不完全な文の構文解析に基づく同時音声翻訳

小田 悠介　Graham Neubig　Sakriani Sakti　戸田 智基　中村 哲
奈良先端科学技術大学院大学 情報科学研究科
{oda.yusuke.on9, neubig, ssakti, tomoki, s-nakamura}@is.naist.jp

1 はじめに

同時音声翻訳は話者の音声を他言語の音声へと連続的に変換し，発話からの時間差を可能な限り小さく保ちながら聞き手へ提示するシステムである．通常の音声翻訳とは異なり発話の終了を待たないため，講演や日常会話など，発話単位が明瞭でない場面における音声翻訳の手法として重要である．

同時音声翻訳は音声認識，機械翻訳，音声合成の順に処理することで実現されるが，特に音声認識結果をどのような形で，どのようなタイミングで機械翻訳に渡すかが重要となる．まず，音声認識器から次々に得られる単語列をどのような基準で区切り，翻訳単位とするかを判断する必要がある．このための文分割アルゴリズムが複数提案されている [9, 16, 3, 14]．これにより得られる翻訳単位は個別に翻訳器へ入力されるが，分割位置によっては重要な文脈情報が失われてしまい，訳出の精度が下がってしまう可能性がある．このため，今までに得られた翻訳単位から次回の入力に関する情報を推定し，翻訳時の参考とすることが考えられる．このような手法としては，独英翻訳において未来の動詞を予測し，不完全な文を補完する手法が提案されている [4]．しかしこの手法では翻訳単位の構文的な役割までは考慮しておらず，前後の文に対して不整合な訳出を生成する可能性がある．一方，構文を考慮する翻訳手法としては Tree-To-String 翻訳 [6] があるが，完全な文のに対する構文解析が想定されるため，文分割を前提とする同時音声翻訳に直接適用することは好ましくない．

これらの問題を解決するために，本研究では翻訳単位に隣接する文法要素を予測し，得られた結果を文の一部として扱う手法を提案する．これにより翻訳単位に対して正しい構文解析結果を得られるようになるだけでなく，Tree-To-String 翻訳の訳出を調べることで後続の翻訳単位を待つべきかどうかを判断することが可能となる．評価実験により，提案法は従来法と同等以上の精度で訳出が可能であることが分かった．

2 不完全な文の構文解析

通常の構文解析は与えられた単語列で文が完結することを前提としている．例えば「this is a pen」という英文の句構造は図 1(a) で表され，実際の句構造解析器 [17] でもこの結果が得られる．しかし同時音声翻訳の場合，音声認識器から出力される翻訳単位は完全な文ではなく，本来得られるべき文の分割された一部となる．このため「this is」「is a」といった不完全な

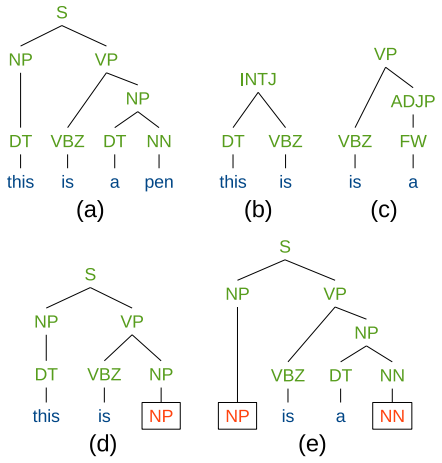

図 1: 不完全な文の構文解析

単語列が構文解析器に入力される可能性があり，これらをそのまま構文解析しても図 1(b)，(c) のように本来の構文木を想定すると不適切な結果となってしまう．この問題は，翻訳単位の前後に適切な文法要素を補うことで回避することができる．例えば「this is」の場合，図 1(a) では後続の要素として NP(名詞句) が存在する．このため「this is NP」のように単語列に文法要素のブラックボックス NP を追加して構文解析を行うことで，図 1(d) のように望ましい構文木が得られると考えられる．同様に「is a」の場合は「NP is a NN」とすれば良く，構文木は図 1(e) となる．

本節ではまず不完全な文の構文解析全体の定式化を行い，次に本研究で導入する隣接する文法要素の推定について述べる．

2.1 構文解析の定式化

句構造解析の標準的な解析モデルは確率的文脈自由文法 (Probabilistic Context Free Grammar: PCFG) であり，これは式 (2) に示すように，構文木 T の生成確率を与えられた単語列 $\boldsymbol{w} \equiv [w_1, w_2, \cdots, w_n]$ の下で最大化する問題である．

$$T^* \equiv \arg\max_T \Pr(T|\boldsymbol{w}) \quad (1)$$

$$\simeq \arg\max_T [\sum_{(X\to[Y])\in T} \log \Pr(X \to [Y]) + \sum_{(X\to w_i)\in T} \log \Pr(X \to w_i)] \quad (2)$$

ここで $\Pr(X \to [Y])$ は文法要素 X からより下の階層の文法要素の列 $[Y]$ を生成する確率，$\Pr(X \to w_i)$ は X から i 番目の単語 w_i を生成する確率である．

文法要素を追加した文の構文解析について考えるために，文の前方に追加される文法要素の列を $\boldsymbol{L} = [L_{-|\boldsymbol{L}|}, \cdots, L_{-2}, L_{-1}]$，後方に追加される文法要素の列を $\boldsymbol{R} = [R_{n+1}, R_{n+2}, \cdots, R_{n+|\boldsymbol{R}|}]$ とする．「this is」の例では $\boldsymbol{L} = [\,]$，$\boldsymbol{R} = [\,\boxed{\text{NP}}\,]$ となる．$\boldsymbol{L}$ 及び $\boldsymbol{R}$ は構文解析より前に単語列 $\boldsymbol{w}$ から独立に生成されるものと仮定すると，構文解析全体としては式 (3)，(4) による文法要素の推定，式 (5) による文法要素を含む構文解析の組み合わせで表現できる．

$$\boldsymbol{L}^* \equiv \arg\max_{\boldsymbol{L}} \Pr(\boldsymbol{L}|\boldsymbol{w}) \tag{3}$$

$$\boldsymbol{R}^* \equiv \arg\max_{\boldsymbol{R}} \Pr(\boldsymbol{R}|\boldsymbol{w}) \tag{4}$$

$$T^* \equiv \arg\max_{T} \Pr(T|\boldsymbol{L}^*, \boldsymbol{w}, \boldsymbol{R}^*) \tag{5}$$

文法要素を含む構文解析は，単純に各文法要素を単語と同様に扱い，CKY 法などの通常の解析アルゴリズムを実行すれば良い．このとき文法要素に相当する単語の生成確率を式 (6) で定義する．

$$\Pr(X \to \boxed{Y}) \equiv \begin{cases} 1, & \text{if } Y = X \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

以降の節では文法要素の推定方法について述べる．

2.2 隣接する文法要素の推定

まず，どのような文法要素を推定の対象とするのか明確にするために，単語列 $\boldsymbol{w}$ と文法要素の列 $\boldsymbol{L}$，$\boldsymbol{R}$ から生成される構文木が次の条件を満たすものとする．

1. 構文木は本来得られるべき構文木の部分木である．
2. 構文木は $\boldsymbol{L}, \boldsymbol{w}, \boldsymbol{R}$ のみを終端器号とする．
3. 構文木のノード数が最小である．

このようにすると，図 1 の「this is」では先に示した例のみがこの条件を満たし，他に考えられる組み合わせ，例えば $\boldsymbol{R} = [\,\boxed{\text{DT}}, \boxed{\text{NN}}\,]$ などは除外されることが分かる．この条件を満たす文法要素を**隣接する文法要素**と定義する．

図 2 に示すのは，Penn Treebank[8] の構文木を任意の単語列について分割したときの，単語列の前後に隣接する文法要素数の統計である．このように大部分の隣接する文法要素数は前方・後方ともに 2 個以内となるが，まれにより多くの数の文法要素を必要とすることが分かる．

このため，生成する文法要素の数を固定しないために，アルゴリズム 1 に示す反復的な手法で文法要素の推定を行う．ここで $+\!\!+$ は列同士の結合を表す．まず単語列 $\boldsymbol{w}$ をそのまま構文解析し，「望ましくない」構文木 T' を得る．この情報と今までに生成した文法要素の情報を素性として次の文法要素を生成する．この処理を文末記号 (アルゴリズム 1 では nil) が生成されるまで続けることで，単語列ごとに適切な数の文法要素を生成できると考えられる．

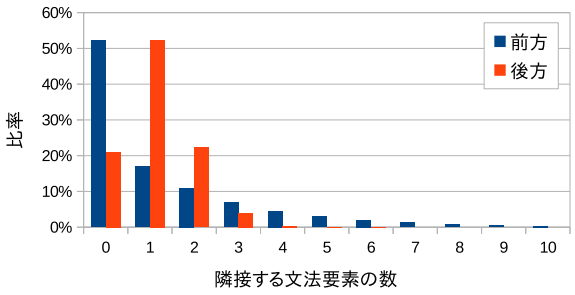

図 2: 前方・後方に追加されるべき文法要素の数

Algorithm 1 後方の文法要素の推定

```
T' ← arg max_T Pr(T|w)                ▷ 事前構文解析
R* ← [ ]
loop
    R+ ← arg max_R Pr(R|T', R*)       ▷ 次の文法要素
    if R+ = nil then
        return R*                     ▷ 推定の終了
    end if
    R* ← R* ++ [R+]                   ▷ 文法要素の追加
end loop
```

本研究では文法要素の推定に線型 SVM[2] による多クラス分類器を使用し，表 1 に示す素性集合によりモデルを構築した．ここで，素性で使用される後方の単語については，既に生成した文法要素がある場合はこれを使用するものとした．例えば「this is a $\boxed{\text{NN}}$」という例に対して，後方 3 単語は「is a $\boxed{\text{NN}}$」を指し，後端の単語は「$\boxed{\text{NN}}$」を指すことになる．なおアルゴリズム 1 及び表 1 では後方の文法要素の推定方法のみを示したが，単語列全体を反転させれば前方の文法要素も同様の手法で推定可能である．

3 文法要素を含む構文木の機械翻訳

式 5 で推定された構文木は通常の対訳コーパスによって学習された Tree-To-String 翻訳器の入力データとして用いる．こうすることで，推定された文法要素を含まない構文木を使用した場合よりも整合性の高い構文木を翻訳に使用することができるため，より高い翻訳精度を実現できると考えられる．[1]

しかし，英語と日本語などの言語対は語順が大きく異なるため，入力を逐次的に翻訳すると目的言語としての自然性が大きく損なわれてしまう．さらに同時音声翻訳では翻訳開始時点で完全な文を得られない可能性が高く，次回以降の翻訳単位が現在の翻訳単位よりも前に訳出されるべきである場合がある．例えば連続した翻訳単位「this is $\boxed{\text{NP}}$」「a pen」はそれぞれ翻訳結果が「これは$\boxed{\text{NP}}$です」「ペン」となり，「ペン」は「です」より前，$\boxed{\text{NP}}$の位置に挿入するべきであることが予想できる．

[1] 文法要素を含む構文木に対して厳密に Tree-To-String 翻訳モデルを適用するには，実際には言語モデルなどの大域素性を文法要素に対応させなければならない．これらの計算は些細な問題ではないため今後の課題とし，本研究では簡単のため，構文木に含まれる文法要素は全て未知語として扱い，訳出にはそのままの形で出力するものとした．

表 1: 文法要素の推定に用いる素性

種別	素性
単語	前方 3 単語の単語，品詞 1, 2-gram 後方 3 単語の単語，品詞 1, 2-gram 前後端の語による単語，品詞の 2 つ組
構文木	根とその子ノードのラベル 根とその子ノードのラベルの 2 つ組
長さ	入力単語数 既に推定された文法要素の数

Algorithm 2 訳出の並べ替えに基づく翻訳待機

$\boldsymbol{w} \leftarrow [\,]$
loop
  $\boldsymbol{w} \leftarrow \boldsymbol{w} + \mathrm{NextSegment}()$ ▷ 単語列の追加
  $\boldsymbol{L}^* \leftarrow \arg\max_{\boldsymbol{L}} \Pr(\boldsymbol{L}|\boldsymbol{w})$ ▷ 前方の文法要素
  $\boldsymbol{R}^* \leftarrow \arg\max_{\boldsymbol{R}} \Pr(\boldsymbol{R}|\boldsymbol{w})$ ▷ 後方の文法要素
  $T^* \leftarrow \arg\max_{T} \Pr(T|\boldsymbol{L}^*, \boldsymbol{w}, \boldsymbol{R}^*)$ ▷ 構文解析
  $\boldsymbol{e}^* \leftarrow \arg\max_{\boldsymbol{e}} \Pr(\boldsymbol{e}|T^*)$ ▷ 機械翻訳
  if $\boldsymbol{R}^*$ の要素が全て $\boldsymbol{e}^*$ の末尾に存在 **then**
    $\mathrm{Output}(\boldsymbol{e}^*)$
    $\boldsymbol{w} \leftarrow [\,]$
  end if
end loop

翻訳単位の後方に隣接する文法要素が翻訳結果の末尾以外に現れた場合，文分割法が適切な位置で単語列を分割しなかったものと考えられる．このため，翻訳結果にこのような現象が生じた場合に後続の翻訳単位を待機し，現在の翻訳単位と併せて翻訳を行うことで翻訳精度を向上することができると考えられる．この手法をアルゴリズム 2 に示す．

4 実験

提案法の効果を調べるために 2 種類の実験を行った．まず，不完全な文に対して前後に隣接する文法要素の推定精度を調べた．次に文法要素を含めた構文解析結果を同時音声翻訳の設定の下で Tree-to-String 翻訳に適用し，その翻訳精度を調べた．以下では実験設定，および実験結果について述べる．

4.1 実験設定

4.1.1 隣接する文法要素の生成

まず英語ツリーバンクを用いて前後に隣接する文法要素の生成器を学習し，その精度を調べた．実験対象として Penn Treebank 中の構文木に含まれる 2 単語以上の全ての部分単語列と，部分単語列に対応する前後に隣接する文法要素の列を抽出した．このうち 9 割を学習データ，1 割をテストデータとし，前述の推定法により生成された文法要素の適合率と再現率を調べた．単語分割には Stanford Tokenizer[2]，句構造解析には Ckylark[17] を使用した．

[2]http://nlp.stanford.edu/software/tokenizer.shtml

表 2: 隣接する文法要素の推定結果

文法要素		適合率 %	再現率 %	F %
$\boldsymbol{L}$	(順序)	31.93	7.27	11.85
	(非順序)	51.21	11.66	19.00
$\boldsymbol{R}$	(順序)	51.12	33.78	40.68
	(非順序)	52.77	34.87	42.00

4.1.2 機械翻訳への適用

次に文法要素の推定結果を英日翻訳に適用し，他の同時音声翻訳の訳出法との精度比較を行った．使用したデータは TED 講演対訳コーパス [1]，及び英辞郎，例辞郎のエントリ [3] である．日本語の単語分割に KyTea[11] を使用し，英語の単語分割，句構造解析は文法要素の生成と同様とした．単語アライメントには GIZA++[13] を使用し，日本語の言語モデルは KenLM[5] による 5-gram モデルを使用した．Tree-To-String 翻訳器には Travatar[10] を使用し，パラメータを MERT[12] により最適化した．最終的な翻訳精度は BLEU[15] によって比較した．また，ベースラインとして句に基づく翻訳器を Moses[7] により構築した．

比較対象とする手法を以下に示す．いずれも文分割法には固定単語数による分割を使用した．

PBMT 句に基づく翻訳・構文木を使用しない

T2S Tree-To-String 翻訳・文法要素の推定をしない

T2S+Tag T2S に文法要素の推定を追加

T2S+Wait T2S に並べ替えに基づく翻訳待機を追加

PBMT-Sent 句に基づく翻訳・文分割なし

T2S-Sent Tree-To-String 翻訳・文分割なし

4.2 実験結果

4.2.1 隣接する文法要素の生成

表 2 に推定された文法要素の適合率，再現率，F 値について，文法要素の生成順序を考慮した場合としない場合の値を示した．

再現率が適合率よりも低いことから，本研究で作成した推定器は全体としてテストデータよりも少ない数の文法要素を生成していることが分かる．これは実際にはテストデータに重要でない文法要素が多く含まれているためであると考えられる．例えば「DT JJ pen」を例とすると，2 番目の文法要素 JJ(形容詞) は存在しなくてもよく，構文上重要ではない．実際にこのような場合，推定器は「DT pen」のようにより少ない構文要素を推定する傾向にある．また，このような重要でない構文要素は単語列の前方に現れやすく，この影響が前方に隣接する要素で順序を考慮する場合としない場合での適合率の差に現れている．

[3]http://eijiro.jp/

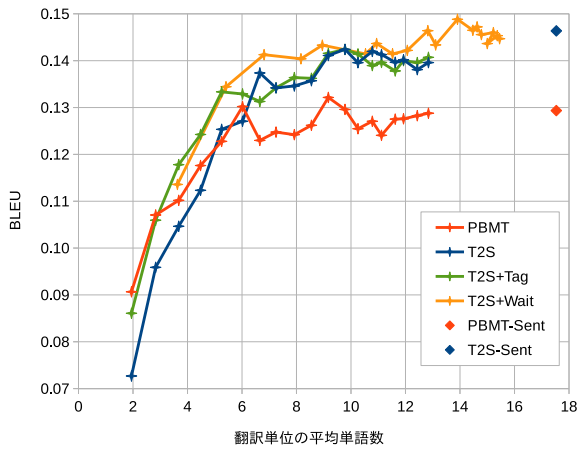

図 3: 各手法の平均単語数の推移と翻訳精度

4.2.2 機械翻訳への適用

図 3 に各翻訳法による翻訳単位の平均単語数と翻訳精度の関係を示す．まず PBMT-Sent と T2S-Sent を比較すると，文分割を全く行わない場合は先行研究 [6] で示されているように Tree-To-String 翻訳の精度が句に基づく翻訳の精度を上回っていることが分かる．しかし文分割の適用により Tree-To-String 翻訳の精度は大きく下がり，翻訳単位の平均単語数が 5 から 6 の付近を境に Tree-To-String 翻訳と区に基づく翻訳の関係が逆転していることが分かる．これは短い翻訳単位では構文情報がうまく捉えられず，Tree-To-String 翻訳器が正しい構文木を用いることができないためであると考えられる．

一方，T2S+Tag では短い単語数でも PBMT と同様の翻訳精度を維持していることが分かる．このことから，本研究の手法で推定された文法要素により文が本来持つ情報が再現され，Tree-To-String 翻訳に対して有効に作用したと考えられる．翻訳単位が長くなると Tree-To-String 翻訳本来の翻訳精度に近づくため，全体として従来法である PBMT と同等以上の翻訳精度を達成している．また T2S-Wait は他のいずれの手法よりも全体的に良い翻訳精度を達成しており，本研究による翻訳待機の手法が有効であることが確認できる．ただし翻訳単位を結合することにより，従来法の利点である翻訳単位ごとの長さの制御は難しくなる．

5 おわりに

本研究では同時音声翻訳において構文を考慮した翻訳手法を実現するために 2 つの手法を提案した．一つは文として不完全な翻訳単位に対する前後の文法要素の推定であり，これを Tree-To-String 翻訳と併せて適用することで，従来法と同等以上の翻訳精度を達成できることが分かった．また文法要素と現在の翻訳結果を用いて翻訳待機を行う手法では，分割位置を修正することで更に良い翻訳結果を得られることが分かった．今後の課題としては文法要素の推定精度の向上，また翻訳待機における翻訳単位の長さ制御などが挙げられる．また本手法では一度の翻訳に構文解析と機械翻訳を数回実行する必要があるため，これらのモジュールの実行時間の短縮や，より少ない回数の解析で同様の結果を得る手法の考案などが実際のシステム作成，運用にあたって重要となる．

謝辞

本研究の一部は，JSPS 科研費 24240032 の助成を受け実施した．

参考文献

[1] Mauro Cettolo, Christian Girardi, and Marcello Federico. WIT^3: Web inventory of transcribed and translated talks. In *Proc. EAMT*, pp. 261–268, 2012.

[2] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 2008.

[3] Tomoki Fujita, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Simple, lexicalized choice of translation timing for simultaneous speech translation. In *InterSpeech*, 2013.

[4] Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. Don ' t until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proc. EMNLP*, October 2014.

[5] Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. Scalable modified Kneser-Ney language model estimation. In *Proc. ACL*, Sofia, Bulgaria, August 2013.

[6] Liang Huang, Kevin Knight, and Aravind Joshi. Statistical syntax-directed translation with extended domain of locality. In *Proc. AMTA*, Vol. 2006, pp. 223–226, 2006.

[7] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL*, 2007.

[8] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The Penn treebank. *Computational linguistics*, Vol. 19, No. 2, 1993.

[9] Evgeny Matusov, Arne Mauser, and Hermann Ney. Automatic sentence segmentation and punctuation prediction for spoken language translation. In *Proc. IWSLT*, 2006.

[10] Graham Neubig. Travatar: A forest-to-string machine translation engine based on tree transducers. In *Proc. ACL*, pp. 91–96, Sofia, Bulgaria, August 2013.

[11] Graham Neubig, Yosuke Nakata, and Shinsuke Mori. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA, June 2011.

[12] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proc. ACL*, pp. 160–167, Sapporo, Japan, July 2003.

[13] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational linguistics*, 2003.

[14] Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Optimizing segmentation strategies for simultaneous speech translation. In *Proc. ACL*, June 2014.

[15] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proc. ACL*, pp. 311–318, 2002.

[16] Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. Segmentation strategies for streaming speech translation. In *Proc. NAACL-HLT*, 2013.

[17] 小田悠介, Graham Neubig, 波多腰優斗, Sakriani Sakti, 戸田智基, 中村哲. 解析失敗の発生しにくい PCFG-LA 句構造構文解析. 言語処理学会第 21 回年次大会予稿集, 3 月 2015.