NAIST-IS-MT1351023

修士論文

同時音声翻訳システムにおける訳出速度と翻訳精度の向上

小田 悠介

2015 年 3 月 12 日

奈良先端科学技術大学院大学
情報科学研究科 情報科学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

小田 悠介

審査委員：

中村 哲 教授	（主指導教員）
松本 裕治 教授	（副指導教員）
戸田 智基 准教授	（副指導教員）
Graham Neubig 助教	（副指導教員）
Sakriani Sakti 助教	（副指導教員）

同時音声翻訳システムにおける訳出速度と翻訳精度の向上 *

小田 悠介

内容梗概

話者同士がお互いの言語を理解しない状況下では通訳者による意思疎通の補助が行われるが，人的資源や金銭面の負担を考慮すると常に通訳者に頼ることはできない．これらの言語の壁を乗り越える技術として音声翻訳システムが研究されている．音声翻訳は話者の音声を計算機により解析し，聴取者の言語による音声に変換，出力するシステムであり，従来のような同時通訳者の仲介なしで多言語間のコミュニケーションを可能とする．この中でも特に同時音声翻訳は，話者の発話終了を待たないで翻訳結果の提示を行うことに特化しており，意思疎通の同時性を維持することができる．本研究では同時音声翻訳システムの訳出速度と翻訳精度を目的とし，機械翻訳システムの改良を中心とした各種手法を提案する．具体的には従来法よりも高精度な文分割法，及び Tree-To-String を用いたより高精度な同時音声翻訳の手法について述べる．既存手法との実験を通した比較により，提案法による文分割は既存手法と同程度の翻訳精度を 2 倍以上の速度の訳出で実現できることを示す．また Tree-To-String 翻訳を用いる手法では，文分割により生成された翻訳単位の文法的な問題を補正する効果があることを示す．これらの手法により，従来よりも同時性が高く，なおかつ翻訳の精度も高い同時音声翻訳システムを実現することができる．

キーワード

自然言語処理, 機械学習, 機械翻訳, 同時音声翻訳, アルゴリズム

*奈良先端科学技術大学院大学 情報科学研究科 情報科学専攻 修士論文, NAIST-IS-MT1351023, 2015 年 3 月 12 日.

Improving the Speed and Accuracy of Simultaneous Speech Translation Systems*

Yusuke Oda

Abstract

Interpreters aid communication in situations where speakers do not understand each other's languages, but this approach is not reliable everywhere, considering the required human resources and money. Speech translation systems are one way to get over this "language barrier." Speech translation is a tool allowing us to communicate with speakers of a foreign language without mediation of a simultaneous interpreter, by analyzing utterances of the speaker and converting them into an other utterance in the listener's language. Especially, simultaneous speech translation preserves the simultaneity of communication by proposing translation results without waiting until the end of the speech. This study proposes methods to improve the speed and translation performance of simultaneous speech translation systems. Specifically, this paper proposes a segmentation strategy with higher performance than conventional methods and high-performance method for simultaneous speech translation using tree-to-string translation techniques. Experiments show that our new segmentation strategies reduce the unit for translation to half the size as than conventional methods while keeping the same translation performance. Another experiment shows that our new methods of simultaneous speech translation using tree-to-string translation techniques is effective in compensating for syntax problems in the segmentation process. Using

*Master's Thesis, Department of Information Science, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT1351023, March 12, 2015.

these proposed techniques, we are able to create a simultaneous speech translation system with both higher simultaneity and higher translation performance.

Keywords:

natural language processing, machine learning, machine translation, simultaneous speech translation, algorithm

目 次

図 目 次

表 目 次

1. 緒言

1.1 背景

音声を用いた他者との意思疎通は日常生活のあらゆる場面で行われ，人間にとって最も重要なコミュニケーション手段のひとつである．これは同時に，音声による意思疎通に何らかの制限があると，コミュニケーションを行う上で大きな支障となることを示している．このような問題の具体例としては話者同士の用いる言語の相違が挙げられる．言語の違いを越えてコミュニケーションを行う場合，現状では人間の通訳者が両者の仲介を行う．しかし人件費や通訳者自体の全体数といった制約により，全ての他言語コミュニケーションの場面で通訳者を利用するのは現実的ではない．このような制約を取り除くための技術として，コンピュータによる**音声翻訳 (Speech Translation: ST)** [21] が研究されている．これはある言語の発話をプログラムにより他言語の発話に変換するもので，従来通訳者が行っている作業をコンピュータ上で実現する技術である．現在実用化されている音声翻訳システムは，**音声認識 (Automatic Speech Recognition: ASR)**，**機械翻訳 (Machine Translation: MT)**，**音声合成 (Text-To-Speech: TTS)** の 3 つの技術が連携して動作するものであり，この手法を用いた携帯デバイス上で動作する音声翻訳システム [1] などが発表されている．

人間による通訳では，通訳者が翻訳を開始するタイミングにより大きく 2 種類の通訳手法が存在する．**逐次通訳 (Consecutive Interpretation)** は，通訳者は対象話者の発話が終わるまで待ち，発話文全体を一度に翻訳する手法である．この手法では翻訳対象として完全な文を扱うことができるため，比較的高い品質を維持した通訳が可能である．しかし，話者と通訳者が交互に発話する形となるため，通常の会話に対して約 2 倍の時間を要することとなる．これに対して**同時通訳 (Simultaneous Interpretation)** は，適当な単位の発話を得た時点で即座に通訳を提示する手法である．この場合，話者と通訳者は数秒のタイムラグの下で同時に喋ることになるため，逐次通訳に比べて訳出時間が短いといった利点がある．しかし，同時通訳技術の習得には通常の翻訳技術に加えて専門の訓練が必要

[1] http://www.nict.go.jp/press/2010/08/05-1.html

であり，通訳の品質も逐次通訳に比べて低くなる傾向がある．

実用化されている音声翻訳システムでは機械翻訳の入力として完全な文を扱う [18, 14] ため，逐次通訳と同様に話者の発話を何らかの方法で判定するか，話者自身がシステムに対して発話終了の合図を送る必要がある．このような手法は音声チャットや質問応答などの一問一答型のシステムでは有効であるが，講義や日常会話のような発話の終了が明確に示されない状況では使用することができないため，同時通訳のように発話の終了を待たずに翻訳を開始できる手法を採用する必要がある．このような，翻訳開始のタイミングに関する基準をシステム自身が持つような音声翻訳システムを本論文では**同時音声翻訳 (Simultaneous Speech Translation: SST)** と呼ぶ．同時音声翻訳における翻訳開始タイミングの制御法としては，音声認識器から連続的に入力される単語列を適切な単位で分割し，得られた単語列を順に機械翻訳の入力とする手法が複数提案されている [7, 1, 32, 8]．本論文ではこれらを総称して**文分割 (Segmentation)** と呼ぶ．

1.2 目的

同時音声翻訳システムでは利用者同士の意思疎通の観点から，翻訳結果の正しさ（翻訳精度）に加えて，入力音声を得てから翻訳結果を出力するまでにかかる時間（訳出速度）を可能な限り短くすることが要求される．訳出速度を向上するためには高頻度な分割を行う文分割法を採用する必要があるが，文分割の頻度が多くなると翻訳単位に含まれる単語数が少なくなり，文全体としての情報が欠落してゆくため，翻訳精度は低下してしまう．このように，一般に翻訳精度と訳出速度はトレードオフの関係にある．

このため，翻訳精度を可能な限り維持しながら訳出速度を向上させられるような文分割法を設計する必要がある．従来の文分割法はいずれも何らかの人手で決められた基準に従って分割位置を選択しており，文分割の結果が翻訳精度にどのような影響を及ぼすのかは考慮されてこなかった．これを踏まえ，本論文の最初の提案では翻訳精度を直接考慮するような文分割法の構築手法を述べる．具体的には機械翻訳の自動評価尺度を最大化するという戦略に基づき，貪欲探索と動的計画法に基づく手法で文分割法を明示的に最適化することで，与えられた機械翻

訳システムに対して最適な文分割法を構築するアルゴリズムを述べる．

また従来の文分割法を用いた同時音声翻訳では，文分割により生成される翻訳単位を直接機械翻訳の入力としているが，機械翻訳に用いられる手法は基本的に完全な文が入力されることを前提としており，文として不完全な翻訳単位は妥当な翻訳が行われない可能性がある．これは文分割法に本質的な問題であるが，本論文では文分割法を施した上で，分割により失われた情報を再現しながら翻訳を行う手法を述べる．具体的には，既存の構文木から単語列の周囲に続くと考えられる構文情報を予測するモデルを構築し，この情報を元に Tree-To-String 翻訳の枠組みで処理を行うことで，構文的に正しい翻訳結果を生成する手法を述べる．また，この手法により生成された翻訳結果を評価することで分割位置が文法的に問題がないかどうかを再評価し，翻訳の段階で分割位置を修正する手法についても述べる．

Tree-To-String 翻訳を同時音声翻訳に使用するには，構文解析器が失敗せず，常に結果を返す必要がある．既存の高精度な構文解析モデルは一定の割合で解析に失敗するため，この用途に適さないという問題があった．このため，本論文では既存の構文解析モデルの解析失敗を抑制するモデルの修正についても述べる．

1.3 本論文の構成

2 章 では同時音声翻訳システムについての説明と定式化を行う．3 章 では本論文の中心的な話題となる統計的機械翻訳の各種要素技術についての解説を行う．次に，本論文の提案する 2 種類の手法について述べる．具体的には**翻訳結果の評価尺度最大化に基づく文分割法 (4 章)** と**不完全な文の構文解析に基づく同時音声翻訳 (5 章)** について順に述べる．6 章で本論文の総括を行う．

2. 同時音声翻訳システム

本章では，本論文の対象とする技術である同時音声翻訳システムの概要と構成について述べる．また，本論文による提案法の基礎を与える機械翻訳の各要素技術についても述べる．

2.1 音声翻訳の概要

同時音声翻訳を含む音声翻訳システムの共通の目的は，特定の言語に基づく発話の音声信号が与えられたとき，これに対応する翻訳対象の言語に基づく音声信号に変換することである．ここで入力側の言語を**原言語 (Source Language)**，翻訳対象となる言語を**目的言語 (Target Language)** と呼び，それぞれ記号 f と e を用いて表現することとする．具体的な音声翻訳システムの実装としては，入力された特定言語の音声信号からテキストを書き起こす**音声認識 (Automatic Speech Recognition: ASR)**[31]，原言語のテキストから目的言語の対応するテキストを生成する**機械翻訳 (Machine Translation: MT)**[18, 14]，テキストから特定言語の音声信号を合成する**テキスト音声合成 (Text-To-Speech: TTS, T2S)**[34, 37] の各モジュールを用意し，これらを直列に繋ぎ合わせる手法が主に採用される．具体的な音声翻訳システムの処理手順を図 1 に示す．

2.2 同時音声翻訳

図 1 の音声翻訳システムは，一度に翻訳する単位を発話の終了によって識別している．このような翻訳手法は，音声によるチャットや質問応答のような利用者の発話が比較的短い文と仮定できる場合に有効である．しかし，講義や討論，会談，日常会話のような話者同士が連続的に発話するような場面では，発話ごとの区切り位置を明示的に得ることはできず，また得られたとしても翻訳単位が長くなってしまう．このような場合，図 1 のような音声翻訳システムでは話者の発話開始から翻訳結果の提示まで長い時間を要することとなり，会話の同時性が損なわれてしまう．この問題を解決するには，システム側が発話中の適切な位置を発

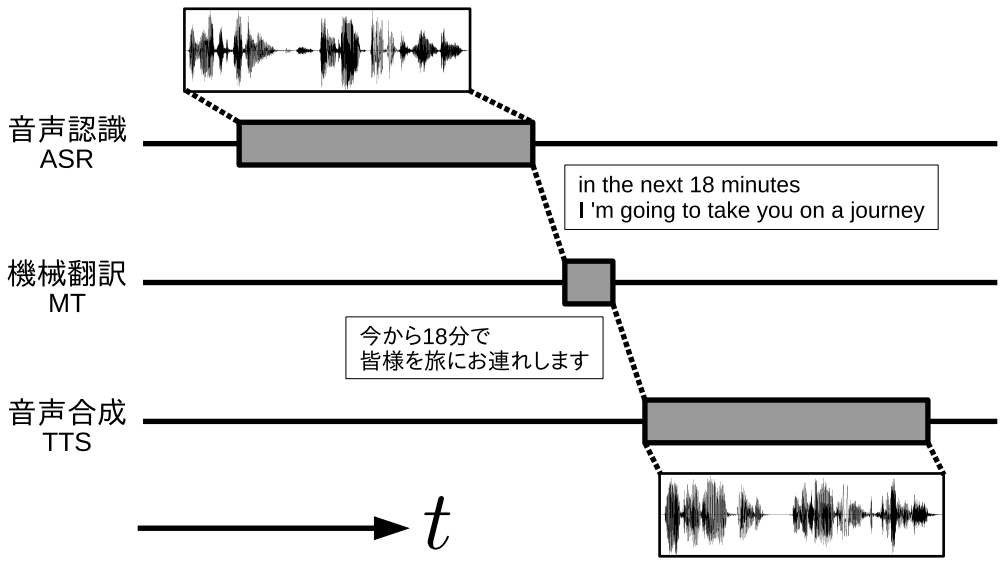

図 1　音声翻訳システムの基本的な処理手順

話の区切りとして認識することで，発話全体の終了を待たずに翻訳を開始する手法が考えられる．このような戦略を用いる音声翻訳を**同時音声翻訳**と呼ぶ．同時音声翻訳の処理手順を図 2 に示す．

通常の音声翻訳では話者による発話と翻訳結果の提示が交互に発生するため，聞き手には話者の発話終了までの時間と翻訳結果の提示に要する時間を順に要することとなる．一方，同時音声翻訳では話者の発話と翻訳結果の提示が一定時間のずれを挟んで同時に行われるため，通常の音声翻訳と比較して聞き手の待ち時間を短縮することができる．また同時音声翻訳では発話の終了を待つ必要がないため，話者が一方的に話し続けているような場面でも使用することができる．

このように同時音声翻訳は音声翻訳よりも時間に関する制約を緩和することができるが，発話の終了を待たずに翻訳を開始するということは文として成立していない単位を翻訳対象とすることとなり，文全体を考慮して翻訳を行う通常の音声翻訳に比べて翻訳結果の精度が低下してしまう．このため，同時音声翻訳では通常の機械翻訳の手法に加えて，不完全な文に対していかにして妥当な翻訳結果を生成するかが重要となる．

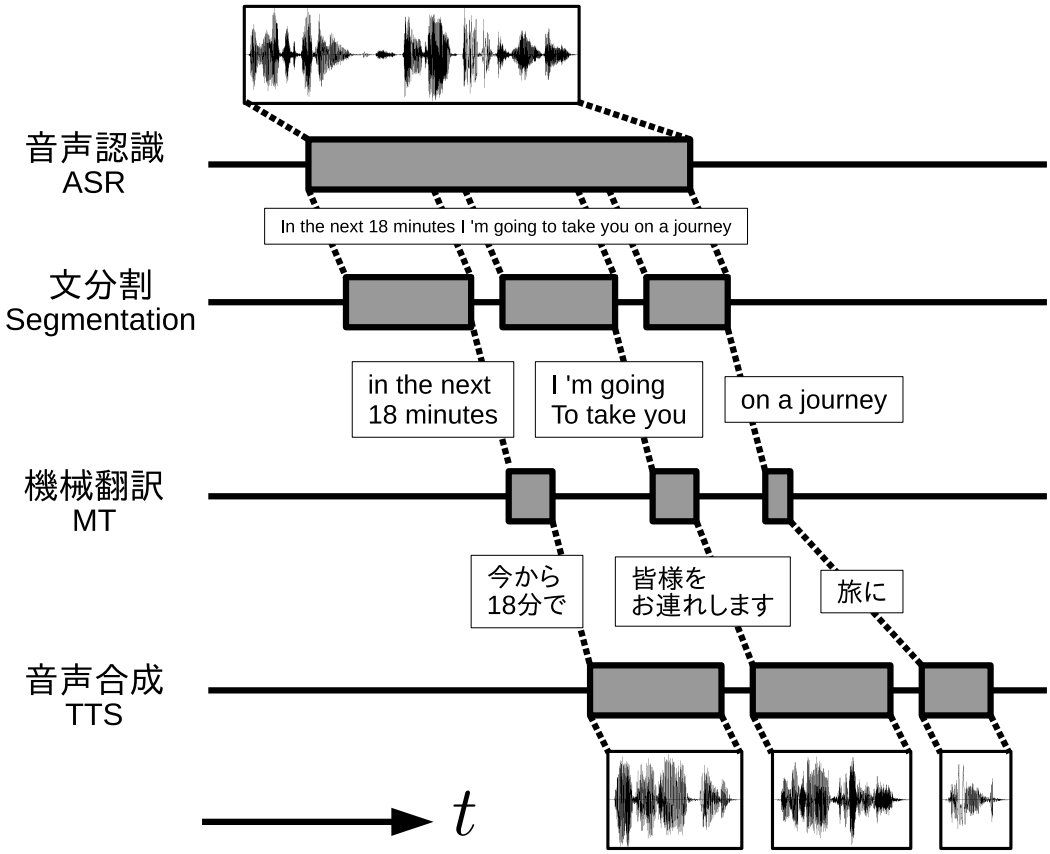

図 2　同時音声翻訳システムの処理手順

2.3 同時音声翻訳の定式化

音声翻訳システムの内部で使用される音声認識，機械翻訳，テキスト音声合成の各システムは，確率モデルの形で定式化することが可能である．すなわち，音声翻訳システム $ST_{f\to e}$ は原言語による発話 U_f が与えられた下での目的言語の発話 U_e の事後確率最大化問題として考えられ，式 (3) の形で表現することができる．

$$
\begin{aligned}
U_e^* &= ST_{f\to e}(U_f) && (1)\\
&\equiv \arg\max_{U_e} \Pr(U_e|U_f) && (2)\\
&= \arg\max_{U_e} \sum_{\boldsymbol{e},\boldsymbol{f}} \Pr(U_e|\boldsymbol{e},\boldsymbol{f},U_f)\Pr(\boldsymbol{e}|\boldsymbol{f},U_f)\Pr(\boldsymbol{f}|U_f) && (3)
\end{aligned}
$$

ここで $\boldsymbol{f}$ 及び $\boldsymbol{e}$ は原言語と目的言語の単語列を表す．式 (3) の各項は左から順にテキスト音声合成，機械翻訳，音声認識の各モジュールに対応する確率分布であり，音声翻訳システム全体はこれらの結合確率で与えられる．ここでモデルを簡単にするために，テキスト音声合成は目的言語の単語列のみ，機械翻訳は原言語の単語列のみに依存するという仮定を導入する．この場合，音声翻訳システムは式 (4) のように表現される．

$$
ST_{f\to e}(U_f) \simeq \arg\max_{U_e} \sum_{\boldsymbol{e},\boldsymbol{f}} \Pr(U_e|\boldsymbol{e})\Pr(\boldsymbol{e}|\boldsymbol{f})\Pr(\boldsymbol{f}|U_f) \qquad (4)
$$

式 (4) にはモジュール間の依存関係が残っている．このためシステム全体の最適化を同時に行わなければならず，システムの構築が複雑になる．音声認識，機械翻訳，テキスト音声合成の各モジュールを個別に最適化するためには，機械翻訳の入力として音声認識の最尤推定値，テキスト音声合成の入力として機械翻訳の最尤推定値を使用し，各モジュールを個別の事後確率最大化問題として表現する．この場合，音声翻訳システムは式 (5) の形で表現される．

$$
ST_{f\to e}(U_f) \simeq (TTS_e \circ MT_{f\to e} \circ ASR_f)(U_f) \qquad (5)
$$

ここで $\circ$ は関数の合成を表す．ASR_f ， $MT_{f\to e}$ ， TTS_e はそれぞれ原言語の音声認識器，原言語から目的言語への機械翻訳器，目的言語のテキスト音声合成器

であり，式 (7)，(9)，(11) で表現される．

$$\begin{aligned}
\boldsymbol{f}^* &= ASR_f(U_f) && (6)\\
&\equiv \arg\max_{\boldsymbol{f}} \Pr(\boldsymbol{f}|U_f) && (7)\\
\boldsymbol{e}^* &= MT_{f\to e}(\boldsymbol{f}^*) && (8)\\
&\equiv \arg\max_{\boldsymbol{e}} \Pr(\boldsymbol{e}|\boldsymbol{f}^*) && (9)\\
U_e^* &= TTS_e(\boldsymbol{e}^*) && (10)\\
&\equiv \arg\max_{U_e} \Pr(U_e|\boldsymbol{e}^*) && (11)
\end{aligned}$$

式 (5) を基に同時音声翻訳の定式化を考える．同時音声翻訳では，音声認識と機械翻訳の間に音声認識結果の適切な区切り位置を判断する**文分割法**が追加される．よって，同時音声翻訳システム $SST_{f\to e}$ は式 (13) の形で定式化される．

$$\begin{aligned}
SST_{f\to e}(U_f) &\equiv \arg\max_{U_e} \Pr(U_e|U_f) && (12)\\
&\simeq (\mathrm{concat}\circ TTS_{[e]}\circ MT_{[f]\to[e]}\circ SEG_{f\to[f]}\circ ASR'_f)(U_f) && (13)
\end{aligned}$$

ここで ASR'_f は音声認識器だが，文分割法への入力のために単語単位の出力が可能であるという制約が加わる．$SEG_{f\to[f]}$ は文分割法であり，原言語の単語列 $\boldsymbol{f}$ を受け取り，$\boldsymbol{f}$ を適切な区切り位置で分割した小さな単語列のリスト $[\boldsymbol{f}^{(i)}]_{i=1}^K$ を返す関数である．K は分割により生成される単語列の数であり，これが大きいほど高頻度の分割となり，より短い単語列を翻訳単位とする手法となる．$K=0$ のときは文分割を行わず，通常の音声翻訳と同じ処理となる．単語列の最適な分割位置は式 (15) の事後確率最大化問題として定式化される．$MT_{[f]\to[e]}$ 及び $TTS_{[e]}$ は単語列のリストを入出力とする機械翻訳器とテキスト音声合成器であり，式 (18) 及び式 (11) で定義される．concat は出力を結合する仮想的な関数である．機械翻訳，及びテキスト音声合成は式 (17)，式 (20) のように出力済みの翻訳単位を履歴として考慮することでより文脈的に一貫性のある出力を得ることができると考えられるが，本論文では簡単のためにそれぞれ独立に翻訳，音声合成が行われるものとする．

Algorithm 1 同時音声翻訳の処理手順

$\boldsymbol{f}^* \leftarrow [\,]$
loop
  $w_f^* \leftarrow$ 次の音声認識結果の単語
  $\boldsymbol{f}^* \leftarrow \boldsymbol{f}^* +\!\!+ [w_f^*]$
  if $\boldsymbol{f}^*$ 中に区切り位置が存在 **then**
    $i \leftarrow$ 区切り位置の手前の単語番号
    $\boldsymbol{w}'_f \leftarrow \boldsymbol{f}^*[1:i]$
    $\boldsymbol{f}^* \leftarrow \boldsymbol{f}^*[i+1:|\boldsymbol{f}^*|]$
    $\boldsymbol{e}^* \leftarrow MT_{f\to e}(\boldsymbol{w}'_f)$
    $U_e^* \leftarrow TTS_e(\boldsymbol{e}^*)$
    U_e^* を出力
  end if
end loop

$$
\begin{aligned}
[\boldsymbol{f}^{*(i)}]_{i=1}^K &= SEG_{f\to[f]}(\boldsymbol{f}) && (14)\\
&\equiv \arg\max_{[\boldsymbol{f}^{(i)}]_{i=1}^K} \Pr([\boldsymbol{f}^{(i)}]_{i=1}^K|\boldsymbol{f}) && (15)\\
[\boldsymbol{e}^{*(i)}]_{i=1}^K &= MT_{[f]\to[e]}([\boldsymbol{f}^{*(i)}]_{i=1}^K) && (16)\\
&\equiv [MT_{f\to e}(\boldsymbol{f}^{*(i)}; [\boldsymbol{f}^{*(j)}]_{j=1}^{i-1})]_{i=1}^K && (17)\\
&\simeq [MT_{f\to e}(\boldsymbol{f}^{*(i)})]_{i=1}^K && (18)\\
[U_e^{*(i)}]_{i=1}^K &= TTS_{[e]}([\boldsymbol{e}^{*(i)}]_{i=1}^K) && (19)\\
&\equiv [TTS_e(\boldsymbol{e}^{*(i)}; [\boldsymbol{e}^{*(i)}]_{j=1}^{i-1})]_{i=1}^K && (20)\\
&\simeq [TTS_e(\boldsymbol{e}^{*(i)})]_{i=1}^K && (21)
\end{aligned}
$$

実用的には全ての発話 U_f が得られた下で翻訳を行うのではなく，入力される発話を次々に翻訳して出力してゆくこととなる．アルゴリズム 1 に同時音声翻訳の実際の処理手順を示す．なお，$+\!\!+$ はリスト同士の結合を表す．

2.4 本章のまとめ

本章ではまず従来の音声翻訳システムが音声認識，機械翻訳，テキスト音声合成の各モジュールの連結で実現されていることを述べ，次に音声認識と機械翻訳の間に文分割法を導入することで本論文の対象とする同時音声翻訳システムを定義した．また，文分割法の導入により翻訳開始の時間的制約が解決されること，及び通常の音声翻訳とは異なる翻訳戦略が必要であることを述べた．最後に同時音声翻訳システムの処理全体に関して確率モデルの観点から定式化を行い，モジュール間の独立性に関する仮定を導入することでモデルの簡略化を行った．

3. 統計的機械翻訳に関する諸技術

計算機により処理を行う翻訳手法を総称して**機械翻訳 (Machine Translation: MT)** と呼ぶが，このうち式 (9) のように，原言語の単語列 $\boldsymbol{f} = [f_1, f_2, \cdots, f_{|\boldsymbol{f}|}]$ が与えられた下で事後確率が最大となるような目的言語の単語列 $\boldsymbol{e} = [e_1, e_2, \cdots, e_{|\boldsymbol{e}|}]$ を求める探索問題として定式化されるものを**統計的機械翻訳 (Statistical Machine Translation: SMT)** と呼ぶ．本章では，本論文の中心的な話題である統計的機械翻訳に関わる諸技術の解説を行う．最初に，最も基本的な翻訳手法であり，より高度な翻訳手法の基礎となる**単語アライメント**について解説する．次に単語アライメントを応用する翻訳手法，特に**句に基づく翻訳**と **Tree-To-String 翻訳**について解説する．また，これらの翻訳手法で使用される要素技術として**構文解析**，**言語モデル**，**対数線型モデル**についての解説を行う．最後に，機械翻訳により生成された翻訳文の品質を評価する手法について解説する．

3.1 単語アライメント

考えられる翻訳手法で最も単純なものは,「this」は「これ」,「pen」は「ペン」といったように翻訳対象となる言語対の対訳辞書を用意し，原言語の単語を対訳辞書に従って目的言語の単語へ順次置き換えるものである．このような対訳辞書は人手により用意することも可能であるが，翻訳対象の言語対に関する大量の対訳文の集合が与えられた下で機械学習により辞書を生成する手法が考えられる．このような手法は**単語アライメント (Word Alignment)** または単に**アライメント (Alignment)** と呼ばれる．単語アライメントの学習対象となる対訳文の集合を**対訳コーパス**と呼ぶ.

単語アライメントの代表的な手法は IBM モデル [2] と呼ばれるもので，翻訳の過程を式 (24) に示す**雑音のある通信路モデル (Noisy Channel Model)** により

表現する．

$$
\begin{aligned}
MT_{f\to e}(\boldsymbol{f}) &\equiv \arg\max_{\boldsymbol{e}} \Pr(\boldsymbol{e}|\boldsymbol{f}) && (22)\\
&= \arg\max_{\boldsymbol{e}} \frac{\Pr(\boldsymbol{f}|\boldsymbol{e})\Pr(\boldsymbol{e})}{\Pr(\boldsymbol{f})} && (23)\\
&= \Pr(\boldsymbol{f}|\boldsymbol{e})\Pr(\boldsymbol{e}) && (24)
\end{aligned}
$$

ここで $\Pr(\boldsymbol{f}|\boldsymbol{e})$ は**翻訳モデル**と呼ばれ，$\boldsymbol{e}$ から $\boldsymbol{f}$ が生成される確率である．また，$\Pr(\boldsymbol{e})$ は後述する**言語モデル**である．今，翻訳モデルとして単語単位の置き換えを考えるとすると，$\boldsymbol{f}$ と $\boldsymbol{e}$ の間には単語同士の対応関係があると考えられる．この対応関係を特にアライメントと呼ぶ．英日翻訳におけるアライメントの例を図 3 に示す．雑音のある通信路モデルの上では，アライメントは一意に定まるものではなく確率的であり，様々な組み合わせが考えられる．このため翻訳モデル $\Pr(\boldsymbol{f}|\boldsymbol{e})$ は式式 (25) に示すように，アライメント $\boldsymbol{a}$ に関する周辺化が行われた確率分布であると考えられる．

$$
\Pr(\boldsymbol{f}|\boldsymbol{e}) = \sum_{\boldsymbol{a}} \Pr(\boldsymbol{f},\boldsymbol{a}|\boldsymbol{e}) \qquad (25)
$$

一つの対訳文 $\boldsymbol{f}$，$\boldsymbol{e}$ が与えられたとき，この対訳文に対して可能なアライメントは任意の単語の組み合わせだけあるため，$O(\exp|\boldsymbol{f}||\boldsymbol{e}|)$ 程度存在することとなる．このため，式 (25) の総和を明示的に計算するのは避け，アライメントに関する制約条件を導入することで計算量の削減を行う．IBM モデルでは $\boldsymbol{e}$ の各単語がただ一つのアライメントを持つという制約条件が用いられる．このため $\boldsymbol{a}$ は $\boldsymbol{e}$ の各要素の対応先である $\boldsymbol{f}$ の単語の位置を表す配列となる．図 3 の場合は $\boldsymbol{a} = [1, 0, 6, 0, 9, 7, 5, 5, 5, 3, 3, 10]$ となる．ただし $\boldsymbol{f}$ の最初の単語を 1 とし，対応のない単語「は」「を」は $\boldsymbol{f}$ の 0 番目に仮想的に挿入される単語への対応（NULL アライメント）としている．

IBM モデルでは EM アルゴリズムを用いて式 (25) の最適化を行うことで，最適なアライメントの組み合わせ (図 3 の黒い四角の組み合わせ) を学習する．このとき式 (25) の分解方法により様々な手法が考えられる．IBM モデルではこの違いにより，最も簡単なモデル 1 から最も複雑なモデル 5 までの 5 種類のモデル式が提案されている．

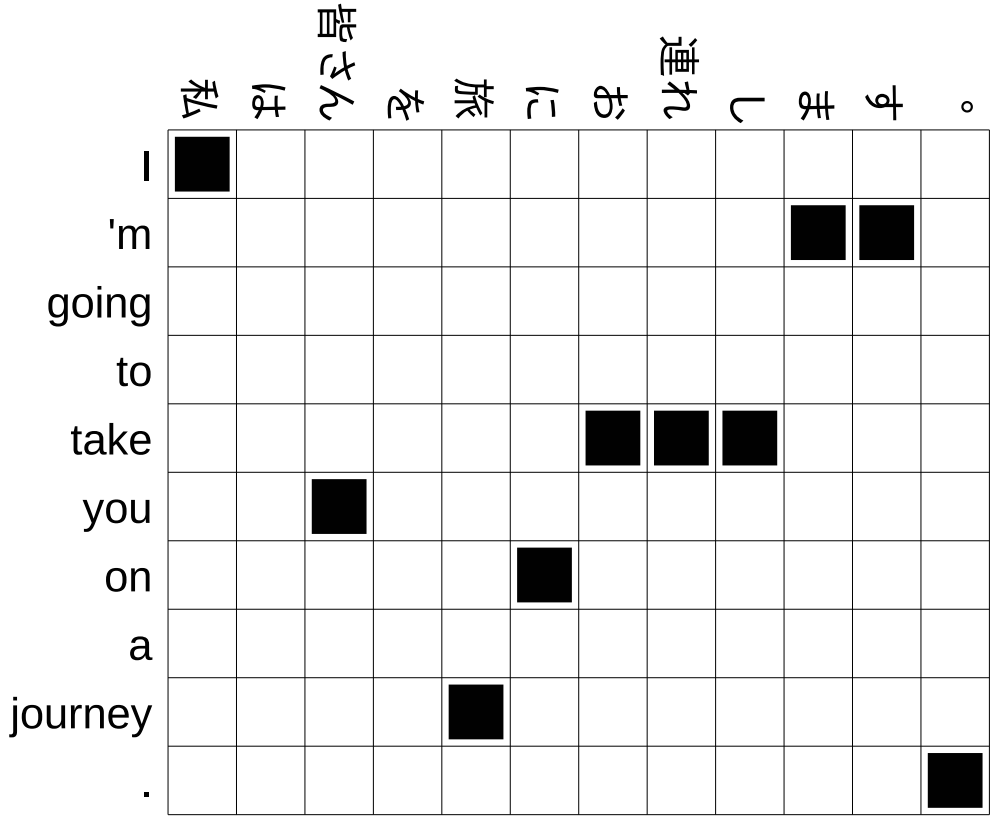

図 3 アライメントの例

IBM モデルのみでは一方向の翻訳についてしか考慮されないため，実際には $\boldsymbol{f}$ と $\boldsymbol{e}$ を逆転させて学習したアライメントを合成することで両方向の翻訳について対等なアライメントを生成する．また IBM モデル以外の単語アライメント学習法も提案されており，単語列以外のより多くの情報を素性として用いることで精度を向上する手法 [33]，反転トランスダクション文法 (Inverse Transduction Grammar: ITG) を用いて一定の単語のまとまりごとにアライメントを生成可能な手法 [26] などが挙げられる．

3.2 句に基づく翻訳

自然言語の文は構文や意味が 1 単語ずつで完結するわけではなく，実際には複数の単語が連結した**句 (フレーズ, Phrase)** が重要であることが多い．このため機械翻訳においても，単語アライメントのような単語同士の対応関係だけでなく，句と句の間の対応関係を用いて翻訳を行うことが有効であると考えられる．翻訳の最小単位として句を用いる翻訳手法を**句に基づく翻訳 (Phrase Based Machine Translation: PBMT)**[18] と呼ぶ．句に基づく翻訳も雑音のある通信路モデルにより翻訳過程を定式化する点は単語アライメントと同様であるが，式 (28) に示すように確率分布を分解する手法が異なる．

$$
\begin{aligned}
MT_{f\to e} &\equiv \arg\max_{\boldsymbol{e}} \Pr(\boldsymbol{f}|\boldsymbol{e})\Pr(\boldsymbol{e}) && (26)\\
&= \arg\max_{\boldsymbol{e}} \sum_{\boldsymbol{\phi},\boldsymbol{a}} \Pr(\boldsymbol{f},\boldsymbol{\phi},\boldsymbol{a}|\boldsymbol{e})\Pr(\boldsymbol{e}) && (27)\\
&\simeq \arg\max_{\boldsymbol{e}} \sum_{\boldsymbol{\phi},\boldsymbol{a}} \Pr(\boldsymbol{f},\boldsymbol{a}|\boldsymbol{\phi})\Pr(\boldsymbol{\phi}|\boldsymbol{e})\Pr(\boldsymbol{e}) && (28)
\end{aligned}
$$

句に基づく翻訳の例，及び式 (28) の各確率分布との対応を図 4 に示す．$\boldsymbol{\phi} = [\langle \boldsymbol{f}^{(a_i)}, \boldsymbol{e}^{(i)} \rangle](1 \le i \le |\boldsymbol{\phi}|)$ は原言語の単語列と目的言語の単語列の組の列であり，各組 $\langle \boldsymbol{f}^{(a_i)}, \boldsymbol{e}^{(i)} \rangle$ は**フレーズペア (Phrase Pair)** と呼ばれる．また $\boldsymbol{a} = [a_i](1 \le i \le |\boldsymbol{\phi}|)$ は目的言語の句 $\boldsymbol{e}^{(i)}$ と対応するアライメントであり，$\boldsymbol{e}^{(i)}$ に対応するフレーズペアの原言語の句 $\boldsymbol{f}^{(a_i)}$ の位置を指す．$\Pr(\boldsymbol{\phi}|\boldsymbol{e})$ は**句翻訳モデル (Phrase Translation Model)** と呼ばれ，目的言語の単語列 $\boldsymbol{e}$ からフレーズペアの列 $\boldsymbol{\phi}$ を生成する過程を表す．$\Pr(\boldsymbol{f},\boldsymbol{a}|\boldsymbol{\phi})$ は**句歪みモデル (Phrase Distortion Model)** と呼ばれ，生成されたフレーズペアの並べ替えにより原言語の単語列が生成される過程を表す．式 (28) の総和は現実的には計算が難しいため，$\boldsymbol{\phi}$，$\boldsymbol{a}$ について最も事後確率が高い候補のみを採用することで問題を簡略化できる．この場合のモデルは式 (29) となり，このモデルにより生成される翻訳文は**ビタビ (Viterbi) 翻訳**と呼ばれる．

$$
MT_{f\to e}(\boldsymbol{f}) \equiv \arg\max_{\langle \boldsymbol{e},\boldsymbol{\phi},\boldsymbol{a} \rangle} \Pr(\boldsymbol{f},\boldsymbol{a}|\boldsymbol{\phi})\Pr(\boldsymbol{\phi}|\boldsymbol{e})\Pr(\boldsymbol{e}) \qquad (29)
$$

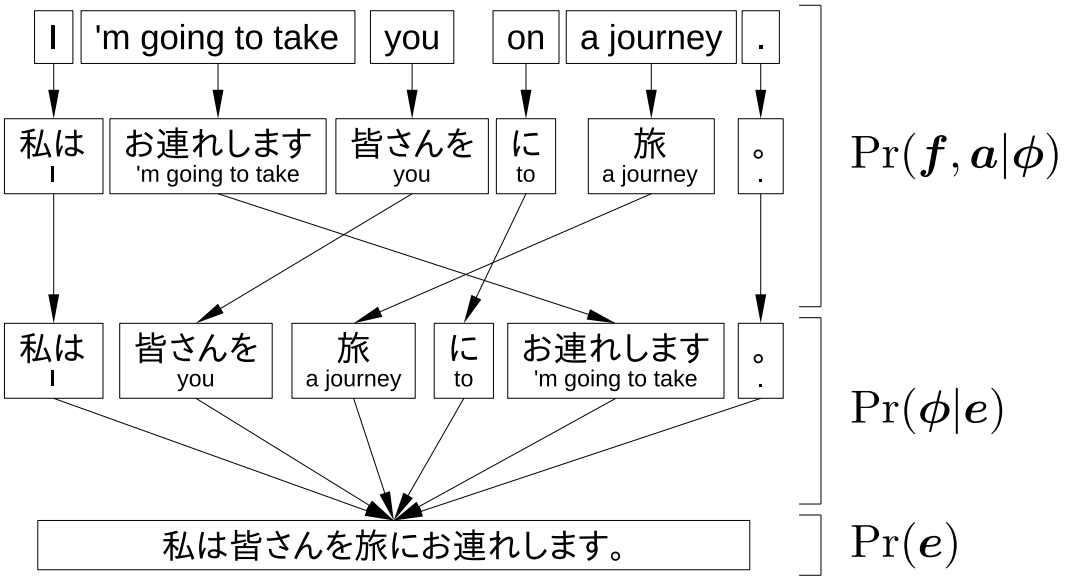

図 4　句に基づく翻訳

句に基づく翻訳の学習も $\Pr(\boldsymbol{f}|\boldsymbol{e})$ を最大化する点では単語アライメントと同様である．しかし句に対するアライメント $\boldsymbol{a}$ の列挙は#P 困難であり直接探索するのは難しいため，$\Pr(\boldsymbol{f}, \boldsymbol{a}|\boldsymbol{\phi})$ と $\Pr(\boldsymbol{\phi}|\boldsymbol{e})$ は独立したモデルであると仮定する．更に $\Pr(\boldsymbol{\phi}|\boldsymbol{e})$ については，式 (30) に示すように $\boldsymbol{\phi}$ に含まれる各フレーズペアについて独立であると仮定する．

$$\Pr(\boldsymbol{\phi}|\boldsymbol{e}) \simeq \prod_{i=1}^{|\boldsymbol{\phi}|} \Pr(\boldsymbol{f}^{(i)}|\boldsymbol{e}^{(i)}) \tag{30}$$

この仮定により，句翻訳モデルは各フレーズペア $\langle \bar{\boldsymbol{f}}, \bar{\boldsymbol{e}} \rangle$ ごとに個別の確率を推定すればモデル化が可能となる．この推定は対訳コーパスからのフレーズペアの抽出と数え上げによって行われる．まず N 文からなる対訳コーパス $\langle \mathcal{F} = [\boldsymbol{f}_i], \mathcal{E} = [\boldsymbol{e}_i] \rangle\, (1 \leq i \leq N)$ に対して単語アライメントの学習を行い，各文に対するアライメント $\mathcal{A} = [\boldsymbol{a}_i] (1 \leq i \leq N)$ を得る．この単語アライメントによる制約の下で，特定のヒューリスティクスを用いて各対訳文 $\boldsymbol{f}_i$ と $\boldsymbol{e}_i$ から可能なフレーズペアの集合 $\boldsymbol{\Phi}_{\langle \boldsymbol{f}, \boldsymbol{e}, \boldsymbol{a} \rangle}$ を得る．最終的に対訳コーパスとアライメントから得られるフレーズペ

アの集合 (フレーズテーブル) は式 (31) となる．

$$\Phi \equiv \bigcup_{\langle \boldsymbol{f},\boldsymbol{e},\boldsymbol{a}\rangle \in \langle \mathcal{F},\mathcal{E},\mathcal{A}\rangle} \Phi_{\langle \boldsymbol{f},\boldsymbol{e},\boldsymbol{a}\rangle} \tag{31}$$

あるフレーズペア $\langle \bar{\boldsymbol{f}}, \bar{\boldsymbol{e}} \rangle$ が対訳コーパスから抽出された数を $c(\bar{\boldsymbol{f}}, \bar{\boldsymbol{e}})$ とすると，フレーズペアに対する句翻訳確率 $\Pr(\bar{\boldsymbol{f}}|\bar{\boldsymbol{e}})$ の最尤推定値は式 (32) となる．

$$\Pr(\bar{\boldsymbol{f}}|\bar{\boldsymbol{e}}) \equiv \frac{c(\bar{\boldsymbol{f}}, \bar{\boldsymbol{e}})}{\sum_{\langle \bar{\boldsymbol{f}}', \bar{e} \rangle} c(\bar{\boldsymbol{f}}', \bar{\boldsymbol{e}})} \tag{32}$$

句に基づく翻訳による翻訳文の生成過程は雑音のある通信路モデルによる信号の復元過程と考えられ，ここから**デコーダ**と呼ばれる．入力単語列 $\boldsymbol{f}$ とフレーズテーブル Φ，各フレーズペアに対する句翻訳確率 $\Pr(\bar{\boldsymbol{f}}|\bar{\boldsymbol{e}})$ が与えられた下で適切なフレーズペアの列 $\boldsymbol{\phi}$ を生成する問題は NP 完全であり，直接解くには適さない．このため動的計画法に基づく手法である **Held-Karp アルゴリズム**により，適用するフレーズペアを一つずつ追加してゆくことで $\boldsymbol{\phi}$ を生成する．

3.3 Tree-To-String 翻訳

原言語の文 $\boldsymbol{f}$ に関して単語列よりも詳しい情報が使用できる場合，これらの情報を考慮した翻訳モデルを用いることでより有効な翻訳文を生成できると考えられる．このような情報としては，$\boldsymbol{f}$ に関する**構文木 (Syntax Tree)** $T_{\boldsymbol{f}}$ が挙げられる．構文木は単語や句などの文に含まれる部分的な構造の役割や相互の関係性を特定の生成規則に基づく木構造で表現したものである．原言語に関する構文情報を考慮して翻訳を行うモデルを**構文に基づく翻訳 (Syntax Based Machine Translation)** と呼び，特に原言語の構文木から目的言語の単語列を生成するモデルは **Tree-To-String 翻訳** [14] と呼ばれる．

Tree-To-String 翻訳のモデルは式 (36) に示すように，式 (9) に対して構文木に

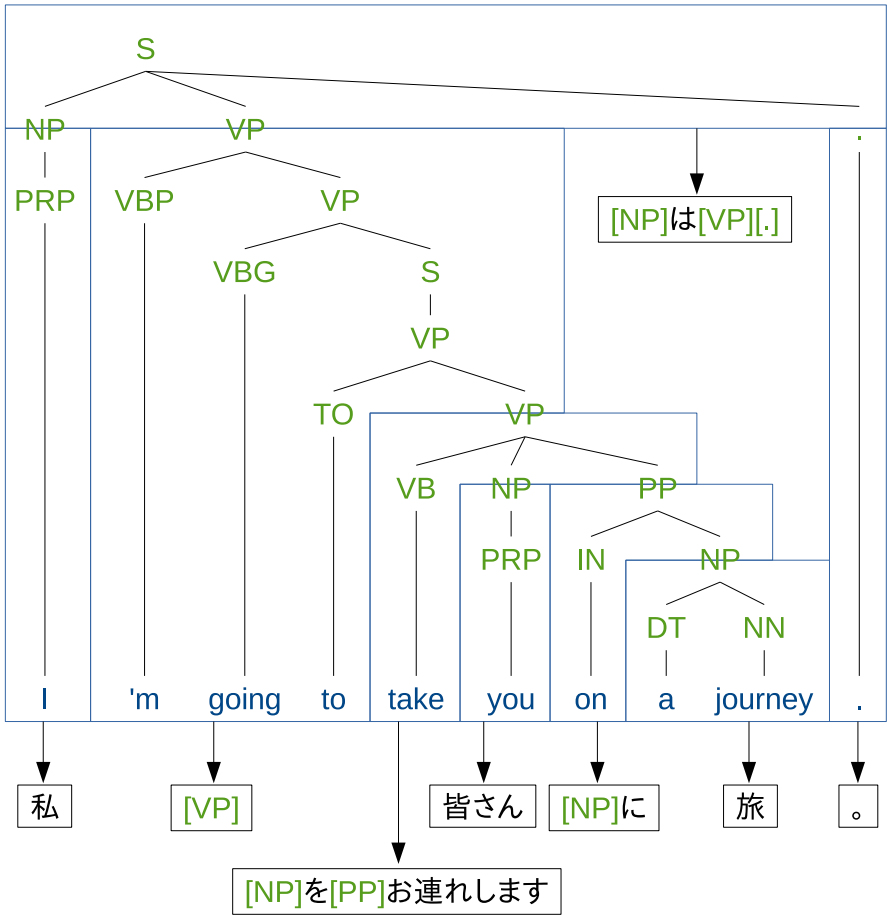

図 5　Tree-To-String 翻訳

関する周辺化を行うことにより導出される．

$$
\begin{aligned}
MT_{f\to e}(\boldsymbol{f}) &\equiv \arg\max_{\boldsymbol{e}} \Pr(\boldsymbol{e}|\boldsymbol{f}) && (33)\\
&= \arg\max_{\boldsymbol{e}} \sum_{T_f} \Pr(\boldsymbol{e}|T_{\boldsymbol{f}}, \boldsymbol{f}) \Pr(T_{\boldsymbol{f}}|\boldsymbol{f}) && (34)\\
&\simeq \arg\max_{\boldsymbol{e}} \sum_{T_{\boldsymbol{f}}} \Pr(\boldsymbol{e}|T_{\boldsymbol{f}}) \Pr(T_{\boldsymbol{f}}|\boldsymbol{f}) && (35)\\
&\simeq \arg\max_{\boldsymbol{e}} \Pr(\boldsymbol{e}|T^*_{\boldsymbol{f}}) && (36)
\end{aligned}
$$

ここで，

$$T^*_{\boldsymbol{f}} \equiv \arg\max_{T_{\boldsymbol{f}}} \Pr(T_{\boldsymbol{f}}|\boldsymbol{f}) \tag{37}$$

であり，$\boldsymbol{f}$ を表現する構文木のうち，最も事後確率の高いものとする．

Tree-To-String 翻訳の例を図 5 に示す．ここで図 5 の構文木は**句構造 (Phrase Structure)** と呼ばれ，文の構造を**文脈自由文法 (Context Free Grammar: CFG)** によるルールの集合として表現する．句に基づく翻訳では原言語の単語列をいくつかの句に分割し，各句を個別に翻訳することで全体的な翻訳文を得ていた．Tree-To-String 翻訳でも同様に原言語の構文木をいくつかの部分木に分割し，各部分木 $\bar{T}_{\boldsymbol{f}}$ と目的言語の句 $\bar{\boldsymbol{e}}$ の組である**同期ルール (Synchronous Rule)** $\langle \bar{T}_{\boldsymbol{f}}, \bar{\boldsymbol{e}} \rangle$ を用いて翻訳文を生成する．同期ルールという名称は，Tree-To-String 翻訳が**同期木置換文法 (Synchronous Tree Substitution Grammar: STSG)** のサブセットであることによる．句に基づく翻訳と異なる点は，$\bar{T}_{\boldsymbol{f}}$ と $\bar{\boldsymbol{e}}$ には構文要素のプレースホルダ (図 5 の [NP] など) を含む点であり，これにより各部分木の子要素の翻訳文が文法的に正しい位置へ挿入される．

Tree-To-String 翻訳の同期ルールは，句に基づく翻訳と同様，事前に単語アライメントを学習した対訳コーパス $\langle \mathcal{T}_{\mathcal{F}}, \mathcal{E}, \mathcal{A} \rangle$ から特定のヒューリスティクスにより抽出される．ここで $\mathcal{T}_{\mathcal{F}}$ は原言語の単語列 $\mathcal{F}$ から生成された構文木の列である．このようなアルゴリズムとしては，生成される部分木の整合性を考慮した **Galley-Hopkins-Knight-Marcu アルゴリズム (GHKM アルゴリズム)** が知られている [9]．

3.4 構文解析

Tree-To-String 翻訳では原言語の単語列 $\boldsymbol{f}$ に対する構文木 $T_{\boldsymbol{f}}$ を事前に生成する必要がある．構文木の生成処理は**構文解析 (Parsing)** と呼ばれ，生成対象である構文木の種類により様々な手法が用いられる．図 5 のような単語列に関する句構造を生成する構文解析は特に**句構造解析**と呼ばれ，単語列 $\boldsymbol{w}$ と解析対象の言語を表す文脈自由文法 $\mathcal{G}_{CFG}$ が与えられた下で，$\boldsymbol{w}$ を $\mathcal{G}$ で受理可能かどうかを判定する問題として定式化される．

文脈自由文法は式 (38) に示す 5 つ組により表現される．

$$\mathcal{G}_{CFG} \equiv \langle \mathcal{V}, \Sigma, \mathcal{R}, S \rangle \tag{38}$$

$\mathcal{V}$ は**非終端記号 (Non-Terminal Symbol)** の集合であり，各非終端記号はその記号が被覆する句の文法的な特徴を表す．Σ は**終端記号 (Terminal Symbol)** の集合であり，これは句構造では単語の集合と等価である．$\mathcal{R}$ は**生成規則 (Production Rule)** と呼ばれ，各生成規則は 1 個の非終端記号 $v \in \mathcal{V}$ から 1 個以上の非終端記号か終端記号の列 $[u_i](1 \leq i \leq N, u_i \in (\mathcal{V} \cup \Sigma)^*)$ への変換 $v \to [u_i]$ を表す．$S \in \mathcal{V}$ は**開始記号 (Start Symbol)** と呼ばれ，$\mathcal{G}_{CFG}$ が生成する全ての構文木の根となる非終端記号である．

自然言語は文に曖昧性を含むため，厳密な $\mathcal{G}_{CFG}$ を定めることはできず確率的となる．この現象を捉えるために，各文法規則 $r = (v \to [u_i]) \in \mathcal{R}$ に対して式 (40) による生成確率 $\Pr(r)$ を付与する．ただし，$c(r)$ は人手により作成された句構造のデータベース (**ツリーバンク (Treebank)**) 中に含まれるルールの数である．

$$\Pr(r) \simeq \Pr([u_i]|v) \tag{39}$$

$$\equiv \frac{c([u_i]|v)}{\sum_{(v \to [u'_j]) \in \mathcal{R}} c([u'_j]|v)} \tag{40}$$

これを用いて，単語列 $\boldsymbol{w}$ から構文木 $T_{\boldsymbol{w}}$ を生成する確率 $\Pr(T_{\boldsymbol{w}}|\boldsymbol{w})$ を式 (42) によるルールごとに独立した確率の積として近似し，構文解析を 式 (42) の最大化問題として定義する．このモデルは**確率的文脈自由文法 (Probabilistic Context Free Grammar: PCFG)** と呼ばれる．

$$T^*_{\boldsymbol{w}} \equiv \arg\max_{T_{\boldsymbol{w}}} \Pr(T_{\boldsymbol{w}}|\boldsymbol{w}) \tag{41}$$

$$\simeq \arg\max_{T_{\boldsymbol{w}}} \prod_{r \in T_{\boldsymbol{w}}} \Pr(r) \tag{42}$$

PCFG の探索は，CFG による単語列の受理問題を解くアルゴリズムである **Cocke-Kasami-Younger アルゴリズム (CKY アルゴリズム)** に確率による距離尺度を導入することで実現できる．CKY アルゴリズムはボトムアップ的に動

作する動的計画法の一種であり，生成規則ごとに確率が独立であれば式 (42) の厳密解を求めることができる．

句構造解析の手法としては，後の章で解説するツリーバンクの構文情報からより詳細な規則を生成する手法が，単純な PCFG よりも高い解析精度を実現することが知られている [35].

3.5 言語モデル

言語モデル (Language Model: LM) は単語列 $\boldsymbol{w}$ の「言語らしさ」を表現するモデルである．ここで言語らしさとは，単語列 $\boldsymbol{w} = [w_i](1 \leq i \leq |\boldsymbol{w}|)$ の文書中での出現しやすさとして表現される．機械翻訳において言語モデルは翻訳文をより目的言語として流暢な文とするために用いられ，言語モデルの品質が翻訳文の品質に大きな影響を与える．

形式的には式 (43) に示すように，言語モデルは $\boldsymbol{w}$ に含まれる全ての単語に関する結合確率として定義される．

$$\Pr(\boldsymbol{w}) \equiv \prod_{i=1}^{|\boldsymbol{w}|} \Pr(w_i|w_1^{i-1}) \tag{43}$$

ただし，w_i^j は $\boldsymbol{w}$ の i 番目から j 番目までの単語による部分列とする．$\Pr(w_i|w_1^{i-1})$ は，**文脈 (Context)** と呼ばれる前方の単語列 w_1^{i-1} が得られた下で，次の単語 w_i が出現する条件付き確率である．$\Pr(w_i|w_1^{i-1})$ が全て与えられれば $\Pr(\boldsymbol{w})$ が計算できるため，言語モデルの実態はこの確率分布であると考えることができる．

最も簡単な言語モデルは式 (44) に示すように，条件付き確率として対象とする言語のコーパス中に含まれる単語列 w_1^i の出現回数 $c(w_i^j)$ による最尤推定値を用いるものである．

$$\Pr(w_i|w_1^{i-1}) \equiv \frac{c(w_1^i)}{\sum_{w'} c(w_1^{i-1} +\!\!+ [w'])} \tag{44}$$

ただし $+\!\!+$ は列の連結を表す．文脈 w_1^{i-1} の長さが増えるに従って単語列がコーパス中に出現する回数が減少するため，式 (44) では長い文脈による条件付き確率が

ほとんど 0 となってしまう．これは**ゼロ頻度問題 (Zero Frequency Problem)** として知られるもので，$\boldsymbol{w}$ が学習対象のコーパスに出現しないような単語列を含む場合に $\Pr(\boldsymbol{w})$ が 0 となってしまい，評価が不可能となってしまうものである．

この問題を回避するために，式 (45) に示すように w_i の出現確率は直前 n 個の単語列 w_{i-n}^{i-1} のみに依存するという仮定を導入する．

$$\Pr(w_i|w_1^{i-1}) \simeq \Pr(w_i|w_{i-n}^{i-1}) \tag{45}$$

連続した n 個の単語列を **n-gram** と呼び，この近似による言語モデルは **n-gram 言語モデル**と呼ばれる．n-gram 言語モデルは単語列全体の確率だけでなく，前後 $n-1$ 単語を記憶しておくことで部分的な確率を求めることができる．このため翻訳文の生成処理において，部分的な n-gram 言語モデル確率を探索時の情報として用いることができる．

式 (45) の近似により確率分布の空間が圧縮されるため，ゼロ頻度問題の多くが回避できる．しかし完全に確率が 0 となる条件が排除できるものではないため，n-gram 言語モデルでは更に**平滑化 (Smoothing)** と呼ばれる確率分布に対する内挿処理により，全ての条件付き確率に 0 以外の値を与える変形が行われる．このような手法で高精度なモデルとして知られているものに **Kneser-Ney 平滑化 (Kneser-Ney Smoothing)**[16] がある．

n-gram 言語モデル以外の言語モデルとしては，生成過程を**ニューラルネットワーク (Neural Network: NN)** でモデル化する手法が知られている [23, 22, 36]．これらのうち**リカレントニューラルネットワーク (Recurrent Neural Network: RNN)** を内部構造として用いる言語モデルは n-gram 言語モデルよりも高品質として知られているが，確率分布の依存する文脈の長さに制限がないため，翻訳過程で使用することは困難である．このため，これらの言語モデルは翻訳器が出力した複数の翻訳文 (**n-ベスト翻訳**) から最も高品質な翻訳文を選択する**再ランキング (Reranking)** において使用される．

3.6 対数線型モデル

翻訳文が生成されるのに要したルールの数や翻訳文の単語数など，翻訳モデルや言語モデル以外にも翻訳時に考慮すると有用であると考えられる統計量が存在する．これらの情報を確率分布に取り入れる手法として，式 (46) と式 (47) に示す**対数線型モデル**が用いられる．

$$\Pr(\boldsymbol{e}|\boldsymbol{f}) \equiv \frac{\exp(\boldsymbol{\lambda}\cdot\boldsymbol{\phi}(\boldsymbol{f},\boldsymbol{e}))}{\sum_{\boldsymbol{e}}\exp(\boldsymbol{\lambda}\cdot\boldsymbol{\phi}(\boldsymbol{f},\boldsymbol{e}))} \tag{46}$$

$$\Pr(\boldsymbol{e}|T_f) \equiv \frac{\exp(\boldsymbol{\lambda}\cdot\boldsymbol{\phi}(T_f,\boldsymbol{e}))}{\sum_{\boldsymbol{e}}\exp(\boldsymbol{\lambda}\cdot\boldsymbol{\phi}(T_f,\boldsymbol{e}))} \tag{47}$$

ここで $\cdot$ はベクトルの内積を表す．$\boldsymbol{\phi}(\cdot)$ は素性関数であり，翻訳器の入出力から有用な情報を抽出して実数ベクトルの形で返す関数である．$\boldsymbol{\lambda}$ は重みベクトルであり，素性関数によって生成された各要素の重要度を与える．式 (46) と式 (47) の分母は分配関数と呼ばれ，入力 $\boldsymbol{f}$ または T_f が与えられた下では定数となる．このため翻訳時の目的関数は式 (48)，式 (49) のように単純化される．

$$\arg\max_{\boldsymbol{e}}\Pr(\boldsymbol{e}|\boldsymbol{f}) = \arg\max_{\boldsymbol{e}}\boldsymbol{\lambda}\cdot\boldsymbol{\phi}(\boldsymbol{f},\boldsymbol{e}) \tag{48}$$

$$\arg\max_{\boldsymbol{e}}\Pr(\boldsymbol{e}|T_f) = \arg\max_{\boldsymbol{e}}\boldsymbol{\lambda}\cdot\boldsymbol{\phi}(T_f,\boldsymbol{e}) \tag{49}$$

対数線型モデルは与えられた統計量の下で得られる情報量の期待値を最大化するモデルであり，**最大エントロピーモデル**とも呼ばれる．対数線型モデルでは重みベクトル $\boldsymbol{\lambda}$ の値によって確率分布の形状が変化するため，最適な $\boldsymbol{\lambda}$ を学習する必要がある．これには重み学習用の評価データ $\langle \mathcal{F}=[\boldsymbol{f}_i]_{i=1}^N, \mathcal{E}=[\boldsymbol{e}_i]_{i=1}^N\rangle$ を用意し，式 (51) に示す損失関数 L が最小となる重みベクトルを学習する**最小エラー学習 (Minimum Error Rate Training: MERT)**[27] が知られる．

$$\boldsymbol{\lambda}^* \equiv \arg\min_{\boldsymbol{\lambda}} L(MT_{[f]\to[e]}(\mathcal{F};\boldsymbol{\lambda}),\mathcal{E}) \tag{50}$$

$$\simeq \arg\min_{\boldsymbol{\lambda}} \sum_{\langle \boldsymbol{f},\boldsymbol{e}^*\rangle\in\langle\mathcal{F},\mathcal{E}\rangle} E\left(\arg\max_{\boldsymbol{e}}\boldsymbol{\lambda}\cdot\boldsymbol{\phi}(\boldsymbol{f},\boldsymbol{e}),\boldsymbol{e}^*\right) \tag{51}$$

ここで $E(\boldsymbol{e}, \boldsymbol{e}^*)$ は各翻訳結果に関する誤差関数であり，$\boldsymbol{e}$ が $\boldsymbol{e}^*$ に近いほど小さな値となる．これは次節に示す自動評価尺度を用いて定義されるのが普通であるが，そのように定義された E は $\boldsymbol{\lambda}$ に対して滑らかな関数である保証がないため，**単体法 (Simplex 法)** に代表されるような勾配に依存しない最適化アルゴリズムにより学習が行われる．

3.7 翻訳結果の品質評価

機械翻訳の手法の良さを調べるには，実際に生成された翻訳文が翻訳として妥当かどうかを一定の評価尺度に従って採点する手法が一般的である． 翻訳文の評価には手法の違いにより，大きく分けて**主観評価**と**自動評価**の 2 種類が存在する．

3.7.1 主観評価

主観評価 (Subjective Evaluation) または**人手評価 (Human Evaluation)** とは，原文とその人手による翻訳文 (**参照訳**)，翻訳器が生成した翻訳文を人間の評価者に提示し，翻訳文の妥当性を予め定めた基準に従って決定してもらう手法である．主観評価に使用される基準としては，翻訳文の文法的，意味的な正しさを評価する Acceptability [10]，原文に含まれる情報の再現性を評価する Adequacy [41]，翻訳文の目的言語としての流暢さを評価する Fluency [41] などがある．人手評価は人間を直接評価に使用する関係上，後述の自動評価に対して評価結果の妥当性は高いと考えられる．しかし時間や費用の面で負担の大きい手法であり，評価者や評価した時期の違いにより同じ文でも評価値が安定しない問題がある．

3.7.2 自動評価

主観評価による評価の負担を軽減するために，数式を用いて計算機により翻訳文を評価する**自動評価 (Automatic Evaluation)** のための尺度が提案されている．自動評価尺度を用いることで高速かつ安価な翻訳の品質評価が可能となるた

め，前述の対数線型モデルの最適化問題など，翻訳結果を何度も評価する必要があるアルゴリズムも実行することが可能となる．

自動評価尺度として主に使用される戦略は，参照訳に対する翻訳文の類似度を調べる手法である．以降の節では参照訳との類似度に基づく自動評価尺度のうち，特に本研究で使用した **BLEU**，**BLEU**+1，及び **RIBES** について説明する．

3.7.3 BLEU

BLEU (Bilingual Evaluation Understudy) [29] は翻訳文と参照訳の間の n-gram 一致率に基づく自動評価尺度である．まず，翻訳器が出力した N 個の翻訳文 $\hat{\mathcal{E}} = [\hat{\boldsymbol{e}}_i](1 \leq i \leq N)$ とその参照訳 $\mathcal{E}^* = [\boldsymbol{e}_i^*](1 \leq i \leq N)$ があるものとする．また，単語列 $\boldsymbol{w}$ に含まれる n-gram の多重集合を $\mathrm{bag}_n(\boldsymbol{w})$ と表記することとする．このとき，評価データ全体で見た翻訳文と参照訳の **n-gram 適合率 (n-gram Precision)** $Prec_n(\hat{\mathcal{E}}, \mathcal{E}^*)$ を 式 (52) で定義する．

$$Prec_n(\hat{\mathcal{E}}, \mathcal{E}^*) \equiv \frac{\sum_{i=1}^{N} |\mathrm{bag}_n(\hat{\boldsymbol{e}}_i) \cap \mathrm{bag}_n(\boldsymbol{e}_i^*)|}{\sum_{i=1}^{N} |\mathrm{bag}_n(\hat{\boldsymbol{e}}_i)|} \tag{52}$$

BLEU では $n = 1, 2, 3, 4$ についての n-gram 適合率を求め，式 (53) に示すこれらの幾何平均をスコアとして用いる．

$$Prec(\hat{\mathcal{E}}, \mathcal{E}^*) \equiv \sqrt[4]{\prod_{n=1}^{4} Prec_n(\hat{\mathcal{E}}, \mathcal{E}^*)} \tag{53}$$

しかし，n-gram 適合率だけでは参照訳に対して極端に短い翻訳文で不当に高いスコアとなってしまう．この問題を避けるために，式 (54) で示される**簡潔ペナルティ (Brevity Penalty)** $BP(\hat{\mathcal{E}}, \mathcal{E}^*)$ を罰則項として追加する．翻訳文が参照訳よりも全体的に短い場合，簡潔ペナルティが 1 よりも小さい係数を与えるため，

不当に高くなった n-gram 適合率を抑制することが期待される．

$$BP(\hat{\mathcal{E}}, \mathcal{E}^*) \equiv \min \left[1, \exp \left(1 - \frac{\sum_{i=1}^{N} |\boldsymbol{e}_i^*|}{\sum_{i=1}^{N} |\hat{\boldsymbol{e}}_i|} \right) \right] \tag{54}$$

以上の各項により，最終的な BLEU スコアは式 (55) によって表される．

$$BLEU(\hat{\mathcal{E}}, \mathcal{E}^*) \equiv Prec(\hat{\mathcal{E}}, \mathcal{E}^*) \times BP(\hat{\mathcal{E}}, \mathcal{E}^*) \tag{55}$$

3.7.4 BLEU+1

BLEU は多数の評価データが一度に与えられた下での評価尺度であり，1 文ずつなどといった少量のデータに対する評価には適さない．これは，データが少ない場合に高次の n-gram 適合率が 0 となる場合が多くなり，適合率の幾何平均を用いる BLEU スコア自体も 0 となってしまうためである．この問題を避けるために，n-gram 適合率に対して式 (56) に示す修正を加える．

$$Prec'_n(\hat{\mathcal{E}}, \mathcal{E}^*) \equiv \frac{\sum_{i=1}^{N} \left(|\mathrm{bag}_n(\hat{\boldsymbol{e}}_i) \cap \mathrm{bag}_n(\boldsymbol{e}_i^*)| + \delta(n) \right)}{\sum_{i=1}^{N} \left(|\mathrm{bag}_n(\hat{\boldsymbol{e}}_i)| + \delta(n) \right)} \tag{56}$$

ここで δ は式 (57) で示される平滑化項である．

$$\delta(n) \equiv \begin{cases} 0, & \text{if } n = 1 \\ \varepsilon, & \text{otherwise} \end{cases} \tag{57}$$

この変形により，少量の評価データでも妥当な評価値を算出することができるようになる．特に式 (57) の ε を 1 としたものを BLEU+1 [19] と呼ぶ．

3.7.5 RIBES

BLEU は文に含まれる単語や句ごとの一致を重視する手法であるが，日本語と英語のように文法的特徴の異なる言語対では単語の順序が大きく入れ替わるため，

部分ごとの一致だけでなく，対応する単語同士の位置関係に関する情報も評価に有用であると考えられる．RIBES [15] は参照訳と翻訳文の単語の一致に加えて，一致した単語に関する順位相関を使用する尺度である．

$\boldsymbol{h}(\hat{\boldsymbol{e}}, \boldsymbol{e}^*)$ を翻訳文 $\hat{\boldsymbol{e}}$ の各単語に対応する参照訳 $\boldsymbol{e}^*$ 中の単語の位置を表す順位ベクトルとする．ただし同一単語が複数ある場合は周囲の単語の一致により対応を調べ，対応する単語が参照訳中に存在しないものは $\boldsymbol{h}$ に含めないものとする．順位ベクトル $\boldsymbol{h}$ から，$\boldsymbol{e}^*$ と $\hat{\boldsymbol{e}}$ に関する順位相関 $Corr(\boldsymbol{e}^*, \hat{\boldsymbol{e}})$ をケンドールの τ **(Kendall's τ)** を用いて式 (59) のように定義する．

$$Corr(\hat{\boldsymbol{e}}, \boldsymbol{e}^*) \equiv \frac{\tau(\boldsymbol{h}(\hat{\boldsymbol{e}}, \boldsymbol{e}^*)) + 1}{2} \tag{58}$$

$$\equiv \frac{\sum_{k=1}^{|\boldsymbol{h}(\hat{\boldsymbol{e}},\boldsymbol{e}^*)|-1} \sum_{k'=k+1}^{|\boldsymbol{h}(\hat{\boldsymbol{e}},\boldsymbol{e}^*)|} \delta(h_k, h_{k'})}{\begin{pmatrix} |\boldsymbol{h}(\hat{\boldsymbol{e}}, \boldsymbol{e}^*)| \\ 2 \end{pmatrix}} \tag{59}$$

また文脈を考慮した 1-gram 一致率 $P_1(\hat{\boldsymbol{e}}, \boldsymbol{e}^*)$，文単位の簡潔ペナルティ $BP'(\hat{\boldsymbol{e}}, \boldsymbol{e}^*)$ を式 (60)，式 (61) で定義する．

$$P_1(\hat{\boldsymbol{e}}, \boldsymbol{e}^*) \equiv \frac{\boldsymbol{h}(\hat{\boldsymbol{e}}, \boldsymbol{e}^*)}{|\boldsymbol{e}^*|} \tag{60}$$

$$BP'(\hat{\boldsymbol{e}}, \boldsymbol{e}^*) \equiv \min\left[1, \exp\left(1 - \frac{|\hat{\boldsymbol{e}}|}{|\boldsymbol{e}^*|}\right)\right] \tag{61}$$

これらの各項により，文に対する RIBES スコアは式 (62) のように定義される．

$$RIBES(\hat{\boldsymbol{e}}, \boldsymbol{e}^*) \equiv Corr(\hat{\boldsymbol{e}}, \boldsymbol{e}^*) \times P_1(\hat{\boldsymbol{e}}, \boldsymbol{e}^*)^{\alpha} \times BP(\hat{\boldsymbol{e}}, \boldsymbol{e}^*)^{\beta} \tag{62}$$

ここで α 及び β は評価対象に依存するハイパーパラメータであるが，一般的には $\alpha = 0.25, \beta = 0.1$ を用いる．また，順位相関係数としてケンドールの τ の代わりにスピアマンの ρ **(Spearman's ρ)** を使用する場合もある．

RIBES は文単位で計算される評価尺度であるが，コーパス全体に対する RIBES スコアは式 (63) に示すように，各文の RIBES スコアの相加平均により求められる．

$$RIBES(\hat{\mathcal{E}}, \mathcal{E}^*) \equiv \frac{1}{N} \sum_{i=1}^{N} RIBES(\hat{\boldsymbol{e}}_i, \boldsymbol{e}_i^*) \tag{63}$$

3.8 本章のまとめ

本章では同時音声翻訳の構成要素であり，本論文が主に対象とする統計的機械翻訳の各種手法について解説を行い，その要素技術である各種技術について述べた．また機械翻訳による翻訳文の評価法について解説し，特に 3 種類の自動評価尺度の定式化を行った．

4. 同時音声翻訳のための文分割法の最適化

4.1 概要

同時音声翻訳では従来の音声翻訳とは異なり、話者が発話の途中であっても適宜部分的な翻訳結果を聞き手に提示することで意思疎通の同時性を高め、また文末が明確に示されないような発話に対しても適切に翻訳結果を提示することを目標とする。このためには、音声認識器から出力された単語列をどのように区切れば、翻訳結果として適切な文が得られるかを判断しなければならない。このような処理を**文分割 (Segmentation)** と呼び、様々な手法が提案されている。本章では最初に従来研究による文分割法の解説を行い、これらの問題点を指摘する。次に本研究の提案する文分割法の学習アルゴリズムの解説を行う。最後に各手法を実際に機械翻訳の前処理として適用した場合の翻訳精度と訳出速度を比較することで提案法の優位性を述べる。

4.2 従来の文分割法

4.2.1 話者の沈黙による分割

最も単純な文分割法としては、音声認識結果がある一定時間得られなかった場合に現時点で得られている結果を翻訳単位とするものである [7, 1]。この手法は文の構造や単語に関する情報を全く使用せず、発話の特徴が分割位置に直接影響するため、一貫性のない翻訳単位を生成してしまう。ただし他の手法とは異なり次の入力単語を待ち続けるような動作はしないため、文の途中で話者が発話を完全に停止した場合のような、例外的な状況においては有効である。

4.2.2 一定単語数による分割

単純な手法として次に考えられるのは、認識された単語数がある一定数 K に達した段階で強制的に翻訳単位とするものである [32]。この手法は翻訳単位の単語数が必ず K となるため、実際に翻訳結果が提示されるまでの時間差の見積もりが

行いやすい。しかし音声認識結果が得られない場合に分割を行う手法と同様に実際の発話内容は考慮しないため、翻訳単位の構文的な一貫性は失われてしまう。

4.2.3 文末・区切り記号の挿入位置の推定による分割

話者の沈黙による分割、一定単語数による分割は文分割のために明確なルールを用いる。これらに対して、翻訳単位が文の構造について一貫性を保てるような分割位置を機械学習により推定し、この結果を分割位置として利用する手法が提案されている。このような分割位置として直感的に理解できるのは、「.」「!」「?」などの文末記号の挿入位置である。従来の音声翻訳ではこれらを推定することにより、仮想的な文を翻訳単位としていた [21]。しかし講演や会話などでは明示的な文末が存在しない場合が多く、文末の推定では長時間の待機を要する可能性がある。

このため、文末記号だけでなく「,」「:」「;」のような文中の区切り記号の位置を文末記号と等価に扱うことで、文としてある程度一貫性のある単語列が認識された時点で翻訳単位を生成することができるようになる [32]。

4.2.4 翻訳結果の並べ替え発生確率に基づく分割

以上に示したいずれの手法も、原言語の情報のみを用いて文分割を行う。このため目的言語側でどのような翻訳結果が生成されるかについては考慮されておらず、翻訳に適した位置で文分割が行われているかどうかは不明である。特に英語と日本語のような、文法構造が大きく異なり、単語の並べ替えが頻繁に発生するような言語対では、原言語で一貫性のある単語列が目的言語でも有効である保証はない。

この問題に対し、対訳文の単語アライメントを元に「分割位置に対する単語の位置関係が翻訳前後で入れ替わる確率」を求め、この確率が閾確率 P_{th} 以下となる場合に文分割を行う手法が提案されている [8]。この手法では構文的な一貫性よりも単語の並べ替えを重視するため、原言語と目的言語で語順が一致しており、得られた単語を順次翻訳しても問題ないような場合と、大きく語順が異なり、あ

る程度のまとまりで翻訳しなければならない場合の両方に対応している。また、P_{th} によって文分割の頻度をある程度調整することができる。

4.3 従来の文分割法の問題点と解決法

前節で示した音声情報に基づく文分割法、句読点の予測に基づく文分割法、単語アライメントに基づく文分割法の欠点は、いずれの手法においても実際に用いる翻訳手法の情報を考慮しておらず、コーパスから得られる情報のみを用いて文分割を行っている点である。このため、翻訳単位が特定の翻訳器への入力として適切かどうかについては、いずれの従来法でも不明なままである。また単語アライメントに基づく手法は閾確率 P_{th} によって分割頻度を変化させることはできるが、分割頻度と P_{th} の関係が不明であるため、P_{th} を用いて分割頻度を直接制御することは難しい。翻訳開始までの時間は翻訳単位の長さに直接影響するため、分割頻度を明示的に制御できない手法では、翻訳開始までの時間に関しても見積もりを与えることができない。

本研究ではこれら 2 点を明示的に考慮する文分割法を提案する。具体的には、学習済みの翻訳器と機械翻訳のための評価尺度を与えたとき、翻訳器にとって評価尺度を最大化するような文分割法を学習する手法を与える。また、文分割法の学習時に翻訳単位の平均単語数 μ をパラメータとして取り込むことにより、学習データと同じドメインの入力発話に対して、実際の文分割後の平均単語数を μ に近似できることを示す。

本手法で与えるのは文分割法を学習するためのアルゴリズムであり、文分割法そのものは後述するように翻訳器と評価尺度による関数となる。このため、ある翻訳器に対して本手法で得られた文分割法は他の翻訳器に対して使用するのは適切ではないことを明記しておく。

4.4 最適化の枠組み

前述したように、本手法では既に存在する翻訳器に対して、評価尺度に関する最適な文分割法を学習する。まず、文分割法の学習に用いるための原言語と目的

言語の対訳コーパスを $\mathcal{F} \equiv [\boldsymbol{f}_j : 1 \leq j \leq N]$、$\mathcal{E} \equiv [\boldsymbol{e}_j : 1 \leq j \leq N]$ とする。ここで N は両コーパスに含まれる単語列の数であり、各単語列は単独で文を構成するものとする。また $MT(\boldsymbol{f})$ を原言語の単語列 $\boldsymbol{f}$ を受け取り、目的言語の単語列 $\hat{\boldsymbol{e}}$ を返す翻訳器とする。翻訳器は事前に学習を完了しており、本手法はこれを固定的な関数として扱う。実際には MT は $\boldsymbol{f}$ だけでなく、これまでに入力された原言語の文による言語モデルの履歴を考慮する方が、より高い翻訳精度を達成できることが知られている [32]。本研究では単純に $\boldsymbol{f}$ のみの関数として MT を定義し、言語モデルの履歴については今後の課題とする。

本手法の目的は、既に文単位の分割が行われた単語列 $\boldsymbol{f}$ に対し、単語列に含まれる分割可能な位置を探索することで $\boldsymbol{f}$ をより細かい要素の列 $[\boldsymbol{f}^{(i)}]$ に分割することである。実際の音声翻訳では文末が明示的に示されないため、文末による分割も併せて実行する必要があるが、これには従来法 [21] を用いることができる。本手法が文末を分割対象としないのは、文末では必ず分割が行われ、文末の前後で単語の並べ替えが起こることはないという仮定に基づく。

以下に本手法による処理の枠組みを順に述べる。

1. 最初に、出力として希望する翻訳単位の平均単語数 μ、及び最適化に使用する機械翻訳の評価尺度 EV を定める。これらは提案法におけるハイパーパラメータである。EV は文単位の評価が可能であればどのような尺度を用いても良いが、計算機上で高速に評価するためには BLEU のような自動評価尺度が望ましい。

   これらのハイパーパラメータが与えられた下で、$\mathcal{F}$ から実際に選び出す分割位置の個数 K を式 (64) により算出する。

$$K \equiv \max\left(0, \left\lfloor \frac{\sum_{\boldsymbol{f}\in\mathcal{F}} |\boldsymbol{f}|}{\mu} \right\rfloor - N\right) \tag{64}$$

2. $\mathcal{S}$ を $\mathcal{F}$ から選出された文分割位置の集合の候補とする。例えば「最初の文の 3 単語目の後ろで分割」「3 番目の文の 5 単語目の後ろで分割」であれば $\mathcal{S} = \{\langle 1,3\rangle, \langle 3,5\rangle\}$ となる。この表現法に基づき、式 (65) に示すように目

的関数 ω を最大化する $\mathcal{S}$ を $\mathcal{S}^*$ とし、本手法の探索対象とする。

$$\mathcal{S}^* \equiv \underset{\mathcal{S}\in\{\mathcal{S}':|\mathcal{S}'|=K\}}{\arg\max}\ \omega(\mathcal{S};\mathcal{F},\mathcal{E},EV,MT) \tag{65}$$

単純に考えると、EV としてコーパス単位の評価尺度 $EV(\hat{\mathcal{E}},\mathcal{E}^*)$ を使用すれば、ω は EV 自身と等価であると見なすことができる。しかしコーパス単位で何度も EV を計算するのは計算量が大きいため、本研究では ω を式 (66) に示す文単位の評価尺度 $EV(\hat{\boldsymbol{e}},\boldsymbol{e}^*)$ の総和として定義する。

$$\omega(\mathcal{S}) \equiv \sum_{j=1}^{N} EV(MT(\boldsymbol{f}_j,\mathcal{S}),\boldsymbol{e}_j) \tag{66}$$

ここで $MT(\boldsymbol{f},\mathcal{S})$ は図 6 に示すように、$\mathcal{S}$ により生成される $\boldsymbol{f}$ 中の各翻訳単位の翻訳結果 $\{MT(\boldsymbol{f}^{(n)})\}$ を順に結合したものである。

式 (66) は対訳コーパス中の全ての文が独立であるという仮定を導入することと等しく、評価尺度 EV を各文に対して独立に計算することができる。これにより、分割位置が更新された少数の文に対してのみ評価尺度を計算することが可能となるため、EV に関する計算量の大幅な削減に繋がる。EV として BLEU のようなコーパス単位の評価尺度は直接使用できないため、最適化の対象として BLEU を使用したい場合は代わりに BLEU+1 を使用することになる。

3. 式 (66) の最大化により得られた $\mathcal{S}^*$ から特徴量を計算し、分割位置を正例、それ以外を負例とする分類器を学習し、これを文分割モデル $M_{\mathcal{S}^*}$ として実際の入力単語列に対する文分割位置の検出に使用する。

   以降の節で示す貪欲探索に基づく手法では、$M_{\mathcal{S}^*}$ の学習に線型 SVM [6] を使用した。このとき素性として、分割位置の周囲に存在する 5 単語に関する単語 1,2,3-gram、及び品詞 1,2,3-gram を使用した。この設定は従来の SVM を用いた分割位置の学習法に基づく [32]。品詞については事前に Stanford POS Tagger [39] による推定を行い、これを使用した。実際の音声翻訳においては Stanford POS Tagger のような文全体が存在する下での品詞推定は

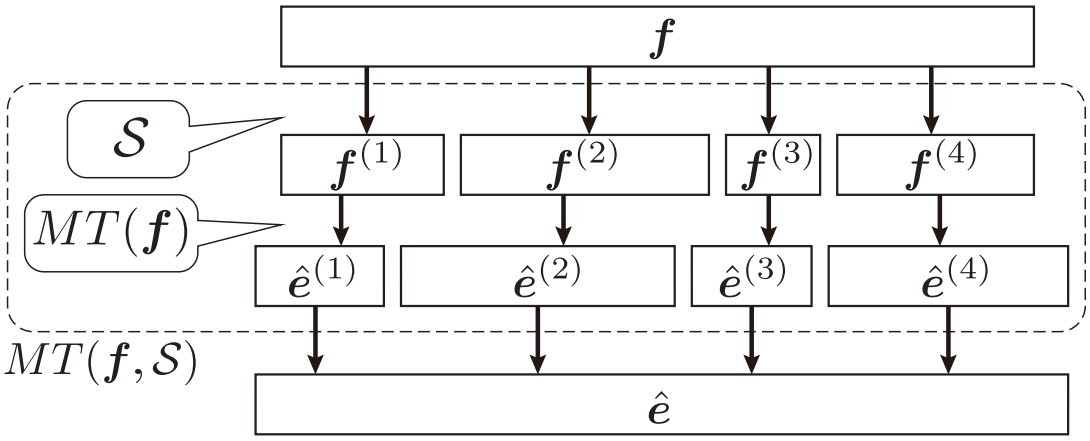

図 6 文分割を適用した場合の翻訳過程 $MT(\boldsymbol{f},\mathcal{S})$

行えないため、音声認識時に単語と品詞を同時に生成するか、音声認識結果の単語列に対して連続的に品詞付与を行う手法が必要となる。

素性による学習器を別に用意する手法は必須ではなく、実際に本手法では分割位置の探索と同時に文分割モデルを学習する手法についても述べる。

以上に示した手順のうち、1. と 3. については些細な問題である。これに対して 2. の $\mathcal{S}^*$ を求める問題は難しく、以降の節における中心的な話題である。式 (65) の厳密解を求めるには可能な $\mathcal{S}$ 全てに対して ω を評価する必要があるが、この計算は NP 困難であり、$\mathcal{F}$ に含まれる単語数の指数に比例する計算量を要するため現実的ではない。これを避けるために、$\mathcal{S}$ と ω の間に何らかの仮定を導入することで、より少ない計算量で近似的に $\mathcal{S}^*$ を求めることとする。次節以降では 2 種類の近似法について述べる。

4.4.1 貪欲探索

最初の近似法は貪欲探索に基づく手法であり、$\mathcal{F}$ 中から分割位置を 1 つずつ順に選択してゆく。今、k 個の既に選択された分割位置 $\mathcal{S}_k$ が存在し、コーパス中でまだ分割されていない位置から $k+1$ 番目の分割位置を選出したい。この選択基

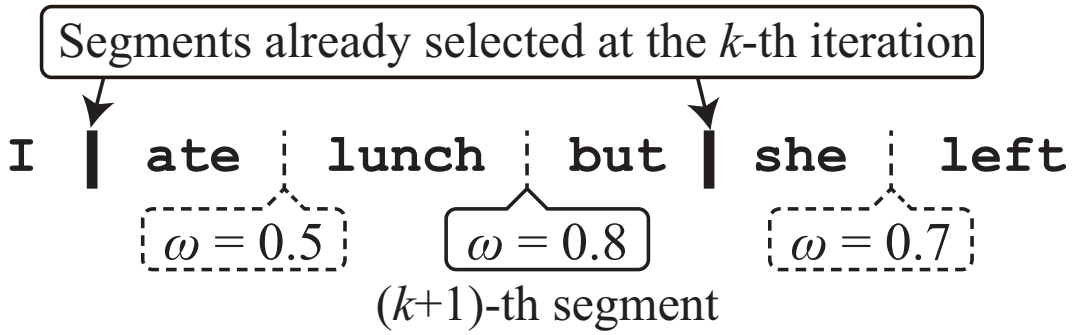

図 7 貪欲探索による分割位置の収集

Algorithm 2 *Greedy* segmentation search

$\mathcal{S}^* \leftarrow \varnothing$

for $k \leftarrow 1 \cdots K$ **do**

$$\mathcal{S}^* \leftarrow \mathcal{S}^* \cup \left\{ \arg\max_{s \notin \mathcal{S}^*} \omega(\mathcal{S}^* \cup \{s\}) \right\}$$

end forreturn $\mathcal{S}^*$

準として、式 (67) に示すように、新たな分割位置 s を $\mathcal{S}_k$ に含めたとき ω を最大化するようなものを選択する。これは前述の文ごとの独立性の仮定により、s を含む単語列に対する EV の変化のみを調べれば良いため、全探索に比べて効率的な探索が可能である。

$$\mathcal{S}_{k+1} \quad = \quad \mathcal{S}_k \cup \left\{ \arg\max_{s \notin \mathcal{S}_k} \omega(\mathcal{S}_k \cup \{s\}) \right\} \tag{67}$$

図 7 に貪欲探索に基づく分割位置の収集法の例を示す。またアルゴリズム 2 に K 個の分割位置を収集するアルゴリズムを示す。

4.4.2 グループ化制約の下での貪欲探索

前節で示した貪欲探索に基づく手法は、学習データに対して高い翻訳精度を実現する分割位置の集合を得ることができる。しかし、翻訳器 MT と評価尺度 EV は複雑であり、ω はこれらの複雑さに由来するノイズを含んでいる。このため、

単純に ω を最大化する貪欲探索では、学習データで偶然評価尺度が高くなるような分割位置を選択してしまい、結果として他のデータに対する文分割モデルの汎化能力を低下させてしまう。

この問題は、学習データから分割位置を選択する際により一貫性のある基準を採用することで回避できると考えられる。2 個目の近似法では、最初に学習データ中で同じ特徴を持つ分割位置をグループ化し、各グループに属する分割位置は必ず同時に選択するという制約を導入することで、単純な貪欲探索に比べてモデルの汎化能力の維持を目指している。図 8 に示すのは、分割位置の素性として左右に隣接する単語の品詞の組を使用し、分割位置に対するグループ化制約を導入した例である。

この制約の導入により、グループに属する分割位置が平均的に評価尺度を高くする場合にのみそのグループが分割位置として選択され、偶然評価尺度を高くするような分割位置の選択を抑制することができると考えられる。また全ての分割位置が素性ごとにグループ化されているため、この制約の下で生成される分割位置は、分割位置として選択するべき素性を情報として持っている。これは分割位置の収集と同時にモデルの学習が行われることを表し、単純な貪欲探索による手法とは異なり、分割位置のモデル化のために追加の学習器は不要である。

分割位置の総数がちょうど K になるような素性の集合を Φ_K とすると、Φ_K を発見する問題は NP 完全であり、Φ_K のうち ω が最大となるものを探索する問題は NP 困難となる。このため、可能な素性集合に対する全探索は避ける必要がある。

アルゴリズム 3 に、本手法で使用するグループ化制約の下での探索アルゴリズムを示す。この手法はアルゴリズム 2 と同様に貪欲探索に基づく手法であり、今までに発見した素性集合に新たな素性を追加してゆくことで目的の素性集合を得るものである。ただし、一度に選択される分割位置は選択する素性によって変化するため、アルゴリズム 2 のように直前の結果のみを記憶しておくのではなく、今までに得られた全ての素性集合を記憶しておく必要がある点が異なる。ここで $c(\phi;\mathcal{F})$ は $\mathcal{F}$ 中に存在する分割位置のうち素性 ϕ を持つものの数であり、$\mathcal{S}(\mathcal{F},\Phi)$ は素性の集合 Φ により $\mathcal{F}$ から抽出された分割位置の集合である。

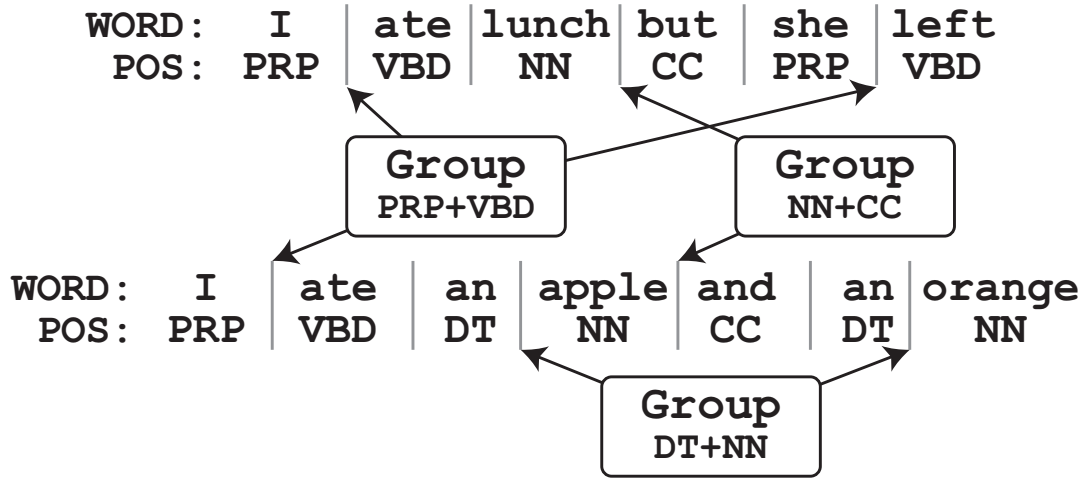

図 8 左右に隣接する単語の品詞による分割位置のグループ化

4.4.3 素性数による正則化

グループ化制約により ω に含まれるノイズは抑制されるが、コーパス中の出現頻度の低い素性に関しては単純な貪欲探索と同様の問題が残っていると考えられる。このためアルゴリズム 3 の目的関数として、式 (68) に示す選択した素性の数による罰則項を加えた評価尺度を考える。

$$\omega_\alpha(\Phi) := \omega(\mathcal{S}(\mathcal{F}, \Phi)) - \alpha|\Phi|. \tag{68}$$

ここで α は罰則の強さであり、大きな正の値を設定するほど素性を選択による ω_α の低下が著しくなり、最終的に選択される素性の数が減少する。$\alpha = 0$ の場合は元の目的関数 ω に一致する。形式的には、式 (68) は Φ に関する重みベクトルの L0 正則化項付き最適化問題に等しい。

4.5 実験

4.5.1 実験設定

本手法により生成される文分割法を適用した場合の翻訳精度を評価する。翻訳対象は英語による TED 講演データ [3] とし、英語からドイツ語、英語から日本語

Algorithm 3 *Greedy+DP* segmentation search

$\Phi_0 \leftarrow \varnothing$

for $k \leftarrow 1 \cdots K$ **do**

  for $j \leftarrow 0 \cdots k-1$ **do**

    $\Phi' \leftarrow \{\phi : c(\phi; \mathcal{F}) = k - j \wedge \phi \not\sqsubset \Phi_j\}$

    $\Phi_{k,j} \leftarrow \Phi_j \cup \left\{ \arg\max_{\phi \in \Phi'} \omega(\mathcal{S}(\mathcal{F}, \Phi_j \cup \{\phi\})) \right\}$

  end for

  $\Phi_k \leftarrow \arg\max_{\Phi \in \{\Phi_{k,j} : 0 \le j < k\}} \omega(\mathcal{S}(\mathcal{F}, \Phi))$

end forreturn $\mathcal{S}(\mathcal{F}, \Phi_K)$

表 1 文分割法の評価に試用したデータ

f*-*e	**Type**	**Dataset**	**#words**	
			f	***e***
En-De	Train *MT*	WIT3, IWSLT2013	21.8M	20.3M
	Train Segmentation	WIT3	424k	390k
	Test	WIT3	27.6k	25.4k
En-Ja	Train *MT*	WIT3, EIJIRO	13.7M	19.7M
	Train Segmentation	WIT3	401k	550k
	Test	WIT3	8.20k	11.9k

の 2 種類の言語対に対して評価を行った。学習データを充実させるために、ドイツ語に関しては IWSLT の評価データ [4] から分野外のデータの抽出 [5] を行って追加した。日本語に関しては英辞郎及び例辞郎のエントリを追加した [2]。表 1 に使用したデータの詳細を述べる。

英語とドイツ語の単語分割と品詞付与には Stanford POS Tagger を使用した。日本語の単語分割には KyTea [25] を使用した。単語アライメントには GIZA++ [28] を使用した。翻訳器 *MT* には句に基づく翻訳法のツールキット Moses [17] を使用し、パラメータを MERT により最適化した。文分割の学習に用いる評価尺

[2] http://eowp.alc.co.jp/info2/

度 *EV* には BLEU+1 を使用した。最終的に生成される翻訳文の評価は BLEU 及び RIBES により行った。

実験の評価対象とした手法を以下に示す。

Greedy 単純な貪欲探索による分割位置の収集を行い、線型 SVM によりモデル化

Greedy+DP 分割位置の素性によるグループ化を導入した貪欲探索によりモデル化。素性は図 8 に示す左右の品詞を使用した。これは同時通訳者が特定の素性の組が現れた場合に分割位置とするという調査結果に基づく [42]。

Punct-Predict 文中の区切り記号の挿入位置の推定による文分割 [32]

RP 単語アライメントによる並べ替えを考慮した文分割 [8]

4.5.2 実験結果

図 9 と図 10 に各評価対象の自動評価尺度による評価結果を示す。横軸は文分割により生成された翻訳単位の平均単語数であり、翻訳が開始されるまでの平均時間に比例する量と考えることができる [32]。

RP、*Greedy* 及び *Greedy+DP* は複数の評価結果が存在するが、これらは分割の頻度を変化させるパラメータを持つ手法であることによる。それぞれの手法について文分割なし（文単位の翻訳）から全て分割（単語単位の翻訳）までパラメータを変化させて評価を行った。

まず、*Greedy* はいずれの手法よりも悪い評価結果となっていることが確認できる。これは既に述べたように、*Greedy* が学習データに対して過学習しやすい手法であるためと考えられる。コレに対して *Greedy+DP* はいずれの従来法よりも BLEU で同等以上、RIBES でより高い評価結果を得ていることが分かる。これにより、*Greedy+DP* で導入した分割位置のグループ化制約が有効に作用していることが確認できる。また *Punct-Predict* と *Greedy+DP* を比較すると、*Greedy+DP* は同様の翻訳精度を 2〜3 倍高速な翻訳開始で実現していることが分かる。

図 11 に示すのは、学習データに対する *Greedy* と *Greedy-DP* による BLEU の変化である。この図から *Greedy* が学習データに対して大きな評価値を得ている

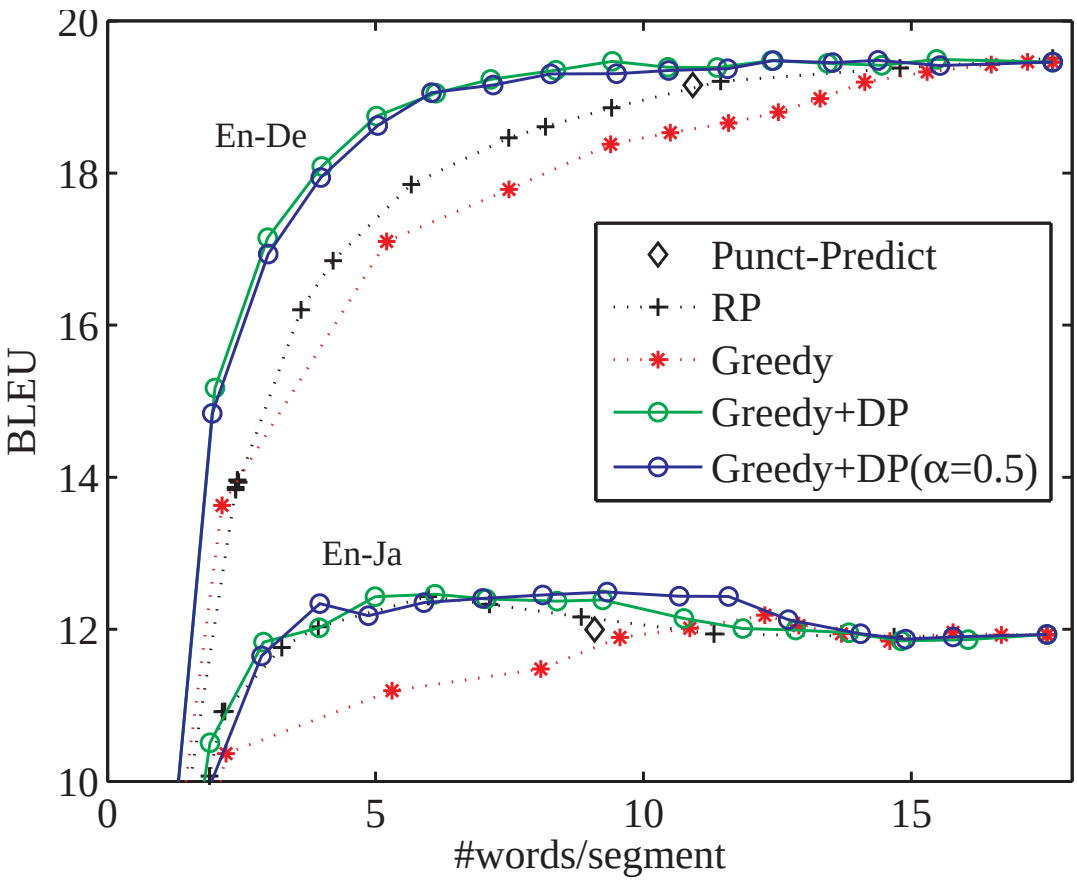

図 9　各文分割法による翻訳結果の BLEU

ことが分かるが、テストデータに対しては同様の効果が得られていないことが分かる。ここから *Greedy* は学習データに強く過学習する手法であることが分かる。一方で *Greedy+DP* は *Greedy* のような特徴は見られず、グループ化制約により過学習を効果的に抑制していることが分かる。

本手法では学習データから分割位置を収集するために、希望する平均単語数 μ を導入した。図 12 に示すのは μ とテストデータで実際に生成された翻訳単位の平均単語数の絶対誤差である。この図から *Greedy+DP* は μ と実際の平均単語数の間の誤差がほぼ 1 単語以下で抑えられていることが分かり、μ に与える値により平均単語数をほぼ制御可能であることが示唆されている。

図 13 に、英語から日本語への翻訳において、式 (68) の α を変化させたときの

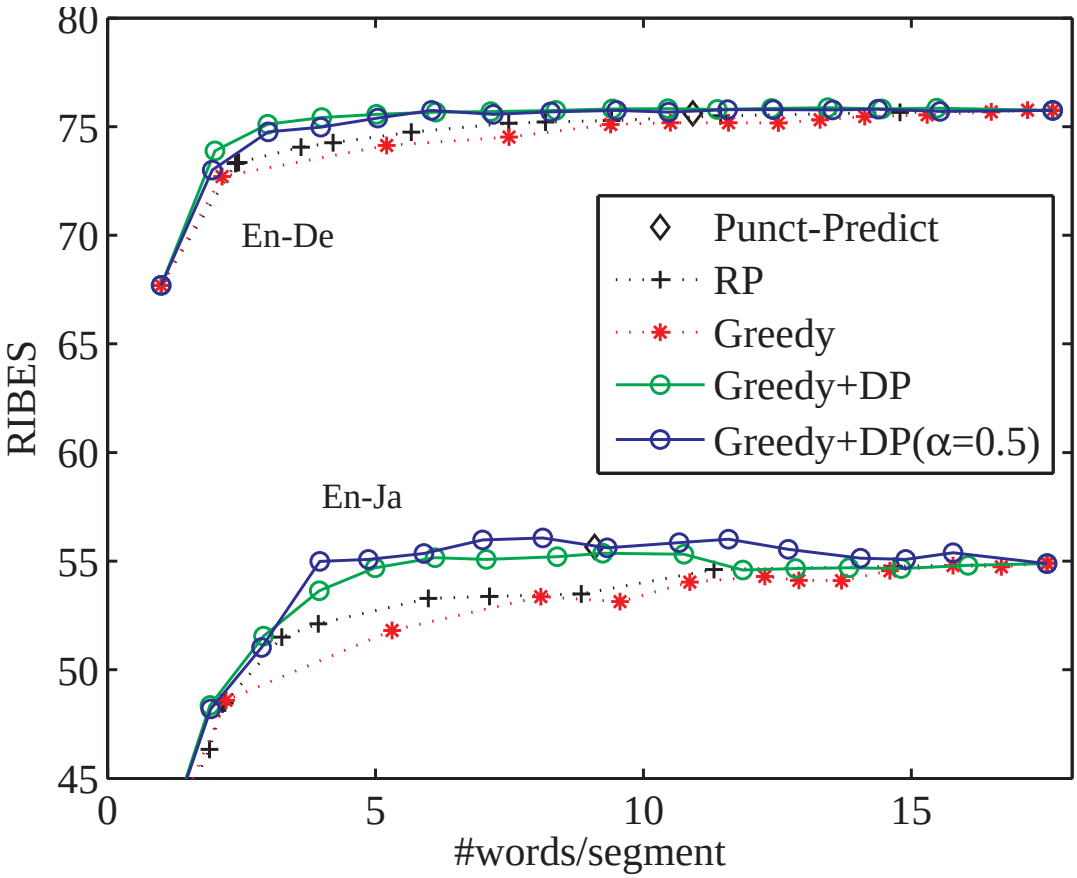

図 10　各文分割法による翻訳結果の RIBES

素性数の変化を示す。ここから、$\alpha = 0$、つまり罰則なしの場合に対して、α を大きくすることで選択される素性の数を抑えることができる。また図 9、図 10 から罰則を与えた場合でも罰則なしと同程度の性能を実現していることから、素性の選択に罰則を与えることで余分な素性が選択されにくくなったものと考えられる。これは品詞の組を用いた本研究ではあまり有効ではないが、より詳細な素性により *Greedy+DP* の探索を行う場合に効果を発揮すると考えられる。

表 2 に、英語から日本語への翻訳において各文分割法により推定された分割位置を示す。

RP として $P_{th} = 0.6$ と 0.7 の 2 種類の場合を示したが、推定された分割位置の数は両者で大きく異なる。ここから、P_{th} に対して文分割の頻度が敏感に変化する

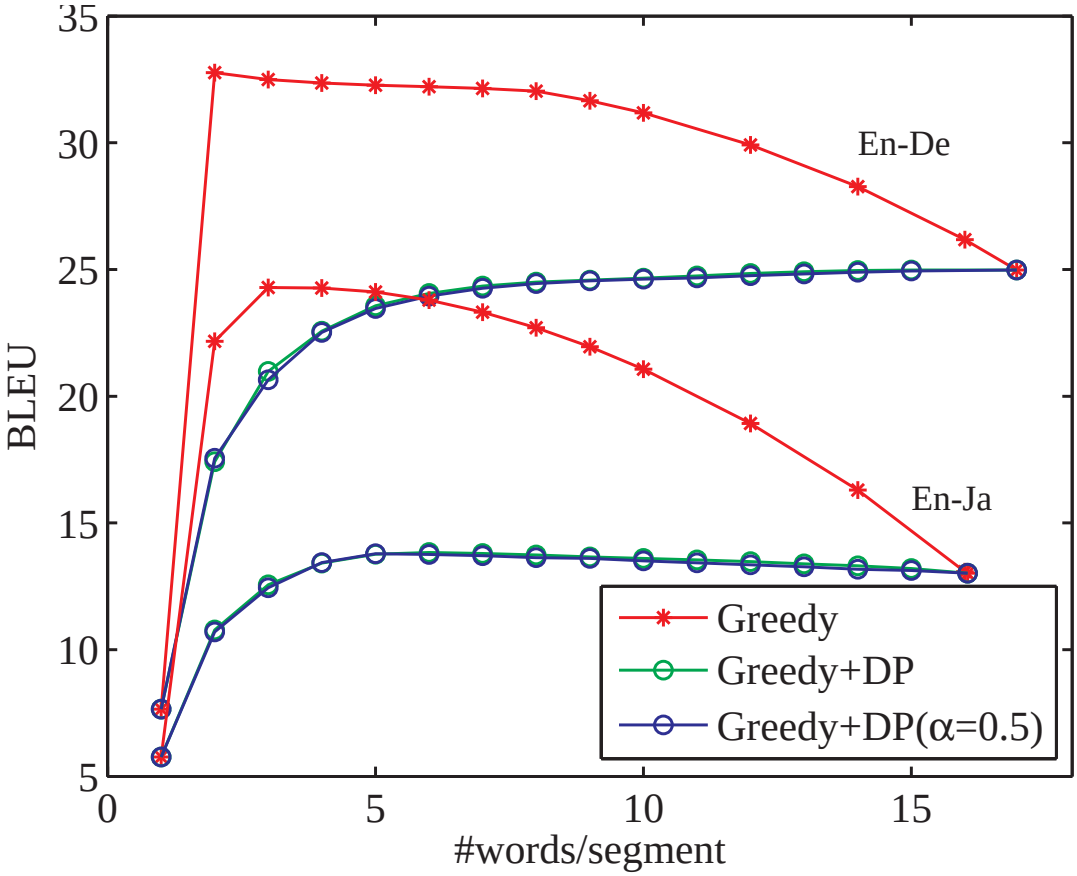

図 11　学習データに対する提案法の BLEU

ことが分かる。また、この文に対する「,」の挿入位置は「minutes」の後ろであると考えられるが、区切り記号の位置を直接推定する *Punct-Predict* と提案法である *Greedy+DP* ではこれを正しく推定できていることが分かる。また *Greedy+DP* では、「you」の後ろのような、文分割を行っても文法的にそれほど問題ないと考えられる位置を分割位置として推定することに成功している。

4.6　本章のまとめ

本章では、既存の文分割法についての解説とその欠点を述べ、この欠点を解決するために翻訳結果の評価尺度に基づき明示的に文分割法を最適化する手法につ

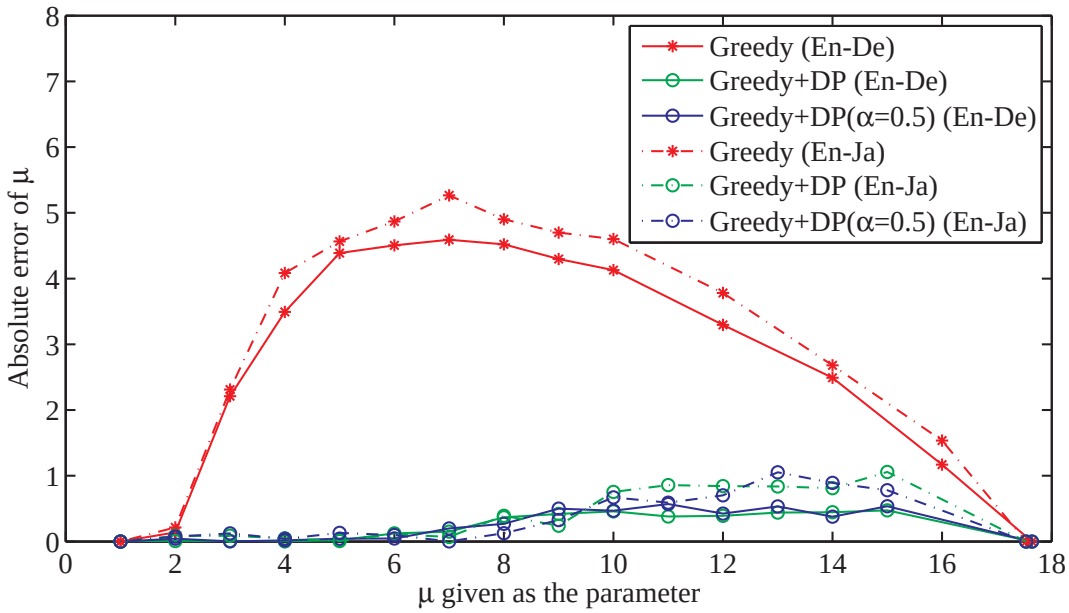

図 12 μ と平均単語数の絶対誤差

いて述べた。実験により、既存の手法よりも評価尺度の面でより高い翻訳精度を達成できることが分かり、また同時に文分割により生成される翻訳単位の平均単語数を制御可能であることを示した。実際の分割位置の比較により、提案法が文法的にそれほど問題のない位置で文分割を行えることを確認した。

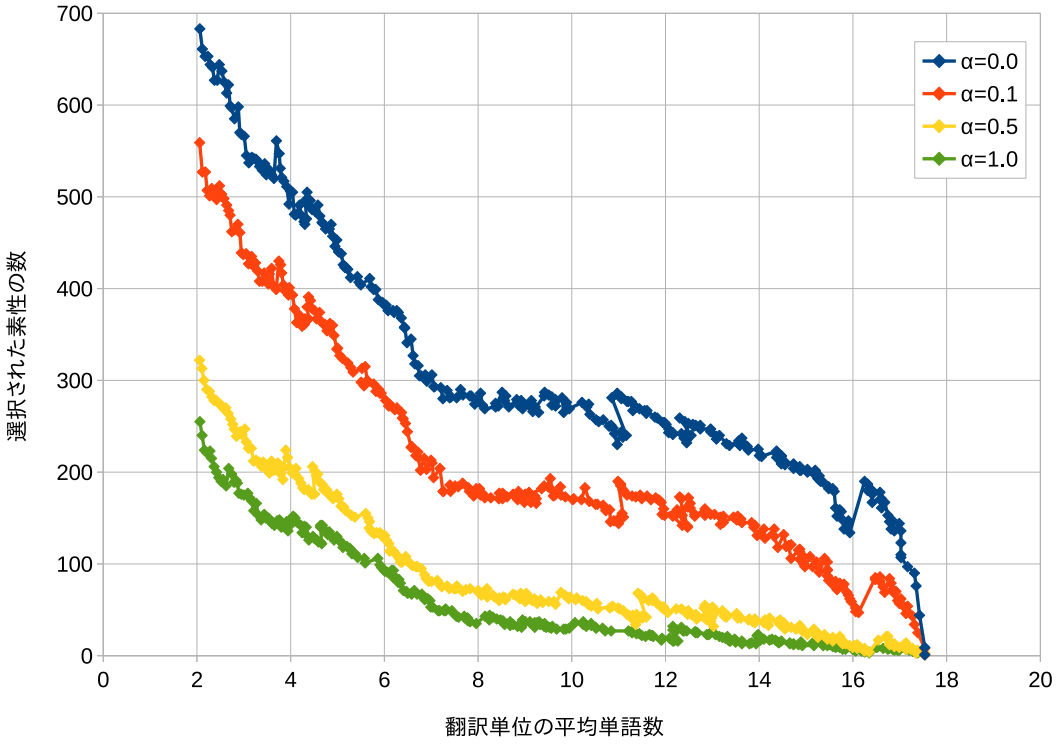

図 13 選択された素性の数の α による変化

Punct-Predict
in the next 18 minutes / I 'm going to take you on a journey
RP ($P_{th} = 0.6$)
in the next 18 / minutes I / 'm going to / take you / on a journey
RP ($P_{th} = 0.7$)
in the next 18 minutes I 'm going to take you on a journey
Greedy ($\mu = 2$)
in / the next 18 minutes I / 'm going to / take / you / on / a journey
Greedy ($\mu = 5$)
in the next 18 minutes I / 'm going to take you on a journey
Greedy+DP ($\alpha = 0.5, \mu = 2$)
in / the next 18 minutes / I 'm going to take you / on / a journey
Greedy+DP ($\alpha = 0.5, \mu = 5$)
in the next 18 minutes / I 'm going to take you / on a journey

表 2 各文分割法により生成された翻訳単位

5. 不完全な文の構文解析に基づく同時音声翻訳

5.1 概要

4 章で述べた文分割法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．この問題への対応として，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 [11]．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 [14] があるが，完全な文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述べる．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．

5.2 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 14(a) で表され，実際の句構造解析器 [44] 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 14(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

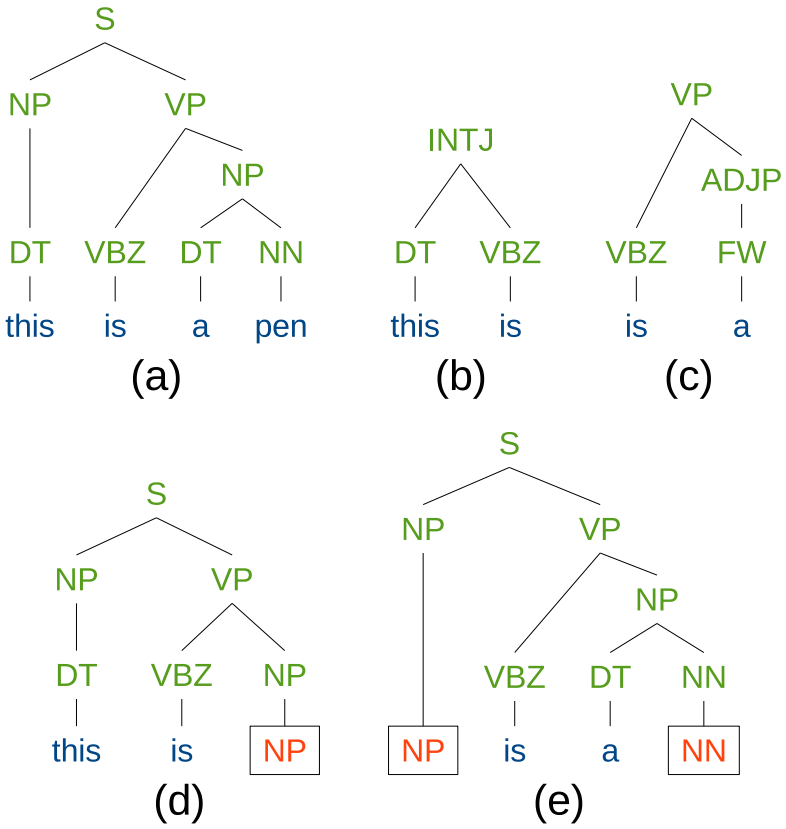

図 14　不完全な文の構文解析

場合，図 14(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 14(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 14(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．

5.2.1　構文解析の定式化

句構造解析の標準的な解析モデルは 3 章で述べた確率的文脈自由文法 (PCFG) である．本手法では PCFG に基づき，文法要素を追加した文の構文解析について

考える．まず，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 (69)，(70) による文法要素の推定，式 (71) 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．

$$\boldsymbol{L}^* \equiv \arg\max_{\boldsymbol{L}} \Pr(\boldsymbol{L}|\boldsymbol{w}) \tag{69}$$

$$\boldsymbol{R}^* \equiv \arg\max_{\boldsymbol{R}} \Pr(\boldsymbol{R}|\boldsymbol{w}) \tag{70}$$

$$T^* \equiv \arg\max_{T} \Pr(T|\boldsymbol{L}^*, \boldsymbol{w}, \boldsymbol{R}^*) \tag{71}$$

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 (72) で定義する．

$$\Pr(X \to \boxed{Y}) \equiv \begin{cases} 1, & \text{if } Y = X \\ 0, & \text{otherwise} \end{cases} \tag{72}$$

$\boldsymbol{L}$ は過去に得られた構文木の一部と考えられるため，前回までの入力を履歴として記憶しておくことで生成可能であると考えられる．しかし履歴に含まれる構文情報のどの部分を $\boldsymbol{L}$ とするかは自明ではなく，また $\boldsymbol{L}$ と $\boldsymbol{R}$ で異なる推定アルゴリズムを使用しなければならなくなる．本論文では問題の簡単化のため，以降の節で示す同一のアルゴリズムを用いて $\boldsymbol{L}$ と $\boldsymbol{R}$ を推定する。

5.2.2 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．

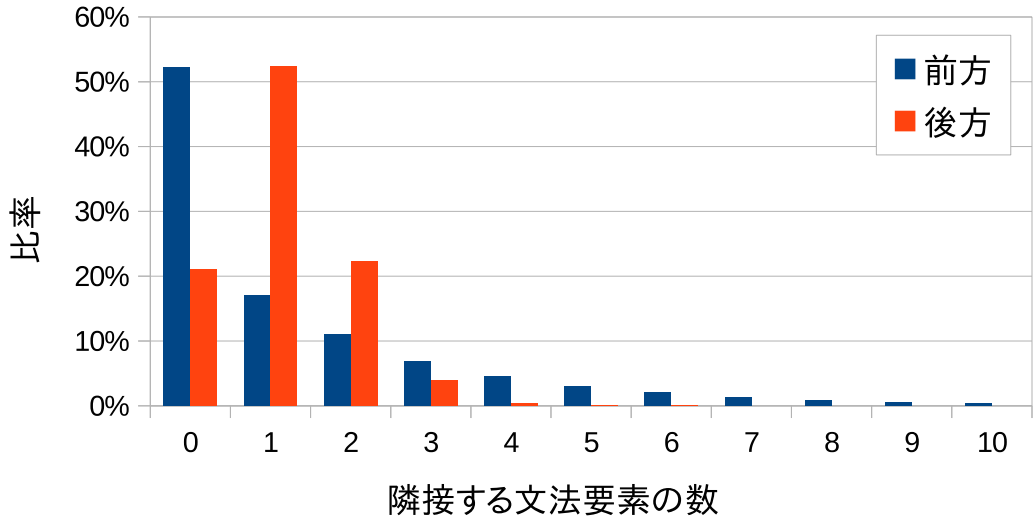

図 15　前方・後方に追加されるべき文法要素の数

3. 構文木のノード数が最小である．

このようにすると，図 14 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 15 に示すのは，Penn Treebank [20] 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 4 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 4 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．

Algorithm 4 後方の文法要素の推定

$T' \leftarrow \arg\max_{T} \Pr(T|\boldsymbol{w})$ ▷ 事前構文解析
$\boldsymbol{R}^* \leftarrow [\,]$
loop
  $R^+ \leftarrow \arg\max_{R} \Pr(R|T', \boldsymbol{R}^*)$ ▷ 次の文法要素
  if $R^+ = \mathrm{nil}$ **then**
    return $\boldsymbol{R}^*$ ▷ 推定の終了
  end if
  $\boldsymbol{R}^* \leftarrow \boldsymbol{R}^* +\!\!+ [R^+]$ ▷ 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 [6] による多クラス分類器を使用し，表 3 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 NN」という例に対して，後方 3 単語は「is a NN」を指し，後端の単語は「NN」を指すことになる．なおアルゴリズム 4 及び表 3 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．

5.3 文法要素を含む構文木の機械翻訳

式 (71) 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．[3]

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

[3] 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 3 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

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 NP」「a pen」はそれぞれ翻訳結果が「これはNPです」「ペン」となり,「ペン」は「です」より前，NPの位置に挿入するべきであることが予想できる．

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 5 に示す.

5.4 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．

Algorithm 5 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

5.4.1 実験設定

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[4]，句構造解析には PCFG-LA[35] を主な解析手法とする Ckylark[44] を使用した．

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 [3] ，及び英辞郎，例辞郎のエントリ [5] である．日本語の単語分割に KyTea [25] 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

[4] http://nlp.stanford.edu/software/tokenizer.shtml
[5] http://eijiro.jp/

ライメントにはGIZA++ [28] を使用し，日本語の言語モデルはKenLM [13] による5-gramモデルを使用した．Tree-To-String翻訳器にはTravatar [24] を使用し，パラメータをMERTにより最適化した．また，ベースラインとして句に基づく翻訳器をMoses [17] により構築した．入力単語列を生成するための文分割法としては、一定単語数による強制的な分割と4章で示した貪欲探索に基づく文分割法の2種類を使用した。最終的な翻訳精度はBLEU [29] 及びRIBES [15] によって比較した．

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

5.4.2 実験結果

隣接する文法要素の生成　表4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．このような事例は学習データ中の一貫性が弱いため，結果として推定器は「DT pen」のようにより少ない構文要素を推定する傾向となる．また，このような重要でない構文要素は英語では単語列の

表 4 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

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．

機械翻訳への適用　図 16〜19 に各手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 [14] 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 *PBMT* と *T2S* の BLEU の大小関係が逆転し、RIBES も小さな平均単語数では同等程度まで低下してしまう．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．

一定単語数による文分割を用いた場合、*T2S-Tag* では短い単語数でも BLEU において *PBMT* と同等の翻訳精度を維持していることが分かる．これは本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，翻訳待機の手法が有効であることが確認できる．

一方、貪欲探索に基づく文分割を用いた場合は *T2S-Tag* 及び *T2S-Wait* は BLEU において *T2S* と近い傾向を示している。これは文分割法が文の特徴に関して一貫

性を保つような分割位置を選択するため、文法要素の推定による効果が一定単語数による文分割に比べて小さいためと考えられる。しかしながら、*T2S-Wait* はRIBES において他の手法よりも良い翻訳精度を達成しており、翻訳待機の手法の語順に対する頑健性を示していると考えられる。

5.5 本章のまとめ

本章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．

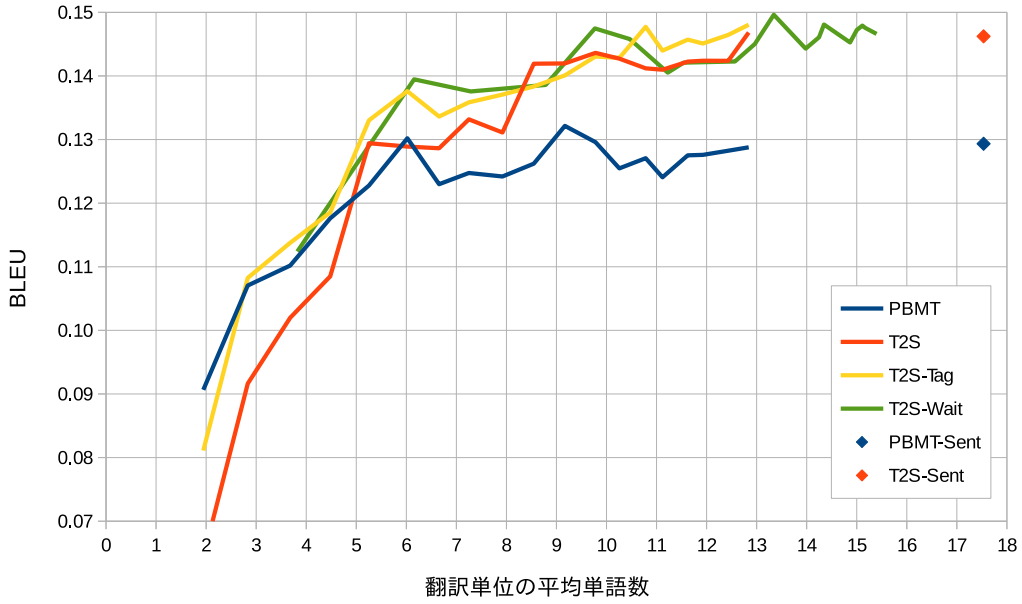

図 16　一定単語数による文分割を用いた場合の平均単語数と BLEU

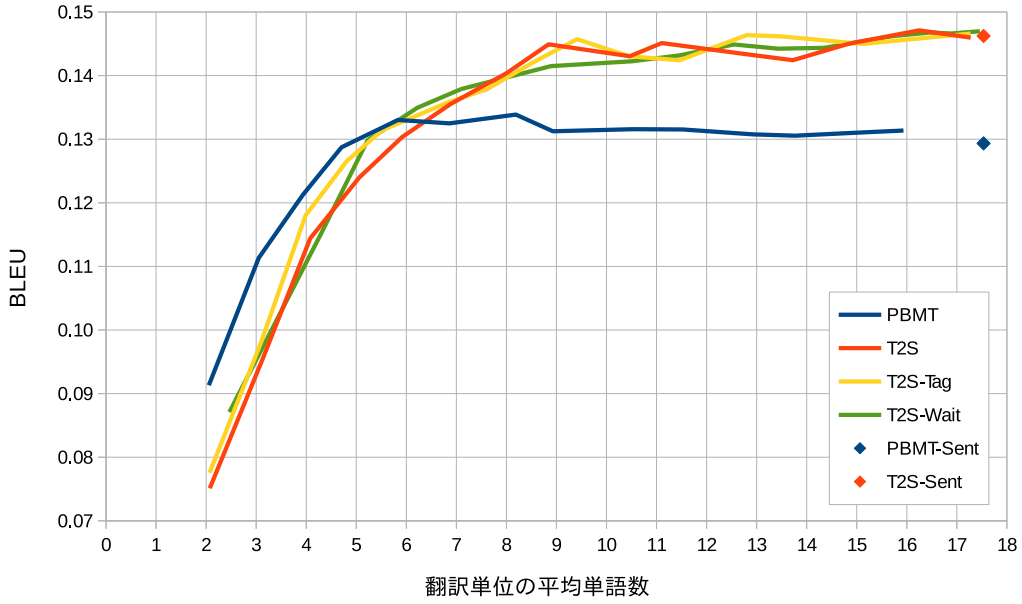

図 17　貪欲探索に基づく文分割を用いた場合の平均単語数と BLEU

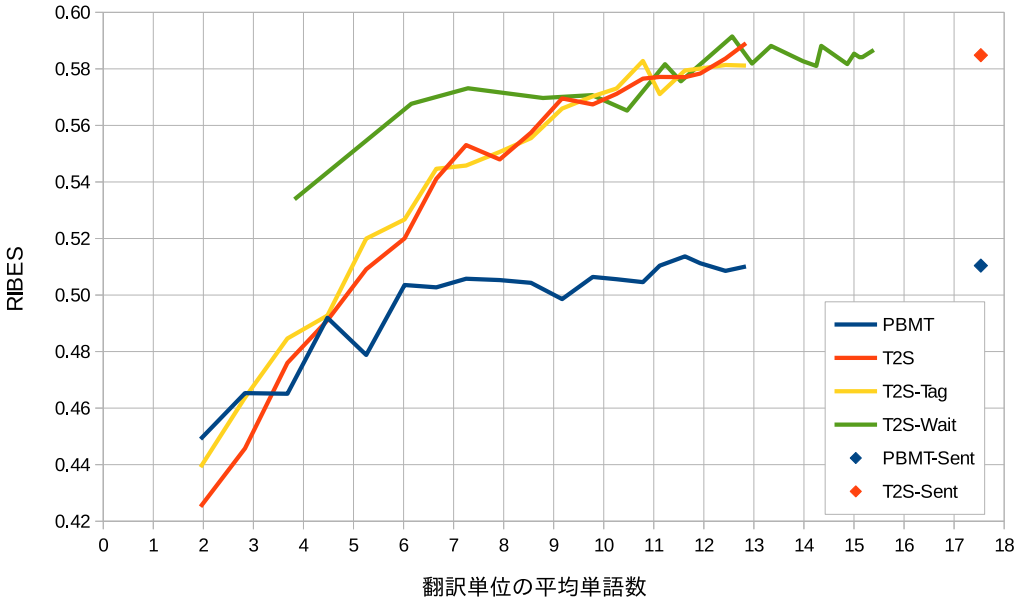

図 18　一定単語数による文分割を用いた場合の平均単語数と RIBES

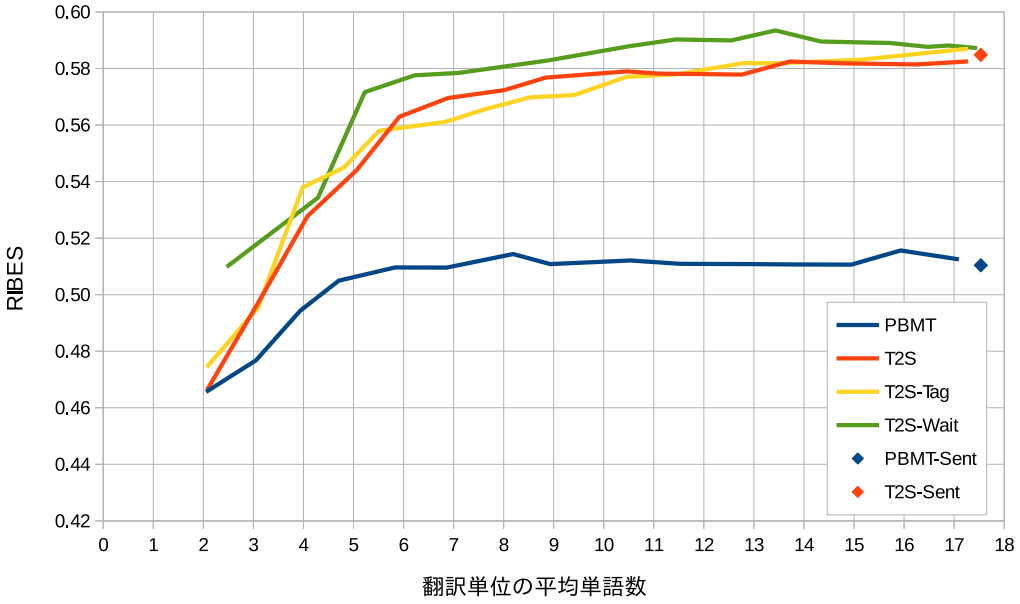

図 19　貪欲探索に基づく文分割を用いた場合の平均単語数と RIBES

6. 結言

6.1 結論

本論文では，同時音声翻訳システムの翻訳精度と訳出速度の向上を目的とし，特に機械翻訳の前処理，後処理に関する 2 種類の改善手法を述べた．

2 章 では，従来の音声翻訳と同時音声翻訳の違いを説明し，同時音声翻訳を実現するための要素技術である文分割法を導入し，システム全体の定式化を行った．

3 章 では同時音声翻訳に関する諸技術について，本研究が特に対象とする機械翻訳を中心に解説を行った．

4 章 では従来の文分割法が翻訳精度を考慮していない点を問題として挙げ，これを解決するための新たな文分割法の学習アルゴリズムを提案した．提案法は翻訳結果の評価尺度の最大化を基準として学習するアルゴリズムであり，これにより翻訳単位の平均単語数を制御しながら，最も高い翻訳精度を実現する文分割戦略を採用できるようになった．具体的には，実験により既存の文分割法と同程度の翻訳精度を，本手法では 2 倍以上の訳出速度で実現可能であることを示した．

5 章 では文分割により失われる構文情報により翻訳精度に影響を及ぼす可能性を指摘し，これを解決するために不完全な単語列に対して前後に続く構文要素を推定する手法を提案した．これにより構文解析が正しく行えるようになり，高精度な翻訳手法である Tree-To-String 翻訳を同時音声翻訳においても実行可能となった．また Tree-To-String 翻訳の翻訳結果を再評価することにより構文的に問題のある分割位置を無視し，次の入力を待機してより良い翻訳を実現する手法についても述べ，実際に翻訳精度が向上することを示した．入力の待機は文分割結果を訂正する効果があるとも言え，文分割法と待機に関する手法を組み合わせることでより効果的な同時音声翻訳を実現できると考えられる．

6.2 今後の課題

本論文では英語から日本語，また英語からドイツ語への翻訳を中心的に扱った．今後はより多くの言語対に対して本論文で提案した手法の検証を行い，優位性，

または問題点を発見する必要がある．

本論文の手法は既存の学習データに依存したものであり，実際の同時通訳者がどのような翻訳戦略を用いているのかは明示的に考慮しているわけではない．このため，本研究の手法による翻訳結果が主観評価に対してどのような影響があるのかは言及できず，今後の調査が必要である．またこれらの研究と並行して，実際の同時通訳者による翻訳事例の収集を実施しており，これらのデータの調査や応用についても今後考慮してゆく必要がある．同時通訳を始めとした音声翻訳のためのデータは現状では少量であり，単独で翻訳器を学習するのに十分な情報量とはならない．このため既に大量に存在する文書翻訳用のコーパスを一緒に用いて翻訳器を学習することとなるが，音声と文書の間の文の特徴の相違が翻訳精度を低下させる可能性がある．この問題の解決に考えられる手法としては，音声翻訳へ特化した分野適応 [12] や，ピボット翻訳と呼ばれる中間言語を介する翻訳手法 [40] の応用などが考えられる．

本論文の手法では既に翻訳された翻訳単位については考慮せず，現在得られている翻訳単位，及び未来に得られると予想される構文に対して重点を置いたものである．未来の情報を予測することに対し，これらの過去に得られた情報は確定しているため，同時音声翻訳の実現の上で強力な事前情報となることが考えられる [32]．今後はこれらの情報についても明示的に考慮するような手法を開発してゆく必要がある．

また，本研究では音声認識，機械翻訳，テキスト音声合成の各モジュールが独立して動作するという仮定の下で各手法の提案を行ったが，実際にはモジュール間のデータ受け渡し時に情報が欠落してしまうため，複数のモジュールを一つの単位として最適化することが望ましいと考えられる．このような手法としてはランク学習を用いた音声認識と機械翻訳の同時最適化 [43] などが提案されている．他にも，機械翻訳による出力は本来の自然言語による文に対して独特の特徴を持っており，これが音声合成時の問題となることが知られている [38]．これらの特徴を考慮することで，より流暢な合成音声を生成する手法なども研究されている [30]．これらの手法は同時音声翻訳の特定の部位についての最適化手法であり，全てのモジュールに跨がるような最適化手法も今後考慮してゆく必要がある．

更には，音響的な情報を機械翻訳の過程で欠落させないような翻訳手法についても研究する必要がある．言い淀みや無言，イントネーションやアクセントなど，発話に含まれる音響的な情報はコミュニケーション上重要な役割を果たしていると考えられるが，現状の機械翻訳ではテキスト化された言語情報のみを扱うため，これらの情報は翻訳結果から欠落してしまう．この問題の解決には，音声認識結果に音響情報を含ませ，直接テキスト音声合成に伝えるか，対応する目的言語の音響情報を翻訳結果の文と同時に生成するような機械翻訳手法を考案する必要がある．

以上に述べた要素などを考慮し，今後も同時音声翻訳システムの性能向上を目指す．

謝辞

本研究を遂行するにあたり，多くの方々から支援を頂きました．

本学情報科学研究科の中村哲教授には，主指導教官として多くの貴重な助言と指導，学会参加を始めとする研究活動の支援を頂きました．本研究は中村教授の指導の下で結実したものであり，他所での実現はなかったものと考えています．心より感謝を申し上げます．

本学情報科学研究科の松本裕治教授には，副指導教官として多方面からの助言を頂きました．心より感謝を申し上げます．

本学情報科学研究科の戸田智基准教授には，副指導教官として研究や発表の方向性，また数学面の詳細など多くの助言を頂きました．心より感謝を申し上げます．

本学情報科学研究科の Graham Neubig 助教には，副指導教官として主な研究の方向性やデータの整備，アルゴリズムに関する助言など，多くの助言，支援を頂きました．また本論文以外の研究活動についても，主な指導者として多くの助言や方向性の提示を行って頂きました．心より感謝を申し上げます．

本学情報科学研究科の Sakriani Sakti 助教には，福指導教官として多くの助言，特に本研究の周辺技術である音声認識の面で多くの助力を頂きました．心より感謝を申し上げます．

本学知能コミュニケーション研究室，及び自然言語処理学研究室の学生，スタッフ諸氏には，研究活動にあたっての多くの支援を頂きました．特に秘書である松田真奈美様にはスケジュール管理や資金の調整など，活動に重要な多くの支援を頂きました．心より感謝を申し上げます．

独立行政法人日本学生支援機構には資金面で多くのご助力を頂きました．感謝を申し上げます．

最後に，研究活動と生活全般を通して陰ながらご助力を頂いた父と母，多くの友人に心から感謝を申し上げます．

参考文献

[1] Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. Real-time incremental speech-to-speech translation of dialogs. In *Proc. NAACL*, 2012.

[2] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19, 1993.

[3] Mauro Cettolo, Christian Girardi, and Marcello Federico. Wit^3: Web inventory of transcribed and translated talks. In *Proc. EAMT*, 2012.

[4] Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 10th iwslt evaluation campaign. In *Proc. IWSLT*, 2013.

[5] Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. Adaptation data selection using neural language models: Experiments in machine translation. In *Proc. ACL*, 2013.

[6] 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*, 9, 2008.

[7] Christian Fügen, Alex Waibel, and Muntsin Kolss. Simultaneous translation of lectures and speeches. *Machine Translation*, 21, 2007.

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

[9] Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In *Proc. NAACL*, 2004.

[10] Isao Goto, Ka Po Chow, Bin Lu, Eiichiro Sumita, and Benjamin K Tsou. Overview of the patent machine translation task at the NTCIR-10 workshop. In *Proc. NTCIR-10*, 2013.

[11] 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*, 2014.

[12] Xiaodong He and Li Deng. Robust speech translation by domain adaptation. In *Proc. Interspeech*, 2011.

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

[14] Liang Huang, Kevin Knight, and Aravind Joshi. Statistical syntax-directed translation with extended domain of locality. In *Proc. AMTA*, 2006.

[15] Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sdoh, and Hajime Tsukada. Automatic evaluation of translation quality for distant language pairs. In *Proc. EMNLP*, 2010.

[16] Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proc. ICASSP*, 1995.

[17] 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.

[18] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proc. NAACL*, 2003.

[19] Chin-Yew Lin and Franz Josef Och. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *Proc. COLING*, 2004.

[20] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The Penn treebank. *Computational Linguistics*, 19, 1993.

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

[22] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proc. Interspeech*, 2010.

[23] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proc. AIStatistics*, 2005.

[24] Graham Neubig. Travatar: A forest-to-string machine translation engine based on tree transducers. In *Proc. ACL*, 2013.

[25] Graham Neubig, Yosuke Nakata, and Shinsuke Mori. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proc. ACL*, 2011.

[26] Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori, and Tatsuya Kawahara. An unsupervised model for joint phrase alignment and extraction. In *Proc. ACL*, 2011.

[27] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proc. ACL*, 2003.

[28] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29, 2003.

[29] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proc. ACL*, 2002.

[30] Alok Parlikar, Alan W Black, and Stephan Vogel. Improving speech synthesis of machine translation output. In *Proc. Interspeech*, 2010.

[31] Lawrence R Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. PTR Prentice Hall Englewood Cliffs, 1993.

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

[33] Jason Riesa, Ann Irvine, and Daniel Marcu. Feature-rich language-independent syntax-based alignment for statistical machine translation. In *Proc. EMNLP*, 2011.

[34] Yoshinori Sagisaka. Speech synthesis by rule using an optimal selection of non-uniform synthesis units. In *Proc. ICASSP*, 1988.

[35] Romain Thibaux Slav Petrov, Leon Barrett and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proc. COLING*, 2006.

[36] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Proc. Interspeech*, 2012.

[37] Keiichi Tokuda, Yoshihiko Nankaku, Tomoki Toda, Heiga Zen, Junichi Yamagishi, and Keiichiro Oura. Speech synthesis based on hidden markov models. *Proc. IEEE*, 101, 2013.

[38] Laura Mayfield Tomokiyo, Kay Peterson, Alan W Black, and Kevin A Lenzo. Intelligibility of machine translation output in speech synthesis. In *Proc. Interspeech*, 2006.

[39] Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. NAACL*, 2003.

[40] Masao Utiyama and Hitoshi Isahara. A comparison of pivot methods for phrase-based statistical machine translation. In *NAACL-HLT*, 2007.

[41] 奥村 学, 渡辺 太郎, 今村 賢治, 賀沢 秀人, Graham Neubig, and 中澤 敏明. **機械翻訳（自然言語処理シリーズ）**. コロナ社, 2014.

[42] 清水 宏晃, Graham Neubig, Sakriani Sakti, 戸田 智基, and 中村 哲. 同時通訳データを利用した同時音声翻訳のための訳出タイミング決定手法. In **言語処理学会第 *20* 回年次大会**, 2014.

[43] 大串 正矢, Graham Neubig, Sakriani Sakti, 戸田 智基, and 中村 哲. 音声認識と機械翻訳のランク学習による同時最適化. In **言語処理学会第 *19* 回年次大会**, 2013.

[44] 小田 悠介, Graham Neubig, 波多腰 優斗, Sakriani Sakti, 戸田 智基, and 中村 哲. 解析失敗の発生しにくい PCFG-LA 句構造構文解析. In **言語処理学会第 *21* 回年次大会**, 2015.

業績一覧

国際会議［査読付］

1. Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Optimizing Segmentation Strategies for Simultaneous Speech Translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)* (Short Papers), Baltimore, USA, June, 2014.

2. Graham Neubig, Katsuhito Sudoh, Yusuke Oda, Kevin Duh, Hajime Tsukada, and Masaaki Nagata. The NAIST-NTT Ted Talk Treebank. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, Lake Tahoe, USA, December, 2014.

3. Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Ckylark: A More Robust PCFG-LA Parser. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)* (Demonstrations), Denver, USA, May, 2015. （投稿中）

4. Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Syntax-based Simultaneous Translation through Prediction of Unseen Syntactic Constituents. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)* (Long Papers), Beijing, China, July, 2015. （投稿中）

全国大会（国内）［査読付］

1. 小田 悠介, 笹倉 隆史, 角森 唯子, 赤部 晃一, 大島 悠司, 鶴田 さくら, 西垣 友理, 田代 光輝, and 中村 哲. オンラインコミュニティの盛衰と挨拶表現

の関連性の調査. 情報社会学会 2014 年度研究発表大会, 神奈川, 5 月, 2014. [最優秀プレゼンテーション賞]

全国大会（国内）[査読無]

1. 小田 悠介, Graham Neubig, 清水 宏晃, Sakriani Sakti, 戸田 智基, and 中村 哲. 翻訳精度の最大化による同時音声翻訳のための文分割法. 言語処理学会第 20 回年次大会, 北海道, 3 月, 2014.

2. 小田 悠介, Graham Neubig, 波多腰 優斗, Sakriani Sakti, 戸田 智基, and 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, 京都, 3 月, 2015. (採録決定)

3. 小田 悠介, Graham Neubig, Sakriani Sakti, 戸田 智基, and 中村 哲. 不完全な文の構文解析に基づく同時音声翻訳. 言語処理学会第 21 回年次大会, 京都, 3 月, 2015. (採録決定)

研究会（国内）[査読付]

1. 小田 悠介 and Graham Neubig. 統計的機械翻訳を用いた自動コメント生成. 情報処理学会ソフトウェア工学研究会ウィンターワークショップ 2015, 沖縄, 1 月, 2015.

研究会（国内）[査読無]

1. 小田 悠介, Graham Neubig, Sakriani Sakti, 戸田 智基, and 中村 哲. 回答選択型 QA システムにおける教師なし意味解析の利用. ALAGIN & NLP 若手の会合同シンポジウム, 東京, 9 月, 2013.

2. 小田 悠介, ニュービッグ グラム, サクティ サクリアニ, 戸田 智基, and 中村 哲. 自動プログラミングへ向けた問題解答コーパスの収集と考察. 情報処理学会第 216 回自然言語処理研究会, 東京, 5 月, 2014.

3. 冨田 隼平, 水上 雅博, 小田 悠介, and 山本 誠一. 人による翻訳文への翻訳自動評価法の適用に関する考察. NLP 若手の会第 9 回シンポジウム, 神奈川, 9 月, 2014.

4. 小田 悠介, 札場 寛之, ニュービッグ グラム, サクティ サクリアニ, 戸田 智基, and 中村 哲. ソースコード構文木からの統計的自動コメント生成. 情報処理学会第 219 回自然言語処理研究会, 東京, 12 月, 2014.

解説記事

1. 小田 悠介. ACL2014 参加報告 (その 1). 言語処理学会ニュースレター Vol. 21 No. 4, 10 月, 2014.

講演

1. 小田 悠介. ACL2014 参加報告 (1). 情報処理学会第 217 回自然言語処理研究会, 北海道, 7 月, 2014.

2. 小田 悠介. ACL 参加報告 (1). NLP 若手の会第 9 回シンポジウム, 神奈川, 9 月, 2014.

特許（国内）

1. 申請中 1 件

ソフトウェア

1. **Ckylark** (句構造解析器)
   http://odaemon.com/?page=tools_ckylark

データセット

1. **NAIST-NTT Ted Talk Treebank**
   http://ahclab.naist.jp/resource/tedtreebank/