平均処理時間を算出。
検索タイプ ：類語コード完全一致検索	類語コード完全一致検索でヒットする 入力 100 件のトータルの処理時間を測定し、 入力 1 件当りの平均処理時間を算出。
検索タイプ ：類似検索	類似検索が行なわれる入力 100 件の 入力 100 件のトータルの処理時間を測定し、 入力 1 件当りの平均処理時間を算出。
検索タイプ ：混在	見出し完全一致検索でヒットする入力、 類語コード完全一致検索でヒットする入力、 類似検索が行なわれる入力が混在している 入力 100 件[1]のトータルの処理時間を測定し、 入力 1 件当りの平均処理時間を算出。

図 5.1: 実験の種類

[検索タイプ：混在] の実験における、見出し完全一致検索でヒットする入力、類語コード完全一致検索でヒットする入力、類似検索が行なわれる入力の割合を図 5.2に示す。用例数が増すにしたがって、見出し完全一致検索でヒットする入力及び類語コー

ド完全一致検索でヒットする入力の割合が増えている。

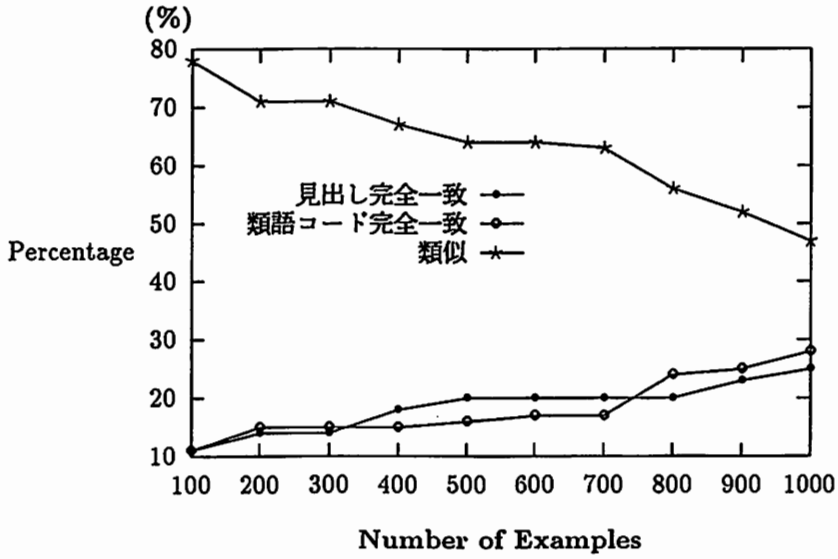


図 5.2: [検索タイプ：混在] における検索タイプの割合

5.2 実験結果

実験結果を図 5.3〜図 5.6に示す。4種類の実験結果のいずれにおいても、IXM2 が SPARC よりも高速であった。特に、[検索タイプ：混在] の実験 (図 5.3) では、用例数 1000 でみると、IXM2 は SPARC に比べて約 12 倍高速となった。

図 5.4では、SPARC の処理時間が上下に揺れているが、これは Lisp の時間関数が 1/100 秒の精度でしか測定できなかったためで、若干の誤差を含んでいる。

図 5.5では、IXM2 において用例数 800 と 900 で処理時間の増加が目立っている。今回の実験では、連想メモリ上の 1 つの用例は、その AC と BC の類語コードの組合せの数だけ展開した形で格納している (4.2節参照)。そのため、入力に対しても同様に展開している。類語コード完全一致検索は、その展開した入力それぞれに対して行なっているため、検索された用例が重複している場合にはその重複を解消する処理を行なっている。用例数 800 と 900 ではその重複の割合が比較的多かったことが、処理時間の増加の原因となっている。

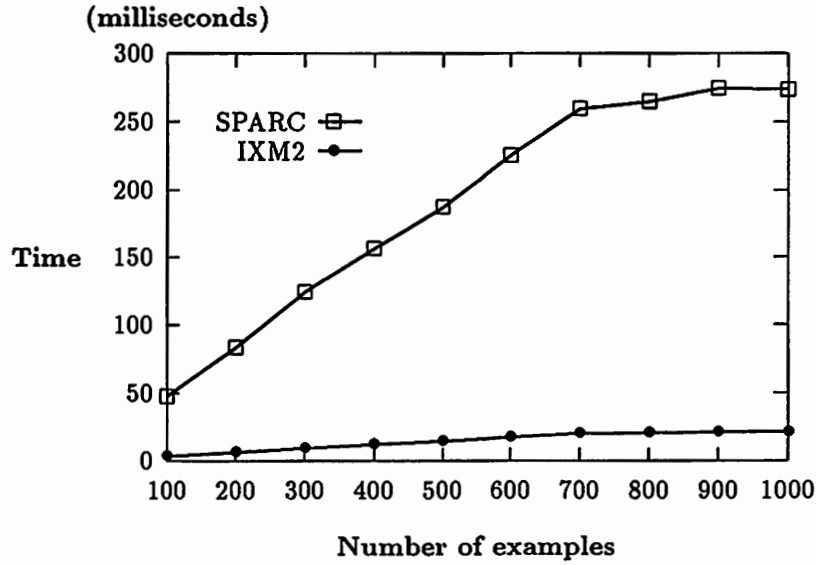


図 5.3: 用例検索の処理時間の比較 (検索タイプ：混在)

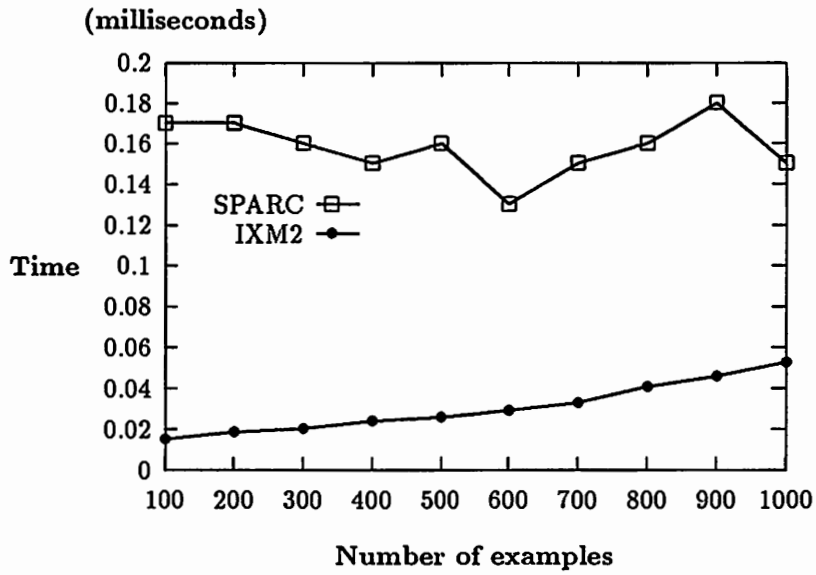


図 5.4: 用例検索の処理時間の比較 (検索タイプ：見出し完全一致検索)

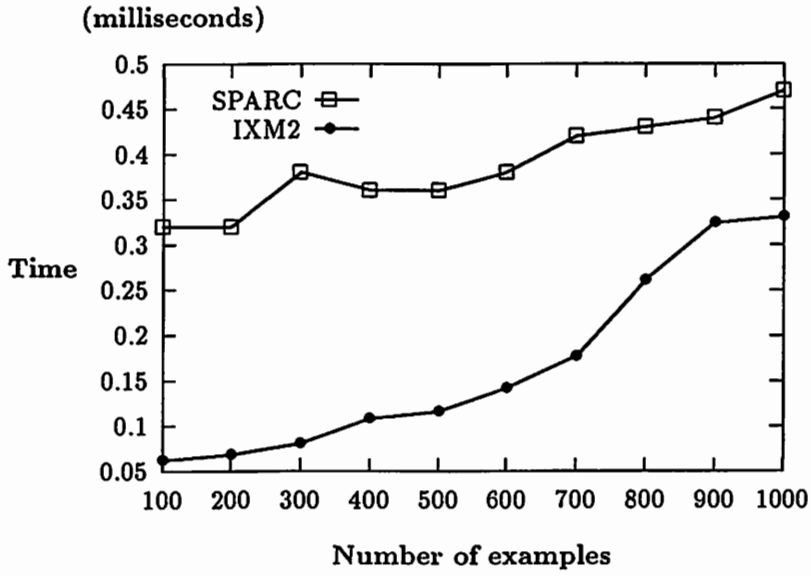


図 5.5: 用例検索の処理時間の比較 (検索タイプ：類語コード完全一致検索)

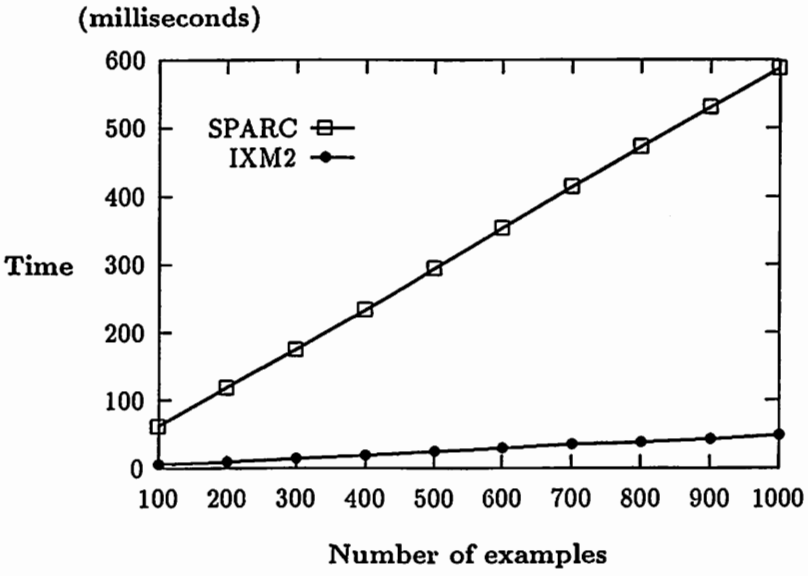


図 5.6: 用例検索の処理時間の比較 (検索タイプ：類似検索)

第 6 章

おわりに

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。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。なお、CM-2(Connection Machine-2) における実験結果に関しては、[9] で報告される予定である。

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] Waltz, D. : "Massively Parallel AI," Proc. of AAAI-90 (1990)

[3] Kitano, H., Hendler, J., Higuchi, T., Moldovan, D. and Waltz, D. : "Massively Parallel Artificial Intelligence," Proc. of IJCAI-91 (1991)

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

[5] Kitano, H. and Higuchi, T. : "Massively Parallel Memory-Based Parsing," Proc. of IJCAI-91 (1991)

[6] Kitano, H. and Higuchi, T. : "High Performance Memory-Based Translation on IXM2 Massively Parallel Associative Memory Processor," Proc. of AAAI-91 (1991)

[7] 大井耕三，隅田英一郎，飯田仁，樋口哲也，北野宏明：『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』, 情報処理学会第 46 回全国大会, 5B-5 (1993.3)

[8] 大野，浜西：『類語新辞典』, 角川書店 (1984)

[9] Sumita, E., Oi, K., Furuse, O., Iida, H., Higuchi, T., Takahashi, N. and Kitano, H. : "Example-Based Machine Translation on a Massively Parallel Processor," Proc. of IJCAI-93, (to appear)