WFST 全展開モデルに基づく統計的機械翻訳

塚田 元 †
tsukada@cslab.kecl.ntt.co.jp

永田昌明 ‡
nagata.masaaki@lab.ntt.co.jp

†NTT コミュニケーション科学基礎研究所
‡NTT サイバースペース研究所

1 はじめに

近年、対訳コーパスの充実や計算能力の向上という背景もあり、Brown ら [1] に端を発した統計的機械翻訳の研究が盛んになってきている。Brown らは翻訳を「雑音のある通信路モデル」(noisy channel model) によってモデル化し、原言語の文は目的言語の文が雑音のある通信路によって変形したものであり、これを元の目的言語の文に復元する処理 (decode) が翻訳であると考える。

機械翻訳の問題は本質的に NP 完全であることが知られており [2]、Brown らのモデルのようにかなり大胆な近似を導入した統計モデルを用いる場合でさえデコーダは全探索が事実上不可能である。そこで、準最適解を探索する手法 [3, 4, 5, 6, 7] が提案されてきているが、デコーダにおける統計モデルの内部表現方法の観点で最適性を求めた研究はまだほとんど見られない。

モデルの内部表現に関して、近年音声認識の分野で重みつき有限状態トランスデューサ (WFST: weighted finite state transducer) に基づく手法が大きな成功を納めている [8]。基本的な考え方は、デコーダが必要とする複数の統計モデルを予め WFST で表現しておき、それを WFST 上に定義される演算によって、最適なものに静的に全展開しておくことで、効率的なデコーディングを実現するというものである。WFST により統計翻訳をモデル化する研究はこれまでも行われているが [9, 10]、おもに定式化に焦点を置いた研究が多く、デコーダの効率の改善という観点でこのアプローチの有効性に注目した統計的機械翻訳の研究はまだほとんど見られない。

本稿では、Brown ら [1] の個々の統計モデルを WFST で表現し、音声認識で広く用いられている前向きビームサーチ・後向き A^* サーチと組み合わせることで、動的なモデル合成と静的なモデル合成 (全展開モデル) の計算量を実験的に比較する。ここでの動的なモデル合成は、これまで多くのデコーダで用いられてきたモデル適用手法に相当している。この結果、静的なモデル合成により劇的に効率が改善することを示す。

2 IBM モデル

まず最初に、翻訳モデルとして用いる Brown ら [1] の IBM モデル 3 について概説する。日本語 f から英語 e への翻訳は、$P(e|f)$ を最大化する e を見付ける問題ととらえる。これはベイズ則により次のように書き換えられる。

$$argmax_e P(e|f) \quad = \quad argmax_e P(f|e)P(e) \quad (1)$$

ここで $P(e)$ を言語モデル、$P(f|e)$ を翻訳モデルと呼ぶ。本稿では、言語モデルを単語 trigram で、翻訳モデルを IBM モデル 3 でモデル化する。翻訳モデルは、考えられる全ての単語対応 a を考慮することで次のように表される。

$$P(f|e) \quad = \quad \sum_a P(f, a|e) \quad (2)$$

IBM モデルにおいては、日本語 f の j 番目の単語が、英語 e の a_j 番目に対応するという 1 対多の単語対応を仮定する。IBM モデル 3 では、$P(f, a|e)$ として次のものを用いる。

$$\begin{aligned} P(f, a|e) \quad = \quad & \left(\begin{array}{c} m - \phi_0 \\ \phi_0 \end{array} \right) p_0^{m-2\phi_0} (1 - p_0)^{\phi_0} \cdot \\ & \prod_{i=0}^{l} \phi_i! \; n(\phi_i|e_i) \cdot \\ & \prod_{j=1}^{m} t(f_j|e_{a_j}) d(j|a_j, m, l) \end{aligned} \quad (3)$$

ϕ_i は**繁殖数**と呼ばれ、e_i に対応する f の単語数を表す。ϕ_0 の場合は NULL に対応する単語数を表す。$n(\phi|e_i)$ は**繁殖確率**と呼ばれ、英語の単語 e_i が ϕ 個の日本語の単語に対応する確率を表す。$t(f_j|e_i)$ は**翻訳確率**と呼ばれ、英語の単語 e_i が日本語の単語 f_j に翻訳される確率を表す。$d(j|i, l, m)$ は**歪み確率**と呼ばれ、英語の文長が l で、日本語の文長が m のときに、英語の単語位置 i が日本語の単語位置 j に対応する確率を表す。式 (3) の詳細については、次節で説明する。

3 WFST カスケードモデル

WFST とは入力だけでなく出力シンボルと出力重みの定義された有限状態機械であり、単純ながらシンボル列変換の非常に強力なモデルになっている。WFST には合成演算 (composition)[11] が定義されており、この演算によって、二つの WFST T_1 と T_2 の入出力を繋いだもの $T_1 \circ T_2$ が得られる。式 (3) の構成要素を複数の WFST で表現したとき、従来のデコーダでは、この合成演算をデコードの過程で実行していた。本稿のアプローチは、これを予め計算しておこうというものである。

WFST による翻訳過程のモデル化は、概ね Knight ら [9] の手法に則り、式 (3) の歪み確率を除いた部分を WFST カスケードによってモデル化する。図 1 にその例を示す。入力となる日本語の文は、あらゆる順序に並び替えられて WFST のカスケードに入力される。まず最初の WFST T **モデル** *(T)* によって、日本語の各単語は英語の単語に直訳される。次に *NULL* **モデル** *(N)* によって単語 NULL を削除する。次に**繁殖モデル** F によって連続した単語を 1 つの単語にまとめあげる。各々の段階で WFST の重みとして定義された確率値が付与される。最後に、**言語モデル** *(L)* によって言語尤度が付与され、最終的な出力単語列とその尤度が決定する。最後の言語モデルでは単語列に修正は加えられない。デコーダの仕事は、このような過程で得られる出力単語列のうち最も確率値の高いものを求めることにある。言い替えると、入力の日本語文のあらゆる並び替えを表す WFST を I とすると、$I \circ T \circ N \circ L$ の最適パスを探索することがデコーダの仕事となる。

図 2 から 5 にそれぞれ T モデル、NULL モデル、繁殖モデル、言語モデルの例を示す。繁殖モデルについては、Knight ら [9] よりも遷移の決定性を高めた表現を用いた。また言語モデルは Mohri ら [8] の表現方法を用いた。図 5 の $b(x)$ はバックオフ係数を表しており、バックオフのための条件分岐を WFST の非決定的な遷移という形で近似している。

ここで個々の WFST モデルと式 (3) の対応を見てみよう。1 つの単語対応があたえられたときそれを実現する並び替えられた入力単語列の場合の数は、${}_{m-\phi_0}C_{\phi_0} \prod_{i=0}^{l} \phi_i!$ 通りある。これを例を使って説明したのが、図 6 から 7 である。図 6 のような単語対応を考えたとき、図 7 に示した入力 j の並び替えがこの単語対応を実現する。繁殖数 3 の “embeded” に対応する “ コード/化/される” の並び替えは全部で 3! 通りある。また、 “NULL” が対応する “は” が入れる入力文の場所は全部で ${}_7C_1 = 7$ 箇所ある。合計 42 通りの並び替え単語列が図 6 の単語対応を実現している。この各々の入力単語列において、$p_0^{m-2\phi_0}(1-p_0)^{\phi_0}$ は NULL モデルで、$\prod_{i=0}^{l} n(\phi_i|e_i)$ は繁殖モデルで、$\prod_{j=1}^{m} t(f_j|e_{a_j})$ は T モデルで実現していることになり、歪み確率を除いた全てが入力単語の並び換えと各 WFST モデルで表現できた。したがって、WFST カスケードは歪み確率を等確率とみなした IBM モデル

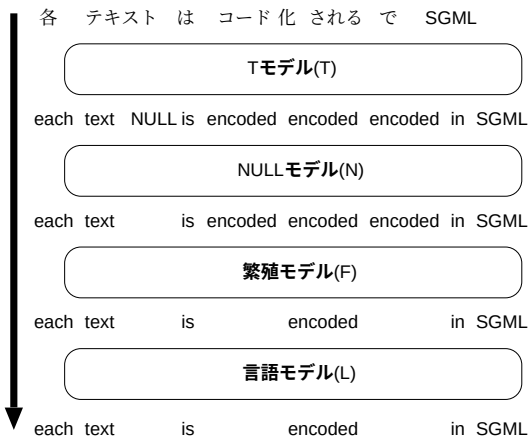

図 1: WFST カスケードによる翻訳過程のモデル化

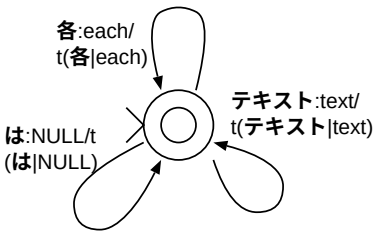

図 2: T モデル

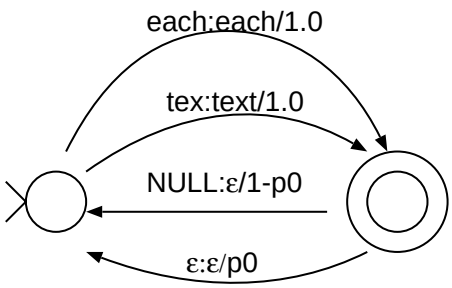

図 3: NULL モデル

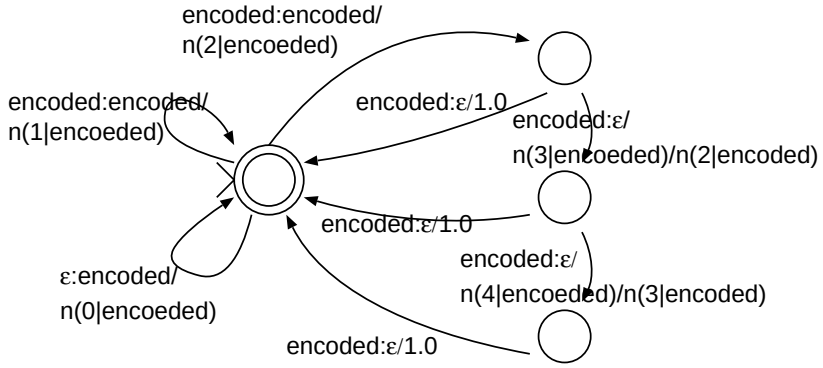

図 4: 繁殖モデル

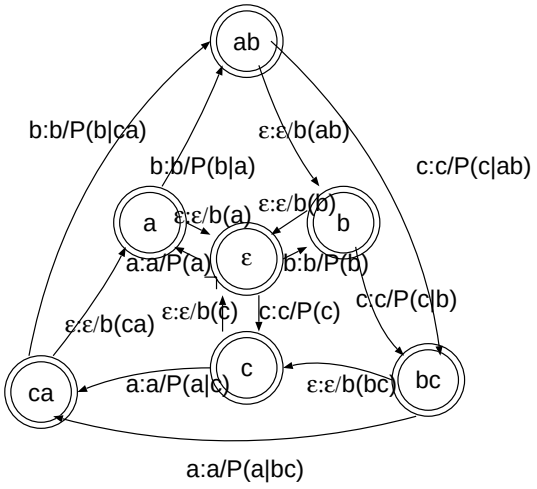

図 5: trigram 言語モデル

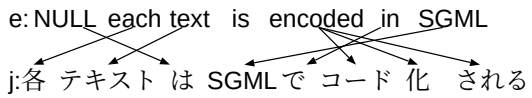

図 6: 単語対応の例

3 の実現になっている。

4 実験

前節で説明した個々の WFST を予め結合演算によって全展開しておく効果を調べるために、同一のデコーダで動的に結合する場合と比較を行った。デコーダのアルゴリズムは、音声認識で広く用いられている前向きビームサーチ、後向き A^* サーチを用いた。前向きサーチにおいては、入力の並び換えを表現する WFST の各状態毎に解の候補を管理し、尤度幅で枝刈りを行った (最上位候補の確率値の 1/10 以下の候補を枝刈り)。入力の並び換えにも制限を設け、最大

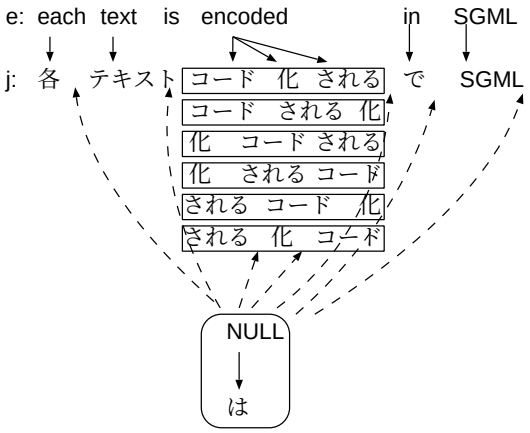

図 7: 入力の並び替えの場合の数

6 単語の範囲内での並び換えに制限した。IBM モデル 3 は GIZA++[12] を、trigram 言語モデルは CMU-Cambridge Statistical Language Modeling Toolkit v2 を用いて作成した。

翻訳実験は、日本語を英語に翻訳した日英対訳コーパスを用いた日英翻訳を行った。対訳コーパスは機械翻訳用格辞書に付属する例文であり、文アライメントがとれているものである。合計約 2 万文の内、17,678 文を学習データ、2,526 文を評価データに使った。文の長さは日本語側平均 8.4 単語、英語側平均 6.7 単語であり、日本語語彙サイズは 15,510 語、英語語彙サイズは 11,806 語である。表 1 に実験に用いた各 WFST モデルのサイズを示す。翻訳結果の評価は NIST スコア [13] および BLUE スコア [14] で行った。

実験結果を表 2 に示す。この実験結果で明らかなように、翻訳精度をまったく犠牲にすることなく、全展開モデルは従来法に相当する動的展開モデルより 10 倍近い速度向上が実現できた。

WFST の種類	状態数	遷移数
T モデル (T)	1	59,024
NULL モデル (N)	2	11,808
繁殖モデル (F)	91,906	19,5146
言語モデル (L)	14,532	30,140
全展開 ($T \circ N \circ F \circ L$)	469,856	7,598,572

表 1: WFST のサイズ

モデルの種類	NIST スコア	BLUE スコア	計算時間) (sec.)
全展開モデル	2.87	0.0332	7,844
動的展開モデル	2.86	0.0332	73,696

表 2: 計算時間の比較

5 まとめ

統計的翻訳手法で用いられる個々の確率モデルを WFST で表現し、あらかじめ合成演算により全展開しておくことで、翻訳精度を保ちながら探索効率を大きく改善できることを示した。一般に、速度とメモリ使用量はトレードオフの関係にある。にもかかわらず、１万を越える語彙サイズの機械翻訳でも、全展開モデルが十分実現可能であることが示せた。

全展開モデルによる機械翻訳は、今後いくつかの方向への発展が期待できる。１つは、モデルとしての等価性を保証しながら探索の観点から最適化する方向、他にはモデルの構造やパラメータを全体最適性の基準で再学習する方向などが考えられる。

謝辞

ngram の WFST への変換プログラムを提供くださった堀貴明氏、対訳コーパスを提供くださったフランシス・ボンド氏、藤田 早苗氏に感謝いたします。

参考文献

[1] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pitra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, Vol. 19, No. 2, pp. 263–311, 1993.

[2] Kevin Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, Vol. 25, No. 4, pp. 607–615, 1999.

[3] Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. Fast decoding and optinal decoding for machine translation. In *Proc. of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 228–235, July 2001.

[4] Christoph Tillmann and Hermann Ney. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, Vol. 29, No. 1, pp. 97–133, March 2003.

[5] Ye-Yi Wang and Alex Waibel. Decoding algorithm in statistical machine translation. In *Proc. of the 35th Annual Meeting of the Association for Computational Linguistics*, 1997.

[6] Franz Josef Och, Nicola Ueffing, and Hermann Ney. An efficient A^* search algorithm for statistical machine translation. In *Proc. of the ACL2001 Workshop on Data-Driven Machine Translation*, pp. 55–62, July 2001.

[7] Taro Watanabe and Eiichiro Sumita. Bidirectional decoding for statistical machine translation. In *Proc. of the 19th International Conference on Computational Linguistics (COLING2002)*, pp. 1079–1085, 2002.

[8] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, Vol. 16, No. 1, pp. 69–88, 2002.

[9] Kevin Knight and Yaser Al-Onaizan. Translation with finite-state devices. In *Proc. of the 4th AMTA Conference*, 1998.

[10] Shankar Kumar and William Byrne. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *Proc. of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pp. 142–149, May - June 2003.

[11] Fernando Pereira and Michael Riley. Speech recognition by composition of weighted finite automata. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*, chapter 15, pp. 431–453. MIT Press, Cambridge, Massachusetts, 1997.

[12] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, Vol. 29, No. 1, pp. 19–51, 2003.

[13] G. Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proc. of HLT 2002*, 2002.

[14] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLUE: a method for automatic evaluation of machine translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 311–318, July 2002.