修　　士　　論　　文

現実世界の時間空間制約を用いた歴史文章解釈

History Text Interpretation by Using Time and Spatial Constraints in the Real World

指導教員　　**鶴岡 慶雅 准教授**

東京大学工学系研究科
電気系工学専攻

氏 名　　**37-156506 村上 優樹**

提 出 日　　**平成 29 年 2 月 2 日**

概 要

近年インターネットの普及により大量のテキストデータが生成されている。その大量のデータを高速に賢くコンピュータに処理させる自然言語処理技術の需要は高まり、盛んに研究されている。自然言語処理の中でも文章の意味をコンピュータに理解させることを目的とした意味理解の分野では、様々なサブタスクやそれに対する手法が提案されている。それらの手法の多くは単語やその品詞、構文情報など文章の表層情報を特徴として機械学習を用いるものであるが、それだけでは文章全体の意味や文脈の整合性を捉えきれておらず、出力結果が実際の解釈と違う場合がある。これは自然言語に潜在的に含まれる曖昧性や一般常識の省略などによって起こる。我々人間は文章を理解する際に、文章そのものだけでなく、今までの経験や背景知識なども活用するため、コンピュータに自然言語の深い理解を実現させるためには文章の表層情報だけでは困難だと考えられる。

このような文章の正しい解釈を目的として、最近は現実世界との対応を取ることを考えた研究が発表されている。例えば2つの固有名詞間の関係性についての知識を機械学習に用いる特徴に加えたり、グラウンディングと呼ばれるテキスト上の表現と実世界のオブジェクトとの対応付けを学習したりする研究がある。しかし現実世界を表現する文章の前後関係や定量的な表現、文章全体の整合性などについて深く考慮された自然言語解釈はまだ実現されていない。

そこで本研究では実世界での出来事をモデル化した世界モデルを利用した自然言語処理に取り組む。この世界モデルでは現実世界での出来事をコンピュータ上でシミュレーションすることを想定したものであるが、自然言語を解釈した結果がそのモデル上での出来事と矛盾がないかどうかを確かめることで、正しい解釈を行うことができる。しかし全ての出来事をモデル化することは非常に困難であるため、最初はある人物の移動にのみ着目した人物移動モデルを扱うことにする。このモデルではある人物に関する時間・空間的な制約について自然言語解釈と整合性がとれるかを確認する。

実際に世界史の幾人かの人物についてのデータを用い、現実世界と整合性のとれた自然言語解釈に取り組み、質問応答や共参照解析、イベント抽出など意味理解に深く関わる重要なタスクについて実験を行った。その結果、質問応答、共参照解析ではそれぞれ既存の従来の表層情報を用いる手法では難しいと考えられる問題について正しい解釈を得られる可能性を示した。イベント抽出では意味役割付与の構文解析結果を用いたベースラインには精度で及ばなかったが、既存手法では抽出困難だと考えられるイベントについて、モデルを利用することで正しく抽出できる例を示した。このことから時間・空間的な情報と人物移動モデルを用いることで、より深い自然言語解釈ができる可能性について示す。

目 次

図 目 次

表 目 次

アルゴリズム，擬似コード

第1章　はじめに

1.1　背景・目的

近年インターネットの爆発的な普及により、WEB 上に大量のテキストデータが生成、蓄積されている。それらのデータを人間が処理するのは一般的に高コストであるため、人間の言葉をコンピュータで自動的に賢く高速に処理することを目的とした自然言語処理技術は需要が高まっており、多くの研究が盛んに行なわれている。自然言語処理の基礎タスクである品詞推定、構文解析などの精度は実用レベルまで達し、最近では主に自然言語の意味を扱う意味理解が研究されている。そもそも自然言語の意味そのものを扱うことは非常に難しいため、さまざまな側面からサブタスク化が行なわれ、多くの手法が提案されている。確かにこれらのタスクは自然言語の意味を扱う上でとても重要だと考えられるが、タスクが解けるようになったとしても必ずしも自然言語の意味を理解したとは言えない可能性について説明する。

自然言語理解のためのサブタスクとして含意関係認識というタスクがある。この含意関係認識は、前提となるテキスト T から仮説 H が意味的に導出できるかどうかを正誤判定するタスクであり、質問応答や情報抽出、自動要約などさまざまなタスクに役立つとされている [12] [35]。

- T: “Tom and Mary play soccer at the park.”
- H: “Tom is in the park.”

上の例のように、“Tom and Mary play soccer at the park”が正しいならば “Tom is in the park” も正しいという含意関係を判定することになる。このタスクが解決されれば 2 つの文同士の意味を比較判定することができるようになり、自然言語で表記された内容を理解しているような作業を行うことができる。このタスクを解決するための主な基本手法として 2 つの文中の単語や構文の重複度がよく利用される。この手法は簡単であるが、PASCAL という共通タスクにおいて 58%の精度（その他のシステムの精度は 50 〜 60%）を出すほど強力である [12]。しかし “Tom and Mary play soccer at the park.” という文の意味そのものを理解しているとは言い難い。我々人間がこの文を読むと、「Tom が平らな地面でボールを蹴る、Mary が飛んでくるボールを足で受け止める、おそらく天気は良いだろう、外は明るいだろう」といった文には直接書かれていない情報まで推測できる。これは人間は今までの経験や知識として持っている情報を総動員して文を理解すると考えられるからである。

他にも、文章を読んで質問に答える読解タスク (Reading Coprehension）が提案されている。これは長文とそれに対する質問文を入力し、システムに正しい答えを出力させるもので例えば MCTest がある [29]。MCTest では 4 択問題形式を採用しており、ランダムに選ぶと 25%の正解率となる。簡単

なベースラインとして、質問文と答え（選択肢の 1 つ）のセットと問題文について単語の出現頻度の一致度の高いものを選ぶ手法が提案されているが、この簡単な手法の正解率は 66%である。7 割弱の読解問題を解けているが、同じような単語が出現するものを選択しているだけで、実際には文章の内容を理解しているわけではない。

もう一つ共参照解析の例を説明する。共参照解析とは自然言語文中に存在する名詞が、実世界のどのエンティティを指示するかを判定するタスクである。

- **Tom** and **Mary** play soccer at the park.
- **He** kicks a ball.

例えば上の例では、代名詞 “He”が “Tom”と “Mary”のどちらを指すかを判定するタスクになる。このタスクを解決する手法として、対象となる 2 つの名詞句の形態的な特徴を利用する方法が提案されている [32]。形態的な特徴とは「男性名詞か女性名詞か」「単数か複数か」といった要素の一致をみるもので、それ以外にも 2 つの名詞句間の距離や文字列としての一致度合などの特徴を利用している。確かにこの方法だと簡単な問題は解けるが、例えば下のような文の意味を理解しないと解決するのが困難であると考えられる例も存在する。

- **Tom** made a call to **Bob** from Tokyo station.
- **He** was surprized in calling from his friend studying in Japan.

“Tom”と “Bob”はどちらも男性名詞であるため形態的な特徴は使えず、既存の手法で解決することは難しい。実際によく使われている有名な Stanford Core NLP の共参照解析器 [22] で上の例を試してみると “He”は “Tom”を指すという結果になる。一つ目の文から日本にいるのは “Tom”だとわかるので、日本にいないと思われる “He”は “Bob”を指すのが正しい共参照解析となるため、この結果は誤りである。このように文の意味を深いレベルまで考慮しなければ解決できない問題もある。

そもそも自然言語は潜在的に曖昧性を含んでおり、我々人間はその曖昧性を今までの経験や知識から排除する術を持っている。確かに自然言語文の単語や構文情報など何らかの表面的な特徴を使い機械学習などでタスクを解決することは汎用的な手法であるが、先ほどの例のような難しい問題を解決するには自然言語文の表層的な情報だけでは不十分だと考えられる。

そこで自然言語文だけでなく、世界知識と呼ばれる我々の身の回りにある常識的な事実を活用する研究が増えてきている。例えば先ほどの共参照解析の例では 2 つの名詞句間の関係をデータベースとして用意し、それを特徴として機械学習に組み込む手法が提案されている [28]。しかし、従来手法の特徴量に知識を組み込むだけでは解決できない問題も多く、現実世界との対応を適切に利用した深い推測による自然言語解釈が必要だと考えられる。

このような背景から自然言語の深い理解を目指した研究が最近になって発表されるようになった。例えば Branavan らは自然言語文をただ解釈するだけでなく、その文章が表す対象であるドメインの環境と何らかの相互作用を行うことで、そのドメイン内の世界と整合性のとれた解釈を学習することに取り組んでいる [8] [9]。Bordes らは “自然言語の一番重要な特徴は実世界における物理的な事柄に対する表現である” [7] という主張から、簡単な家の中の人や家具の配置に関するモデルを利用した、エ

ンティティと自然言語文との対応付けに関する研究を発表している。他にも Hajishirzi らは、“自然言語理解において必要とされる能力はその背後にあるエンティティやそれらの関係性、動作を適切に表し、さらに文全体として整合性が取れるように組み合わせることである” [15]、と主張した上でロボットサッカーのコメントの適切な解釈に関するタスクに挑戦している。その他にも様々な興味深いタスクが提案されているおり、詳細は 2.1 節で説明する。これらの研究はドメイン内における自然言語文とエンティティとの対応についての整合性を考えることが基本的な手法である。特に [15] はテキスト全体での整合性まで考慮する手法であり、自然言語解釈においてとても重要な内容である。しかしこれらの研究は特殊なドメイン内で行われることが多く、現実世界の問題を解くには至っていない。そこで本研究では現実世界での何らかの整合性を考慮した自然言語解釈を実現するために、世界モデルという概念について考える。

世界モデルとは我々の身の回りで起こる出来事をコンピュータ上でシミュレーションすることを想定したものであり、これを利用することで現実世界における正しい自然言語解釈を実現することが可能となる。しかし現実的にあらゆる出来事をシミュレーションすることは不可能であるため、今回は人物についての時間空間情報に着目した人物移動モデルについて考える。[15] では試合中にボールを持てるプレイヤーは一人だけ、という制約を利用していたが、本研究では時間・空間的な制約を利用することで現実世界に関する文章を正しく解釈することを目標とする。またこの人物移動のモデルを利用することでどのような従来の自然言語処理タスクが解決できるかについても考えたい。

1.2 本研究の提案

本研究では正しい自然言語解釈を実現するために、世界モデルの簡略版である人物移動モデルを用いて時間・空間的な制約を利用した新しい自然言語解釈手法を提案する。この手法では自然言語文だけでなく、その内容に関する時間・空間情報を適切に活用することで、従来手法による間違った解釈を抽出することを試みる。そこで本研究では時間・空間情報を活用しやすいと考えられる実際に起こった世界史をテーマに歴史文章解釈に取り組む。さらにこの解釈を通して従来の自然言語処理のタスクである質問応答や共参照解析、イベント抽出などにおいて、本手法がどの程度有効であるかについてその可能性を検証する。

1.3 本研究の貢献

本研究の貢献は以下の通りである。

- 歴史文章中に記述される人物の移動に関する内容について、モデル中でシミュレーションという目に見える形でその移動を再現することにより、自然言語解釈の可能性を提示した。これにより、文章中に含まれる幾つかの情報を全体を通して整合性を考えることで正しい解釈を実現できることが期待される。
- 時間・空間情報を利用して、自然言語処理タスクである質問応答、共参照解析、イベント抽出において、従来手法では困難な問題の解決、および従来手法での誤った解析結果を抽出することに

成功した。これにより人間が自然言語文を正しく解釈するように、コンピュータの背景知識による深い推測による整合性のとれた正しい自然言語解釈の実現が期待される。

1.4 本論文の構成

本論文の構成は以下の通りである。

第 2 章: 関連研究 正しい自然言語解釈を実現するために、自然言語文だけでなく対象となるドメイン中の環境を適切に利用した研究例について紹介する。また、本研究で扱う自然言語処理タスクである質問応答と共参照解析、イベント抽出について、タスクの説明と従来手法について説明する。

第 3 章: 人物移動モデルを用いた質問応答 本研究で提案する時間・空間情報を利用した人物移動モデルについて説明する。またこのモデルがどのようなタスクに適用できるのかを検証するための準備実験として簡単な質問応答例を示し、他タスクへの応用を考察する。

第 4 章: 時間空間情報を用いた共参照解析 第 3 章で述べる準備実験では扱うドメインが固定されていたので、その解決策として Wikipedia を利用した拡張について説明する。また共参照解析の精度向上を実現するために、時間・空間情報を組み合わせた方法と実験について説明する。

第 5 章: 人物移動モデルを用いたイベント抽出 第 4 章の共参照解析の問題点として本文中から特定の情報を抜き出すイベント抽出の精度が良くないことが挙げられる。そこで人物移動モデルを用いたイベント抽出タスクの精度改善とその手法、実験について説明する。

第 6 章: おわりに 最後に本論文の結論と今後の展望について述べる。

第2章　関連研究

2.1　自然言語とそれが表現する環境との対応を考慮した研究

自然言語文で表現される内容の情報は、自然言語解釈を行う上でとても重要である。自然言語文と、その内容となる特定ドメイン内の環境の対応を適切に利用して正しい自然言語解釈に挑戦する論文を紹介するとともに、深い自然言語解釈とは何かについて説明する。

2.1.1　自然言語文とオブジェクトとの対応を考慮した曖昧性解消

自然言語の意味を理解するために、テキストとテキスト中にかかれている背景やオブジェクトとの対応付けを考えた研究を 3 つ紹介する。

まずはテキストから 3D 空間を生成するタスクに取り組んだ chang らの研究を説明する [11]。この論文では例えば “There is a room with a chair and a computer.” という自然言語文から図 2.1 のような 3D 空間を自動生成することを目標としている。この例文には机の存在は一切触れられていないが、生成されたモデルには机が存在している。これを実現するためには、例えば「コンピュータは普通は机の上で使われるものである」という一般常識が必要になるし、床と机の位置関係、机と椅子の位置関係、パソコンの向きなど、テキストでは省略されている多くの知識が必要となる。この論文では大量に用意された 3D モデルのサンプルについて、クラウドソーシングによるアノテーションを行い、自然言語によるラベル付けデータによる機械学習を行うことで、単語やフレーズが持つ 3D 空間上での意味を定量的に対応付けることに成功している。例えば英語には様々な前置詞が存在する。その中でも “on” と

図 2.1. “There is a room with a chair and a computer.” から生成された 3D 空間例 [11]

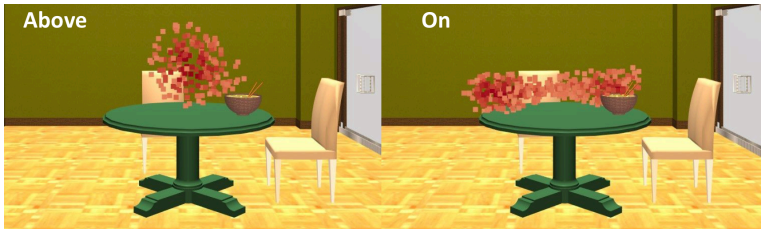


図 2.2. “above” vs “on” [11]

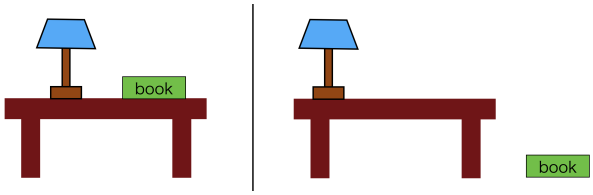


図 2.3. 3D 空間を表現する自然言語の曖昧性

“above”はどちらも「上に」という意味を含むが、そのニュアンスは 3D 空間上では少し異なるという例を図 2.2 に示す。また今論文の今後の課題では構文解析の曖昧性解消についても説明されている。例えば “Put a lamp on the table next to the book”という文が表現する 3D 空間は図 2.3 に示すように 2 通りの解釈ができる。我々人間は「本は普通は床ではなく机の上にのっている」という知識を持っているので図 2.3 の左の解釈が自然であると判断できるが、実際に Stanford の dependency parser [22] で解析してみると図 2.4 のような結果が得られる。この解釈では “next to the book”が “table”にかかっているので、図 2.3 の右側の解釈となり、これは人間から見たら不自然な解釈となってしまう。このような難しい曖昧性を解消するためにもテキストだけでなく、それが表す内容との対応を考える必要があるといえる。

次に Berzak らの映像を利用した曖昧性解消に関する研究を説明する [2]。Berzak らは自然言語に潜在的に存在する曖昧性を除去するために、映像との対応を考慮する手法を提案している。具体的なタスク例を図 2.5 に示す。図 2.5 は “Sam approached the chair with a bag” という文に対する 2 通りの

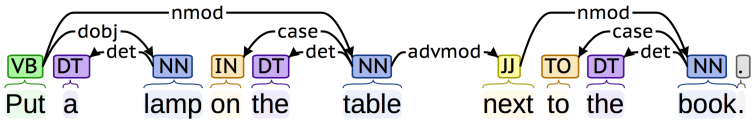
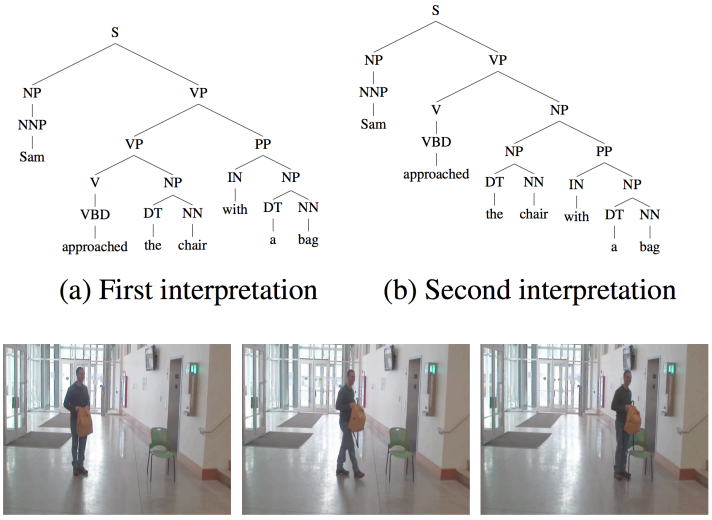


図 2.4. “Put a lamp on the table next to the book”の構文解析例



(a) First interpretation (b) Second interpretation

(c) Visual context

図 2.5. 映像による自然言語の曖昧性解消 [2]

構文解析結果と、文の表す内容を映像で表現したものがセットになったデータを表している。図 2.5(a) の解釈は Sam がバッグを伴って椅子に近づくという解釈で、図 2.5(b) の解釈は Sam がバッグがある椅子に近づくという解釈である。映像データを見ると、Sam という人物がバッグをもって椅子に近づいていることがわかるので、この問題例の場合は (a) の解釈を選ぶのが正解となる。Berzak らはこのタスクを解決するために構文解析結果を一階述語論理の形として意味表現に変換し、その述語論理の変数が表すオブジェクトと、映像データから検出されたオブジェクトとの最適な対応付けをコーパスから学習することで、自然な構文解析を選択する手法を提案している。映像の内容という外部知識を利用することで、自然言語の曖昧性を解消するという、自然言語理解にとってとても重要な研究であり、テキストの表層情報だけでは自然言語を解釈することが困難であることがわかる。

次に Seo らの図形問題と自然言語処理の対応付けに関する研究を説明する [30]。Seo らは図 2.6 に示すような図と自然言語文からなる図形問題に対する解答システムに自然言語解釈を利用し、それを通して自然言語の構文解析の精度向上に成功した。このタスクでは、Seo らは問題を表記する自然言語文を論理表現にマッピングし、図形から読み取れる制約条件と照らし合わせることで、そのマッピングのスコアを更新することにより正しい自然言語解釈を行う方法を提案している。また図形との対応を考えることで、自然言語文の情報のみによるマッピングの間違いを見つけることに成功している。例えば “AB is perpendicular to CD at E” という文について依存構造解析 (dependency parsing) を行い、その結果から論理表現へのマッピングを行ったとする。dependency parsing の結果が誤っている場合は、各記号同士の関係をうまく捉えることができず、この論理表現と図形から読み取られた制約条件

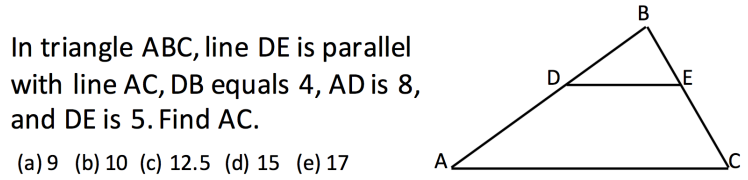


図 2.6. 自然言語解釈による図形問題の解決 [30]

とが矛盾してしまうような場合がある。この場合 dependency parsing の結果がおかしいという仮定のもと、stanford parser の解析結果を選び直す操作を行うことで精度を改善している。

このように自然言語文中に含まれる表層的な情報 (単語や構文情報など) だけでは、その内容を把握することは困難であると考えられ、深い自然言語理解のためにはこのようなテキストが表現する内容と対応を考えることが必要であることがわかる。本研究では実際に現実世界との対応を考慮することで、深い自然言語解釈を目指す。

2.1.2 自然言語文の表現する環境との相互作用を利用した研究

次に、自然言語文が表現している環境との相互作用を利用した研究について説明する。ここでの環境とは、自然言語文の表すドメイン内で定義される状態空間のことを指す。自然言語を理解するためには文単位での解釈も必要だが、文の前後関係などをふくめた文章全体としての解釈も必要である。このような問題を解決するための研究例を幾つか説明する。

まずは Branavan らの Windows マニュアルの強化学習による解釈についての研究 [8] を説明する。この論文では Windows の操作説明に関する自然言語文から実際の画面上でのアクションへのマッピングをタスクとしている。具体的なタスク例を図 2.7 に示す。

文 u からその解釈として画面上でのアクション列 $\overrightarrow{a} = (a_0, a_1, ..., a_{n-1}), a_i = (c, R)$ (c はコマンド、R はコマンドの引数) を求める。その際、自然言語文だけでなく、今の画面のオブジェクトなどの情報 ε も利用した機械学習を行っている。画面内の情報は逐一自然言語解釈の結果によるアクションを実行することで遷移していき、アクションの実行実現性を確かめることで、自然言語解釈の誤りを見つける方法が提案されている。例えば図 2.7 の例では、“click Run”という文について「Run というボタンを押す」以外の解釈をすると想定される画面とは異なる画面が表示されることにより、次の “typing secpol.msc in the open box”を実行できなくなってしまう。この論文ではこの情報をうまく活用することでアノテーションデータなしに、自然言語を正しく解釈することに成功している。

また Branavan らは Minecraft というゲームに関するマニュアル解釈の研究を発表している [9]。Minecraft とは小さな立方体を敷き詰めたグリッドで構成される世界の中で、様々な法則に則りアイ

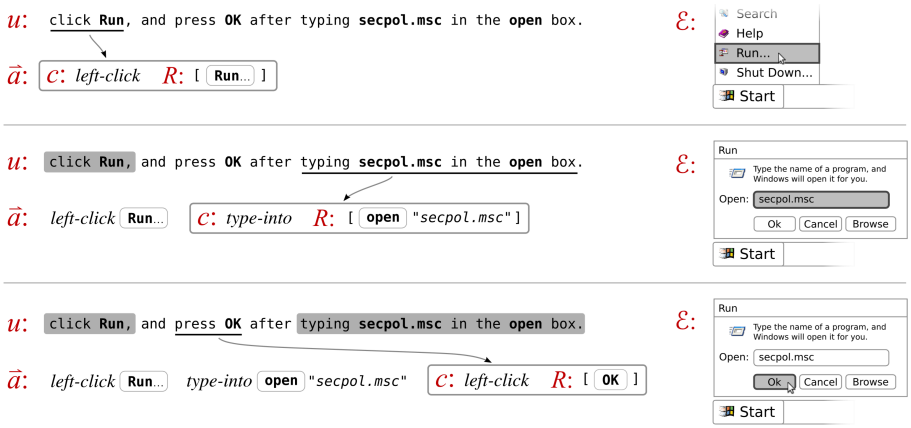


図 2.7. Windows マニュアルの自然言語解釈 [8]

テムを駆使することで世界を作り上げていくシミュレーションゲームである[1]。そのアイテム間の関係は precondition (アイテムを作るために必要となる前提条件) と effect (前提条件を満たしたことにより起こる結果) で定義されており、その関係を説明する wiki 中の自然言語を解釈するタスクに取り組んでいる。この研究では従来手法による自然言語解析の結果得られた precondition と effect に則って、Minecraft 上の世界でそれを実行し、その結果を確認することで強化学習を行う手法を提案している。この手法により、Minecraft の世界と整合性のとれた自然言語解釈を実現することに成功している。

Bordes らはあらかじめシミュレーションにより生成した状態遷移について、そのシミュレーションに関する自然言語中の名詞句とシミュレーション中とのオブジェクトとの紐付けを行うタスクに関する研究を発表している [7]。この論文では家の中に存在する人物や物体をシミュレーションで動かし、それらの関係性を明確にした上で、自然言語文とペアにしたデータを自動的に生成している。このように自然言語の表す対象の情報を利用することは深い自然言語解釈においてとても重要であるが、この研究ではあらかじめシミュレーションで生成した状態遷移を元に問題を生成している。多くの自然言語解釈で求められるタスクでは、自然言語文と一般的な世界の法則を与えられた上で、その自然言語文の背後にある状態遷移などを予測する必要があるが、この研究ではそれがあらかじめ与えられてしまっている。そこで本研究では、このような背後にある状態遷移を時間・空間的な情報に着目し予測する。

他にも Hajishirzi らは RoboCup Soccer という、マルチエージェントシステムによるロボットのサッカーシミュレーション大会をテーマに自然言語解釈を行う研究を発表している [15]。この論文では例えば “Pink7 makes a bad pass that was picked off by Purple 7.” という実際の大会でのコメント文を “badpass(Pink7,Purple7)” という論理表現にマッピングするタスクを扱っている。サッカーコート上での状態 (誰がボールを持っているかなど) をコメント文の解釈から遷移させていくことで、実際のゲー

[1] https://minecraft.net/ja-jp/

ムと整合性のとれない解釈にペナルティーをつけ、正しい解釈ができるような手法が提案されている。

このように自然言語文が表現する環境を状態空間として表現し、それとの対応を考慮することでより正しい解釈を実現することができる。これらの研究では人手により完全に法則が明文化された特殊なドメイン中を対象としている。そこで本研究では、できるだけ人手による世界の法則の生成を減らし、時間・空間情報を適切に用いて我々の現実世界についての深い自然言語解釈に取り組む。

2.2 本研究で扱う自然言語処理タスク

2.2.1 質問応答

質問応答とは、システムに質問が与えられたら、それに対する適切な回答を行うシステムを構成するタスクである。その質問形式は様々で人物、地名など単語で答えられる問題を扱うものもあれば、なぜその事柄が起きたのか、という理由を扱うものもある。以下に問題例を示す。

- Q: 雪国を書いたのは誰か？
- A: 川端康成

正しく応答するために様々な手法が提案されている。例えば Li らは、質問に対する回答候補となる文を仮説として複数用意し、それらを大規模なデータを用いてスコアをつけ、最もスコアが高かったものを答える方法を提案している [21]。この例における仮説とは例えば「川端康成は雪国を書いた」などとなる。IBM が開発した Watson [13] なども基本的な構造はこのようになっている。各仮説を大規模なデータと照らし合わせて評価を行う時に例えば含意関係認識 [12] という二つの文の意味が含意関係にあるか、同じ意味とみなせるか、ということを判断する技術を用いることがある。含意関係認識のタスク例を以下に示す。

- T: 川端康成は雪国の著者である。
- H: 川端康成は雪国を書いた。

テキスト T からその仮説 H が導出できる、つまり T が正しい時 H も正しいかどうかを判断するタスクである。これを解決するため、二つの文の構文情報や単語・品詞が何であるか、同じ単語が使用されているかなど、文の表層的な情報が使われることが多い [12]。しかし、このような表層的な情報だけでは、自然言語の意味を本質的に捉えているとは言えない。一方、自然言語の意味を扱うために、Fowler [14] らは COGEX という論理証明器を用いて、T と H の論理形式から 2 つの文の論理的な正しさを計算する手法を提案している。この方法では自然言語の書き換え規則や一部世界知識に関する変換規則などを用いているが、例えば定量的な内容や深い推測が必要なものは困難だと考えられる。もともと数量的表現を伴う含意関係認識は難しいと考えられている [39]。

このように自然言語の内容に答える、および意味の同一性を判断することは単語のオーバーラップなどを用いるとある程度は実現できるが、たとえば直接表現されない事実など深い推測が必要となる問題は難しいと考えられる。そこで本研究では、直接表現されていない内容についての質問に、時間・空間情報を用いた推測を行うことによって答えることに取り組む。

2.2.2 共参照解析

共参照解析とは、文章中に現れる複数の名詞句のうち、同じエンティティを表しているものを見つけるタスクである。以下の文で太字で表示されたものが名詞句である。

- **Tom** and **Mary** played **soccer**. **She** kicked a **ball**.

これらの名詞句のうち、同じものを表している組を見つけることを考える。この例の場合ではシステムは “She”が “Mary”を示すことを判断することが求められる。基本的な手法として、各名詞句の形態的な特徴を用いて分類器を学習する手法 [32], [27] が知られている。形態的な特徴とは、二つの名詞句が「単数か、複数か」「男性名詞か、女性名詞か」などの特徴のことで、それらの一致を考えることは直感的にも有効な方法だと考えられる。Soon ら [32] はこれらの他にも以下のような特徴を有効だと考え利用している。

- 2 つの名詞句間の距離 (文単位でどの程度はなれているか)
- 代名詞かどうか
- 文字列としての一致
- 固有名詞かどうか
- など

これらはいずれも自然言語文の表層的な特徴しか用いておらず、すべての共参照関係を正しく分類できるとは考えづらい。例えば文中に “He”の候補となる男性名詞が複数存在する場合の曖昧性などは、周辺の単語の利用や、文の意味を考慮する必要がある。また最近では自然言語文の特徴以外に、大規模な世界知識を用いる手法 [26] [28] が提案されている。これは例えば「オバマは大統領である」など二つの名詞句の関係を大量に用意しておいて、それら全てを特徴量として用いることで分類器を学習するものである。これらの研究のエラー分析で自然言語の意味そのものを捉えられていないことが原因の一つとしてあげられており、既存の共参照解析手法では意味を捉えた上で判断することは難しいと考えられる。

例えば冒頭でも挙げた以下のような例では、名詞句間の関係などの世界知識を用いるだけでは解決することができず、ある程度の意味的な推測が必要となる。

- **Tom** made a call to **Bob** from Tokyo station.
- **He** was surprized in calling from his friend studying in Japan.

そこで本研究では現実世界の情報として時間空間制約を利用することで、既存解析器による “He”が示す人物が整合性の取れた正しい結果であるかどうかを判断することで共参照解析の精度向上に取り組んだ。

2.2.3 イベント抽出

イベント抽出とは情報抽出の一つで、自然言語文から 5W1H に相当する情報、つまり「いつ誰がどこで何をどのようにどうした」という文の中心的な内容を自動的に抜き出すタスクで、自然言語理解や質問応答、対話システムなど、より応用的な技術に利用されている [17]。

例えば “Born in Pella in 356 BC, Alexander succeeded his father, Philip II, to the throne at the age of twenty” という文章では “Born” という述語（以下 predicate という）に注目すると、以下のようにそれぞれ該当していることを判断するのが目標となる。

- Who: Alexander (イベントの動作主)
- When: 356 BC (イベントの起きた時間)
- Where: Pella (イベントの起きた場所)
- What: Born (イベント)

短い文だと単純なルールベースによる方法で解決することができるが、少し長い文になると名詞句や修飾部分が複数個出現し、自然言語の曖昧性による解釈ミスが起きる可能性がある。

これを解決するための手法として Surdeanu らは Templette と呼ばれるあるイベントに関するフレームを利用する方法を提案している [33]。このフレームは例えば “Death” というイベントの場合は＜殺された人＞、＜殺した人＞、＜死に方＞、＜死んだ時間＞、＜死んだ場所＞という要素が定義されている。この用意されたスロットに述語項構造解析で得られる主語や目的語を穴埋めすることでイベント抽出を行う。与えられた文がどのイベントクラスに対応するのかについて Naughton らはサポートベクターマシン (以下 SVM) や言語モデルを利用した分類手法を提案している [24]。Wang らはイベントクラスを SVM で分類したのちに、ルールと固有表現抽出 (文中から固有名詞や時間表記などを自動的に抽出する技術、以下 NER) によって 5W1H を決定する手法を提案している [38] [31]。この手法では “When”や “Where”に関する情報は NER で＜ DATE ＞や＜ LOCATION ＞と判定されたものを選択し、“Who”に関する情報は “名詞句＋動詞＋名詞句＋名詞句”のようなパターンマッチで抽出している。他にも意味役割付与 (Semantic Role Labeling 以下 SRL）を利用する手法が提案されている。SRL はある predicate に着目した時に、その主語や目的語、修飾部分 (以下 argument) などを判断するタスクで、これを用いればイベントの中心的役割となる predicate に関する 5W1H を選択するヒントが得られる。例えば Wang らは SRL と NER を利用した手法 [36] [37] を、McCracken らは SRL と簡単な文法的ルールを利用した手法 [23] を提案している。SRL を含めた詳細については 2.2.3.1 節で述べる。

これらの手法は基本的に自然言語文の表層的な特徴をもとにルールや機械学習によってイベント抽出を行うものであるが、その解釈が現実世界と整合性の取れない誤ったものになってしまうことがある。これは文の意味を考えてイベント抽出を行っているわけではないことが原因のひとつであると考

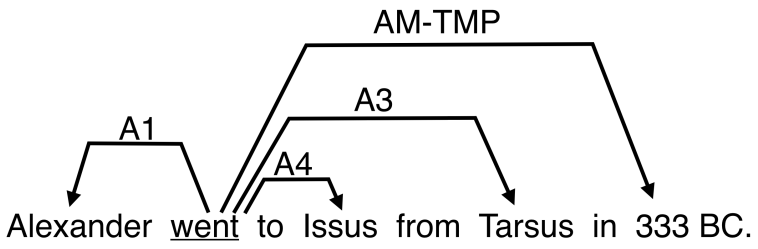


図 2.8. SRL 解析結果例

えられる。そこで本研究では人物移動モデルを用いて時間空間的制約を利用することで誤った解釈を除去しイベント抽出の精度改善を試みる。

2.2.3.1 意味役割付与

意味役割付与とは Semantic Role Labeling と呼ばれるタスクで、述語項構造にラベルを付けた解析結果を出力することが要求される。例えば “Alexander went to Issus from Tarsus in 333 BC.” という文が入力された時の出力を図 2.8 に示す。

この例では predicate が “went” しかないのでこの 1 セット分だけ解析することになるが、文中に複数 predicate が存在するときは、それぞれの predicate ごとに argument とその関係性を表すラベルをつける必要がある。このラベルは Proposition Bank [25] というコーパスのアノテーションに用いられたフレームセットで定義されている。Proposition Bank は、ウォールストリートジャーナルに品詞タグや構文解析アノテーションをつけた Penn Treebank に、意味ラベルを付与して作成されたコーパスで SRL の学習に用いられる。このフレームセット[2]は各単語 (正しくは lemma) ごとに用意されており、例として “go” の場合を図 2.9 に示す。

“go.01” の 01 は、同じ “go” という単語でも複数の意味を持つのでそれを区別するための添字を表す。図 2.9 を見てみると、argument 番号 1 は移動するエンティティ、argument 番号 3 は移動開始位置、argument 番号 4 は移動終了位置と定義されており、図 2.8 の A1,A3,A4 と対応が取れていることがわかる。この argument ラベル (A0 〜 A5) は各フレームごとに役割が定義されており、それぞれが持つ意味は異なる (A0 は主語になりやすい、などの傾向はある) 。これとは別に predicate の修飾を意味する全フレームで共通の argument が定義されており例えば以下の様なものがある。

- AM-TMP: 時間
- AM-LOC: 場所
- AM-NEG: 否定

[2]ブラウザによる閲覧は、http://verbs.colorado.edu/propbank/framesets-english/ xml 形式でのダウンロードは https://github.com/propbank/propbank-frames から行える。

```
<frameset>
<predicate lemma="go">
<note>
Frames file for 'go' based on survey of initial sentences from big corpus
and comparison with 'rise' 'fall' 'become' and 'wander'
</note>

<roleset id="go.01" name="motion" vncls="47.7 51.1-2">
<roles>
  <role descr="entity in motion/goer" n="1">
    <vnrole vncls="47.7" vntheta="Theme"/>
        <vnrole vncls="51.1-2" vntheta="Theme"/></role>
  <role descr="extent" n="2"/>
  <role descr="start point" n="3"/>
  <role descr="end point, end state of arg1" n="4"/>
  <role descr="medium" f="LOC" n="M"/>
  <role descr="direction (usually up or down)" f="DIR" n="M"/>
</roles>
```

図 2.9. フレーム例 “go”

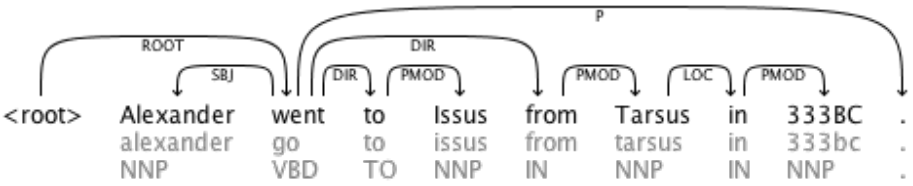


図 2.10. dependency parsing 例

- AM-DIR: 方向
- ...

図 2.8 には、“went”が起きた時間情報として “333BC”に AM-TMP ラベルがつけられている。このように SRL は predicate の抽出と、各 predicate に対応する argument の判定、ラベル付けを行うタスクである。

この SRL については 2004 年, 2005 年, 2008 年, 2009 年に CoNLL の共通タスクとして取り上げられ様々な手法や評価方法が提案され、その精度を競っている。ここでは 2008,2009 年に行われたタスク [34] [16] について説明する。2008 年には英語、2009 年には英語以外にも中国語、日本語、チェコ語、ドイツ語、スペイン語などのデータも追加されているが、ここでは英語についてのみ扱う。2008 年には dependency parsing も精度評価の対象となっている。dependency parsing とは SRL を行う上で非常に大切とされている構文解析であり、図 2.10 に示すように、単語間の係り受けに注目した解析で、中心となる predicate “went”との係り受け関係にある単語に矢印が伸びている。2009 年では dependency parsing の結果と predicate の抽出まで完了した状態で SRL のみの評価タスクとなっている。CoNLL のデータは具体的には表 2.1 に示すような形式で与えられており、セルを埋めることが要求される。表の各項目はそれぞれ以下のような意味を表している。また頭に P がついているもの (PPOS, PHead な

ID	Form	PLemma	PPos	PHead	PDeprel	FillPred	Pred	APred1	...
1	Alexander	alexander	NNP	2	SBJ	-	-	A1	...
2	went	go	VBD	0	ROOT	Y	go.01	-	...
3	to	to	TO	2	DIR	-	-	-	...
4	Issus	issus	NNP	3	PMOD	-	-	-	...
5	from	from	IN	2	DIR	-	-	A3	...
6	Tarsus	tarsus	NNP	5	PMOD	-	-	-	...
7	in	in	IN	6	LOC	-	-	-	...
8	333BC	333bc	NNP	7	PMOD	-	-	-	...
9	.	.	.	2	P	-	-	-	...

表 2.1. CoNLL フォーマット例

ど) は解析器によってよくされたものであることを表し、あらかじめ正解データとして与えられているものは頭に P がついていない項目として与えられる (表 2.1 では省略している) 。

- ID: 単語の ID　1 から順に番号付けされる
- Form: 単語そのもの
- PLemma: 予測された単語の標準的な表記 (三人称単数の s や過去形などを元型に戻したもの)
- PPos: 予測された単語の品詞
- PHead: 予測された dependency parsing 結果における親ノードの ID
- PDeprel: 予測された dependency parsing 結果における親ノードとの関係ラベル
- FillPred: この単語が predicate の場合は “Y” マーク、それ以外は “-”
- Pred: この predicate がどのフレームに属するか
- APredn: この単語が n 番目の predicate のどの argument に相当するか (表では 1 個のみだが、predicate が複数の場合は表の... 以降に APred2, Apred3.. と続く) (ただし基本的に argument 部分の head の単語につく)

2008 年のタスクでは ID 〜 PPos までが与えられており、dependency parsing により PHead 〜 PDeprel を埋め、さらに predicate を判別し (Y かどうかを判定) 各 predicate ごとの argument を求める。それに対して 2009 年の方では、ID　FillPred までが与えられており、純粋に SRL を行い Pred と APredn を埋めることになる。

2.2.3.2 SRL の解決方法

SRL タスクの解決方法としては dependency parsing の結果を中心に特徴量として機械学習を行うものが多く、ここでは 2009 年に全体で 2 位の精度をだした Björkelund らの手法 [5] の詳細を説明する。本研究ではこの論文で説明された SRL 解析器 [4] を利用している。

Björkelund らは、SRL 解析を次の 3 つの段階にわけている。

- Predicate Disambiguation (PD)　predicate がどのフレームに属するのかを判別する (例：“go” が “go.01” なのか、“go.04” なのかを判別)
- Argument Identification (AI)　各 predicate について、どの単語が argument となるかを判別する
- Argument Classification (AC)　AI で判別された argument が predicate の属するフレームのどのラベルに対応するのかを判別する

PD の結果を AI の入力として用い、AI の結果を AC の入力とし、最終的に AI と AC の候補についてビームサーチを行う。ビームサーチとは幅優先探索において、保存するべきノード数を削減する目的で、評価が高いノード上位 N 個のみをメモリに残す手法のことである。PD, AI, AC では対応する分類器を用いて、単語の判別、分類を行う。分類にはロジスティック回帰を用いる。

ロジスティック回帰 [3] とは特徴量 ϕ を持つデータがクラス C_1 とクラス C_2 どちらに分類されるかについて、その確率を推定する手法である。クラス C_1 に分類される確率を $P(C_1|\phi)$、クラス C_2 に分類される確率を $P(C_2|\phi) = 1 - P(C_1|\phi)$ とした時に、シグモイド関数 σ と重みベクトル $\mathbf{w}$ を使って

$$P(C_1|\phi) = \sigma(\mathbf{w}^{\mathrm{T}}\phi) \tag{2.1}$$

と表すことができる。データ全体 $(\phi_n, t_n), t_n \in \{0, 1\}, n = 1...N$ ($t_n = 1$ のときクラス C_1 に分類されるとする) での尤度関数は

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \sigma(\mathbf{w}^{\mathrm{T}}\phi_n)^{t_n}\{1 - \sigma(\mathbf{w}^{\mathrm{T}}\phi_n)\}^{1-t_n} \tag{2.2}$$

で表すことができる。この尤度関数について対数尤度を考えると、

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln \sigma(\mathbf{w}^{\mathrm{T}}\phi_n) + (1 - t_n)\ln(1 - \sigma(\mathbf{w}^{\mathrm{T}}\phi_n))\} \tag{2.3}$$

となる。この関数は交差エントロピー誤差関数と呼ばれ、この値が小さくなるように $\mathbf{w}$ を更新していけばよい。この式を $\mathbf{w}$ について微分すると

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(\sigma(\mathbf{w}^{\mathrm{T}}\phi_n) - t_n)\phi_n \tag{2.4}$$

が得られる。あとは学習率 α に従って $\mathbf{w}$ を更新することで、各データがクラス C_1 に分類される確率を推定することができる。

$$\mathbf{w} \leftarrow \alpha\nabla E(\mathbf{w}) \tag{2.5}$$

AI では各単語が argument になるかならないかの 2 値分類を行うため、上のロジスティック回帰で実現できる。一方 PD, AC では複数あるフレームやラベルから該当するものを選ぶ多クラス分類を行うことになる。

多クラスロジスティック回帰では、特徴量 ϕ を持つデータがクラス C_k に分類される確率をソフトマックス関数を用いて

$$p(C_k|\phi) = \frac{\exp(\mathbf{w_k}^{\mathrm{T}}\phi)}{\sum_j \exp(\mathbf{w_j}^{\mathrm{T}}\phi)} \tag{2.6}$$

と表すことができる。データ全体 $(\phi_n, t_n), n = 1...N$ ($t_n = (0, 0, ...1, ..., 0)$ (k 番目のみ 1) のときクラス C_k に分類されるとする) での尤度関数は

$$p(\mathbf{T}|\mathbf{w}_1, ..., \mathbf{w}_k) = \prod_{n=1}^{N}\prod_{k=1}^{K} \frac{\exp(\mathbf{w_k}^{\mathrm{T}}\phi_n)}{\sum_j \exp(\mathbf{w_j}^{\mathrm{T}}\phi_n)}^{t_{nk}} \tag{2.7}$$

であり ($\mathbf{t}$ は t_{nk} を要素とする $N \times K$ 行列) 、対数尤度は

$$E(\mathbf{w}_1, ..., \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, ..., \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln(\frac{\exp(\mathbf{w_k}^{\mathrm{T}}\phi_n)}{\sum_j \exp(\mathbf{w_j}^{\mathrm{T}}\phi_n)}) \tag{2.8}$$

であり、$\mathbf{w}_i$ について微分すると

$$\nabla_{\mathbf{w}_i} E(\mathbf{w}_1, ..., \mathbf{w}_K) = \sum_{n=1}^{N} (\frac{\exp(\mathbf{w_i}^{\mathrm{T}}\phi_n)}{\sum_j \exp(\mathbf{w_j}^{\mathrm{T}}\phi_n)} - t_{nj})\phi_n \tag{2.9}$$

となる。これにより各データの正解クラスについてそれぞれ $\mathbf{w}_i$ を更新することで、あるデータがクラス C_k に分類される確率を推定することができる。

表 2.1 の例で説明すると、PD では predicate “go” がどのフレームに属するかのクラス分類を多クラスロジスティック回帰によって行う。その結果 predicate “go” は “go.01” というフレームに該当するとする。それを受けて次の AI では、文中のどの単語が predicate “go” の argument となるかをそれぞれ 2 値分類によって判定する。これはロジスティック回帰によって行う。この時点での上位 k 個を候補として保持する。その k 個の結果について次の AC では、各候補ごとにそれぞれの argument がどのラベルに属するのかを多クラスロジスティック回帰によって判定する。その結果について上位 l 個を候補として保持すると、結果として $k \times l$ 個の候補が残ることになる。評価はすべてロジスティック回帰によって行われている為、それぞれの候補は確率として評価されている。そのため $k \times l$ 個の候補は AI と AC での確率を掛け合わせることで最終的な尤度を求めることができる。この値が最も大きいものが SRL 解析結果として出力されることになる。

最終的に学習するべき分類器は以下のようになる。

- PD 用に lemma の数だけ (“go”だったら “go”用のフレームに対応した分類器が必要)
- AI 用に 1 つ
- AC 用の 1 つ

各分類器のロジスティック回帰で使用された特徴量は、Björkelund らの方法では [18] を参考にして以下のようなものを用いている。

- PredPos: predicate の品詞
- PredDeprel: predicate の dependency parsing での親ノードとの関係ラベル
- PredParentWord: predicate の dependency parsing での親ノードの単語
- PredParentPos: predicate の dependency parsing での親ノードの品詞
- など

この論文では用意した特徴量について効率よく計算を行い精度をだすために特徴量選択を行っている。また今まで説明した方法に加えて AI と AC を同時に学習し rerank する方法を追加し精度を上げている。

第3章　人物移動モデルを用いた質問応答

3.1　人物移動モデル

自然言語を正しく解釈するためには、文章の表層的な情報だけでなく現実世界の常識、法則などを適切に扱うことで、現実世界との対応を考慮する必要があることはすでに述べた。幾つかの先行研究でも限定されたドメイン中で状態を人手によって定義する方法が提案されている。しかし実際に現実世界の状態全てを世界のモデルとして正確に表現しシミュレーションすることは、現在のコンピュータでは現実的でない。そこで今回はその簡易版として、ある人物に関する時間・空間情報を用いた「人物移動モデル」を利用することで対象となる人物に関する正しい自然言語解釈を目指す。

人物移動モデルとは、時間変化とともに対象となる人物が無向グラフ上のノードを移動するものを想定する。これを図で表すと例えば図 3.1 のようになる。このモデルを利用することで、ある人物の時間情報と位置情報が分かれば、以下に示すような時間空間情報と整合性のとれた質問応答や共参照解析、イベント抽出が可能になる。

- 質問応答例
  - α は時刻 $t = 0$ の時、A にいました。α は時刻 $t = 3$ の時、F にいました。
  - α は時刻 $t = 1$ の時、どこにいましたか？
- 共参照解析例
  - α は時刻 $t = 0$ の時、A にいました。α は時刻 $t = 3$ の時、F にいました。
  - β は時刻 $t = 2$ の時、D にいました。
  - その後、**彼**は $t = 3$ の時、C にいました。
- イベント抽出
  - α は時刻 $t = 0$ の時、A にいました。α は時刻 $t = 3$ の時、F にいました。
  - β は時刻 $t = 4$ の時、C にいました。
  - 抽出対象:α は β を探しに $t = 4$ で E に行きました。
  - 抽出結果: Who: α,When: $t = 4$, Where: E

説明を簡単にするためにあえて不自然な文章にしている。例えば最初の質問応答の例であるが、本文から α が $A \rightarrow F$ という移動を行っていることがわかり、$t = 1$ ではその中継地点である B のあたり

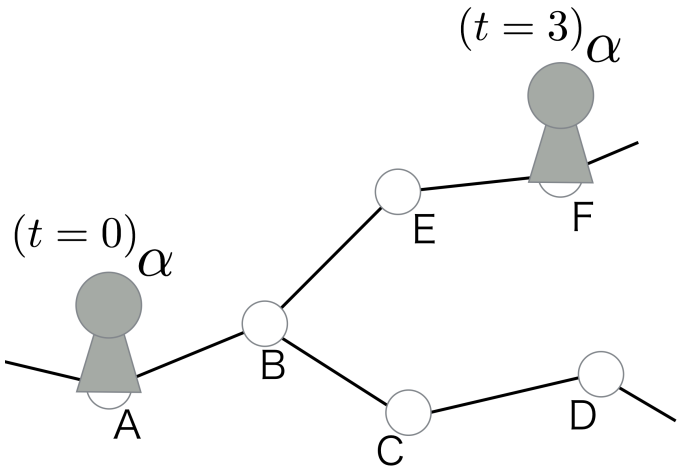


図 3.1. 人物移動モデル

にいることが推測できる。ただ、従来手法のように単語のオーバーラップなどで質問文と問題文を比較するだけでは解決することはほぼ不可能である。この推測を人物移動モデルを用いて実現することができれば、時間・空間情報に関する整合性を考えた簡単な質問応答が実現出来る。

次の共参照解析の例では、文中の “彼”が α と β のどちらを指しているかを判断することになるが、今回はどちらも男性名詞であることを仮定する。するとこの文章では α と β ではほとんど形態的な特徴に差がなく、自然言語の表層情報だけによる共参照解析は難しいと考えられる。そこでこの人物移動モデルにより時間・空間的な制約を考えることで α と β の位置関係などから “彼”が β であることが推測できる。このようにして意味を考えた共参照解析が実現出来る。

最後にイベント抽出の例では、抽出対象となる文 “α は β を探しに $t=4$ で E に行きました。”から「誰がいつどこに」に該当する Who,When,Where の要素を抽出することを考える。この短い文でも Who に該当する候補として α と β の 2 通りが考えられる。長い文になると存在する名詞句の数は増え、曖昧性が多くなる。関連研究で説明したように、従来手法では単語の情報や構文情報など、自然言語の表層情報を利用した機械学習による手法がメインであるが、この人物移動モデルを用いて時間・空間的な整合性を捉えることができれば Who に該当するのは α だとわかる (β は $t=4$ には C にいるため)。

このように人物移動モデルを利用し、ある人物に関する時間・空間的な情報を適切にモデル化することで、現実世界の出来事などに関する自然言語文を時間・空間的な側面からの整合性を考慮することで、正しい自然言語解釈を実現できる。

先行研究と異なり、対象となる人物に関する位置情報だけ得られれば人物移動モデルを構築することができる。まずはこの人物移動モデルを利用することで上に示したことが本当に実現出来るのかを確認する目的で簡単な予備実験を行った。この予備実験では対象とする人物として実在した歴史上の人物 “アレキサンダー大王”を取り上げ、彼に関する文章解釈を行った。彼を選んだ理由としては、ア

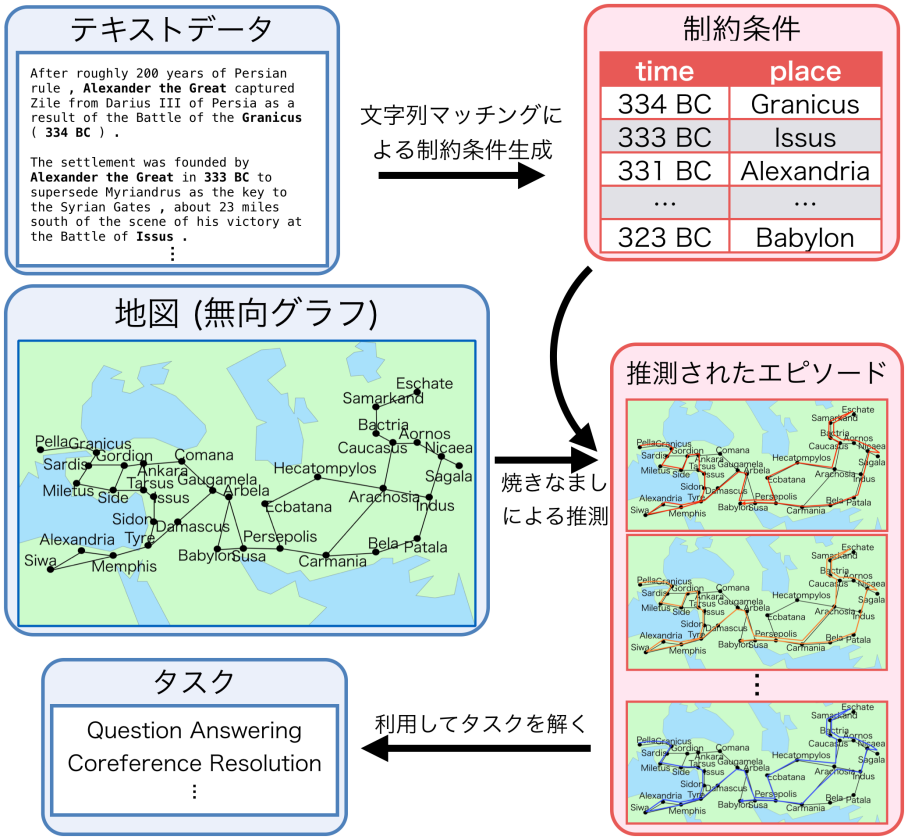


図 3.2. 提案手法　全体像

レキサンダー大王は長い時間をかけて広大な地域を遠征しており、時間・空間情報を最も利用しやすいと考えられることが挙げられる。

3.2　提案手法

実際に先に述べた人物移動モデルを用いてタスクを解決する手法について説明する。提案手法の全体像を図 3.2 に示す。手法の流れは以下の様である。

- まずは対象となる人物 (ここではアレキサンダー大王) に関する自然言語文を用意する。
- また、対象となる人物と関連のある地名を集めて生成した地図を用意する。
- 次に自然言語文から簡単な文字列マッチングとルールにより、「いつ、誰が、どこにいたか」という 3 つの情報を 1 組にした制約条件を生成する。

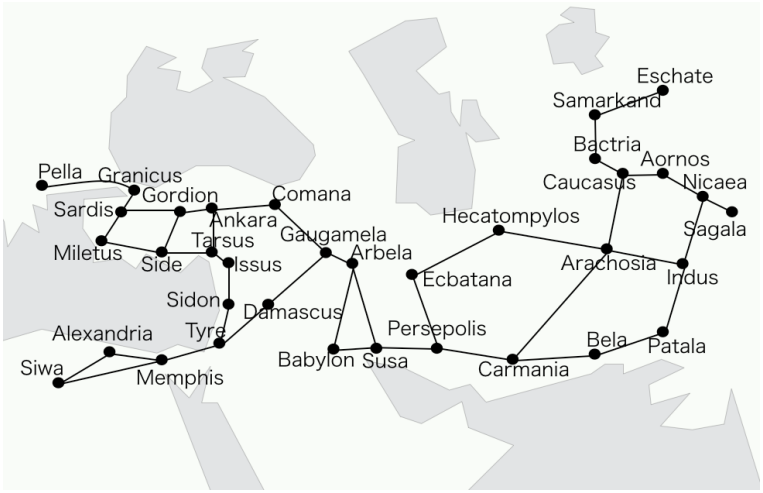


図 3.3. 人物移動モデルで利用する地図

- この制約条件を利用し、先ほどの人物移動モデルにおいて人物移動を推測する。
- その推測結果を利用して様々なタスクを解決する。

各要素について詳細を説明する。

3.2.1 地図について

対象となる人物と関連のある地名を人手により抽出し、当時の情報を参考に地図を作成する。この地図の作成については後で 4 章で述べるように自動化を行うが、準備実験の段階では人手によるものを利用する。実際にアレキサンダー大王の時代における人の移動や道を表した図[1]を参考に、無向グラフを人手により生成した。理想的には実際の地理環境（距離、高低差、山脈があるかどうかなど）を反映させるべきであるが、今回は簡単のため、グラフの各エッジの距離は全て 1 で統一した。そのグラフを図 3.3 に示す。

3.2.2 人物移動モデルの定式化

図 3.3 に示すグラフを $G = (V, E)$ とする。ここでノードの集合を V、エッジの集合を E とする。このときアレキサンダー（以下エージェントと呼ぶ）はある離散的な時間 t が進むごとに各ノード $h_t \in V$ を移動する。その際、現在のノードか隣接したノードに移動するようにする。理想的には距離、高低差、山越えなどノード間を移動する際にかかるコストは一定ではないが、今回は簡単のためにコストをすべて 1 に統一した。つまり単位時間 t の間に 1 ノードしか移動できない、あるいは移動しないこ

[1] https://en.wikipedia.org/wiki/Alexander_the_Great#/media/File:MacedonEmpire.jpg

とになる。ある一定の時間範囲 $t = 0...T$ で移動を行い、その移動の履歴 $< h_0, h_1, ..., h_T >$ をエピソードと定義する。

3.2.3 制約条件の生成

制約条件の生成は基本的に単純な文字列マッチングによって行う。例えば “Alexander the Great won a battle near Granicus river in May 334 BC.” という文があった時に生成される制約条件は「時間：334BC、位置：Granicus」となる。“Who”に関する情報はすべて対象となるエージェントに関するもので統一されているので省略する。抽出された時間情報は適切に $t = 0...T$ の範囲に変換する。ただし、ある程度時間に幅があるような表現を扱うために、範囲を指定するような設定にする。自然言語には同じ表記でも意味がことなる曖昧性があるため、正確に制約条件を生成するためには SRL を用いたイベント抽出などが必要となる。今回はこれらの処理を行わないため、ある程度間違った制約条件が含まれることになるが、全体的な移動の整合性をとることでこれらの間違った解釈をしてしまった制約条件を発見することができる可能性があることを後で説明する。

3.2.4 移動履歴の推測

人物移動モデルと生成された制約条件を利用して、エージェントの移動履歴であるエピソードを推測する方法について説明する。まずは求めるべき適切なエピソードを定義する。生成された制約条件の集合を $c_i \in C, c_i = \{t_{i,begin}, t_{i,end}, h^i\}$ とした時に、この時間情報と位置情報をできるだけ満たすようなエピソード $< h_0, h_1, ..., h_T >$ を求めるべき最適なエピソードとする。制約条件が満たされる条件は、エピソード中に制約条件を定義する時間と位置情報の組み合わせがふくまれている、つまり $h^i \in \{h_{t_{i,begin}}, ..., h_{t_{i,end}}\}$ となる時に制約条件 c_i は満たされるとする。エピソードの最適性を評価するために以下の目的関数を考える。

$$O(Episode) = \frac{|C'|}{|C|} \tag{3.1}$$

ここで $|C|$ は生成された制約条件の総数、$|C'|$ はエピソード $Episode$ によって満たされた制約条件の総数を表す。つまり満たされた制約条件の割合をそのエピソードの評価値と定義する。あとは図 3.3 に示される移動の制約下における、この目的関数 $O(Episode)$ が最大となるエピソードを求めるために焼きなまし法 [19] による最適化を行う。

焼きなまし法は巡回セールスマン問題などの組み合わせ最適化問題を解くのによく用いられる手法である。通常の山登り法は常に目的関数が大きくなるように近傍解へ遷移を繰り返すのに対して、焼きなまし法はある一定確率 P で目的関数が小さくなる近傍解への遷移も行う。P はイテレーションを繰り返すごとに式 3.2 に従って小さくなっていき、焼きなまし法の終わりの方では P がとても小さくなり通常の山登り法と同じように収束する。

$$\begin{aligned} P_t &= e^{\frac{V_t - V_{t-1}}{T}} \\ T &= C\alpha^{\frac{iter}{maxIter}} \ (0 < \alpha < 1) \end{aligned} \tag{3.2}$$

この T とは温度を意味し、イテレーションが大きくなればなるほど小さくなる (C と α は定数)。基本的に目的関数を小さくする方向に動くときに確率 P を考えるので、遷移先の近傍解の評価値 V_t より今の解の評価値 V_{t-1} の方が大きく、$V_t - V_{t-1}$ は常に負であるため、T が小さくなると P も小さくなる。これにより局所的な最適解に収束してしまい本来の最適解が得られなくなる可能性を減らすことができる。

近傍解とは今得られている解を一部改変した解候補のことで、例えば巡回セールスマン問題では隣接する都市を訪れる順番を逆にするなどの操作によって求める。

以上の焼きなまし法を用いて最適なエピソードを求める具体的なアルゴリズムを Algorithm 3.1,3.2 に示す。2 つの関数で構成されており、Algorithm 3.1 が最適なエピソードを求める関数で、Algorithm3.2 が近傍解を求める関数である。エピソードを求める関数では焼きなまし法を繰り返し使い、最適なエピソードをいくつか生成する。複数生成する理由としては以下に示す 2 つがある。

- 山登り法や焼きなまし法による最適化は、初期解にかなり依存することが知られているため、初期解を繰り返し生成する方が正確に最適化が行えるから。
- 一般的に、テキストに書かれている情報は歴史を完全に再現するほど詳細には書かれていないため、制約を満たすエピソードは複数存在する可能性があるから。

初期解については、h_0 に関する情報を与えたうえでランダムに隣接する地名へ移動、あるいは同じ場所にとどまることで生成する。アルゴリズム中の $Val(NextE)$ と $Val(currentE)$ はそれぞれ近傍解の目的関数の値と、今の解の目的関数の値を表し、目的関数は式 3.1 で定義される。

次に近傍解を求める方法について説明する。関数 GetNeighborEpisode では、以下の 4 つの操作をそれぞれ確率 50%で行うことで現在の解から近傍解を生成している。

- 同じ場所にとどまるタイミングをずらす。例えば図 3.3 では $< Ankara \rightarrow Ankara \rightarrow Tarsus \rightarrow Issus >$ という移動が $< Ankara \rightarrow Tarsus \rightarrow Tarsus \rightarrow Issus >$ という移動に変更される。
- 寄り道となる地名を挿入する。例えば図 3.3 では $< Ankara \rightarrow Tarsus >$ という移動が $< Ankara \rightarrow Gordion \rightarrow Ankara \rightarrow Tarsus >$ という移動に変更される。
- 寄り道となる移動を削除する。例えば図 3.3 では $< Ankara \rightarrow Gordion \rightarrow Ankara \rightarrow Tarsus >$ という移動が $< Ankara \rightarrow Tarsus >$ という移動に変更される。
- 複数通りある進路を変更する。例えば図 3.3 では $< Caucasus \rightarrow Aornos \rightarrow Nicaea >$ という移動が $< Caucasus \rightarrow Arachosia \rightarrow Indus \rightarrow Nicaea >$ という移動に変更される。具体的にはグラフ全体からループを形成するノード集合を選択し、そのループ中の移動の向きを逆向きに変更する。

このような方法によって、制約条件をできるだけ満たすエピソード (図 3.2 の feasible episode) を複数生成する。この生成されたエピソードは、入力として与えられた自然言語文で表現されるエージェントの時間・空間の部分的な情報を全体的に解釈した結果となる。

Algorithm 3.1 制約条件を満たすエピソードの探索アルゴリズム

```
1: function findFeasibleEpisodes(maxR,maxIter,α)
2:     feasibleEpisodes ← {}
3:     for round = 1 to maxR do
4:         currentE ← getRandomEpisode()
5:         bestE ← currentE
6:         for iter = 1 to maxIter do
7:             nextE ← getNeighborEpisode(currentE)
8:             if Val(currentE) < Val(nextE) then
9:                 currentE ← nextE
10:                if Val(nextE) > Val(bestE) then
11:                    bestE ← nextE
12:                end if
13:            else
14:                temperature ← Cα^(iter/maxIter)
15:                                          ▷ 0 < α < 1 : α is constant
16:
17:                if rand(0,1) ≤ e^((Val(nextE)−Val(currentE))/temperature) then
18:                    currentE ← nextE
19:                end if
20:            end if
21:        end for
22:        feasibleEpisodes.insert(bestE)
23:    end for
24:    return feasibleEpisodes
25: end function
```

Algorithm 3.2 近傍解を求めるアルゴリズム

```
1: function getNeighborEpisode(currentEpisode)
2:     e ← currentEpisode
3:     if rand(0, 1) < 0.5 then
4:         p1, p2 ← getConsecutiveSameStates(e)
5:         e.remove(p2)
6:         p3 ← getRandomState(e)
7:         e.insert(p3)                                  ▷ at next p3
8:     end if
9:     if rand(0, 1) < 0.5 then
10:        p1 ← getRandomState(e)
11:        p2 ← getAdjacentState(p1)
12:        e.insert(p2, p1)                              ▷ at next p1
13:    end if
14:    if rand(0, 1) < 0.5 then
15:        p1, p2, p3 ← getDetour(e)                     ▷ p1 = p3
16:        e.remove(p2, p3)
17:    end if
18:    if rand(0, 1) < 0.5 then
19:        loop ← getRandomLoop(G)                       ▷ G is the graph
20:        if loop contains some state in e then
21:            e.reverseInLoop(loop)
22:        end if
23:    end if
24:    return e                                          ▷ as a neighbor episode
25: end function
```

3.2.5 人物移動モデルによる質問応答

先ほどまでの説明で得られたエピソードを利用して時間・空間情報に関する質問応答を行う方法を説明する。ここでの質問応答は以下のような時間情報から位置情報を当てる例を想定する。

- アレキサンダー大王は 330BC にどこにいましたか？

この質問に答えるためには入力として与えられた自然言語を解釈した結果である、エピソードを参照する必要がある。具体的には得られた複数のエピソードのうち、330BC にエージェントがいた地名の出現頻度によって求めることができる。もちろん、与えられた入力文に 330BC にどこにいたかを明記する文が含まれていれば、従来の単語のオーバーラップなどの方法で解けるかもしれないが、直接そういった文が含まれていない場合、この質問に正しく答えることは不可能であると考えられる。また、この質問に正しく答えることができれば、自然言語文で表現されるエージェントの移動を正しく再現できている、つまり時間・位置情報に関して正しい解釈ができているといえる。

3.3 実験

3.3.1 コーパスと実験設定

対象とするエージェントはアレキサンダー大王のみとする。人物移動モデルで使用する地図は図 3.3 を用いた。使用するテキストデータには英語の Wikipedia データすべて[2]の本文部分のみを抽出し使用した。事前処理として、“Alexander the Great” と “BC” という二つの文字列が含まれている文を抽出したところ 482 文を取得した。これらのうち、図 3.3 に示される地名を含むものは 87 文あり、そこから 3.2.3 節で説明したように文字列マッチで制約条件を生成した。同じ時間、同じ地名を含む制約条件は重複しているため同一の制約条件 1 つと見なした結果、39 個の制約条件が生成された。このうち人手によるアノテーションによって 32 個正しいものが含まれていることがわかった。エピソード生成の元となるシミュレーションの設定は以下のように設定した。

- 初期位置は “Pella” で固定 ($h_0 = Pella$)
- 現実時間にして 2 ヶ月を 1 ステップ（単位時間）とする
- 1 ステップごとにエージェントは今のノードか隣接するノードに移動する
- 各エピソードは 72 ステップで構成される（実際のアレキサンダー大王の遠征期間と一致）

焼きなまし法については、式 3.2 を用い、式中の定数には $C = \frac{1}{30}, \alpha = 0.001$ を用いた。ランダムな初期解を与え、制約条件をみたすエピソードを得ることを 1,000 回繰り返し (Algorithm 3.1 中にて $maxR = 1,000$ とした)、1,000 通りのテキストと整合性のとれたエージェントの移動を取得することにする。

[2]2013 年 11 月にダウンロード

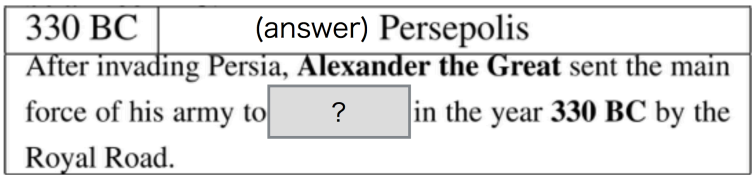


図 3.4. 質問応答の問題例

3.3.2 質問応答の問題生成

はじめに説明したようにテキストに直接書かれていない内容に関する質問に回答することは従来の自然言語処理では難しい。テキストに書かれていること以外の情報を適切に利用する必要があるからである。今回はテキストから直接的に読み取れる情報だけでは決して解けない問題を用意して、モデルを使用して解くことを行った。各問題は 3.2.5 節で説明したように、時間情報とエージェントの情報から地名情報をあてることを想定する。この問題を生成するために、先ほど説明した Wikipedia データから生成した制約条件のうち、時間情報と位置情報の組み合わせが人手によるアノテーションで正しいことが保証されている 32 個について、位置情報を隠すことで質問応答の問題とした。その問題例を図 3.4 に示す。図 3.4 は制約条件 $(h = Persepolis, t = 330BC)$ から生成された問題で、表示されている文は制約条件生成の元となった自然言語文である。

システムが問題に回答する際に、テキスト自体に問題の直接的な答え（この例の場合では Alexander the Great が 330 BC に Persepolis にいたという内容) が含まれる文があると、従来手法でも解答可能だと考えられるが、それは今回の趣旨に反する。そこで、それぞれの問題を解く際に、入力されるテキストから問題に関する直接的な情報を削除して実験を行った。具体的には例えば、この問題例に解答する場合では、制約条件 $(h = Persepolis, t = 330BC)$ が生成されるような文をあらかじめ削除した。これにより、直接的な情報がなくなるため、周りの制約条件と地図の情報から推測することで回答する必要がある、つまり従来手法では解けなくなる。

問題の回答方法は、シミュレーションによって得られた 1,000 個のエピソードについて、時間情報（この例の場合は 330 BC に相当する時間）に出現する地名の頻度を求め、その値が高いほど正解の地名であると判断する。解候補となる地名は図 3.3 に使用されているものとする（35 個)。実験では上位 N 個の正解候補を求め、その中に真の正解が含まれていれば正しく回答できたものとする評価方法とした。また、焼きなまし方の最大イテレーション数を変えながら実験を行った。

3.3.3 実験結果と考察

3.3.3.1 エピソードの推測

図 3.5 にアレキサンダー大王の実際の移動履歴を、図 3.6 に人物移動モデルと入力された自然言語文の内容から推測したエピソード例を示す。両方の図から多少の違いはあるが、ほぼ正確にアレキサン

図 3.5. エージェントの正しいエピソード

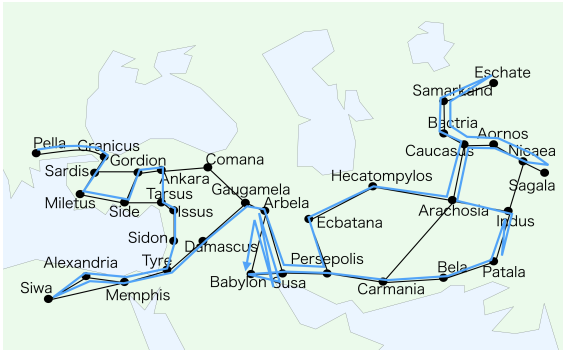


図 3.6. エピソードの推測結果

ダー大王に関する移動を再現することに成功している。このことから自然言語文の内容のうち時間・空間的な情報に関して正しく解釈できていることがわかる。

3.3.3.2 質問応答の結果

結果を図 3.7 に示す。横軸が上位何個の正解候補を出力したかを示し、縦軸が全問題の正答率を示す。各色の線は焼きなまし法における最大イテレーション数を示す。点線で示されているものはベースラインの結果を示しており、従来の自然言語の表層的な情報を用いた方法がどれだけ正解できるかを表している。このベースラインでは、求められた制約条件と地図から、できるだけ時間情報、位置情報が近いものを選択するようにしている。例えば、“330BC”という時間情報に関する問題の場合は、生成された制約条件のうち、時間的に近い条件が含む地名について上位 N 個を選択するものとする。ただし、正解そのものの情報はテキストから削除されているため、この方法で正解できることはほとんどなく、その正答率はチャンスレートとほぼ変わらないものとなった。

イテレーション数を増やせば全体の正答率が上がっていることから、より詳細なシミュレーションを行うほど、テキストとより整合性の取れた移動を把握することができることがわかる。$N = 5$ あたりの結果を見てみると正答率は 6 割近くになっており、直接的な情報がなくとも周りの制約条件と地図データから、テキストに書かれていない間接的な情報をある程度正しく把握できていることがわかる。しかし、$N = 1$ の精度は 10%を下回っているため、正解の地名そのものを当てることは極めて難しいことがわかった。これは、その周辺の直接的な情報に影響を受けるためだと考えられる。

以上の結果から、的確に問題に回答することは困難であるが、なんとなくこの周辺にエージェントが存在しているのではないか、という推測は可能であることがわかった。このことから自然言語で直接的に表現される内容だけでなく、文章全体における時間・空間的な情報の流れをある程度把握できることがわかった。

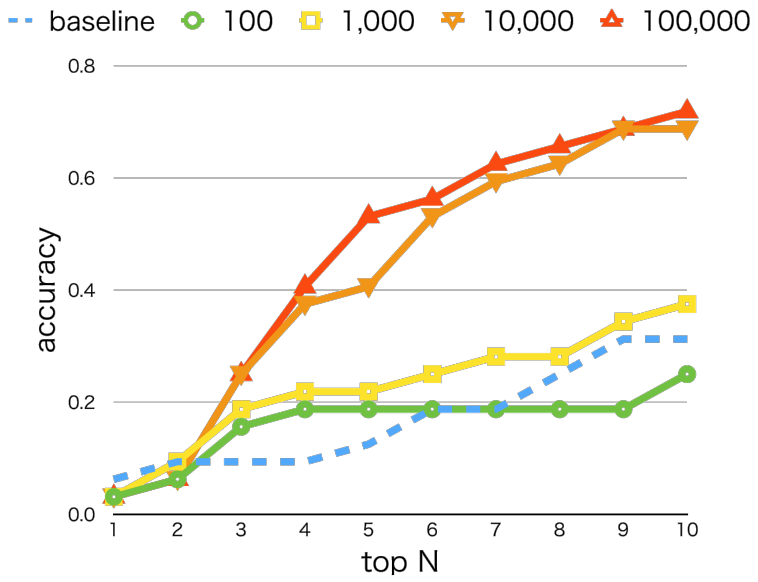


図 3.7. 質問応答の実験結果

3.3.4 他タスクへの応用

この実験では背後にある意味を考えないと解けない質問応答をタスクとして扱った。しかし 3.1 節で説明したように別のタスクでも時間・空間情報の整合性に関する情報を利用することはできる。

ここではテキストから生成された制約条件のうち、各エピソードが満たすことができなかった制約条件について考察する。満たされなかった制約条件と元の文章のペアを以下に示す。

- 制約条件: 334 BC, Alexandria
- 本文: The port of **Alexandria**, founded by **Alexander the Great** in **334 BC**, was a hub for Mediterranean trade for centuries.

- 制約条件: 323 BC, Memphis
- 本文: Arrhidaeus, one of **Alexander the Great**'s generals, was entrusted with the conduct of Alexander's funeral to **Egypt** in **323 BC**.

1 つ目の文は、"Alexandria" という単語をエジプトにあるアレクサンドリアという都市と解釈して生成された制約条件であるが、これを満たすエピソードは得られなかった。この原因は、"Alexandria" がエジプトのアレクサンドリアという解釈そのものが間違っていることにある。当時「アレクサンドリア」と名のつく都市はたくさん存在しており、その曖昧性をうまく捉えられてなかったことがこの制約条件からわかる。2 つ目の文は、アレキサンダー大王が 323 BC にエジプトにいたと解釈されて生成

された制約条件であるが、文をよく見てみると主語がアレキサンダー大王の部下のアリダエウスであることがわかる。

このように単純な文字列マッチだけでは間違った解釈をした制約条件が生成されてしまうことがあるが、空間・時間的な制約を用いて文を解釈しようとすることで、このような誤りを検出できる可能性があることがわかった。このように整合性を考慮することで後で述べるように共参照解析やイベント抽出の精度を改善できる可能性がある。例えば共参照解析の場合、“He”を誰かのエージェントと解釈した場合について、その人物の制約条件が満たされなかった場合は、その共参照関係が間違った解釈であることがわかる。イベント抽出の場合、イベント抽出を利用して制約条件を生成すれば、それが満たされなかった場合にその解釈が間違っていたことがわかる。以降はこれらのタスクについて時間・空間的な整合性を利用する手法について説明する。

第4章　時間空間情報を用いた共参照解析

4.1　準備実験

3章の最後で述べた人物移動モデルを用いて共参照解析の精度を改善できる可能性があることについて説明する。まず、表4.1に示すような例を考える。

この問題では “the area”が、その他の太字で示される4つの地名 “Bela”, “Arachosia”, “Carmania”, “Babylon”のうちどれを指すのかを判断する（この問題の正解は “Bella”）。この4つの地名は形態的な特徴の代表である「男性名詞か、女性名詞か」「単数か、複数か」などの情報はすべて等しく、またすべて地名であるため、周りの単語の出現頻度など従来の手法ではうまく処理することができないと考えられる。この問題を解決する方法として、文中に存在するエージェントの情報である “Alexander the Great”と時間情報である “325BC”に着目する。先述の質問応答と同じように推測されたエピソード中においてエージェントが “325BC”にいた地名の出現頻度を求め、候補となる先行詞から出現頻度が最も高いものを選択することで、時間・空間的な意味を捉えた共参照解析が可能になると考えられる。

3章の実験で得られた1,000個のエピソードを用いて実験を行ったところ、表4.2のような結果が得られた。表中の数値は取得した1,000個のエピソードのうち、“325BC”に先行詞の候補となる地名が出現しているエピソードの数を示す。また2行目の1,000 10,000 100,000という値は、焼きなまし法の最大イテレーション数を表す。この結果から正解の先行詞である “Bela”を正しく選択することができており、自然言語文において時間・空間的な整合性を考慮した共参照解析が可能であると言える。

しかし、準備実験の手法には以下のような問題点がある。

時間情報	325 BC
照応詞	the area
正しい先行詞	Bela
他の先行詞候補	Arachosia, Carmania, Babylon

Bela is directly to the south of the ancient provinces of **Arachosia** and Drangiana, to the east of **Carmania** and due west of the Kingdoms of Ancient India. In **325 BC**, **Alexander the Great** crossed **the area** on his way back to **Babylon** after campaigning in the east.

表 4.1. 共参照解析例

先行詞候補	最大イテレーション数		
	1,000	10,000	100,000
Bela (正解)	**247/1,000**	**547/1,000**	**745/1,000**
Carmania	210/1,000	454/1,000	640/1,000
Arachosia	154/1,000	404/1,000	651/1,000
Babylon	35/1,000	8/1,000	1/1,000

表 4.2. 人物移動モデルによる共参照解析結果例

1. アレキサンダー大王に関する文章しか扱えない
2. 問題を解くのにかなりの時間がかかる
3. 制約条件生成が簡単な文字列マッチのため正確でない可能性がある

まず一つ目の問題点であるが、これは地図を生成する過程を人手によって行っていることが原因である。そのためこの問題を解決し、多くの歴史上の人物をエージェントの対象として扱うためには、その人物に関する地名を自動的に集取し、地図を自動生成する必要がある。

二つ目の問題点について、まず焼きなまし法を用いて 100,000 イテレーションで 1,000 個のエピソードを生成するのに 8 コアで約 30 分ほどの計算時間を要する。このため多くの一般的な人物を扱えるようになったとしても共参照解析を行うためにはあまりに多くの時間がかかってしまう。

これらの問題を解決するためにこの章では、人物移動モデルを簡略化することにした。今まで焼きなまし法を用いてシミュレーションにより時間・空間的な整合性を考えていたが、時間・空間制約を用いたルールによって解決することにする。具体的にはテキストから生成された制約条件、対象となる共参照解析に関する制約条件、それぞれに関する時間・空間情報を地図上にマップし、それらの時間の差、距離の差を利用してルールから判断することにする。詳細は 4.4 節で説明する。また、対象となるエージェントに関連する地名は Wikipedia の Infobox から自動的に緯度経度を抽出することで地図上に表現することにする。これについては 4.2 節で詳細を説明する。

三つ目の問題点について、自然言語文から制約条件を生成する方法であるが、関連研究でも述べたように SRL とルールによるイベント抽出を行うことで生成する方法をとることにした。詳細は 4.4 節で説明する。

4.2　扱うエージェントの一般化のための拡張

対象となるエージェントが与えられた際に、そのエージェントに関連する制約条件をもとに自動的に地図を生成する手法を説明する。

この章で扱うモデルでは、人物の移動をシミュレーションすることはせず、与えられた制約条件の時間・空間情報をルールで解決する。そのためには以下の 2 つが必要である。

- 対象となるエージェントに関連する地名を自動的に取得する方法
- 各制約条件のもつ空間情報、つまり地名の位置情報を自動的に取得する方法

まずエージェントに関連する地名を自動的に取得する方法であるが Wikipedia のリンク情報をもとに行うことにする。具体的には英語版 Wikipedia の xml ファイルを利用する。この xml ファイルには InfoBox や Persondata など、地名の位置情報や人物のデータが含まれている。テーマの中心となるエージェントをアレキサンダー大王とし、“Alexander the Great”の記事から幾つかハイパーリンクを経由することで得られるページのうち、Persondata を持つものをエージェント、経度緯度情報をもつものを地名として採用することにした。

次に各制約条件の地名の位置情報を自動的に取得する方法であるが、先ほど得られた地名に関する Wikipedia のページには経度緯度情報が載っているので、それを利用して地図上にマッピングすることにする。

4.3 タスク

今までの説明を踏まえ、時間・空間情報を用いた共参照解析に関するタスクについて説明する。この章では表 4.1 に示すような一般的な共参照解析ではなく、代名詞 “He”に関する共参照解析を扱うことにする。これは対象となるエージェントを複数扱えるようになったことから、“He”に関する曖昧性を除去することができるのではないかと考えられたからである。ただ “He”が誰であるかを的確に当てることが難しいことは 3 章で説明した通りなので、このタスクでは時間・空間情報をつかった整合性の考慮により、既存の解析器による結果の正誤を判定することを考える。ある共参照解析器が出した共参照関係に注目し、“He”の解釈のうち誤っているものを除去することで共参照解析の精度の向上を図ることがこのタスクの目標である。

実際に Stanford の共参照解析器 [20] により、対象となるテキストのすべての “He”について共参照関係の代表としてエージェント名を求め、そのうち誤っているものを分類する 2 値分類問題としてタスクを生成した。解析結果についてはすべて人手により、その正誤判定ラベルを付与した。

4.4 提案手法

4.4.1 概要

自然言語の意味を捉える際に、背後にある世界の常識・法則などの一部として人物移動における空間・時間的な制約を考える。これは例えば「ある人物は、短い期間中に離れた場所に存在できない」などの基本的なルールを利用する。これらの情報は文の表層情報には現れないため、これを利用することで共参照解析を適切に処理することを目指す。

この章では Wikipedia における人称代名詞 “He”に関する共参照解析の精度改善を行う。既存の解析器の共参照解析結果を、あるエージェントに関する時間と空間の制約を利用し “He”の先行詞が不適切だと判断できる候補を除去することで精度向上をめざす。

扱う内容は世界史 (主にアレキサンダー大王の時代) をテーマにする。具体的な手法を以下に示す。

- 対象とするエージェント (人物) 、地名を Wikipedia を用いて選択
- 対象とするエージェントと関連があると考えられる Wikipedia 本文の代名詞 “He”について共参照解析 (照応解析) を既存ツールで行う
- 対象とするエージェント名が明記された文章から制約条件 $c_i \in C$ (いつ誰がどこにいたか) を生成する
- “He”の解析結果からエージェント名を置換し制約条件 $c'_i \in C'$ を生成し、結果が適切であるかどうかを制約条件の集合 C から判断する

例えば、“(1) Alexander found Alexandria in 332 BC.” “(2) He invaded India in 333 BC.”というテキストがあり、既存の共参照解析器により “He”が “Alexander”を示す結果が得られた場合は、(1) と (2) からそれぞれ制約条件 c_1 $(agent = Alexander, location = Alexandria, time = 332BC)$ と c'_1 $(agent = Alexander, location = India, time = 333BC)$ が生成され、Alexandria と India の距離と時間条件から共参照解析結果が妥当か判断する。

4.4.2 制約条件生成

制約条件は「いつ、誰が、どこにいたか」という情報をもつ。制約条件は “Alexander the Great”のページから幾つかのハイパーリンクで繋がるエージェントのページすべての本文から生成することにする。関連研究で紹介したように、各文について SRL を行い、その結果とルールによりイベント抽出を行うことで制約条件を生成する。例えば “He invaded India in 333 BC.”の解析例は、predicate“invade.01” (A0 (invader) : He, A1 (place invaded) : India, AM-TMP (time) : 333 BC) となる。SRL によって付与されたラベルは各 predicate の属するフレームごとに定義される “A0,A1,...,A5”などと、共通の修飾用のラベル “AM-TMP,AM-LOC など”がある。これらのラベルは “A0 や A1 は主語になりやすい”、“時間修飾は AM-TMP”、“場所修飾は AM-LOC”というように役割がアノテーションガイドライン [1] で決められている。これらの要素からルールを用いてエージェントが 333 BC にどこにいたか、という情報に関する制約条件が生成できる。その際、本文中の表現がどのエージェント、どの地名に対応するかについては文字列の一致度合いをスコア化することで決定する。

4.4.3 共参照解析の誤り検出

本文中にエージェント名 f が明記された文から生成された制約条件の集合 C と、エージェント名が “He”であり、既存の共参照解析器で “He”が f と判断され生成された制約条件 $c'_i \in C'$ から、c'_i に関する “He”が表すエージェントが適切であるかを判断する。

まず Wikipedia から得られた地名の経度緯度情報をもとに地図上に $c_j \in C$ $(j = 1..|C|), c'_i$ をそれぞれ点 $P_{c_j}, P_{c'_i}$ としてマッピングする。次にその 2 点間の距離 $L = |P_{c_j} P_{c'_i}|$ と時間条件の差 Δt を求める

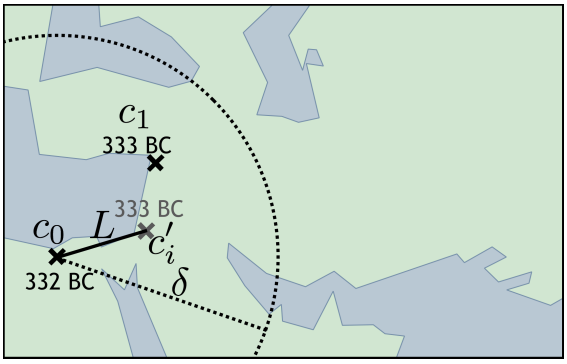


図 4.1. 適切な共参照解析だと判断する例

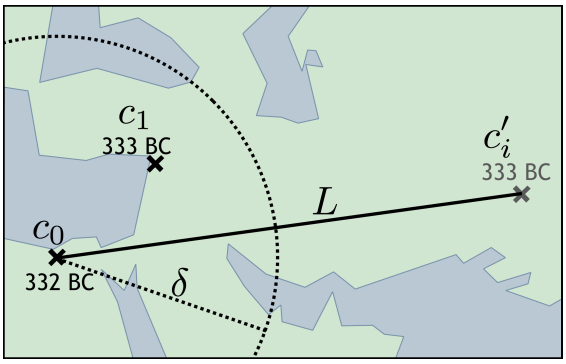


図 4.2. 不適切な共参照解析だと判断する例

(例えば 332 BC と 331 BC なら $\Delta t = 1$) 。それらを用いて制約条件 c_i' に関するスコアを以下のように定義する。

$$Score(c') = \frac{\sum_i Val(c_i)}{|C|} \tag{4.1}$$

$$Val(c_i) = \begin{cases} 0 & (\text{if } L <= (1+\Delta t) \times \delta) \\ -1 & (\text{otherwise}) \end{cases} \tag{4.2}$$

$$L = r \arccos(\sin(lat_1)\sin(lat_2) + \cos(lat_1)\cos(lat_2)\cos(long_2 - long_1)) \tag{4.3}$$

ここで δ はエージェントが 1 年間に移動できる限界値を表す閾値とする。L は地球の半径 $r = 6378.137(km)$ と 2 地点の緯度 (lat (rad)) 経度 (long (rad)) から式 4.3 を用いて計算される。最終的にこのスコアが全体的な閾値 θ を上回れば制約条件として適切、つまり “He”の共参照解析は適切であると判断する (図 4.1) 。逆に下回れば “He”の共参照解析結果が不適切であると判断する (図 4.2) 。なお、$|C| = 0$ となる場合については判断材料がないため適切と判断する。

4.4.4　評価手法

共参照解析では精度を確かめるために様々な評価手法が提案されている。共参照解析では文中の名詞句がどのエンティティを指すかを判断することになるが、その分類のされ方によって評価手法も異なる。例えばペアワイズスコアは、文中の各名詞句間どうしのすべてのペアについて、共参照関係があるかどうかを 2 値分類する際の F 値を算出する。

$$F1 = \frac{2PR}{P+R} \quad (P : \text{再現率}, R : \text{適合率}) \tag{4.4}$$

F 値とは適合率 (precision) と再現率 (recall) から算出される値である。適合率とはシステムが適切だと判断した事例のうち真に適切だったものの割合を表し、再現率とは全体の真に適切な事例のうちシ

ステムが適切だと判断できた事例の割合を表す。一般的に適合率と再現率はトレードオフの関係になっていることから、その調和平均を表す F 値が利用されることが多い。F 値の式を以下に示す (適合率と再現率の平均の割合が等しい時 F1 値と呼ばれる)。

ペアワイズスコアは各名詞句間ペアを独立に扱っていたが、MUC スコアや B^3 スコア,CEAF スコアは同じエンティティを指す名詞句の集合を単位としてその分類精度を F 値で評価する手法もある。このように共参照関係の分類精度を直接評価する手法が一般的であるが、本研究では既存の共参照解析器の出力結果から時間・空間的な整合性がまちがっているものを除去することで精度向上を図ることが目的のため、既存の解析器に対する相対的な F 値で評価することにした。つまり解析器の出した結果のうち “He”が正しいエージェントに結びついているものを正確に分類する分類問題を F 値で評価することになる。その際の適合率と再現率は

- 適合率: システムが正しいと判断した既存の解析器の結果のうち、真に正しかったものの割合
- 再現率: 既存の解析器の真に正しい結果のうち、システムが正しいと判断したものの割合

で定義される。

4.5 実験

対象とするテキストデータ中の “He”が誰を表しているのかを Stanford の共参照解析器で求め、その精度 (F1 値) を時間・空間制約を利用し改善する実験を行った。

4.5.1 対象とするデータ

対象の中心的なエージェントとして “Alexander the Great”を選択する。対象となるエージェントは、Wikipedia の xml データ[1]のハイパーリンクを “Alexander the Great”から 2 つたどり到達する全ページのうち Persondata の項目が含まれるページの人物 (1,469 人) とする。対象となる地名は、同じくリンクを 2 つたどり到達する全ページのうち、経度緯度情報を持つページの地名 (6,579 箇所) とする。制約条件生成のもととなるテキストデータについては、対象となるエージェントの全ページの本文 (119,882 文) とする。

4.5.2 制約条件生成

制約条件はエージェント、地名、時間の要素を持ち、例えば “Alexander found Alexandria in 332 BC.” というテキストから “agent=Alexander, location=Alexandria, time=-0331” という制約条件が生成される。“-0331” という表記は Chang らの SUTime [10] という時間表現抽出システムの定義によるものである。制約条件生成の具体的な実装とルールを以下にまとめる。

[1] 2015/9/1 にダウンロード

実装方法について

- Stanford Core NLP [22] によりテキストに対して tokenize を行う
- Bohnet [6] らと Björkelund [5] らのパイプラインツール[2]を用いて SRL を行う
- Chang らの SUTime [10] により時間表現を抽出し、SRL 結果とルールにより制約条件を生成する
- Stanford Core NLP [20] によりテキストの共参照解析を行う

SRL 結果による生成ルール

- ある動詞 (predicate) の SRL 結果ごとに制約条件を生成する
- “A0”ラベルが付いた文字列をエージェントとする
- “AM-TMP”ラベルが付いた文字列を時間情報とする
- “AM-LOC” > “A4” > “A3” > “A2” > “A1” のラベル優先度で位置情報とする
- SUTime による時間表現が “-NNNN”で表現されないものは除去する

“-NNNN”の形式で表現される時間表現は紀元前何年という情報を持ち、今回はこれに限定した。またエージェントと位置情報についてはあらかじめ決定されたエージェントリストと地名リストの中から文字列間の一致度合いをスコア化しその最大のものを選択する。具体的には (全体一致) $\times 1000 +$ (単語一致) $\times 50 -$ (文字列長の差) の重みで算出する。 今回はエージェント、地名ともにスコアが 40 以上となるものを制約条件生成の条件とした。その結果生成された制約条件は 1,214 個で、そのうち “He” を含むものは 78 個生成された。この 78 個について人手によるアノテーションを行い、Stanford の共参照解析器による結果の正誤についてラベルを付与した。この 78 個の問題に対してシステムがそれぞれスコアを 4.4.3 節で述べた手法で算出し、その値が 0 以上ならば Positive、0 未満ならば Negative とする。これにより、空間・時間的な制約を満たせないような共参照解析結果を Negative として除去することで Stanford の共参照解析結果を改善することができる。ただし、Stanford の共参照解析器が出力した正しい候補を除去する場合もあるため、本実験では 4.4.4 節で述べたように、相対的な F 値によって評価する。

4.5.3 実験結果

エージェントが 1 年間に移動できる限界値を表す閾値 δ を変化させた時の “He”に関する共参照解析の F1 値を表 5.1 と図 4.3 に示す。

まず $\delta = 10,000$km (制約をかけない場合) から本実験の Stanford Core NLP による共参照解析の精度はおよそ 7 割程度であることがわかる。形態素解析や構文解析などのタスクの精度が 9 割を超えるものがあるのに対して、共参照解析の精度は一般的に 7 割程度とされている。F1 値が最も高かったのは $\delta = 2,500$km 付近で、Stanford の共参照解析器による F1 値をわずかであるが改善する結果となっ

[2]https://code.google.com/archive/p/mate-tools/からダウンロード可能

δ(km)	0	200	400	600	...	2,800	3,000	3,200	3,400	...	10,000
True positive	0	29	34	37	...	47	48	48	48	...	50
False positive	0	11	11	11	...	18	18	18	19	...	21
True negative	50	21	16	13	...	3	2	2	2	...	0
False negative	21	10	10	10	...	3	3	3	2	...	0
Precision	-	0.725	0.756	0.771	...	0.723	0.727	0.727	0.716	...	0.704
(Relative) Recall	-	0.580	0.680	0.740	...	0.940	0.960	0.960	0.960	...	1.000
F1	-	0.644	0.716	0.755	...	0.817	**0.828**	**0.828**	0.821	...	0.826

表 4.3. δ を変化させた時の F1 値

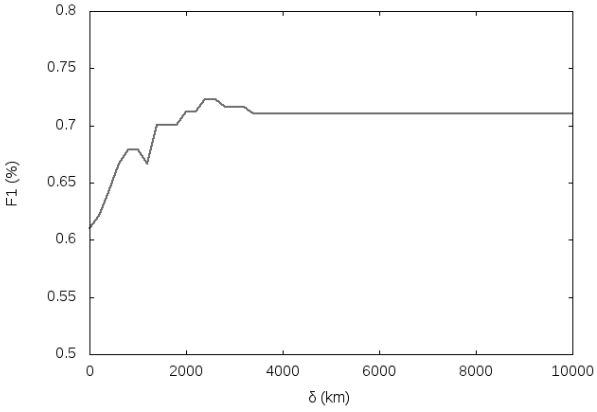


図 4.3. δ を変化させた時の F1 値

た。この 3,000km という値は人間の 1 年間に歩く平均距離が 1,800km であることから現実世界とそれほど矛盾しない結果となった。このことは現実の空間・時間的な制約をモデル化して用いていることから説明できる。

False negative の行は、Stanford Core NLP が間違えた結果について、正しく誤りを検出できていることを示す。具体的にどういう誤りをしているのか例を以下に示す。

- “After the death of Alexander the Great, Perdiccas expelled the ⋯ to leave. **He** then founded a school in Lampsacus before returning to Athens in 306 BC where he remained until his death.”

この例では “he” を Alexander だと解釈しているが、Alexander の死後のことについて書かれているため、明らかに共参照解析が間違っていることがわかる。このような誤りを時間・空間的な制約を利用することで除去することができた。

この共参照解析で精度があまり改善しなかった理由のひとつに制約条件生成の精度、つまりイベント抽出の精度が良くないことがあげられる。文章から “いつ誰がどこに” という情報を取得する際、時間空間的な整合性がおかしな解釈が起きてしまうことがある。そこで次の 5 章ではこのイベント抽出の改善について説明する。

第5章　人物移動モデルを用いたイベント抽出

5.1　タスク

共参照解析では自然言語文からイベント抽出を行うことで制約条件を生成したが、そのイベント抽出自体の精度が良くない可能性について述べた。そこで本章ではイベント抽出に関して人物移動モデルを用いた改善について説明する。例えば以下の文についてイベント抽出することを考える。

- It was originally thought to have been the sarcophagus of Abdalonymus (died 311 BC), the king of Sidon appointed by Alexander immediately following the battle of Issus in 331.

この例は Wikipedia の “Alexander the Great”の記事から持ってきた文である。実際に Bohnet [6] らと Björkelund [5] らのパイプラインツールと Stanford の NER、簡単なルールを利用して上の例についてイベント抽出を行ったところ、例えばアレキサンダー大王というエージェントに注目すると以下のような抽出結果となる。

- Who: Alexander the Great
- Where: Battle of Issus (歴史的な戦争などは経度緯度情報が付いていることが多く、地名として扱っている)
- When: 311BC

この結果は誤りで、311BC は “Abdalonymus”という人物と結びつく情報である。このように自然言語の表層情報だけでは間違った構文解析や、間違った SRL 解析結果により、最終的に現実と整合性の取れないイベント抽出を行ってしまうことがある。そこで、本研究では人物移動モデルを利用し時間・空間的な整合性を考慮することで誤ったイベント抽出を除去することにより、その精度改善に挑戦する。

5.2　提案手法

5.2.1　概要

ある人物についての自然言語文から 5W1H に相当する情報を抽出する際に、その人物の移動に関する空間・時間的な整合性が取れているかどうかを考慮した方法を提案する。4 章の共参照解析の際の距離と時間から計算する方法とは異なり、今回はイベント抽出によって生成された制約条件が人物移動のシミュレーションによって満たされたかどうかで判断する。また 3 章ではシミュレーションを繰り

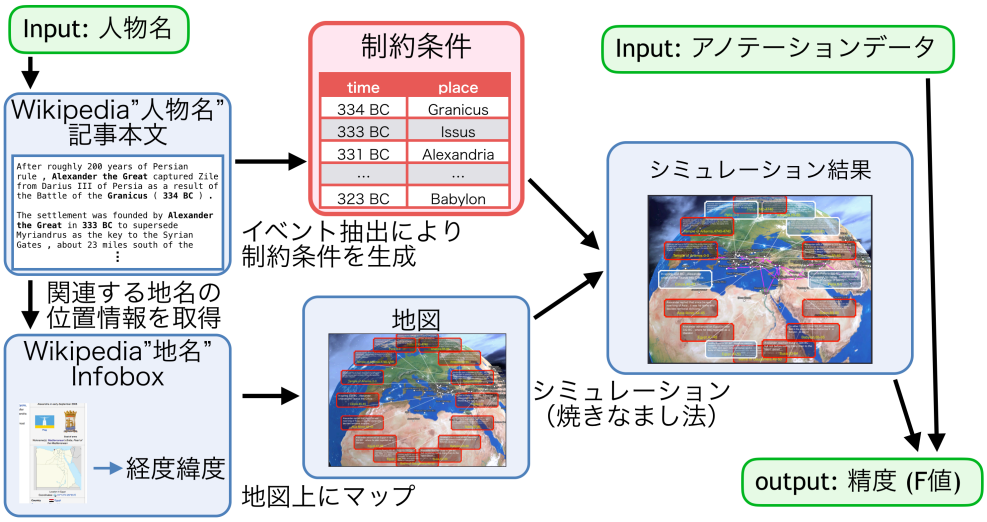


図 5.1. イベント抽出　提案手法　全体図

返すことでエピソードを大量に生成していたが、計算時間のことを考えシミュレーションは 1 回とし、実験を繰り返して精度の平均を求めることにした。これはモデルで利用する地図自体も Wikipedia の情報を利用することで自動生成するよう改変することにより、準備実験で扱ったものよりも規模が大きくなっていることも理由である。制約条件は従来手法の SRL と NER、ルールを用いて生成したものをベースラインとし、このうちシミュレーションで満たされなかった制約条件を不適切なものとして除外することで、イベント抽出の精度向上を目指す。実際に “Alexander the Great”, “Augustus”, “Napoleon” など歴史上の偉人についての Wikipedia テキストでイベント抽出の実験を行った。以上全ての機能をまとめた全体像を図 5.1 に示す。

5.2.2　地図について

基本的には 3.2.1 節で述べた方法と同様である。今回は対象となる人物のページからいくつかリンクをたどることで到達する経度緯度情報をもつページを地名とし、それらを地図上にマッピングした後、それらのうちの 2 点間の距離がある一定値以下である場合にそのノード間をエッジで接続するという方法をとる。準備実験と異なるのは、今回の地図は各ノードのコストが 1 に設定されているのではなく、現実世界と対応する距離を持っている点である。

5.2.3　制約条件生成 (イベント抽出)

基本的には 4.4.2 節と同じであるが、NER を導入した点が異なる。SRL とルールによって得られた「いつ、誰が、どこに」に相当する情報について、NER の結果との積集合を考慮することでより正確にイベント抽出を行うように改変を行った。具体的には SRL とルールによって得られた各要素 “Who”

と “Where”,“When”に該当するフレーズそれぞれについて、NER の< PERSON >, < DATE >, < LOCATION >タグのついているフレーズを抽出することで、イベントの要素を決定する。さらにイベント抽出によって得られた時間情報について、シミュレーションを行うためにあらかじめ決められた単位ステップ時間に変換を行う。単位ステップ時間はシミュレーション中の時間粒度を決めるものであり、対象とするエージェントの生存期間 (Wikipedia の PersonData 中から抽出する) から決定する。例えば Alexander の場合だと 1 ステップが 3ヶ月に相当し、(Alexander, Babylon, 323BC) というセットから (Alexander, Babylon, 130 - 131) という制約条件が生成される。このように各制約条件はエージェント名 f、地名 h、時間 t_{begin} 〜 t_{end} の 3 つの要素のタプルで表現される。

5.2.4 シミュレーション

以下の目的関数が最大となるエージェント f の移動 $< h_0, h_1, ..., h_T >$ (エピソード) を求めることがシミュレーションの目的である。

$$O = \frac{|C'|}{|C|} - \beta \sum_{t=0}^{T-1} (\frac{d(h_t, h_{t+1})}{\gamma})^2 \tag{5.1}$$

ただし、$c_i \in C$ はイベント抽出によって生成された制約条件、そのうち、シミュレーションによって求められたあるエージェント f についてのエピソード $< h_0, h_1, ..., h_T >$ (h_t はステップ t に f がいた場所 h, T は総ステップ数) によって満たされた制約を $c'_i \in C'$ とする。制約 $c_i = (f, h, t_{begin} - t_{end})$ が満たされる条件は、エピソードの $h_{t_{begin}}, ..., h_{t_{end}}$ 中に h が含まれることとする。よって目的関数の第 1 項目は生成された全制約条件のうち、エージェントの移動によって満たされた制約条件の割合を表すことになる。$d(h_t, h_{t+1})$ は地名 h_t と h_{t+1} 間の距離を表しており Wikipedia の経度緯度情報を用いて式 4.3 によって求められ、目的関数の第 2 項目は距離によるペナルティー項を表す。α と β は満たした制約条件の割合とのバランスをとる定数である。

この目的関数が最大になるエピソードはできるだけ無駄なく、多くの制約を満たそうとするエージェントの移動を表すことになる。距離をペナルティーとして定義しなければ実際の移動とかけ離れたエピソードが得られてしまう可能性がある。またペナルティーを距離の 2 乗で定義したのは、例えば地図上でほぼ一直線上に A, B, C という地名が並んでいたとする。もし距離のペナルティーが線形に定義されている場合、$A \to C$ と $A \to B \to C$ という二つの移動についてペナルティーがほぼ変わらないという状況が発生する。今回の人物移動モデルでは、「$A \to C$ という移動をしたのならばその間の B も通過しただろう」という時間・空間的な推測を利用することができるので、これを実現するために $A \to B \to C$ という移動になるようペナルティーを非線形に定義することにした。この移動で満たされない制約条件は、例えば時間情報が周囲の制約条件と大幅にずれていたり、地名が全く関係なかったりするものが当てはまることが想定される。そのため、このシミュレーションで満たされなかった制約条件はその元となっているイベント抽出が整合性の取れない解釈となっていると考えられるため、これを除去することで精度を改善できる。

5.2.5 評価手法

この実験で扱う文章について、入力されるテキストごとに正しく生成されるべき制約条件について人手によるアノテーションにより正解データを作成する。ベースライン含め各手法によって生成された制約条件と、この正解データの比較により F1 値を算出する。各エージェントにつき数回ずつシミュレーションを繰り返し、それらの全体的なマクロ平均をとることで最終的な F 値とする。なお正解データとの比較については、“Who”, “Where”, “When”の 3 つの情報が完全に一致した場合のみ正解として扱うことにする。

5.3 実験

対象となるテキストデータ中から、既存の解析器による SRL とルール、NER を用いてイベント抽出を行い制約条件を生成し、シミュレーションを行い不適切な制約条件を除去し精度を改善する実験を行った。

5.3.1 対象とするデータ

“Alexander the Great”, “Augustus”, “Napoleon”の 3 人の歴史上の偉人についての Wikipedia 本文 (1,600 文) を対象のテキストデータとする。このテキストデータについて、あらかじめ人手でどの文から “いつ誰がどこにいたか”という要素のセットが正しい解釈のもと抽出できるかどうかをラベル付けするアノテーションを行った。その結果 54 個の正解データが生成された。これは 30 文に 1 個の割合で制約条件が生成されたことになる。また、xml データを利用して[1]対象となる地名は各人物のページからリンクを 1 つたどることで到達する全ページのうち、経度緯度情報をもつページのタイトルとした。また、シミュレーションのステップ数を決定するために必要な各エージェントの生存期間は、エージェントの記事の personData というボックス中の情報から抽出した。基本的にイベント抽出は、対象とするエージェントごとに行うこととする。

5.3.2 制約条件生成

制約条件は基本的には 4.4.2 節と同様に Stanford Core NLP [22], SRL 解析器 [5,6], SUTime [10] を用いてイベント抽出を行うことで生成する。ただし共参照解析は行わない。なお NER の LOCATION 判定の精度が良くなかったため、本実験では地名についてのみ使用する文について NER のアノテーションを行いそのデータを利用することにする。総ステップ数 N ができるだけ 240 となるように、1 ステップあたりの時間粒度を決定する。例えば “Napoleon”の場合だと 1 ステップが現実世界での 6ヶ月という単位になった。

[1] 2015/9/1 にダウンロード

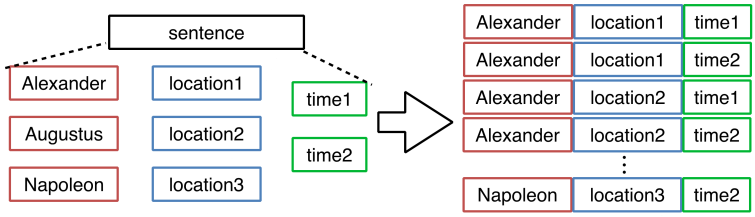


図 5.2. NER のみのベースライン手法の説明図

5.3.3 シミュレーション設定

シミュレーションでの目的関数の最適化は式 3.2 で表される焼きなまし法を用いた。地名の増加により計算負荷を減らすため、Algorithm3.2 の 18 行目から 23 行目で表される近傍状態生成のループの進行方向を逆にする操作を行わないことにした。また、エピソードを求める際に使用する地図については、2 地点間の距離が $1,000km$ 以内であればその間にエッジがあるものとして作成した。式 5.1,3.2 の α, β, γ, C はそれぞれ $\alpha = 1,000, \beta = 0.3, \gamma = 0.001, C = \frac{1}{30}$ とし、各エージェントについて 100 回ずつシミュレーションを行い、満たされなかった制約条件を除去したのち、アノテーションデータとの比較で F1 値を求める。このパラメータについてはいくつか試した結果良さそうなものを選択した。

5.3.4 ベースラインについて

まずは単純に SRL とルール、NER によってイベント抽出を行う方法をベースライン 1 とし、これによって抽出された “いつ誰がどこにいたか” という情報から生成された制約条件と、アノテーションデータとの比較によって F1 値を求める。これが自然言語の表層情報のみを用いた解析結果による解釈となる。基本的には提案手法はこれとの比較になるが、この手法は構文解析結果や NER などの積集合によってイベント抽出が行われるため、一般的に考えると適合率が高くなり再現率が低くなることが予想される。本手法は誤りを除去することにより精度を改善する方法をとるため、除去の対象となるイベント抽出結果は基本的に再現率が高い方が都合が良い。そこで SRL などの構文解析結果を用いずに NER のみによるベースライン 2 を考える。このベースライン 2 は各文について NER によって抽出された< PERSON >, < DATE >, < LOCATION >の情報の全ての組み合わせによって制約条件を生成する。例えばある文から< PERSON >として “Alexander”,“Augustus”,“Napoleon”、< LOCATION >として “location1”,“location2”,“location3”、< DATE >として “time1”,“time2 が抽出された場合に生成される制約条件の様子を図 5.2 に示す。このようにすべての組み合わせがカバーされるため、再現率はとても高くなるが、適合率は低くなると考えられる。このベースライン 2 に本手法を適用したものを提案手法 1 として F 値がどの程度改善されるのかを評価する。さらにこの提案手法 1 とベースライン 1 の和集合を求める、つまりどちらかが正しいと判断したイベント抽出結果を全体として正しいと判断し出力する手法を提案手法 2 とする。

図 5.3. システム実行中の様子

表 5.1. イベント抽出　実験結果

	Precision	Recall	F1
baseline 1	**0.706**	0.444	0.545
baseline 2	0.246	**0.870**	0.384
propose 1	0.474 ± 0.007	0.525 ± 0.011	0.498 ± 0.008
propose 2	0.474 ± 0.004	0.679 ± 0.007	$\mathbf{0.559 \pm 0.005}$

5.3.5　実験結果

本システムの実行中の様子を図 5.3 に示す。図中の丸く並んでいる四角形は、白文字が元となる自然言語文を表し、黄色文字が生成された制約条件を表している。地球儀上にマッピングされた各地名について対象となるエージェント (この例ではアレキサンダー大王) に関するエピソードをピンク色の線で表す。制約条件を表す四角形のうち、白く表示されているものはエピソードによって満たされた制約条件で、赤く表示されているものはエピソードによって満たされていない制約条件を表す。最初はすべて赤く表示される制約条件がシミュレーションが進むにつれて白くなっていき、最後まで赤いままであった制約条件は、その解釈がエピソードと矛盾していることを示す。実際の実行中のデモ動画は `http://www.logos.t.u-tokyo.ac.jp/~murakami/demo` で確認することができる。

各手法における適合率 (precision)、再現率 (Recall)、F1 値を表 5.1 に示す。提案手法 1 は、ベース

ライン 2 でのイベント抽出結果のうち、整合性がとれないと判断したものを除去するため、precision は上がる可能性があるが、recall 自体は上がらない。表 5.1 から確かにベースライン 2 の適合率が改善されていることがわかる。さらに SRL (ベースライン 1) による構文解析情報を使った方法よりも再現率は高くなり、SRL による抽出に失敗しているイベントを抽出することに成功している。最終的な F1 値としてみると SRL を使った方法には及ばなかったが、構文解析を行っていないにもかかわらずほぼ半分のイベント抽出に成功している。また提案手法 2 より、ベースライン 1 と組み合わせることで F 値が有意に改善されることがわかった。F1 値があまり高くならなかったのは時間・空間情報による矛盾を発見できないケースによるもので、これを改善するためには移動以外の実際の世界史に関する情報をいれてモデルをより豊かなものになるように拡張する必要がある。

ベースライン 2 と比べて Recall が下がってしまった原因については、時間粒度の問題があげられる。今回はシミュレーションに用いる制約条件の数などから総ステップ数を 240 としたが、これが小さいために、短時間で長距離を移動した区間での正しい制約条件が満たせなくなってしまう現象が見られた。その他の原因としては制約条件の少なさが挙げられる。距離的に離れた制約条件は、焼きなまし法のような近傍探索ではたどり着けないことがあり、データスパースによって正しい制約条件を満たすことができない現象が見られた。

最後に SRL によるベースライン 1 が間違えた文についてモデルでは間違っているイベント抽出だと判断できた例 (1) と、ベースライン 1 ではイベント抽出できなかったがモデルでは正しく抽出できた例 (2) を以下に示す。これは現状の表層情報のみを用いた自然言語処理による解釈ミスを、モデルが検出できた例である。

1. It was originally thought to have been the sarcophagus of Abdalonymus (died **311 BC**), the king of Sidon appointed by **Alexander** immediately following **the battle of Issus** in 331.

2. Serving in the French army as an artillery officer , **Napoleon** supported the Revolution from the outset in **1789** and tried to spread its ideals to **Corsica** , but was banished from the island in 1793 .

(1) の例では末尾の “331” という表記が紀元前 331 年を表す “BC” が省略されたことにより、時間表現として認識されず、SRL 解析器が間違って時間情報を Abdalonymus の死んだ時間を意味する “311BC” と解釈してしまったために起きたイベント抽出ミスである。(2) の方では SRL による解析結果で位置情報である “Corsica” が脱落してしまうことによるミスを、NER の組み合わせから正しいものを選択することで、構文解析を使わずにイベント抽出に成功していることを示す。

このように自然言語の表層情報のみによる処理だけでは到達できないレベルでの解釈ミスを判断するためには、空間・時間のような身の回りの世界との整合性を考える仕組みが必要であることがわかる。

第6章　おわりに

6.1　本研究の結論

自然言語理解には、文の表層的な情報だけでなく、身の回りに溢れる様々な情報を総動員する必要がある。そのため、本研究ではそのような情報を現実世界でうまく扱えるような世界モデルの実現を目指し、単純な人物移動モデルを用いて実験を行った。最初に地図データを人手によって生成して行った実験では、直接文章に書かれていない内容に関する質問応答に答えるタスクに取り組んだ。この実験により文章全体で表現されている人物の移動に関してある程度正しい推測に成功し、また質問応答に関しても直接書かれていない事実に関して推測することができた。またこの推測を利用することで共参照解析やイベント抽出などその他のタスクにも応用できる可能性を示した。

次の共参照解析の実験では時間・空間情報をルールで処理することによって共参照解析の精度向上に取り組んだ。この実験によって時間・空間情報を利用することで意味を考えないと解けないと考えられる難しい共参照解析も解ける可能性があることを示した。

3つ目のイベント抽出の実験では人物移動モデルを用いたイベント抽出に取り組んだ。この手法では抽出された結果が時間・空間的な制約と整合性が取れているかどうかを確認することで、既存の構文解析などを用いた手法だけでは難しいと考えられるイベント抽出ができることを示した。

これらの実験を通して現実世界の時間空間制約を用いた推測を行うことは、現実世界と整合性の取れた正しい自然言語解釈を行う上で有用であることを示した。

6.2　今後の課題

最後に今後の課題と展望について説明する。

本研究では主に歴史上の人物についてその時間・空間的な制約を扱う人物移動モデルによる自然言語解釈に取り組んだが、各タスクにおいて精度を上げるためには、それ以外の要素を考慮する必要がある。例えば人物の移動についても距離がコストとして捉えられている場合、特に動機がない場合は移動しない方向に推測されてしまう。このような動機に加え、地図データについても山脈や砂漠といった要素を考慮することでより正確な人物移動を再現することができ、世界モデルに一歩近づくことができる。最終的にすべてをシミュレーションすることは難しいと考えられるが、例えば世界史についてはオントロジーなどを利用することで、自然言語文で書かれている内容をわかりやすい形で再現することができると思われる。これは自然言語解釈でとても有用であり、自動要約などその他のタスクでも応用が期待される。

また、本研究では時間空間制約を用いた手法がどの程度文章解釈で有用かを検証するために様々なアノテーションデータを生成したが、一般的な人工知能分野や自然言語処理分野の研究と比較するとサイズが小さく、正確な評価が難しいという問題点もある。これを解決するためにも現実世界における情報を何らかのモデルで利用する様々な手法の提案や共通のフレームワークなどの研究が必要である。

参考文献

[1] Olga Babko-Malaya. Propbank annotation guidelines. *URL: http://verbs. colorado. edu*, 2005.

[2] Yevgeni Berzak, Andrei Barbu, Daniel Harari, Boris Katz, and Shimon Ullman. Do you see what i mean? visual resolution of linguistic ambiguities. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1477–1487, Lisbon, Portugal, September 2015.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] Anders Björkelund, Bohnet Bernd, Love Hafdell, and Pierre Nugues. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstrations*, pp. 33–36, Beijing, China, August 2010.

[5] Anders Björkelund, Love Hafdell, and Pierre Nugues. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pp. 43–48, Boulder, Colorado, June 2009.

[6] Bernd Bohnet. Efficient parsing of syntactic and semantic dependency structures. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pp. 67–72, Boulder, Colorado, June 2009.

[7] Antoine Bordes, Nicolas Usunier, Ronan Collobert, and Jason Weston. Towards understanding situated natural language. In *AISTATS*, pp. 65–72, 2010.

[8] S.R.K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 82–90, Suntec, Singapore, August 2009.

[9] S.R.K. Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. Learning high-level planning from text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 126–135, Jeju Island, Korea, July 2012.

[10] Angel X Chang and Christopher D Manning. Sutime: A library for recognizing and normalizing time expressions. In *LREC*, pp. 3735–3740, 2012.

[11] Angel X Chang, Manolis Savva, and Christopher D Manning. Learning spatial knowledge for text to 3D scene generation. In *Proceedings of Empirical Methods in Natural Language Processing, EMNLP*, pp. 2028–2038, 2014.

[12] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges*, pp. 177–190, 2006.

[13] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, Vol. 31, No. 3, pp. 59–79, 2010.

[14] Abraham Fowler, Bob Hauser, Daniel Hodges, Ian Niles, Adrian Novischi, and Jens Stephan. Applying cogex to recognize textual entailment. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pp. 69–72, 2005.

[15] Hannaneh Hajishirzi, Julia Hockenmaier, Erik T. Mueller, and Eyal Amir. Reasoning about robocup soccer narratives. In *Proceedings of the 27th conference on Uncertainty in Artificial Intelligence*, pp. 291–300, 2011.

[16] Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pp. 1–18, Boulder, Colorado, June 2009.

[17] Sangdo Han, Kyusong Lee, Donghyeon Lee, and Gary Geunbae Lee. Counseling dialog system with 5w1h extraction. In *Proceedings of the SIGDIAL2013 Conference*, pp. 349–353, 2013.

[18] Richard Johansson and Pierre Nugues. Dependency-based syntactic–semantic analysis with propbank and nombank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pp. 183–187, Manchester, England, August 2008.

[19] Scott Kirkpatrick, MP Vecchi, et al. Optimization by simmulated annealing. *Science*, Vol. 220, No. 4598, pp. 671–680, 1983.

[20] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Stanfordâ$^{\mathrm{TM}}$s multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pp. 28–34, Portland, Oregon, USA, June 2011.

[21] Xinjian Li, Ran Tian, Ngan L. T. Nguyen, Yusuke Miyao, and Akiko Aizawa. Question answering system for entrance exams in QA4MRE. In *Working Notes for CLEF 2013 Conference*, Valencia, Spain, September 2013.

[22] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60, 2014.

[23] Nancy McCracken, Necati Ercan Ozgencil, and Svetlana Symonenko. Combining techniques for event extraction in summary reports. In *Proceedings of the 2006 AAAI Workshop on Event Extraction and Synthesis*, pp. 7–11, 2006.

[24] Martina Naughton, Nicola Stokes, and Joe Carthy. Investigating statistical techniques for sentence-level event classification. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pp. 617–624, Manchester, UK, August 2008.

[25] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, Vol. 31, No. 1, pp. 71–106, 2005.

[26] Haoruo Peng, Daniel Khashabi, and Dan Roth. Solving hard coreference problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 809–819, 2015.

[27] Altaf Rahman and Vincent Ng. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pp. 968–977, 2009.

[28] Altaf Rahman and Vincent Ng. Coreference resolution with world knowledge. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 814–824, 2011.

[29] Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 193–203, Seattle, Washington, USA, October 2013.

[30] Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. Solving geometry problems: Combining text and diagram interpretation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1466–1476, September 2015.

[31] Smriti Sharma, Rajesh Kumar, Pawan Bhadana, and Sumita Gupta. News event extraction using 5w1h approach & its analysis. *International Journal of Scientific & Engineering Research*, Vol. 4, No. 5, pp. 2064–2068, 2013.

[32] Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, Vol. 27, No. 4, pp. 521–544, 2001.

[33] Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 8–15, Sapporo, Japan, July 2003.

[34] Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pp. 159–177, Manchester, England, August 2008.

[35] Marta Tatu and Dan Moldovan. A semantic approach to recognizing textual entailment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 371–378, 2005.

[36] W. Wang, D. Zhao, and D. Wang. Chinese news event 5w1h elements extraction using semantic role labeling. In *Proceedings of the third International Symposium on Information Processing*, pp. 484–489, 2010.

[37] Wei Wang. Chinese news event 5w1h semantic elements extraction for event ontology population. In *Proceedings of the 21st International Conference on World Wide Web*, pp. 197–202, 2012.

[38] Wei Wang, Dongyan Zhao, Lei Zou, Dong Wang, and Weiguo Zheng. Extracting 5w1h event semantic elements from chinese online news. In *International Conference on Web-Age Information Management*, pp. 644–655, 2010.

[39] 成澤克麻, 渡邊陽太郎, 水野淳太, 岡崎直観, 乾健太郎. 数量表現を伴う文における含意関係認識の課題分析. 言語処理学会第１８回年次大会 発表論文集, pp. 1087–1090, 2012.

本研究に関する発表文献

国際会議論文 (査読付き)

1. Yuki Murakami, Yoshimasa Tsuruoka. Where Was Alexander the Great in 325 BC? Toward Understanding History Text with a World Model. *In Proceedings of the First Workshop on Linking Computational Models of Lexical, Sentential and Discourse-level Semantics*, pp. 83–88, Lisbon, Portugal, September 2015.

国内会議論文 (査読なし)

1. 村上優樹, 鶴岡慶雅. アレキサンダー大王の遠征は再現できるか? ～人物移動モデルによる歴史文章解釈～. 言語処理学会第 21 回年次大会発表論文集, pp. 361–364, 京都, 2015 年 3 月.

2. 村上優樹, 鶴岡慶雅. 現実世界の時間・空間制約を用いた共参照解析の精度向上. 言語処理学会第 22 回年次大会発表論文集, pp. 250–253, 仙台, 2016 年 3 月.

論文誌 (査読付き)

1. 村上優樹, 鶴岡慶雅. 現実世界の時間空間制約を用いた歴史上の人物に関する情報抽出. 人工知能学会論文誌. 投稿予定.

謝辞

本研究を進めるにあたり多くの方々にお世話になりました。

指導教員である鶴岡慶雅准教授には、卒論生の時から 3 年間にわたりお世話になりました。配属最初のチュートリアルから、研究の進め方、テーマ決め、論文の紹介、実験のやり方、論文の書き方まで研究生活を送る上で必要な非常に多くの知識やアドバイスをいただきました。特に修士 1 年の時に書いた英語論文では、内容だけでなく細かい文法ミスやおかしな言い回しなど細部まで何度も丁寧に添削していただきました。締め切りぎりぎりなことが多く大変ご迷惑をおかけいたしましたが、英語が苦手な私にとってとても勉強になりました。本当にありがとうございます。

研究室の先輩である江里口瑛子さん、亀甲博貴さん、橋本和真さん、水上直紀さんには、いつも一緒にご飯に誘っていただき研究に関する様々な知識を教えていただきました。私にとってとても頼もしい博士課程の先輩方でした。

研究室の後輩の皆さんにもたまに週末のボードゲームにつきあっていただいたり、一緒にお昼ご飯につき合っていただいたりし、最後まで楽しい修士生活を送ることができました。ありがとうございます。

同期は私一人でしたが、多くの先輩後輩に囲まれ寂しい思いをすることなく最後まで研究生活を無事続けることができました。いろいろご迷惑をおかけすることもあったかもしれませんが、とても楽しい 3 年間でした。本当にありがとうございました。

最後に今まで長いこと学生であることを認め、支えてくれた家族に感謝いたします。ありがとうございます。

平成 29 年 2 月 2 日