話し言葉の依存構造解析における音声認識誤りの影響の評価

吉川 将司　　進藤 裕之　　松本 裕治
奈良先端科学技術大学院大学 情報科学研究科
{masashi.yoshikawa.yh8, shindo, matsu}@.is.naist.jp

概要

対話システムなどにおける自然言語理解のためには、文に対して構文構造の解析を行うことが重要である。しかし、話し言葉には言い淀みや自動音声認識による認識誤りなどの特徴があり、このような文を構文解析することは困難であると思われる。実際に既存研究では、状態の遷移に基づく係り受け解析において言い淀みに対して特別な処置を取ることで高い構文解析の精度を示している。しかしながら、これらの研究では人手によって書き写された文に対して実験を行っており、実際の利用場面で発生する自動音声認識による誤りは考慮に入れられていない。本研究ではそのような自動音声認識による誤りを含む文に対しても頑健に構文解析をすることを目指す。本論文では、そのような誤りを含むコーパスを構築し、それに対して既存の手法を用いて構文解析の実験を行った。その結果、構文解析の精度は音声認識の誤りによって大きく悪影響を受け、それに対して処置を行う必要があることが明らかとなった。

1　はじめに

対話システムなどにおいては、誰がいつどうしたなどの述語や項の間の関係を把握する自然言語理解 (Natural Language Understanding) の技術が重要である。このようなシステムでは人間の話し言葉を処理しなければならないが、話し言葉は自然言語処理において中心的に研究されている書き言葉と比較して以下のような特徴を持っている。1) 言い直し、繰り返し、談話標識、間投詞などの言い淀みを含んでいる。2) 自動音声認識の認識の誤りによるノイズを含む。

自然言語理解のためには、係り受け解析を行うことが一般的であり、その係り受け解析においても上に挙げたような話し言葉の特徴は精度の低下につながることが予測される。

実際に 1) の言い淀みについては構文解析の精度に悪影響を与えることが知られており、Honnibal ら [2]、Wu ら [4]、Rasooli ら [3] の研究では、従来の状態の遷移に基づく係り受け解析の手法に、新たに言い淀みに対処するためのアクションを追加することでこの問題に対処している。この言い淀みを対処するための新しいアクションにより、話し言葉においても高い性能で係り受け解析を行うことができている。

しかしながら、これらの研究では音声会話コーパスの人手によって書き写されたデータで実験が行われており、実際の応用の場面においてはそのような言い淀みに加えて、2) の自動音声認識の失敗によるノイズも発生することで、そこで示されたような理想通りの構文解析をすることができないだろうと考えられる。

例えば、実際に音声認識を行った場合は、音声認識モデルの語彙に出現しない固有表現などはそれと音的に似た語、もしくは語列に置き換わってしまうが、このような状況では構文解析に悪影響を起こしてしまうであろう。

本研究は、そのような自動音声認識による誤りを含むような文に対しても頑健に構文解析をすることを目指すものである。ここでは、自動音声認識の誤りに頑健であるとは、1) 構文解析がそのような誤りによって失敗しない。2) そのような誤りを検知し処理を行う。ようなことを言う。

この論文では、Switchboard コーパスの音声データに自動音声認識を適用し、その結果に対して人手による書き出し文に付与されている構文木の情報を対応付け、自動音声認識による誤りを含むようなデータを作成した。そのデータに対し既存の手法で構文解析を行い、音声認識の誤りによるノイズが構文解析にどのような影響を与えるかを調査した。

本論文の構成は以下のとおりである。2 章で実験のために自動音声認識を用いて作成したデータについて紹介し、3 章で用いた解析手法について述べ、4 章でそれを用いた係り受け解析の実験について述べる。5 章でまとめと今後の課題について述べる。

2　データの作成

本論文では Switchboard コーパス [1] を用いて、自動音声認識システムの認識誤りを含むような係り受け解析の実験用データを作成した。

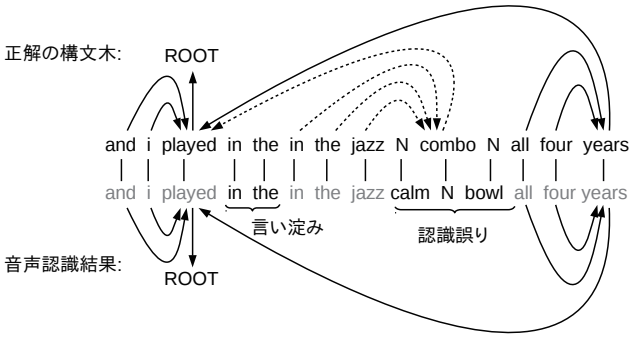

図 1　作成したデータの例。N はアライメントの相手がいないということを表す。combo という語が自動音声認識結果では消えてしまったため、作成したデータではこれの子孫にあたる語の係り受け関係を除いた。(点線)

2.1　Switchboard コーパス

Switchboard コーパスは、与えられたテーマについての二人の人間による数分間の会話 2438 件の音声データとそれを人手によって書き出したテキストから構成されている。2438 件のうち 1129 件には品詞の情報、1126 件には言い淀みについて情報、またそのうち 642 件には句構造木の情報がアノテートされている。

2.2　自動音声認識

以下では、実験に用いる自動音声認識の誤りを含むコーパスデータの作成について説明する。

上述の通り Switchboard コーパスにおいて構文木の情報が付与されているのは全体の一部であり、実際に構文解析の実験に利用するのもその部分である。よって、Switchboard コーパスの残りのすべて部分は音声認識のための学習データとすることができる。

具体的には、Switchboard コーパスの会話データ約 2438 件のうち、構文木の情報が付与されていない 1796 件の会話はすべて音声認識のモデル学習に用いる。

学習データの量の不足を防ぐために、これに構文情報が付与されている部分の 3 分の 2 を音声認識モデルの学習のために追加し、残りの 3 分の 1 についてそのモデルを使い音声認識結果をデコードさせた。これを 3 度行うことで Switchboard コーパスの構文情報が付与された部分全体についての音声認識結果を得た。

自動音声認識システムには Kaldi [*1]を用い、音響モデルには GMM-HMM モデル、言語モデルには n-gram 言語モデルを用いた。自動音声認識の単語誤り率 (Word Error Rate) はそれぞれ約 28% となった。

2.3　係り受け情報の付与

この自動音声認識の結果の文それぞれに対して、人出による書き出しの文と Levenshtein 距離が最小になるようにアライメントをとり、それに付与されていた構文の情報を音声認識結果に対応付けた。

このとき対応付けた構文の情報は、先行研究の Honnibal ら [2] に従い、Switchboard コーパスに Stanford Dependency Converter[*2]を用いて得た係り受け木である。

図 1 に作成した音声認識の誤りを含む構文木データの例を示す。

アライメントの結果、自動音声認識の出力に含まれるある語に対して書き出しの文に対応相手がない、または書き出し文のある語について対応相手がいない、という状況が発生する。図 1 は自動音声認識結果の calm と bowl に対応相手がない状況であるが、この場合付与する係り受けの情報がなく、親となる語はないことになる。また、人出による書き出し文の combo という語についても対応相手がない。この場合、この語は音声認識結果に存在しないので作成したデータにはこの語に関わる係り受け関係も含めないということにした。よって作成したデータには点線で示す in the jazz のそれぞれの語から combo にかかる係り受け関係は除かれている。

作成したデータのうち、このようにして親を持たないような語トークンの割合は約 32% となった。

表 1 に Switchboard コーパスと今回作成したデータについての統計量を示す。

コーパス	文長 ≥2	学習	開発	評価
Switchboard	78207	65579	8432	4196
作成データ	64171	53569	7135	3467

表 1　構文解析の実験のデータ量。Switchboard コーパスには yes などの 1 語文が多く存在するため文の長さが 2 以上のものだけを構文解析の実験に用いる。

2.4　POS タグ付け

この作成したデータに対して Stanford POS タガー[*3]を用いて品詞情報を付与した。品詞タガーの学習用のデータとして、Switchboard コーパスにおいて品詞の情報がアノテートされていながら、構文木の情報が付与されていない部分を使った。その部分は 485 件の会話からなり、56601 文が含まれている。

POS タガーの性能を、Switchboard コーパスの構文情報の付与されている部分において評価してみたところ正解率として 95.2% となった。

2.5　考察

上で述べたようにアライメントにおいて人出による書き出し文のある語に対応する認識結果の語がない場合、その語がかかわる係り受け関係は取り除かれることになる。この場合明らかに、そのような語が構文木のなかで葉に位置している場合に比べ、構文木の中間の節点である場合はその語の子にあたる語の係り受け関係も一緒に取り除いてしまうことになるので、データ作成における

[*1] http://kaldi-asr.org/

[*2] http://nlp.stanford.edu/software/stanford-dependencies.shtml

[*3] http://nlp.stanford.edu/software/tagger.shtml

影響が大きい。そのような影響を測るため、書き出し文にありながらアライメントの結果取り除かれた語について、構文木の中で葉に位置するものと中間の節点に位置するものの割合を調べた。結果として、書き出し分からアライメントの結果取り除かれた語のうち、構文木の中間の節点に位置するような語の割合は 44% であった。

3 解析手法

3.1 従来手法

Honnibal ら [2] によって提案されている手法は、状態の遷移に基づく係り受け解析に、解析中の部分木を削除するという動作を加えたものである。ここで言う削除とは、最終的な解析結果となる構文木にその語を含めないと判断することである。構文解析の手法の基本となる部分はシフト・リデュース法によるもので、Zhang ら [5] の arc-eager システムの構成を踏まえており、デコードにビームサーチ、学習には平均化パーセプトロンを用いる。

構文解析は、以下に挙げるアクションを繰り返し行いながら「スタック」と「バッファ」を操作し構文木を漸次的に完成させる。

SHIFT: バッファの先頭の語をスタックに入れる。

RIGHT-ARC: バッファの先頭の語をスタックに入れ、元のスタックのトップの語からその語に弧を張る。

REDUCE: スタックの先頭の語をポップする。

LEFT-ARC: スタックの先頭の語をポップし、バッファの先頭の語からその語に弧を張る。

EDIT: スタックの先頭の語とその右の部分木に含まれる語すべてを削除する。左の部分木はすべてスタックに入れる。このとき削除した語に関するすべての弧を消す。

標準的な arc-eager システムと異なる点は、最後の EDIT アクションである。EDIT アクションについて図 2 で図示する。これによって、構文解析の結果、言い淀みの部分は含めない構文木を出力することができる。

パーセプトロンを用いて各状態から次に行うアクションを決定するが、その時に用いる素性は主に Zhang ら [5] で示されているスタックとバッファの内容に関する 73 のテンプレートからなる素性である。また言い淀みを検知するために、ある範囲と別のある範囲の語の系列が似ているか、ある語の近隣の語が言い淀みであったかなどの素性が加えられている。

手法の詳細については文献 [2] に譲る。

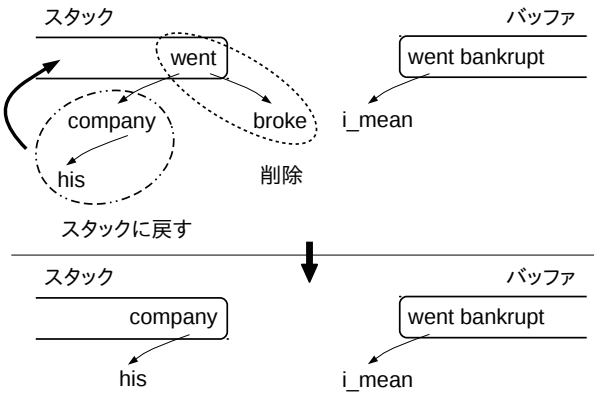

図 2 edit アクション。スタックのトップとその右の部分木は削除し、左の部分木はスタックに戻す。

3.2 従来手法の音声認識誤りに対する適用

本研究では、従来手法を音声認識の誤りに対しても拡張するため、Honnibal ら [2] の手法に手を加え上述の EDIT アクションを 2 つに分けるということを行った。具体的には EDIT アクションの代わりに言い淀み用 EDIT アクション、音声認識誤り (とそれにより親を失った語) 用の EDIT アクションの 2 つを用意した。それらの動作は本来の EDIT アクションと同じであるが、削除した対象が言い淀みであった、もしくは音声認識誤りであったということを分類することができる。

4 実験

4.1 実験方法

本研究では、自動音声認識の認識誤りを含むような話し言葉の文に対し既存手法の性能を評価するため、今回作成したデータに対して上述の Honnibal ら [2] の手法を適用した。この実験では、言い淀みに加え音声認識による認識誤りも削除の対象と考えた。また、データ作成の過程で発生した親となる語がない語についても削除の対象とする。

具体的に図 1 では、実線で示された部分木を構文解析の結果出力したい木とみなし、言い淀みである in the や、自動音声認識の誤りである calm bowl は削除の対象とする。また認識の結果消えてしまった combo を親する in the jazz も同じく削除対象とみなした。

また、この実験ではどの語の間に係り受け関係があるかのみを予測させ、係り受け関係のラベルについては予測させないことにした。よって Honnibal ら [2] の手法から、係り受けのラベルについての素性は除いた。

実験 1 では、Honnibal ら [2] の手法をそのまま適用し、言い淀みと自動音声認識の誤りについて区別せずどちらの場合もただ削除できれば良いとした。実験 2 では、3.2 で紹介した Honnibal ら [2] の手法に手を加えたものを用いて言い淀みと音声認識誤りの分類も行うことができるか実験を行った。

コーパス	UAS	誤りなし	誤りあり
Switchboard	86.0	-	-
作成データ (実験 1)	62.1	78.8	55.9
作成データ (実験 2)	62.8	79.2	58.7

表 2　係り受け解析の結果。評価データ全体、そのうち音声認識の誤りのない部分のみ、音声認識のあやまりを 1 つ以上含む部分のみでの UAS を示す。

		適合率	再現率	F 値
実験 1	全体	58.5	53.6	55.9
実験 2	言い淀み	62.5	49.3	55.1
	認識誤り	54.8	52.0	53.4
	全体	55.6	51.7	53.6

表 3　言い淀みと音声認識誤りの検出の結果。言い淀みのみ、音声認識の誤りのみ、両方を含めたものの結果を示す。

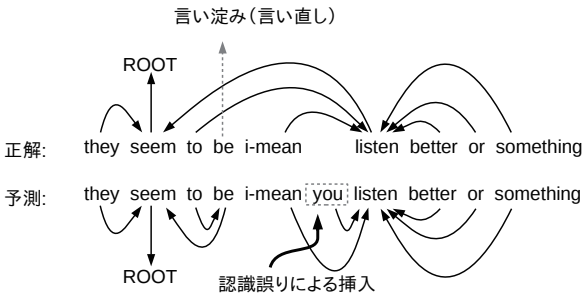

図 3　実験 1 における構文解析の結果の例。正解では be は言い淀みであり、削除されるべきであった。音声認識の結果の場合、認識誤りによる you の挿入が起こり、これにより惑わされてしまったと思われる

係り受け解析の実験の評価指標として文のそれぞれの語のうちいくつ正しくその親の語を予測できたかを測る Unlabeled Attachment Score(UAS) を用いる。また、言い淀みと認識誤りの検出についての評価は、それぞれの語について削除するかどうかを判断する二値分類の問題と考え、適合率、再現率、F 値で示す。

実験の結果を表 3 に示す。人手によって書きだした Switchboard コーパスに対して Honnibal ら [2] の従来手法を適用した結果も表 3 に示す。この結果も同様に係り受けのラベルについての素性を含めないものである。これは、表 1 に示しているようにデータの量が異なるので厳密な比較とはならない。

4.2　考察

データ作成を通して、約 3 割の語トークンについては、それに親がないという状況になったが、そのようなデータでも表 2 に示すように入力の文が綺麗な文であれば正しく係り受け解析を行うことができており、係り受け解析の学習ができていることがわかった。

しかしながら、自動音声認識の誤りを含むような文では、構文解析がうまく行っていない例が多く見られた。

音声認識の結果に惑わされた例を図 3 に示す。この例では be が言い淀みであり、あとで listen に言い直されている。よって be に対して適切に判断し、解析途中に削除しなければならない。しかし、自動音声認識の結果、誤って you という語が挿入され、このことにより構文解析に悪影響が起こっていると考えられる。なぜなら、挿入によって文の you 以降の語列がひとつの節を構成してしまい、全体として they seem to be と you listen better or something の 2 つの節からなるような文に見えてしまい、to be i-mean listen が言い直しであると判断できなかったようであるからである。

実験 2 から、言い淀みと音声認識誤りの区別は今回の実験では困難であることがわかった。特に今回の実験では素性は従来研究と同じものであったため、音声認識誤りを区別できる素性の追加が必要であると思われる。

5　終わりに

話し言葉は言い淀みや自動音声認識による認識誤りなどの特徴を含みこのような文を係り受け解析することは困難であるが、既存研究では言い淀みに対して特別な処置を取ることで高い性能を示している。しかしながらこれらの研究では人手によって書き出された文を用いて実験しており、実際の利用場面で発生する自動音声認識による誤りは考慮に入れられていない。本論文では自動音声認識による誤りを含むコーパスを構築し、構文解析の実験を行った。その結果、構文解析の精度は音声認識の誤りによって大きく悪影響を受け、それに対して処置を行う必要があることが明らかとなった。

今後の課題としては、今回の研究を踏まえこのような自動音声認識による誤りに対しても頑健な構文解析を構築することである。

参考文献

[1] Godfrey J. J. Holliman E. C. and McDaniel J. “switchboard: Telephone speech corpus for research and development”. 1992.

[2] Mark Johnson Matthew Honnibal. Joint incremental disfluency detection and dependency parsing. 2014.

[3] Mohammad Sadegh Rasooli and Joel Tetreault. Non-monotonic parsing of fluent umm i mean disfluent sentences. 2014.

[4] Shuangzhi Wu, Dongdong Zhang, Ming Zhou, and Tiejun Zhao. Efficient disfluency detection with transition-based parsing. July 2015.

[5] Yue Zhang and Joakim Nivre. Transition-based dependency parsing with rich non-local features. 2011.