学位論文　博士（工学）

深層学習を用いた
一階述語論理による推論に関する研究

2021 年度

慶應義塾大学大学院理工学研究科

本田　裕

目次

図目次

表目次

第１章 序論

１.１　はじめに

本研究の目的は，深層学習による記号推論を行うことで，ビッグデータを扱うことを可能とするロバストで解釈可能な推論手法を確立することである．従来の人工知能による記号推論は，人が中身を解釈することは容易であるが，ビッグデータに含まれるデータの曖昧性に対処することは困難である．他方で，深層学習はデータの曖昧性に対してロバストであるものの，人が中身を解釈することは困難である．そこで本研究では，従来の人工知能による記号推論と深層学習の手法を融合することにより，ロバスト性と解釈可能性の両方を備える推論手法を確立することを目的とする．

前述の推論手法を確立するために，曖昧性を含む大量のデータから知識ベースを構築し，それらを一階述語論理で表現し，深層学習により学習を行う．具体的には，(1) Web 上のデータから構築した知識ベースをもとに演繹推論を行う手法（第 2 章）, (2) Web 上のデータから構築した知識ベースをもとに類推を行う手法（第 3 章）, (3) 学習済みの深層強化学習モデルから構築した知識ベースをもとに演繹推論を行う手法 （第 4 章）を提案することによって実現する．

本章は以下のように構成される．まず，1.2 節では記号推論の研究，1.3 節ではニューラルネットワークの研究について概説する．1.4 節では，1.2 節と 1.3 節を踏まえて，ニューラルネットワークを用いた記号推論の研究について述べる．1.5 節で本研究の位置づけについて述べた後，1.6 節で本論文の構成についてまとめる．

1.2　記号推論

古代ギリシャ哲学における論理学の研究

論理を用いた推論の研究は古く，古代ギリシャ哲学が起源である．初めての系統的な研究はアリストテレスによるオルガノンであり，三段論法について論じられている．アリストテレスの三段論法とは，以下に示すような３つの命題による演繹推論である．

大前提　：全ての人間は死すべきものである
小前提　：ソクラテスは人間である
結論　　：ゆえにソクラテスは死すべきものである

古代ギリシャ哲学における論理学の研究においては，現代の記号論理に用いられているような記号による表現は行われていなかった．

近代における記号論理の研究

近代に入ると George Boole によりブール論理の研究 [1] が行われ，現代の命題論理にほぼ近い体系が形作られた．ブール論理では，初めて記号を用いた表現が導入され，記号論理の始まりとなった．記号論理では，命題は記号に置き換えられ，論理演算記号を用いた論理式の表現が行われるようになった．式(1.1)～(1.3)は三段論法における命題論理を論理式で表した例である．P ならば Q，Q ならば R であれば，P ならば R が成り立つことを意味している．

$$P \to Q \tag{1.1}$$

$$Q \to R \tag{1.2}$$

$$P \to R \tag{1.3}$$

その後，Gottlob Frege によって述語論理の研究 [2] が行われた．述語論理は論理式に含まれる変数を量化することができ，命題論理よりも表現力が高い．例えば述語論理に導入されている量化子には「すべて」を意味する全称量化子∀や「ある」を意味する存在量化子∃がある．

命題論理の表現能力では，「全ての人間は死すべきものである」，「ソクラテスは人間である」，「ゆえにソクラテスは死すべきものである」を記号で表現することはできない．これらは述語論理を使用することで式(1.4)～(1.6)のように記号で表現することが可能である．$human(X)$は「X は人間である」という述語，$mortal(X)$は「X が死すべきものである」という述語を意味する．$socrates$ はソクラテスを意味する．

$$\forall(human(X) \rightarrow mortal(X)) \tag{1.4}$$

$$human(socrates) \tag{1.5}$$

$$mortal(socrates) \tag{1.6}$$

述語論理には個体の量化のみを許す一階述語論理，述語の量化も可能とする高階述語論理がある．一階述語論理は健全性と完全性を備えていることが Gödel により証明 [3] されたが，高階述語論理は健全性と完全性は保証されていない．このことから，記号論理における推論や証明には一階述語論理が使用されることが多い．

AI への記号論理の導入

1950 年代には， McCarthy により一階述語論理が初めて AI の分野に導入 [4] され，一階述語論理を利用した推論システムの研究が行われるようになった．1972 年には，Alain Colmerauer によって一階述語論理を基礎とする論理プログラミング言語 Prolog が開発 [5] され，AI における記号推論の研究に多く用いられるようになった．Prolog のプログラムは事実と規則，質問の 3 つの要素から構成されている．式(1.7)，(1.8)に Prolog の事実の例を示す．式(1.7)は「Tom は男性である」という事実を意味し，式(1.8)は「Tom は Mary の親である」という事実を意味する．

$$male(tom). \tag{1.7}$$

$$parent(tom, mary). \tag{1.8}$$

式(1.9)に Prolog の規則の例を示す．規則は事実と事実の関係性を示すもので，式(1.9)は「X

が男性で，X は Y の親であれば，X は Y の父親である」という規則を意味する．

$$fatehr(X,Y):-male(X),parent(X,Y). \tag{1.9}$$

尚，Prolog の規則は一階述語論理の部分集合であるホーン節により記述される．式(1.10)にホーン節の例を示す．これは $body_1$ から $body_n$ までが成り立てば，$head$ が成り立つことを意味する． $head$ を頭部と呼び， $body_1$, $body_2$, …, $body_n$ を本体と呼ぶ．

$$head:-body_1,body_2,\dots,body_n. \tag{1.10}$$

式(1.11)に Prolog の質問の例を示す．式(1.11)は「Mary の父親は誰か」という質問を意味し，Prolog 処理系は知識ベースに記述されてている事実や規則に基づいて回答を導く．

$$fatehr(X,mary). \tag{1.11}$$

式(1.12)は式(1.11)の回答である．Mary の父親を問われ，「Tom」と回答をしている．

$$X=tom. \tag{1.12}$$

1970 年以降の研究

1970 年から 1980 年にかけて，論理プログラミングによる記号推論は，自動定理証明 [6]やエキスパートシステム [7] といった研究に適用された．自動定理証明は計算機により定理の証明を行う研究であり，エキスパートシステムは人間の専門家の意思決定を模擬する研究である．これらの初期の研究においては知識ベースの構築を人手で厳密に行う必要があった．その後，知識ベースが不完全な場合にルールを生成する研究 [8-9] や，データが不足している場合に，仮説をおいて推論を行う研究 [10-12] が行われてきた．しかしながら，これらの技術では Web 上にあるような大量なデータを扱うことは困難であり，限定した分野での活用に留まっていた．

1.3 ニューラルネットワーク

第一次ニューラルネットワークブーム

ニューラルネットワークの研究は，1943 年に McCulloch と Pitts によって発表された形式ニューロン [13] が始まりである．形式ニューロンは人間のニューロンをモデル化したものである．その後，1957 年に Rosenblatt によりパーセプトロン [14] が発明され，第一次ニューラルネットワークブームが始まった．しかしながら，Minsky と Papert によりパーセプトロンは線形分離不可能な問題を解けないことが示され [15]，第一次ニューラルネットワークブームは終焉した．

第二次ニューラルネットワークブーム

1986 年に Rumelhart により誤差逆伝搬法 [16] が発明された．誤差逆伝搬法を用いることで，多層ニューラルネットワークを学習することが可能となり，パーセプトロンでは実現できなかった線形分離不可能な問題に対応することができるようになった．これにより，第二次ニューラルネットワークブームが始まった．しかしながら，過学習や勾配消失問題といった学習能力の限界により，第二次ニューラルネットワークブームも終焉することとなった．

第三次ニューラルネットワークブーム

2006 年に Hinton により多層ニューラルネットワークの勾配消失問題を解決する手法として Autoencoder [17] と Deep Belief Network [18] が発表された．これらは後に深層学習（ディープラーニング）と呼ばれる研究の始まりであった．その後，Hinton らの AlexNet [19] が ILSVRC（ImageNet Large Scale Visual Recognition Challenge）で大差で優勝したことにより，一躍深層学習が注目を浴びた．以降，深層学習の研究は活発となり，第三次ニューラルネットワークブームが始まった．深層学習は画像認識への適用に留まらず，自然言語処理やグラフ構造のデータ処理，強化学習への研究にも応用されるようになった．

1.4　ニューラルネットワークを用いた記号推論

1.2節では記号推論の研究について，1.3節ではニューラルネットワークの研究についてそれぞれを概説した．本節では記号推論とニューラルネットワークの研究の関わりについて述べる．

これまで，AI による記号推論の研究とニューラルネットワークの研究は長い間交わることがなかった．第二次ニューラルネットワークブームであった 1990 年代には，コネクショニズムによって，多層ニューラルネットワークを利用した記号処理の研究 [20-21] が行われた．しかしながら当時のハードウェア能力と多層ニューラルネットワークの学習能力の限界により，手法の提案に留まっており，実用的なシステムの構築には至らなかった．

第三次ニューラルネットワークブームでは，深層学習が登場し，それまでのニューラルネットワークの学習能力が飛躍的に向上した．特に，深層学習によってグラフ構造の学習 [22-23] が可能となったことをきっかけに，再びニューラルネットワークを利用した記号推論の研究 [24-25] が行われるようになった．一階述語論理の知識表現をグラフ構造として捉え，実際に深層学習により演繹推論や帰納推論が可能なことが示された．ニューラルネットワークを用いた記号推論は，曖昧なデータに対するロバスト性や学習モデルの解釈可能性を解決する可能性があり，研究が注目され始めている．

1.5　本研究の位置づけ

本節では本研究と目的及び手法が類似している，ニューラルネットワークを用いた記号処理の既存研究との比較を述べる．

1.5.1　Web 上のデータから構築した知識ベースをもとに演繹推論を行う手法

Web 上にあるような大量なデータから知識ベースを構築するには，様々なデータ形式を取り扱うための高い表現力，誤りや未学習データに対して高いロバスト性，小規模なデータからの学習が必要となる．

そこで，本研究ではマッチング，導出及びリスト処理といった記号処理を深層学習により実

現し，

1) ネットワークの構成に依らず，原子論理式に含む項の数に制約を設けない
2) ネットワークの構成に依らず，論理式に含む原始論理式の数に制約を設けない
3) ネットワークにメタルールを提供しない
4) 原子論理式が使用するデータ構造にリストを許容する
5) 未学習の定数記号が入力されても解を導く
6) 小規模なデータからも学習可能とする

という従来研究にはない特長を持った記号処理の学習を行う．本提案手法は，単語分散表現の利用によって未学習のデータを含んでいても深層学習による演繹推論を可能とした初めての事例である．更に，深層学習による記号処理を質問応答システムに組み込むことでその実用性を示す.

1.5.2 Web 上のデータから構築した知識ベースをもとに類推を行う手法

これまで深層学習を用いた記号推論の研究は演繹推論と帰納推論が中心であった．しかしながら，Web 上にある膨大なデータから効率よく推論を行うには，演繹推論や帰納推論だけでなく，従来の人工知能研究で行われてきた類推や仮説推論といった手法も重要になる.

そこで本研究ではオブジェクト間の関係性の類似及びオブジェクト間の類似に着目することで，一階述語論理の規則の類推を深層学習により実現し，

1) 既知の規則を含む知識ベースから未知の規則の推論を可能とする
2) 知識ベースのサイズが大きい場合でも未知の規則を効率的に抽出可能とする

ことに取り組む.

類推とは，ある事例と他のある事例の類似性から新しい事例の解決を図る推論である．これまでにニューラルネットワークを用いた類推の研究も行われているが，記号処理に一階述語論理を用いて類推を行うのは本研究が初めてである.

1.5.3　学習済み深層強化学習モデルから構築した知識ベースをもとに演繹推論を行う手法

これまでの説明可能な深層強化学習の研究では，エージェントが個々人の感性に合わせて情報を伝えることや，人がエージェントに規則を追加することは困難であった．

そこで本研究では深層学習による記号処理を用いて，

1) 人の解釈が困難な出力を行う学習済みの強化学習モデルから個々人が理解しやすい記号表現を出力するモデルを再現して適用
2) 人が表現しやすい知識表現を用いて人が意図を伝えることが可能

を特長とするエージェントの構築に取り組む．本提案手法は，学習済みの強化学習モデルから人とコミュニケーション可能なエージェントを構築する初めての事例である．

1.6　本論文の構成

本論文は全 5 章から構成される．

第 2 章では，深層学習を用いて一階述語論理による演繹推論を行い，知識ベースから質問応答システムを構築する手法を提案する．

第 3 章では，深層学習を用いて一階述語論理による類推を行う手法を提案する．

第 4 章では，深層強化学習モデルをファジィ表現と一階述語論理を組み合わせた記号表現モデルに置き換えることで，人とコミュニケーションを可能とする手法を提案する．

第 5 章では，本研究において取り組んだ深層学習を用いた一階述語論理による推論に関する成果をまとめる．また，今後の展望についても述べる．

第２章 深層学習による記号処理を用いた知識ベースからの質問応答システム構築

２.１　序論

これまで，ニューラルネットワークを利用した記号処理の研究 [20-21] が行われてきた．深層学習が登場し，グラフ構造の学習 [22-23] を行うことが可能となり，ニューラルネットワークを利用した記号処理の研究 [24-26] が活発に行われるようになった．しかしながら，Web 上にあるような大量なデータから知識ベースを構築するには，様々なデータ形式を取り扱うための高い表現力，誤りや未学習データに対して高いロバスト性，小規模なデータからの学習が必要となる．そこで，本章における研究ではマッチング，導出及びリスト処理といった記号処理を深層学習により実現し，

1) ネットワークの構成に依らず，原子論理式に含む項の数に制約を設けない
2) ネットワークの構成に依らず，論理式に含む原始論理式の数に制約を設けない
3) ネットワークにメタルールを提供しない
4) 原子論理式が使用するデータ構造にリストを許容する
5) 未学習の定数記号が入力されても解を導く
6) 小規模なデータからも学習可能とする

という従来研究にはない特長を持った記号処理の学習を行う．本提案手法は，単語分散表現の利用によって未学習のデータを含んでいても深層学習による演繹推論を可能とした初めての事例である．

更に，深層学習による記号処理を質問応答システムに組み込むことでその実用性を示す．従来の質問応答システム [27-28] の多くは未知語を含む Web 上の大量なデータから推論を行って回答を導くことはできない．提案するシステムは未知語を含んだ事実から一階述語論理に基づく推論を行い，質問に対する回答を導く．本システムを適用することにより，Web 上のデー

タを利用した高機能なチャットボット [29-30] や意思決定支援システム [31-32] の構築が可能になる．これらのアプリケーションを実現できれば，ユーザは新たな概念の探索や隠れた情報の発見ができるようになる．

1990 年代のコネクショニズム以降，従来の人工知能研究の領域とニューラルネットワーク研究の領域が交わって研究が行われることは非常に少なかった．本章における研究は従来の人工知能研究と深層学習の研究に跨る新たな研究領域といえる．また，深層学習を用いてPrologライクな処理系を実現しており，従来にはない新規性の高いアプリケーションとなっている．

本章ではまず，2.2節で本章における研究と関連する研究について述べる．2.3節では，学習対象である記号処理について述べる．2.4節では，2.3節で述べた記号処理の学習手法並びに、それらを利用した質問応答システムを提案する．2.5 節では，提案手法の実験結果について報告する．尚，本章における研究は [33] の研究をベースとして行った．

2.2 関連研究

2.2.1 ニューラルネットワークによる記号推論

深層学習の登場以前より，ニューラルネットワークを用いて記号処理を学習し，推論を行う研究 [20-21] が数多く行われてきた．命題論理の学習を対象とした研究 [34-36] だけでなく，本章における研究のように一階述語論理を対象とした研究 [37-39] や単一化を学習する研究 [40-41] も行われてきた．しかしながら，これらの研究は手法の提案に留まっており，実用性は低かった．

深層学習の登場後は，グラフニューラルネットワーク [24-25,42-44] やフィードフォワードネットワーク [26] を利用した記号処理の研究が行われるようになった．グラフニューラルネットワークを利用した記号学習の場合，事前に原始論理式や論理式の形式をネットワークに与えておかなければならず，データに含まれる論理式の形式を事前に想定しておく必要がある．フィードフォワードネットワークを用いた記号学習の場合は，原子論理式や論理式の形式はネットワークの構成に依存し，事前にネットワークにメタルールを提供する必要がある．本論文の提案手法は原子論理式に含む項の数や論理式に含む原始論理式の数に制約を設けておらず，事前にネットワークに与える必要もない．

一方で，深層学習による GPT-3 [45] や Transformer [46]のような高性能な言語モデルを用いて自然言語による推論タスクを解く研究 [47] が行われている．しかしながら，高性能な言語モデルの回答は一貫性がなく，矛盾が発生する課題があった．そこで，高性能な言語モデルと記号推論を組み合わせて性能を向上させる研究 [48] が行われている．

2.2.2 質問応答システム

深層学習の登場後、深層学習を用いた質問応答システム [28,45] の研究が行われ，システムのパフォーマンスが向上した．これらのシステムは，入力した事実に基づいて回答候補の検索を行い，その候補の中から回答を選択して応答する．提案システムとは異なり，回答の導出に推論は用いずに，単に事実に基づいて回答をするのみである．

また，質問文をベクトル表現に Embedding し，深層学習により質問応答を行う研究 [50-52] も行われている．しかしながら，我々の知る限り，質問文に未学習の記号が含まれていた場合でも，本論文での提案手法のように未学習の記号を内部表現に Embedding して回答を推論する研究はない．

2.2.3 プログラム合成

ニューラルネットワークによる記号推論に関連する研究の一つに深層学習によるプログラム合成がある．プログラム合成の研究では自然言語をもとにしたソースコードの生成や，あるプログラミング言語から他のプログラミング言語のソースコードへの変換が扱われている．プログラム合成は一種の翻訳と捉えることができるため，Neural Machine Translation (NMT)の技術 [46,53-54] が使われている．プログラム合成ではモデルに Seq2Seq [53-54] をベースとしたものを適用する研究 [55-56] が多く行われている．一方で Transformer [46] を適用する研究 [57-58] は少ない．Seq2Seq は Decoder の LSTM [59] の隠れ状態により，プログラミング言語の構文構造の学習が容易であるが，Transformer は隠れ状態が存在しないため，学習が困難なためと考えられる．

```
?- male(tom) = male(tom) .
true.

?- male(tom) = male(bob) .
false.

?- male(X) = male(tom).
X = tom.
```

図 2.1　単一化の例

2.3　記号処理

本論文では記号処理に Prolog ライクなシステムを用いる．Prolog 処理系 [60] は質問を受け取ると，知識ベースに格納されている事実と規則の集合を参照し，回答を導き出す．質問は 1 つもしくは複数個の目標から成る．Prolog 処理系は質問が入力されると，後ろ向き推論を行うことで目標が充足するか否かを導く．以降に Prolog 処理系が推論を行う際に実行している処理について簡単に説明する．

2.3.1　単一化

単一化は与えられた 2 つの項に対して，変数への代入を行い同じ項を得る処理である．Prolog プログラムでは 2 つの節をオペレータ“=”で接続することで，単一化を実行することができる．項の中に変数が含まれる場合は，変数が置換される．

図 2.1 に単一化の例を示す．“male(tom)”とオペレータ“=”の右側が同一の場合は“true.”，異なる場合は“false.”を回答する．変数を含む“male(X)”の場合は，変数への代入が行われ，“X=tom.”を回答する．

尚，2 つの項が与えられたとき，それらが同一であるか，もしくは両者の項中の変数のオブジェクトを代入した後にそれらの項が同一となるように変数のオブジェクトへの具体化ができることをマッチングと呼ぶ．

2.3.2　導出

導出は与えられた目標と規則から新たな目標を導く処理である．Prolog では，ホーン節に限定した SLD 導出（Selective Linear Definite clause resolution）と呼ばれる導出を行う．ホー

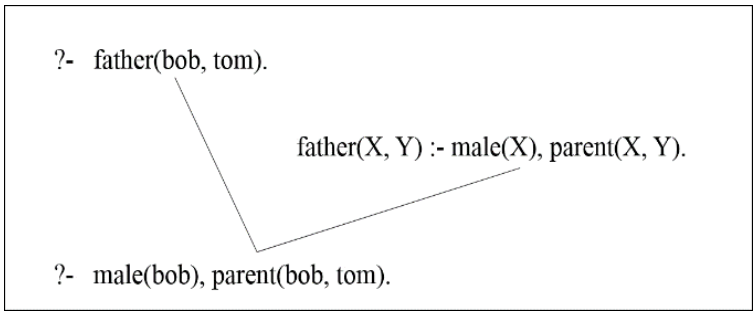


図 2.2　SLD 導出の例

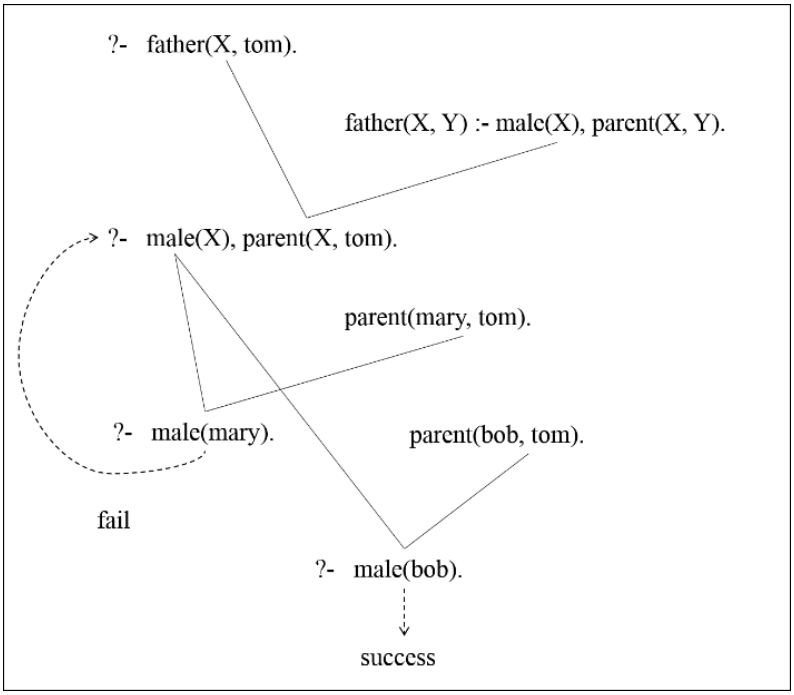


図 2.3　バックトラックの例

ン節とは，一つ以下の肯定形のリテラルから成る節である．

図 2.2 に SLD 導出の例を示す．目標“father(bob, tom).”に規則“father(X, Y) :- male(X), parent(X, Y).”を適用し、新たな目標“male(bob), parent(bob, tom).”を導いている．

2.3.3　バックトラック

Prolog処理系は，知識ベースから事実や規則を取り出し，マッチング及び導出を実行する．マッチングや導出が失敗した場合は成功した時点まで後戻りした後，新たな事実や規則を取り出して単一化や導出を繰り返す．Prolog処理系が実行する後戻りの処理はバックトラックと呼

```
?- member(kansas, [colorado, nebraska, kansas]).
true.

?- member(georgia, [colorado, nebraska, kansas]).
false.
```

図 2.4　メンバーシップ関係の例

ばれる．

図 2.3 にバックトラックの例を示す．規則“father :- male(X), parent(X, Y).” により目標“father(X, tom).”から，“male(X), parent(X, tom).”が導出される．続いて目標“parent(X, tom)”は事実“parent(mary, tom).”とマッチングすると，“X = mary.”を得るが，目標“male(mary).”とマッチングする事実が存在しないため失敗し，バックトラックを行う．次に事実“parent(bob, tom).”を取り出し，マッチングすると“X = bob.”を得るが，事実“male(bob).”が存在するため，成功となる．

2.3.4　リスト

Prolog ではデータ構造にリストを扱うことができる．Prolog プログラムでは，リストに新たなオブジェクトを加える，２つのリストを結合するといった操作を行うことができる．

図 2.4 にリスト処理の一つであるメンバーシップ関係の例を示す．メンバーシップ関係とはリスト中に指定したオブジェクトが存在するか確認する処理である．リスト“[colorado, nebraska, kansas]”は“kansas”を含む．これにより，“member(kansas, [colorado, nebraska, kansas]).”は“true.”となる．一方，リスト“[colorado, nebraska, kansas]”は“georgia”を含まないため，“member(kansas, [colorado, nebraska, kansas]).”は“false.”となる．

2.4　提案する深層学習による記号処理と質問応答システム

本節では，深層学習を用いて記号処理を学習する手法について説明する．続いて，深層学習による記号処理を利用して，知識ベースから質問応答システムを構築する手法について説明す

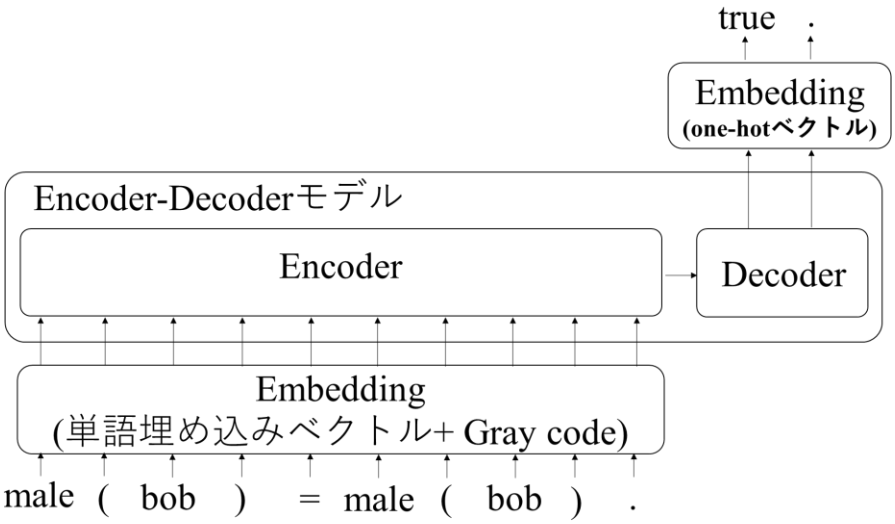


図 2.5　提案するマッチングの学習モデル

る．

2.4.1　深層学習による記号処理

NMT の手法の一つである Encoder-Decoder モデル [46,53-54] と単語分散表現 [61-63] を組み合わせて，2.3 節で述べたマッチング，導出並びにメンバーシップ関係を学習する手法を提案する．Encoder-Decoder モデルはある系列を他の系列に変換することを学習可能なモデルである．Encoder-Decoder モデルを用いることにより，Graph Convolution [22-23] を使用した際に生じる原子論理式に含む項の数，及び論理式に含む原始論理式の数の制約を設ける必要がなくなる．事前にメタルールをネットワークに与える必要もない．また前述のように，単語分散表現を用いることで，未学習の定数記号を意味的に近い既知の定数記号に置き換えて扱うことができる．

(1) マッチングの学習

図 2.5 に提案システムで用いるマッチングの学習モデルを示す．本モデルへの入力はマッチング可否の質問である．マッチングした場合は“true.”，マッチングしなかった場合は“false.”を出力するように学習を行う．Prolog 処理系では，オペレータ”=”は単一化を意味するが，本研究においてはマッチングを意味する．質問には変数を含むことを許容するが，

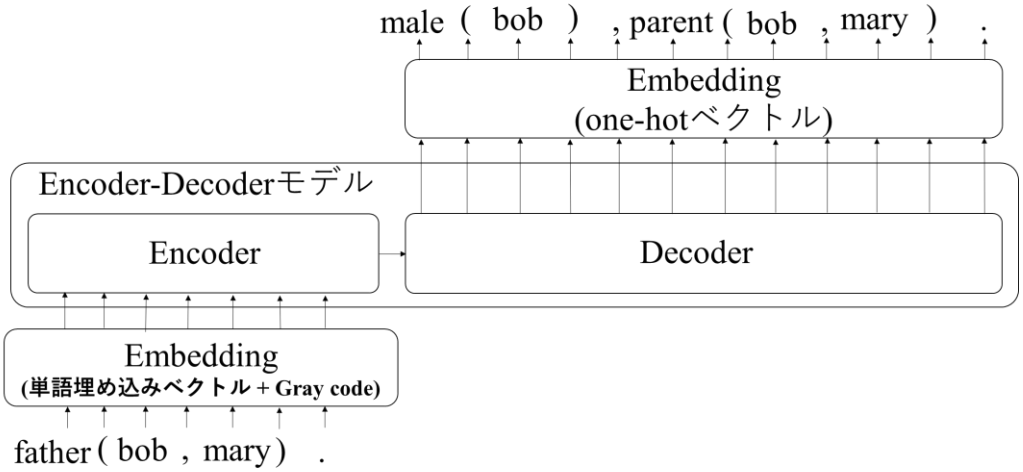


図 2.6　提案する導出の学習モデル

Prolog 処理系とは異なり変数の代入は行わず，マッチングの可否判定のみを行う．例えば“male(X) = male(tom).”は“X = tom.”ではなく，“true.”を出力するようにする．代入はマッチングの可否判定後に，モデル外で処理を行うことで対応する．

マッチングの学習モデルへマッチング可否の質問単語列を入力すると，入力用 Embedding 層で単語埋め込みベクトルによる 300 次元の Word Embedding と Gray code [64] による Word Embedding の結合ベクトルに変換される．“male”などの固有名詞ではない定数記号は単語埋め込みベクトルで Word Embedding し，“(”や“)”，“,”，“.”などの論理記号，及び“bob”のような固有名詞の定数記号は Gray code で Word Embedding する．Gray code は，隣接する符号間のハミング距離が 1 という特性を持っている．これは，論理記号及び固有名詞の定数記号は単語埋め込みベクトルで表現できないことによる処置である．Gray code の代わりに PN（Pseudo random Noise）符号を用いることも考えられるが，本章における研究では実装を簡単にするため Gray code を適用する．定数記号の表現に単語埋め込みベクトルを用いるため，学習データに含まれない単語が入力された場合でも出力を得ることが出来る．

続いて単語埋め込みベクトルと Gray code の結合ベクトルは Encoder-Decoder モデルの Encoder へ渡される．Encoder-Decoder モデルの Decoder からの出力は，出力用 Embedding 層で one-hot によって Word Embedding され，マッチング結果の単語列が出力される．

Data Augmentation前のTraining Set

head	father(bob , X).
body	male(bob), spouse(bob, X).

Data Augmentation後のTraining Set

head	father(bob , X).
body	male(bob), spouse(bob, X).
head	father(bob-1 , X).
body	male(bob-1), spouse(bob-1, X).
head	father(bob-2 , X).
body	male(bob-2), spouse(bob-2, X).

⋮

図 2.7　Data Augmentation の例

(2) 導出の学習

図 2.6 に提案システムで用いる導出の学習モデルを示す．本モデルへ規則の頭部が入力されると，規則の本体を出力するように学習が行われる．例えば規則“father(bob , mary) :- male(bob), parent(bob, mary).”の場合，提案モデルへの入力は “father(bob , mary).”，出力は“male(bob), parent(bob, mary).”となる．規則に変数を含むことを許容する．

導出学習モデルへ規則頭部の単語列を入力すると，マッチングの学習と同様の方法で入力用の Embedding 層で単語埋め込みベクトルによる Word Embedding と Gray code による Word Embedding の結合ベクトルに変換される．固有名詞ではない定数記号は単語埋め込みベクトルで Word Embedding され，論理記号及び固有名詞の定数記号は Gray code で Word Embedding される． 続いて単語埋め込みベクトルと Gray code の結合ベクトルは Encoder-Decoder モデルの Encoder へ渡される．Encoder-Decoder モデルの Decoder からの出力は出力用の Embedding 層によって one-hot で Word Embedding され，規則本体の単語列が出力される．

既存の知識ベースを用いて Encoder-Decoder モデルで導出を学習する場合，学習データの数が十分ではない場合がある．このような場合は，固有名詞を増やすことにより Data Augmentation を行う．例えば， “bob”から“bob-1”，“bob-2”といった架空の固有名詞を生成し，図 2.7 に示すように学習データを増加させる．

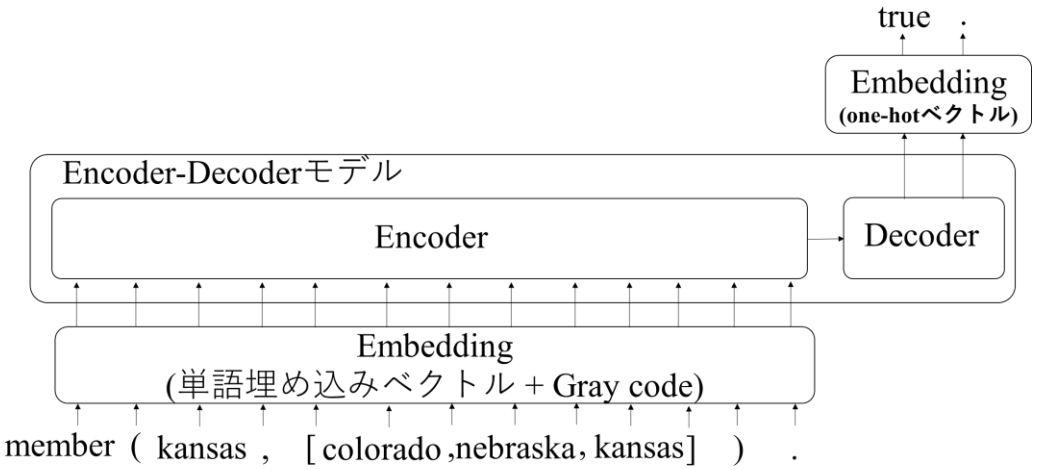


図 2.8　提案するメンバーシップ関係の学習モデル

(3) メンバーシップ関係の学習

図 2.8 にメンバーシップ関係の学習モデルを示す．本モデルへの入力は“member(kansas , [colorado, nebraska, kansas]).”のようなメンバーシップ関係の質問である．リスト中にオブジェクトが存在した場合は“true.”，存在しなかった場合は“false.”を出力するように学習が行われる．なお質問に変数は含めない．

メンバーシップ関係の学習モデルへメンバーシップ関係の質問単語列が入力されると，マッチングの学習と同様の方法で入力用 Embedding 層で単語埋め込みベクトルによる Word Embedding と Gray code による Word Embedding の結合ベクトルに変換される．続いて単語埋め込みベクトルと Gray code の結合ベクトルは Encoder-Decoder モデルの Encoder へ渡される．Encoder-Decoder モデルの Decoder からの出力は，出力用 Embedding 層で one-hot によって Word Embedding され，オブジェクトの存在可否結果の単語列が出力される．

また，学習データの数が十分ではない場合は，導出の学習と同様の手法で Data Augmentation が行われる．

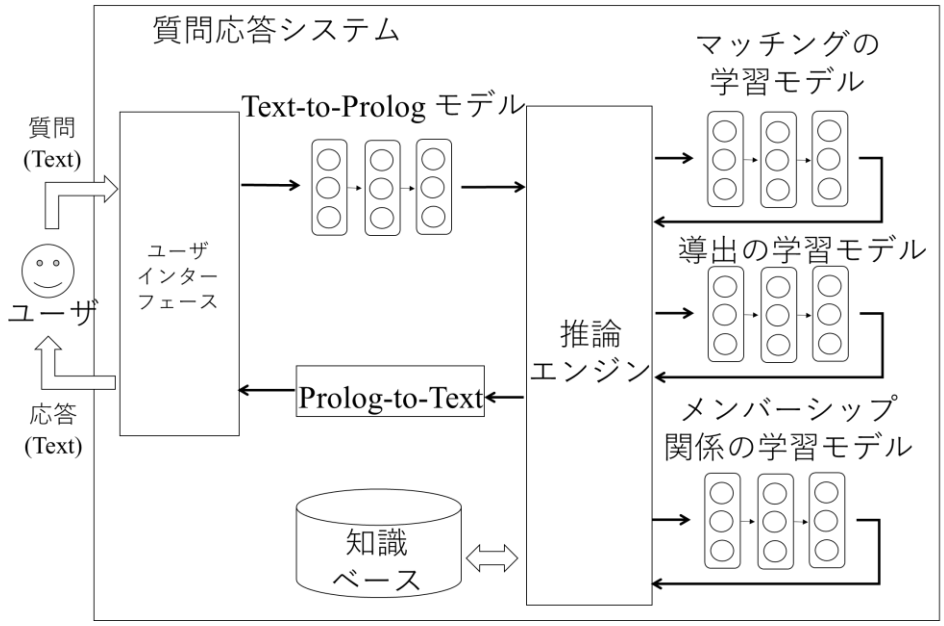


図 2.9　提案する質問応答システムの構成

2.4.2　深層学習による記号処理を用いた質問応答システム

2.3.1 節で示した記号処理の学習モデルを使用して質問応答システムを構築する手法を提案する．ユーザは質問応答システムにテキストを入力することで，一問一答型の質問応答を行うことができる．質問は事実か否かを問う Yes-No 型質問 [65] 及び，What 型質問 [66]，Why 型質問 [67] に対応する．

図 2.9 に提案する質問応答システム構成を示す．図 2.9 中のユーザインターフェースはユーザから入力されたテキストの質問を受け取ると，Text-to-Prolog へ渡す．Text-to-Prolog ではユーザ質問をテキストから Prolog の質問へ変換し，推論エンジンへ渡す．推論エンジンは Prolog で記述された質問に対して回答を推論し，結果を Prolog-to-Text へ渡す．推論エンジンは推論の際に，Prolog で記述された事実が格納されている知識ベースを参照する．Prolog-to-Text は Prolog で記述されたシステム応答をテキストに変換し，ユーザインターフェースへ渡す．ユーザインターフェースはテキストのシステム応答をユーザに提示する．以降で質問応答システムの各構成要素の詳細について説明する．

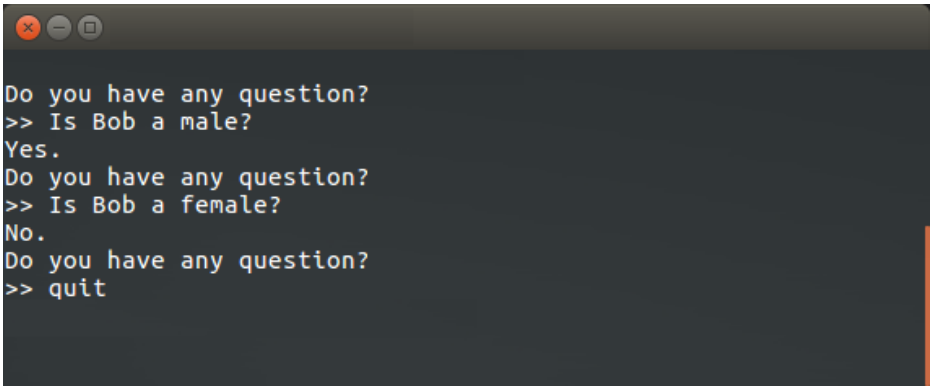


図 2.10　ユーザインターフェース

Yes-No型質問	
Text	Is Bob Tom's father?
Prolog	father(bob, tom).
What型質問	
Text	Who is Tom's father?
Prolog	father(X, bob).
Why型質問	
Text	Why is Bob Tom's father?
Prolog	trace , father(bob, tom).

図 2.11　テキストから Prolog への変換例

(1) ユーザインターフェース

図 2.10 に質問応答システムのユーザインターフェースを示す。ユーザは自然言語で“Is Bob a male?”と質問を入力すると，質問応答システムは“Yes.”と自然言語で応答を返している．一対の質問応答が完了すると，ユーザは続けて次の質問を返す．

また，Text-to-Prolog や推論エンジンで誤った文法の Prolog が出力され，質問応答システムで回答を導くことができなかった場合は，ユーザインターフェースでは“I cannot answer.”と出力される．

(2) Text-to-Prolog

Text-to-Prolog はテキストで記述されたユーザからの質問を Prolog に変換する．質問パターン別の変換例を図 2.11 に示す．“Is Bob Tom's father?”のような Yes-No 型質問の場合は，

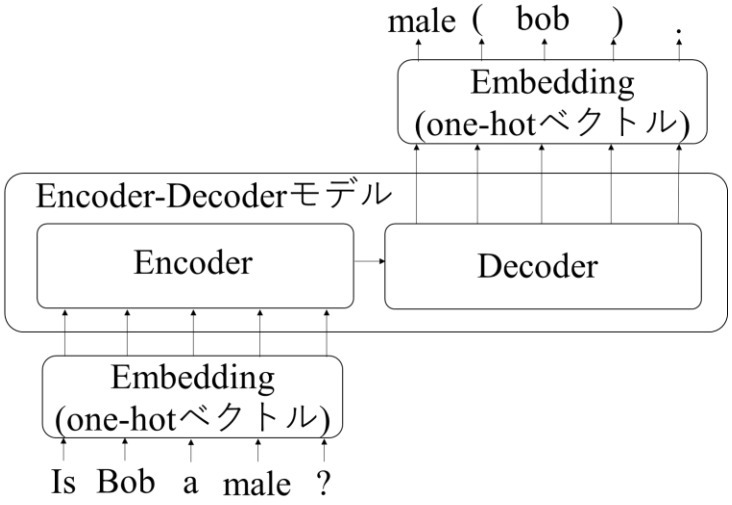


図 2.12　提案する Text-to-Prolog 学習モデル

“father(bob,tom).”のような変数を含まない Prolog の質問文へ変換される．“Who is Tom’s father?”のような What 型質問の場合は，質問対象を変数に置き換えて“father(X, tom).”のように変換される．“Why is Bob Tom’s father?”のような Why 型質問の場合は，“trace, father(bob,tom).”のように先頭に“trace”を付与した Prolog の質問文へ変換される．“trace”は目標の全てのトレースを行う組込み述語を意味している．

テキストから Prolog への変換は，Encoder-Decoder モデルを用いた学習が行われる．深層学習を用いることで，ユーザの様々な言い回しでの質問に対応した，Prolog 質問文への変換が可能となる．

図 2.12 に Text-to-Prolog の学習モデルを示す．本モデルへユーザからの質問をテキストで入力すると，それに対応する Prolog の質問文を出力するように学習が行われる．

Text-to-Prolog 学習モデルへテキストの単語列を入力すると，入力用の Embedding 層で one-hot による Word Embedding が行われる．one-hot ベクトルは Encoder-Decoder モデルの Encoder へ渡される．Encoder-Decoder モデルの Decoder からの出力は出力用の Embedding 層によって one-hot で Word Embedding され，Prolog で記述された質問文の単語列が出力される．

(3) 推論エンジン

推論エンジンは 2.3.1 節で学習したモデルを使用して記号処理を行う．推論エンジンが質問

Algorithm 2.1: 推論エンジンのアルゴリズム

```
Input: a question written in Prolog  GoalList
Output: an answer written in Prolog  FinalGoal
1: NewGoalList ← []
2: HasTrace ← false
3: while not  empty(GoalList) do
4:       Goal ← head(GoalList)
5:       if Goal == "trace"
6:            HasTrace ← true
7:            continue
8:       IsMatching ← false
9:       for Num = 0 to  size(KnowledgeBase) do
10:           if Goal == KnowledgeBase[Num] or matching_model(Goal, KnowledgeBase[Num])
11:                IsMatching ← true
12:                break
13:      if not IsMtaching
14:           NewGoals ← resolution_model(Goal)
15:           MembershipGoal, NotMembershipGoals ← extract ("member", NewGoals)
16:           if not empty(MembershipGoal)
17:                if membership_model(MembershipGoal)
18:                     append(NewGoalList, NotMembershipGoals)
19:                else if
20:                     return false
21:           else if
22:                append(NewGoalList, NewGoals)
23:      else
24:           return false
25: FinalGoal = backtrack(NewGoalList, KnowledgeBase)
26: if HasTrace and FinalGoal
27:      return NewGoalList
28: return FinalGoal
```

図 2.13　推論エンジンの質問応答生成アルゴリズム

文から応答を生成するアルゴリズムを図 2.13 に示す.

本アルゴリズムでは，Text-to-Prolog から受け取った目的リスト *GoalList* をマッチングの学習モデル（図 2.13 10 行目 関数 *matching_model*()），導出の学習モデル（図 2.13 14 行目 関数 *resolution_model*()），メンバーシップ関係の学習モデル（図 2.13 17 行目 関数 *membership_model*()）を用いてマッチング，導出，メンバーシップ関係のチェックを行い，新たな目的リスト *NewGoalList* を得る．（図 2.13 3-24 行目）Text-to-Prolog から受け取った目的リストに未学習の定数記号が含まれていたとしても，マッチングの学習モデルや導出の学習モデルにより既知の定数記号のみから成る新たな目的リストに変換される．以降は Prolog

Yes-No型質問

Prolog	true.
Text	Yes.

What型質問

Prolog	X=tom.
Text	Tom.

Why型質問

Prolog	male(bob), spouse(bob, tom).
Text	Because Bob is a male and Bob is Tom’s spouse.

図 2.14　Prolog からテキストへの変換例

処理系と同様に新たな目的リストに対してバックトラック（図 2.13 25 行目 *backtrack*()）が行われ，その結果 *FinalGoal* を返す．なお，目的リストに“trace”を含む場合は，Why 型質問の回答として，前出の新たな目的リスト *NewGoalList* を返す．

(4) Prolog-to-Text

Prolog-to-Text はシステム応答の Prolog をテキストに変換する．質問パターン別の変換例を図 2.14 に示す． Prolog からテキストへはルールベースでの変換が行われる．

２.４.３　Encoder-Decoder モデル

Encoder-Decoder モデルは NMT で使用されるモデルの一つであり，あるシーケンスから別のシーケンスへの変換に使用される． Attention 付き Seq2Seq [54] および Transformer [46] をマッチングの学習モデル，導出の学習モデル，メンバーシップ関係の学習モデル，および Prolog-to-Text モデルの内部で使用する Encoder-Decoder モデルに適用する． Attention 付き Seq2Seq と Transformer は，よく使用される典型的な Encoder-Decoder モデルである．

Encoder-Decoder モデルを使用して記号処理を行う場合，グラフネットワークやフィードフォワードネットワークとは異なり，原子論理式に含まれる項の数と式に含まれる原子論理式の数に制限はなく，メタルールは不要となる．これは，Encoder-Decoder モデルがシーケンシャルデータを処理できるためである．

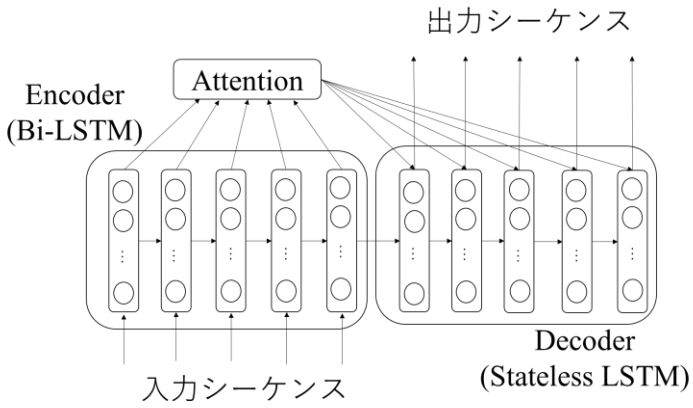


図 2.15　Attention 付き Seq2Seq の構成

(1) Attention 付き Seq2Seq

図 2-15 に Attention 付き Seq2Seq [54] の構成を示す．Attention 付き Seq2Seq は，Encoder，Decoder，Attention 機構の 3 つのブロックで構成されている．Encoder は入力シーケンスを受け取ると，圧縮されたベクトルを返す． Attention 機構は，出力シーケンスのコンテキストに基づいて，入力シーケンスの各単語に与えられる注意の程度を計算する．次に、圧縮されたベクトルが注意によって重み付けされ，加えられる．Decoder は Encoder と Attention 機構から結果の出力ベクトルを受け取ると、出力シーケンスを生成する．

Encoder と Decoder は，リカレントニューラルネットワーク（RNN） [68] の一種である長短期記憶（LSTM） [59] で構成されている． LSTM は従来の RNN では学習できない長期的な依存関係を持つシーケンシャルデータを処理できる．Encoder は，過去の情報だけでなく将来の情報も使用する Bi-LSTM [69] を適用する．使用する Bi-LSTM には，128 次元の出力と 3 つのレイヤーを備える．Decoder は短期記憶を引き継がないステートレス LSTM を適用する．ステートレス LSTM は 128 次元の出力を持ち，活性化関数として Maxout [70] を使用する．

Dropout 率は 0.1，バッチサイズは 128 とし，20 エポックのトレーニングを実施する．最適化には Adam [71] を使用し，パラメータは α =0.001，β_1=0.9，β_2=0.999，$\varepsilon = 1.0 \times 10^{-8}$ とする．

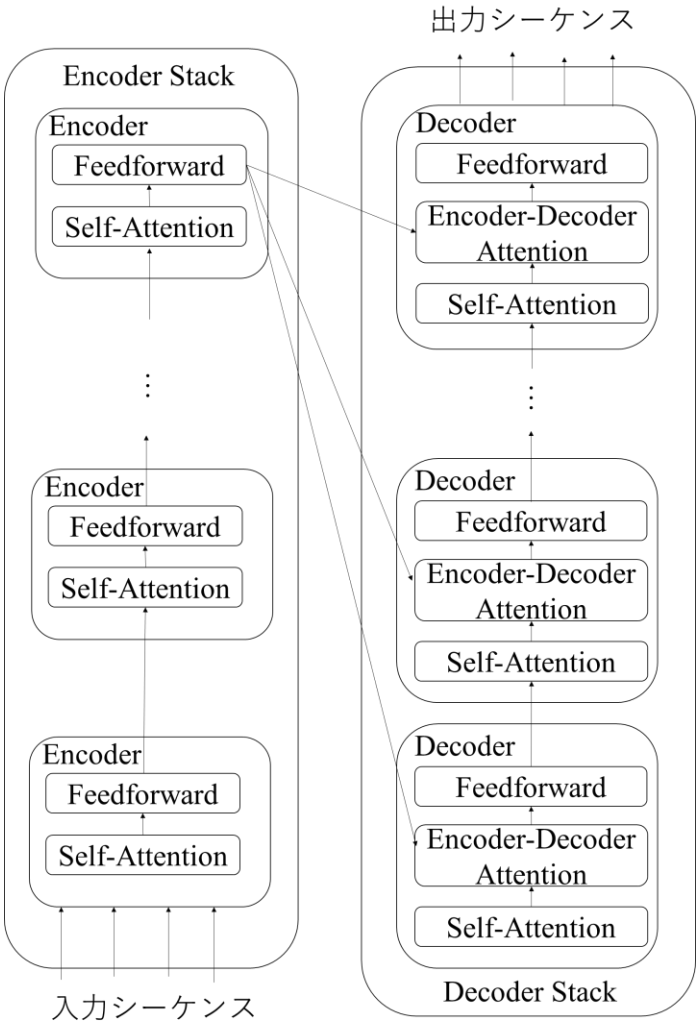


図 2.16　Transformer の構成

(2) Transformer

図 2-16 に Transformer [46] の構成を示す．Transformer は、Encoder Stack と Decoder Stack の 2 つのブロックで構成されている．Encoder Stack は入力シーケンスを受け取り，Decoder Stack は出力シーケンスを返す．Encoder Stack と Decoder Stack は，それぞれ 6 つの Encoder と 6 つの Decoder を備える．

Encoder は，512 次元の出力と 6 つのレイヤーから成る Feedforward と Self-Attention によって構成される．Decoder は，512 次元の出力と 6 つのレイヤーから成る Feedforward，Encoder-Decoder Attention，Self-Attention によって構成される．

```
father ( p3876 , p3877 ) :- male ( p3876 ) , parent ( p3876 , p3877 ) .
mother ( p1013 , p1097 ) :- female ( p1013 ) , parent ( p1013 , p1097 ) .

son ( p4806 , p4791 ) :- male ( p4806 ) , parent ( p4791 , p4806 ) .
daughter ( p3428 , p3357 ) :- female ( p3428 ) , parent ( p3357 , p3428 ) .

husband ( p2968 , p3063 ) :- male ( p2968 ) , spouse ( p2968 , p3063 ) .
wife ( p4801 , p804 ) :- female ( p4801 ) , spouse ( p4801 , p804 ) .
```

図 2.17　Kinsources の知識ベース

Self-Attention は，単一シーケンス内でどの位置に注意を向けるか関連付けるために使用される． Encoder-Decoder Attention は，Decoder が入力シーケンスのどの位置に注意を向けるか関連付けるために使用される． Feedforward は活性化関数に *Leaky ReLU* [72] を使用する．

Dropout 率は 0.1，バッチサイズは 48 とし，100 エポックのトレーニングを実施する．最適化には Adam [71] を使用し，パラメータは α=5.0 × 10^{-5}，β_1=0.9，β_2=0.98，$\varepsilon = 1.0 \times 10^{-9}$とする．

2.4.4　単語分散表現

単語分散表現は単語を高次元のベクトルで表現する技術である．単語分散表現を実現するツールの一つに Word2Vec がある．Word2Vec は、ニューラルネットワークを使用して大量のテキストデータから各単語のベクトル表現を得る．Word2Vec には，ある単語から周辺の単語を予測する Skip-Gram と周辺の単語から中心の単語を予測する CBOW（Continuous Bag-of-Words）の２種類のモデルがある．Skip-Gram 及び CBOW は文の文脈情報を用いない単語分散表現である．一方で，文の文脈情報を用いる単語分散表現に BERT [73] がある．

本研究では， GoogleNews-vectors-negative300 [61-63] と呼ばれる Word2Vec による文の文脈情報を用いないモデルをマッチングの学習モデル，導出の学習モデル，およびメンバーシップ関係の学習モデルに適用する．GoogleNews-vectors-negative300 は，約 1,000 億語のデータセットでトレーニングされている．モデルは，300 万の単語とフレーズについて 300 次元のベクトルを保持している．尚，GoogleNews-vectors-negative300 は Skip-Gram と CBOW のどちらのモデルを採用しているかは明らかにされていない．

知識ベースの中には，知識ベースに格納されている知識を説明する文が存在するものがある．特にそのような知識ベースについては，文脈情報を利用可能な BERT を適用することで性能

表 2.1　Kinsources によるモデルの学習結果（未学習データは含まない）

	マッチング	導出	Text-to-Prolog
Training Set	81,115	51,278	80,000
Validation Set	10,140	6,411	10,000
Test Set	10,139	6,409	10,000
従来のProlog処理系による正答率	1.0000	1.0000	-
Attention付きSeq2Seqを採用したモデルの正答率	1.0000	0.9995	0.9988
Transformerを採用したモデルの正答率	1.0000	0.8582	0.5653

が向上する可能性があるが，本研究ではまずは比較的単純な Word2Vec のツールを用いたモデルを適用する．

2.5　評価実験

Prolog で記述された知識ベースを利用してモデルの学習及び質問応答システムの構築を行ない，その評価実験を行った．具体的には，グラフ構造を持つ 2 種類の知識ベースを用いた実験を行った．

2.5.1　Kinsources を用いた実験

知識ベースの作成に，Kinsources［74］のデータセットを用いた実験を実施した．Kinsources は血族関係を表すデータの集合体であり，知識はグラフ構造に表現されている．図 2.17 に知識ベースの例を示す．Prolog で記述された知識ベースは 5,887 のアトムと 10 種類の述語から成り，リストは含まない．

この知識ベースをもとに，モデルの学習データを生成した．Kinsources の学習データ数は十分大きかったため，Data Augmentation は行わなかった．Text-to-Prolog の学習データは Yes-No 型質問，主語が質問対象の What 型質問，目的語が質問対象の What 型質問，Why 型質問を含み，それぞれ 2 パターンずつの言い回しを用意した．

モデルの学習結果を表 2.1 に示す．表 2.1 中の Test Set には未学習データは含まれていない．尚，正答率は式（2.1）によって計算される．推論結果に文法的な誤りが含まれる場合は，*正答に正確に一致した数*に含めない．

表 2.2　マッチングの学習モデルにおける未学習データの正答率

	male → man	female → woman
未学習データのTest Set	1,150	1,198
従来のProlog処理系による正答率	0.0000	0.0000
Attention付きSeq2Seqを採用したモデルの正答率	0.9478	0.8389
Transformerを採用したモデルの正答率	0.9861	0.4992

表 2.3　導出の学習モデルにおける未学習データの正答

	father → dad	mother → mom
未学習データのTest Set	1,272	1,215
従来のProlog処理系による正答率	0.0000	0.0000
Attention付きSeq2Seqを採用したモデルの正答率	0.9969	1.0000
Transformerを採用したモデルの正答率	0.7547	0.7720

$$正答率 = 正答に正確に一致した数 / Test\ Set の数 \tag{2.1}$$

Attention 付き Seq2Seq と Transformer を Encoder-Decoder モデルに適用し，それぞれの正解率を算出した．更に，従来の Prolog 処理系で同様の実験を行い，提案システムと正解率を比較した．

表 2.1 に示したマッチングの学習モデルの Test Set の内，述語に“male”を含むものを“man”，“female”を含むものを“woman”に置き換えて未学習データを生成した．未学習データ を用いたマッチングの結果を表 2.2 に示す．

表 2.1 に示した導出の学習モデルの Test Set の内，述語に“father”を含むものを“dad”，“mother”を含むものを“mom”に置き換えて未学習データを生成した．未学習データを用いた導出の結果を表 2.3 に示す．

表 2.1 に示した各モデルを利用して質問応答システムを構築した．本実験は Encoder-Decoder モデルに Attention 付き Seq2Seq を適用した．実験では，質問種別と応答種別の組み合わせごとに質問文を生成し，質問応答システムに与えた．質問種別は Yes-No 型質問，What

表 2.4　質問応答システムの正答率

Yes-No型質問	**Positive**	**Negative**
Test Set	1,000	1,000
正答率	0.985	0.749
What型質問	**Positive**	**Negative**
Test Set	1,000	1,000
正答率	0.907	0.614
Why型質問	**Positive**	**Negative**
Test Set	1,000	1,000
正答率	0.894	1.000

表 2.5　GPU メモリ使用量

	マッチング	**導出**	**Text-to-Prolog**
学習時のGPUメモリ使用量（MB）	775	691	649
推論時のGPUメモリ使用量（MB）	387	395	404

型質問，Why 型質問の 3 種類とした．応答種別は Positive（Yes-No 型質問は “Yes.”， What 型質問もしくは Why 型質問は”Bob.”等の具体化された応答が得られる場合）と Negative（Yes-No 型質問は “No.”， What 型質問もしくは Why 型質問は応答が不可能な場合）の 2 種類とした．質問文は表 2.2 及び表 2.3 に示した未学習データを含み，質問種別ごとに 2 パターンずつの言い回しでランダムに生成した．質問種別を分けずに，ランダムに質問文を生成すると，Negative な質問文がほとんどを占めてしまうため，Positive と Negative に分けて実験を行うこととした．質問文を入力した際の正答率を表 2.4 に示す．

また，本実験における学習時及び推論時の各モデルの GPU メモリ使用量を表 2.5 に示す．尚，本実験では CPU は Intel Core i7-7700K，GPU は Nvidia GeForce GTX1080 Ti のマシンを使用した．

2.5.2　Geoquery を用いた実験

知識ベースの作成に，Geoquery [75] のデータセットを用いた実験を実施した．Geoquery はアメリカの地理データであり，知識はグラフ構造に表現されている．図 2.18 に知識ベースの例を示す． Prolog で記述された知識ベースは 574 のアトムと 8 種類の述語から成り，本体

```
adjacent ( alabama , tennessee ) :-
        border ( alabama , [ tennessee , georgia , florida , mississippi ] ) ,
        member ( tennessee , [ tennessee , georgia , florida , mississippi ] ) .

located ( colorado-river , colorado ) :-
        river ( colorado-river , [ colorado , utah , arizona , nevada , california ] ) ,
        member ( colorado , [ colorado , utah , arizona , nevada , california ] ) .

located ( mckinley-mountain , alaska ) :- mountain ( mckinley-mountain , alaska ) .

located ( route-15 , montana ) :-
        road ( route-15 , [ montana , idaho , utah , arizona , nevada , california ] ) ,
        member ( montana , [ montana , idaho , utah , arizona , nevada , california ] ) .

located ( superior-lake , michigan ) :-
        lake ( superior-lake , [ michigan , wisconsin , minnesota ] ) ,
        member ( michigan , [ michigan , wisconsin , minnesota ] ) .
```

図 2.18　Geoquery の知識ベース

表 2.6　Geoquery によるモデルの学習結果（未学習データは含まない）

	導出	メンバーシップ関係	Text-to-Prolog
Data Augmentation前のTraining Set	496	1,256	8,000
Data Augmentation後のTraining Set	45,136	38,936	-
Validation Set	63	158	1,000
Test Set	62	157	1,000
従来のProlog処理系による正答率	1.0000	1.0000	-
Attention付きSeq2Seqを採用したモデルの正答率	0.8226	0.9172	1.0000
Transformerを採用したモデルの正答率	0.4603	0.9873	0.4100

にリストを含む．また，図 2.18 中で述語“located”を頭部に持つ規則が示すように，本知識ベースの表現には選言が含まれる．

この知識ベースをもとに，モデルの学習データを生成した．導出モデル及びメンバーシップ関係モデルは，学習データ数が少ないため，Data Augmentation を行った．また，導出モデルの学習データでは，リストを変数Lに置き換えを行っている．これは，リストが規則本体に現れる論理記号から一意に決定することができることから，学習を容易にするための処置であ

表 2.7　導出の学習モデルにおける未学習データの正答

	adjacent→adjoining	located→situated
未学習データのTest Set	18	44
従来のProlog処理系による正答率	0.0000	0.0000
Attention付きSeq2Seqを採用したモデルの正答率	1.0000	0.7273
Transformerを採用したモデルの正答率	0.0556	0.2667

表 2.8　質問応答システムの正答率

Yes-No型質問	Positive	Negative
Test Set	100	100
正答率	0.66	0.96

表 2.9　GPU メモリ使用量

	導出	メンバーシップ関係	Text-to-Prolog
学習時のGPUメモリ使用量（MB）	601	959	599
推論時のGPUメモリ使用量（MB）	408	423	358

る.

Text-to-Prolog の学習データは Yes-No 型質問を含み，2 パターンの言い回しを用意した.

モデルの学習結果を表 2.6 に示す．表 2.6 の Test Set には未学習データは含まれていない．Attention 付き Seq2Seq と Transformer を Encoder-Decoder モデルに適用し，それぞれの正解率を算出した．更に，従来の Prolog 処理系で同様の実験を行い，提案システムと正解率を比較した.

表 2.6 に示した導出モデルの Test Set の内，述語に“adjacent”を含むものを“adjoining”，“located”を含むものを“situated”に置き換えて未学習データを生成した．未学習データを用いた導出の結果を表 2.7 に示す.

表 2.6 に示した各モデルを利用して質問応答システムを構築した．本実験は Encoder-

Decoder モデルに Attention 付き Seq2Seq を適用した．実験では，質問種別と応答種別の組み合わせごとに質問文を生成し，質問応答システムに与えた．質問種別は Yes-No 型質問の 1 種類とした．応答種別は Positive(応答が“Yes.”の場合)と Negative(応答が“No.” の場合)の 2 種類とした．質問文は表 2.6 に示した未学習データを含み，質問種別ごとに 2 パターンずつの言い回しでランダムに生成した．質問文を入力した際の正答率を表 2.8 に示す．

また，本実験における学習時及び推論時の各モデルの GPU メモリ使用量を表 2.9 に示す．尚，本実験では CPU は Intel Core i7-7700K，GPU は Nvidia GeForce GTX1080 Ti のマシンを使用した．

2.5.3 Attention 付き Seq2Seq と Transformer の正答率比較

記号処理の学習モデルで，Encoder-Decoder モデルに Attention 付き Seq2Seq と Transformer を採用した場合，正答率に差があるか分析を行った．分析には表 2.1，表 2.2，表 2.3，表 2.6，表 2.7 に含まれるマッチングの学習モデル，導出の学習モデル，メンバーシップ関係の学習モデル，及び Text2Prolog 学習モデルの各正答率を用いた．Attention 付き Seq2Seq を適用した場合の正答率の平均は 0.9374，標準偏差は 0.0887 であった．一方，Transformer を適用した場合の正答率の平均は 0.6346，標準偏差は 0.2943 であった．ウィルコクソン符号付順位和検定の結果，Attention 付き Seq2Seq を採用したモデルと Transformer を採用したモデルの正答率には，有意な差が見られた （$p=0.0076$）．ここでの有意水準は $p<0.05$ とした．

尚，Prolog 処理系の正答率の平均は 0.3333，標準偏差は 0.4714 であった．ウィルコクソン符号付順位和検定の結果，Attention 付き Seq2Seq を採用したモデルと Prolog 処理系の正答率にも，有意な差が見られた （$p=0.016$）．ここでの有意水準は $p<0.05$ とした．

2.5.4 考察

2.5.3 項の結果より，記号処理の学習モデルに Transformer を採用した場合よりも，Attention 付き Seq2Seq を採用した場合の方が，パフォーマンスが良かった．記号処理の学習はプログラム合成と類似しており，構文構造の学習を必要とする．このため，Decoder に隠れ状態を持つ Attention 付き Seq2Seq の方が Transformer よりもパーフォーマンスが良い結果

になったと考えられる．実際，マッチングの学習モデルと導出の学習モデルにTransformerを採用した場合，正答できなかった結果のほとんどは構文の誤りであった．

Attention付きSeq2Seqを使用したマッチングの学習モデルの正答率は1.00，導出の学習モデルの正答率は0.823以上であった．何れのモデルに対しても原始論理式や論理式に事前に制約を与えずに高い正答率を実現した．メンバーシップ関係の学習モデルは正答率が0.917であり，リストを含むデータ構造を扱うことが可能といえる．以上より，本提案手法は，様々なデータ形式を取り扱うための高い表現力を備えていることが示された．

Attention付きSeq2Seqを使用した未学習データに対するマッチングの学習モデルの正答率は0.839以上，導出の学習モデルの正答率は0.727以上と高い値であった．メンバーシップ関係の学習モデルについては，リストの要素が固有名詞のみであったため，今回の実験では未学習データの正答率を測定することができなかった．しかしながら，メンバーシップ関係の学習モデルは，マッチングの学習モデルや導出の学習モデルと同様の学習モデルを採用していることから，同様に未学習データに対応可能なことが示唆される．また，本実験でのマッチングの学習モデル及び導出の学習モデルは述語にのみ未学習データが含まれるケースであったが，実験の結果からアトムに未学習データが含まれる場合でも対応可能なことが示唆される．以上より，単語分散表現を利用することで，本提案手法の，未学習データに対する高いロバスト性が示唆される．

Geoqueryにおける導出モデルの学習においては，もともと使用可能なデータは496，メンバーシップ関係モデルの学習に使用可能なデータは1,256と非常に少ない数であった．しかしながら，Attention付きSeq2Seqを使用した場合，正答率はそれぞれ0.823，0.913であり，提案手法により，小規模なデータからも学習可能なことが示された．

更に，提案システムを従来のProlog処理系と比較した場合，前者が未知のデータを処理できることは明らかである．従来のProlog処理系には手動でルールが組み込まれているため，既知のデータを使用する場合，マッチングの学習モデル，導出の学習モデルおよびメンバーシップ関係の学習モデルの正答率は1.00となる．しかし，未知のデータを使用した場合，生徒率は0.00となった．提案システムは，既知のデータに対して1.0未満の正答率とはなるものの，高いパフォーマンスを達成できるだけでなく，未知のデータを処理することもできる．

質問応答システムの実験結果は，正答率の平均値が0.846，標準偏差が0.141であった．既存の知識ベースをもとにして，実用性の高い質問応答システムを構築することができたと考えられる．今回システムに組み込んだ導出モデルは，導出が不可能な事例の学習は行っていない．

もし導出モデルで導出不可能か否かの判定が可能になれば，今回最も低かった What 型質問の Negative な文を入力した際の正答率は改善する可能性がある．

また，各記号処理の学習モデルについて学習時及び推論時の GPU メモリ使用量を測定した結果，比較的小規模なメモリ量で実現することができている．一方で，本研究で推論エンジンが使用するメモリ使用量に制限を設けていない．今後，大規模な知識ベースを対象として推論を行おうとした際に，バックトラックを大量に繰り返す場合は，Prolog処理系と同様に推論エンジンにガベージコレクション[76-78] 等のメモリ管理の実装が必要になると考えられる．

2.6 まとめ

本章では，深層学習を用いて記号処理を学習する手法並びに，その学習済みモデルを使用して質問応答システムを構築する手法を提案した．記号処理の学習モデルに関する実験結果から，提案手法は高い表現力及び高いロバスト性を備え，小規模なデータからも学習可能なことを示すことができた．本提案手法は，単語分散表現の利用によって未学習のデータを含んでいても演繹推論を可能とした初めての事例でもある．また，質問応答システムの実験結果から，Prolog で記述された知識ベースから実用に即した質問応答システムが構築可能なことを示すことができた．これは，従来のコネクショニズムによる手法では非常に実現が困難である．

本章における研究は，従来の人工知能研究の領域とニューラルネットワーク研究の領域に跨る新たな研究領域といえる．実験結果を通し，この新たな研究領域に対する研究の有用性を示唆することもできたと考えられる．

第３章 深層学習による記号処理を用いた類推

３.１　序論

これまで深層学習を用いた記号推論の研究 [24-26,33,42-44,79] は演繹推論と帰納推論が中心であった．しかしながら，これらの研究では規則のテンプレートを事前に手動で作成して，モデルに与える必要がある．テンプレートには例えば，”#1(X,Y):-#2(X,Z),#3(Z,Y).” といったものがある．これらの研究では，テンプレートに一致する規則のみが推論の対象となるため，Web 上の大量のデータから未知の規則を推測するには不十分であると考えられる．また，Web 上にある膨大なデータから効率よく推論を行うには，演繹推論や帰納推論だけでなく，従来の人工知能研究で行われてきた類推や仮説推論といった手法も重要になる．そこで，本章における研究では深層学習を用いた類推に取り組むこととした．類推とは，ある事例と他のある事例の類似性から新しい事例の解決を図る推論である．これまでにニューラルネットワークを用いた類推の研究 [80-83] も行われているが，記号処理に一階述語論理を用いて類推を行うのは本章における研究が初めてである．

一階述語論理を使用した類推に関連する研究の 1 つに，Probabilistic Logic Model（PLM）[84-85] を使用した転移学習がある． PLM を使用した転移学習は，あるドメイン用に学習したモデルを使用して，全く異なるドメインの学習モデルを生成することを目的としている．PLM を使用した転移学習は，学習したモデルに含まれる全ての規則を転移して別のドメインの仮説を生成するため，仮説の数は膨大になる．一方，本章における研究のアプローチは，学習済みモデルを使用して未知の規則を発見することを目的としている．類推は，類似性を使用することにより，新しい規則を発見するための仮説空間の検索を限定することができる．転移学習とは異なり，類推は特定のドメインのほとんど全ての規則を発見することはできないが，新しい規則を効率的に発見することが可能である．更に，類推はあるドメインで学習したモデルから別のドメインの新しい規則を発見するだけでなく，同じドメイン内の未知の規則も発見することができる．

そこで，本章における研究ではオブジェクト間の関係性の類似及びオブジェクト間の類似に着目することで，一階述語論理の規則の類推を深層学習により実現し，

1）既知の規則を含む知識ベースから未知の規則の推論を可能とする

2）知識ベースのサイズが大きい場合でも未知の規則を効率的に抽出可能とする

という従来研究にはない特長を持った推論を行う．本提案手法は，深層学習を使用した一階述語論理に基づく類推の初めての事例である．

本章ではまず，3.2 節で本章における研究と関連する研究について述べる．3.3 節で学習対象である類推について述べ，3.4 節では，深層学習によって類推を行うシステムを提案する．3.5 節では，提案手法の実験結果について報告する．尚，本章における研究は [86] の研究をベースとして行った．

3.2 関連研究

3.2.1 類推

これまで人工知能や認知科学の研究では，人間が行うような類推を機械に行わせる研究 [87-94] が行われてきた．オブジェクト間の関係の類似性をもとに類推を行う構造写像理論 [88] が提案され，構造写像理論に基づいて類推を行う推論エンジン [89] が開発された．更に，オブジェクト間の関係の類似性に加えて，オブジェクト同士の類似性や問題解決方法の類似性を用いた推論エンジン [90] の開発も行われた．また，問題解決の目標の類似性に着目したプラグマティックな類推の研究 [95-96] も行われた．

近年は，深層学習を用いた類推によって知識グラフを学習する研究 [96-97] が行われている．しかしながら，これらの研究では一階述語論理を扱うことは困難である．

3.2.2 ニューラルネットワークによる記号推論

深層学習が登場する以前より，多層ニューラルネットワークを用いて記号処理を学習し，推論を行う研究が行われてきた．命題論理を対象とした推論の研究 [34-36] や一階述語論理を対象とした推論の研究 [37-41] が行われてきた．また，類推を扱った研究 [82-83] も行われた．

深層学習登場後は，グラフニューラルネットワークを用いて，一階述語論理に基づく演繹推論や帰納推論の研究 [24-25,42-44] が行われた．その後，フィードフォワードネットワークを用いた研究 [26] や再帰型ニューラルネットワークを用いた研究 [33,79] も行われるようになった．これらの研究では，事前にモデルに規則のテンプレートや規則に関する情報を与える必要があり，Web 上の大量のデータから未知の規則を推論するには十分とは言えない．また，類推については，画像や単純な記号を対象とした研究 [80-81] は行われているものの，一階述語論理を対象として推論を行うものはない．

3.2.3 転移学習

類推に関連する研究の一つに転移学習がある．2000 年代以降，記号表現を用いた転移学習に関する研究 [84-85,99-100] が行われてきた．その中で述語論理を用いた PLM [84-85] による転移学習の研究がある．PLM を使用した転移学習では，あるドメインの PLM に基づいて，別のドメインの PLM が作成される．PLM を用いた転移学習では，モデルに含まれる全ての規則を使用して別のドメインのモデルを生成するため，仮説の数が増大するという問題がある．手作業による仮説探索の範囲をヒューリスティックに制限するための研究 [101] も行われている．このため，PLM を使用した転移学習を使用して、Web 上の膨大な情報から新しい規則を発見することは困難である．

3.2.4 ゼロショット学習

類推に関連するもう一つの研究にゼロショット学習 [102-103] がある．ゼロショット学習は，未知のカテゴリが含まれている場合，それを分類することを目的としている．入力データと既知のカテゴリ間の類似性を測定することにより，未知のカテゴリを含むデータか分類する．ゼロショット学習のいくつかの研究では，類似性を測定するために Word Embedding を使用している [104-105]．また，類似性を測定するために人間の知識を用いる研究もおこなわれている [106-107]．しかしながら，ゼロショット学習では未知のカテゴリを分類できるが，知識ベースから未知の規則を推論することは困難である．

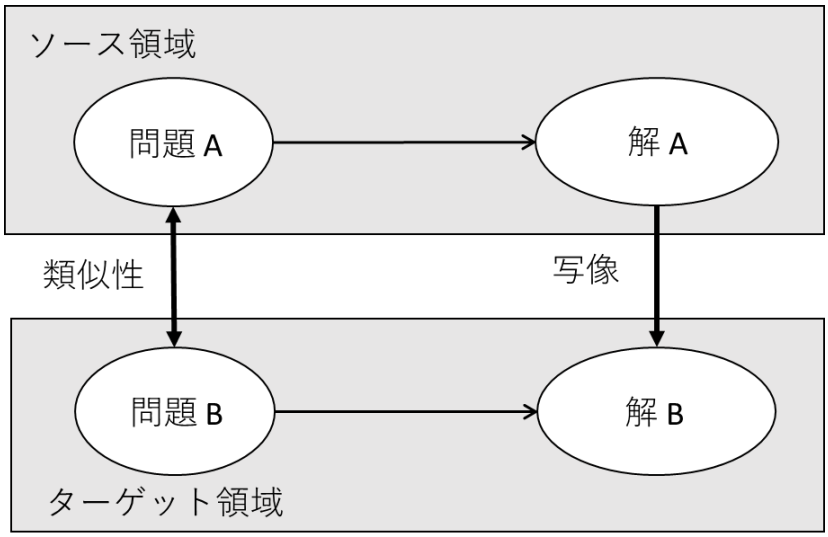


図 3.1　類推の概念図

3.3　記号処理による類推

本論文では記号処理に一階述語論理のサブセットである Prolog [60] を用いて類推を行う．本節では Prolog の論理式をもとに類推を行う方法について述べる．

3.3.1　類推の仕組み

図 3.1 に類推の概念図を示す．ターゲット領域は性質を明らかにしたい領域であり，ソース領域は類推の対象となる領域である．ソース領域の問題 A とターゲット領域の問題 B に類似性がある場合，問題 A の解である解 A は問題 B の解である解 B に写像される．ターゲット領域とソース領域の問題の類似性を発見することにより，類推が行われる．

類似性には，(1) ターゲット領域とソース領域内のオブジェクト間の類似性，および (2) オブジェクト間の関係の類似性の 2 種類がある．規則の類推は，関係の類似性に基づいて類推を実行し，マッチングの類推は、オブジェクトの類似性に基づいて類推を実行する．提案システムでは，規則の類推とマッチングの類推の両方を組み合わせることにより，推論能力を向上させる．また，提案システムは単語分散表現を使用，オブジェクトの類似性と関係の類似性を

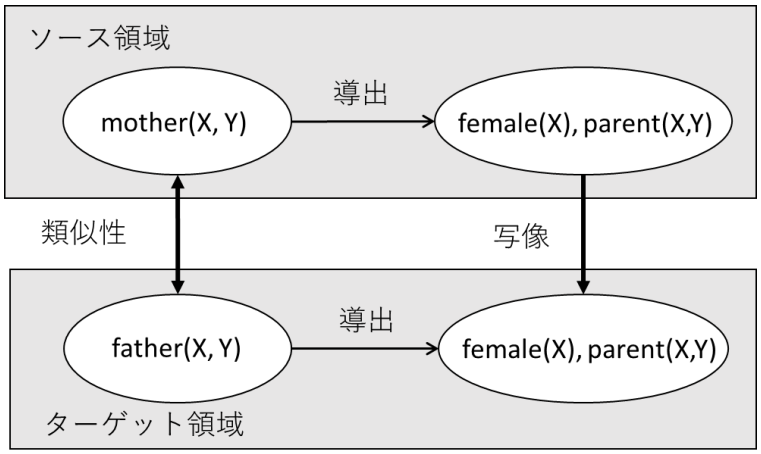


図 3.2　規則の類推の例

測定する．

3.3.2　規則の類推

類推の仕組みを Prolog の規則に適用する方法について述べる．図 3.2 に規則の類推の例を示す．ターゲット領域には式(3.1)に示す規則が存在し，ソース領域に式(3.2)に示す事実が存在する場合を考える．式(3.1)は X が女性かつ，X は Y の両親であれば，X は Y の母親であるという規則である．Prolog プログラムでは，X や Y などの大文字は変数を意味する．演算子“:-”の左側にある部分を頭部，右側にある部分を本体と呼ぶ．規則は，本体が真である場合，頭部も真となる．演算子“,”は，節の連言を示す．頭部が真となるには，連言で連結された全ての節が真でなければならないことを意味する．式(3.2)は X は Y の父親であるという事実である．

$$mother(X,Y) :- female(X), parent(X,Y). \tag{3.1}$$

$$father(X,Y). \tag{3.2}$$

ここで，式(3.1)の頭部を問題，式(3.1)の本体を解と考える．このとき，式(3.1)の頭部と式(3.2)の間に類似性を見つけることによって，式(3.1)の本体は式(3.2)を頭部に持つ節の本体へ写像される．すなわち，式(3.3)に示す規則を類推することができる．この類推方法を以降，規則の類推と呼ぶことにする．

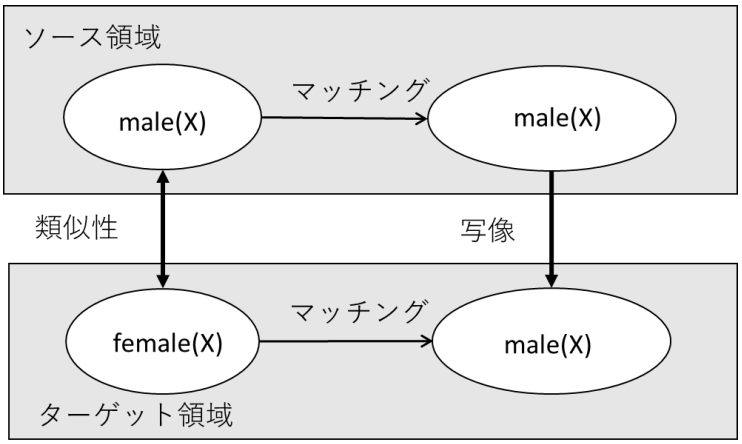


図 3.3　マッチングの類推の例

$$father(X,Y) :- female(X), parent(X,Y). \tag{3.3}$$

ここで，式(3.3)の規則を参照すると，父親は女性であることになり，これは事実に反している．規則の類推だけでは，類似した規則を得ることができるものの必ずしも正しい規則を得ることができるとは限らない．そこで，次節で説明するマッチングの類推を利用する．

3.3.3　マッチングの類推

類推の仕組みを Prolog のマッチングに適用する方法について述べる．マッチングとは与えられた目標が事実と同一かどうか判定を行う処理である．Prolog プログラムでは 2 つの節を演算子“=”で接続することで，2 つの節がマッチングしているか否かを質問することができる．図 3.3 にマッチングに関する類推の例を示す．ターゲット領域では式(3.4)に示す質問が“true”であるとし，ソース領域には式(3.5)に示す事実が存在する場合を考える．

$$male(X) = male(X). \tag{3.4}$$

$$female(X). \tag{3.5}$$

ここで，式(3.4)の左辺の節を問題，式(3.4)の右辺の節を解と考える．このとき，式(3.4)の左辺の節と式(3.5)の間に類似性を見つけることによって，式(3.4)の右辺の節は式(3.5)の解に写像される．す

なわち，式(3.6)の質問が“true”となることを類推することができる．以降，この類推方法をマッチングの類推と呼ぶことにする．

$$female(X) = male(X). \tag{3.6}$$

規則の類推により得られた式(3.3)の規則に対して，式(3.6)のマッチングを適用することにより，式(3.7)の規則を導くことができる．式(3.7)の規則では父親は男性となっており，正しい規則となっている．

$$father(X,Y) :- male(X), parent(X,Y). \tag{3.7}$$

3.4 提案する深層学習による記号処理を用いた類推

本節では，まず深層学習によって 3.3 節で述べた類推を行う手法について説明する．次に，これらの手法を利用して，事実集合から規則を類推する提案システムについて説明する．

3.4.1 深層学習による記号処理

Attention 付き Seq2Seq [54] と単語分散表現 [61-63] を組み合わせたモデル [33] を用いて規則の類推とマッチングの類推を学習する．このモデルは単語分散表現を用いることにより，未学習データを入力した場合でも導出やマッチングが可能な特長がある．類推のポイントであるソース領域にある問題とターゲット領域にある問題の類似性の発見には，単語分散表現を用いたモデルが有効と考える．類似性は Word Embedding されたベクトル同士の距離によって計測される．

(1) 規則の類推の学習

深層学習による規則の類推モデルを図 3.4 に示す．本モデルに規則頭部の単語列を入力すると規則本体の単語列を出力するように学習する．規則には変数を含むことを許容する．

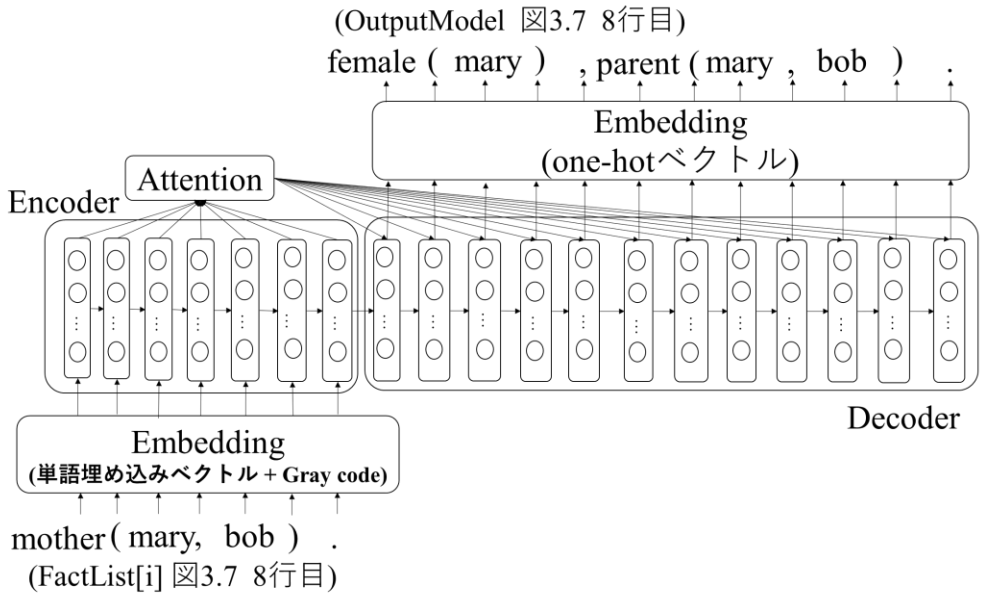


図 3.4　提案する規則の類推モデル

まず，入力用 Embedding 層では，規則頭部の単語列が単語埋め込みベクトルによる Word Embedding と Gray code [64] による Word Embedding の結合ベクトルに変換される．Gray code は，前後に隣接する符号間のハミング距離が必ず 1 という特徴を持っている． Gray code の代わりに PN 符号を用いることも考えられるが，本章における研究では実装を簡単にするため Gray code を適用する．“(” ，“,” ， “)” ， “.” といった論理記号や“mary”，“bob”といったアトムは Gray code により Embedding され，“mother”などの固有名詞ではない述語は単語埋め込みベクトルにより Embedding される．

単語埋め込みベクトルの生成には学習済みの Word2Vec である GoogleNews-vectors-negative300 [61-63] を用いる．これは約 1000 億語のデータセットで学習されている．単語は 300 次元のベクトルで表現され，300 万語のベクトルを保持している．

続いて，入力用 Embedding 層で生成された結合ベクトルは Attention 付き Seq2Seq へと渡される．Attention 付き Seq2Seq は Encoder，Decoder，Attention から構成される．Encoder は入力系列を受け取ると圧縮したベクトルを出力する．Attention は出力系列のコンテキストに基づいて，入力系列内の各単語に与える注意の度合いを計算する．圧縮ベクトルには注意の度合いによる重みが加えられる．Decoder は Encoder と Attention からベクトルを受け取ると出力系列を生成する．

Encoder と Decoder には Long Short Term Memory（LSTM） [59] を用いる．Encoder に

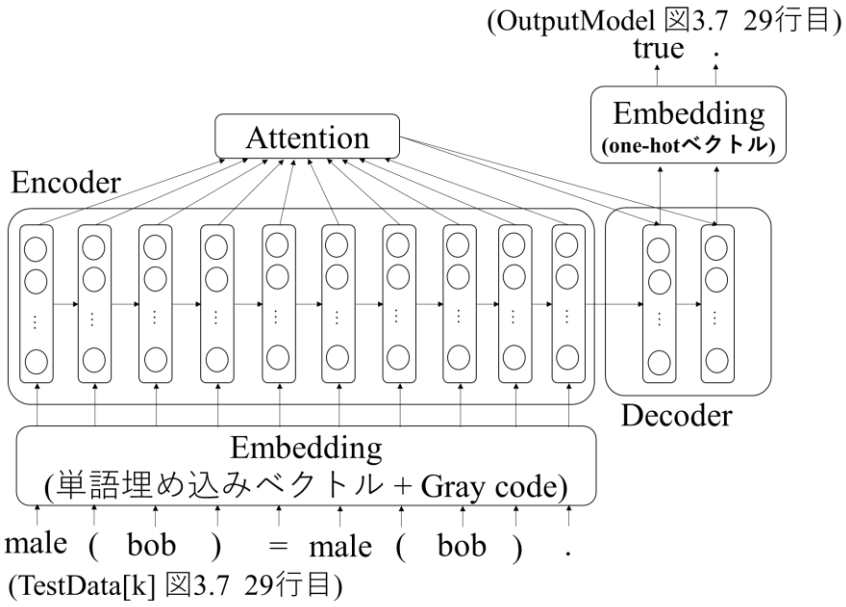


図 3.5　提案するマッチングの類推モデル

は過去の情報と同様に未来の情報も扱える Bi-LSTM [69] を適用する．Bi-LSTM は 3 層から成り，出力層は 128 次元である．Decoder には短期記憶を引き継がない Stateless LSTM を適用する．Stateless LSTM は出力層が 128 次元であり，活性化関数に Maxout [70] を用いる．

最後に，Attention 付き Seq2Seq の出力は出力用 Embedding 層に渡され，one-hot で Embedding されて規則本体の単語列として出力される．

本モデルの学習には，ソース領域の規則を学習データとして与える．本論文中の学習に関しては，Dropout 率は 0.1，バッチサイズを 128 とし，20 エポックの学習を行った．最適化には Adam [70] を用い，$\alpha = 0.001$，$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\varepsilon = 1.0 \times 10^{-8}$とした．学習済みのモデルに対してターゲット領域の事実を入力することで規則の類推を行う．

(2) マッチングの類推の学習

深層学習によるマッチングの類推モデルを図 3.5 に示す．本モデルにマッチング可否の質問を入力するとマッチング可能な場合は“true.”，マッチング不可能な場合は“false.”を出力するように学習する．質問には変数を含むことを許容するが，Prolog処理系とは異なり変数の具現化は行わず，マッチング可否を出力する．例えば式(3.8)の質問がモデルに入力された場合，“X = bob.”ではなく“true.”を出力する．

$$male(X) = male(bob). \tag{3.8}$$

まず，入力用 Embedding 層では，質問の単語列が単語埋め込みベクトルによる Word Embedding と Gray code [63] による Word Embedding の結合ベクトルに変換される．“(”，“,”，“)”，“=”，“.”といった論理記号や“bob”といったアトムは Gray code により Embedding され，“male”などの固有名詞ではない述語は単語埋め込みベクトルにより Embedding する．単語埋め込みベクトルの生成は規則の類推モデルと同じく GoogleNews-vectors-negative300 を用いる．

続いて，入力用 Embedding 層で生成された結合ベクトルは Attention 付き Seq2Seq へと渡される．Attention 付き Seq2Seq の構成は規則の類推モデルと同じである．

最後に，Attention 付き Seq2Seq の出力は出力用 Embedding 層に渡され，one-hot で Embedding されて“true.”もしくは“false.”が出力される．

本モデルの学習には，ソース領域に存在するマッチング可否の質問を学習データとして与える．本論文中の学習に関しては，Dropout 率は 0.1，バッチサイズを 128 とし，20 エポックの学習を行った．最適化には Adam を用い，$\alpha = 0.001$，$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\varepsilon = 1.0 \times 10^{-8}$とした．学習済みのモデルに対してターゲット領域の事実を入力することでマッチングの類推を行う．

3.4.2 提案する深層学習による記号処理を用いた類推システム

本節では，規則の類推モデルとマッチングの類推モデルを利用した提案する類推システムについて説明する．本システムに，ある事実を入力すると，Prolog で記述された知識ベースから類推を行い，その事実を頭部に持つ規則を出力する．例えば，本システムへ式(3.9)に示す事実を入力すると，知識ベースに式(3.9)を頭部に持つ規則がない場合でも， 類推を行い式(3.10)の規則を出力する．

$$father(X, Y). \tag{3.9}$$

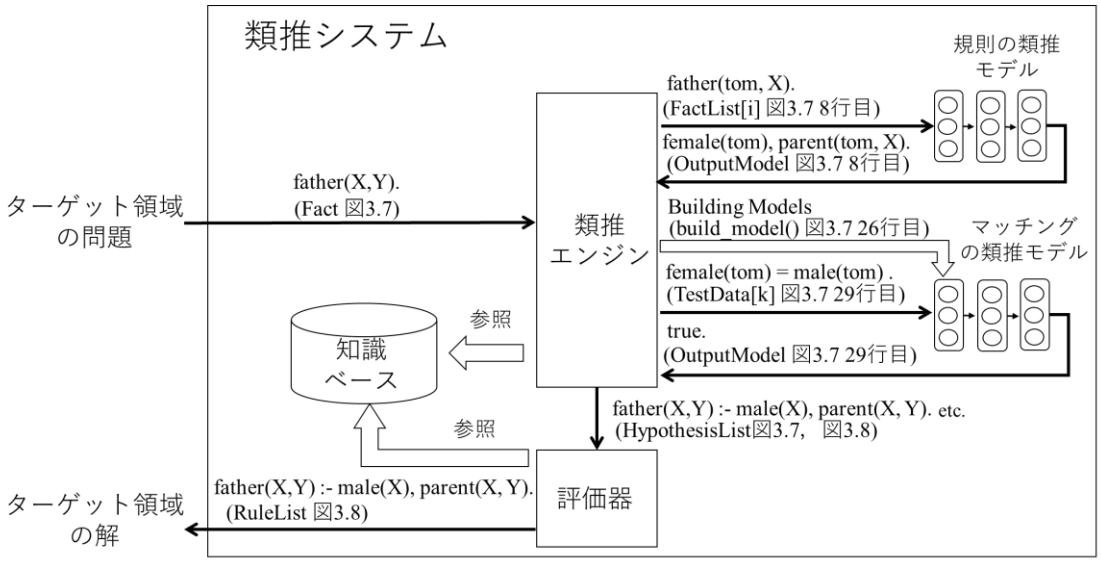


図 3.6　提案する類推システムの構成

$$father(X,Y) : - male(X), parent(X,Y). \tag{3.10}$$

図 3.6 に提案する類推システムの構成を示す．本システムは類推エンジンと評価器から構成されている．類推エンジンは事実を受け取ると，これが頭部となるような規則の仮説を複数生成し評価器に渡す．ここで規則の類推モデルとマッチングの類推モデルが使用される．評価器は仮説を受け取ると，知識ベースを参照して検証を行い，採択した仮説を出力する．以降では類推システムの各構成要素の詳細について説明する．

(1) 類推エンジン

類推エンジンは 3.4.1 項(1)で説明した規則の類推モデルと 3.4.2 項(2)で説明したマッチングの類推モデルを用いて類推を行う．規則の類推モデルは，知識ベースに記述されている規則を使用してあらかじめ学習したものを使用する．マッチングの類推モデルは，類推エンジンが類推の過程において，知識ベースに記述されている事実を使用して生成したものを使用する．

Algorithm 3.1: 類推アルゴリズム

```
Input: a fact written in Prolog  Fact
Output: rules written in Prolog  HypothesisList
1:  FactList ← []
2:  for i = 0 to  size(KnowledgeBase) do
3:      if unify(KnowledgeBase[i], Fact)
4:              append(FactList, KnowledgeBase[i])
5:              append(FactList, variableization(KnowledgeBase[i]))
6:  RuleResultList ← []
7:  for I = 0 to  size(FactList) do
8:      OutputModel ← rule_analogy_model(FactList[i])
9:      append(RuleResultList, OutputModel)
10: RuleResults ← extract_high_correct_rate(RuleResultList , α)
11: HypothesisList ← []
12: for i = 0 to  size(RuleResults) do
13:     AtomList ← []
14:     AtomList ← split_into_atoms(RuleResults[i])
15:     ConvertAtomList ← []
16:     for j = 0 to  size(AtomList) do
17:             FactList1 ← []
18:             FactList2 ← []
19:             TrainingData ← []
20:             TestData ← []
21:             for k = 0 to  size(KnowledgeBase) do
22:                     if unify(KnowledgeBase[k], AtomList[j])
23:                             append(FactList1, KnowledgeBase[j])
24:                     else
25:                             append(FactList2, KnowledgeBase[j])
26:             TrainingData ← convert_training_data(FactList2)
27:             TestData ← convert_test_data(FactList1, FactList2)
28:             matching_analogy_model ← build_model(TrainingData)
29:             UnificationResultList ← []
30:             for k = 0 to  size(TestData) do
31:                     OutputModel ← unification_analogy_model(TestData[k])
32:                     append(UnificationResultList, OutputModel)
33:             UnificationResults ← extract_high_correct_rate(UnificationResultList, β)
34:             append(ConvertAtomList, UnificationResults)
35:     Hypothesis ← convert_atom(RuleResult, ConvertAtomList)
36:     append(HypothesisList, Hypothesis)
37: return HypothesisList
```

図 3.7　類推アルゴリズム

図 3.7 に類推エンジンが類推を行うアルゴリズムを示す．初めに，類推エンジンは知識ベースを参照し，入力された事実とマッチング可能な事実を *FactList* に収集する．（図 3.7 2-4 行目）このとき，マッチング可能な事実を関数 *variabilization*()で変数化したものも *FactList* に追加する．（図 3.7 5 行目）ここで，変数化とは事実の引数のうち 1 つを変数 X に置き換える

操作である．例えば，式(3.11)を変数化した場合，式(3.12)と式(3.13)が生成される．

$$father(tom, bob). \tag{3.11}$$

$$father(X, bob). \tag{3.12}$$

$$father(tom, X). \tag{3.13}$$

続いて，先ほど作成した *FactList* を規則の類推モデル（関数 *rule_analogy_model*()）へ入力し，類推結果を得る．（図 3.7　7-9 行目）類推結果から，本体の引数が頭部の引数によって全て説明されている規則の中で，最も割合が多いものを抽出する．（図 3.7　10 行目）例えば頭部が式(3.14)の場合，本体 (3.15)の引数は頭部と同じく“tom”と“bob”であるため本体の引数は全て説明されている．一方，本体(3.16)の引数には“jim”と“ken”があるため本体の引数は説明されていない．規則の抽出は関数 *extract_high_correct_rate*()によって行われ，抽出結果は *RuleResults* に格納される．関数 *extract_high_correct_rate*()の第二引数のパラメータ α は抽出する規則の数を意味する．

$$father(tom, bob). \tag{3.14}$$

$$female(tom), parent(tom, bob). \tag{3.15}$$

$$female(tom), parent(jim, ken). \tag{3.16}$$

次に，*RuleResults* の規則から仮説を生成する．（図 3.7　12-36 行目）関数 *split_into_atoms*()によって，規則の本体を項に分解する．（図 3.7　14 行目）例えば式(3.17)の規則があった場合，式(3.18)と式(3.19)の項を得る．分解した項は *AtomList* に格納される．

$$father(X, Y) :- female(X), parent(X, Y). \tag{3.17}$$

$$female(X). \tag{3.18}$$

$$parent(X, Y). \tag{3.19}$$

分解した項ごとに知識ベースの事実とマッチングを行う．（図 3.7　16-25 行目）*AtomList* とマッチング可能な事実は *FactList1* に格納され，マッチング不可能な事実は *FactList2* に格納される． *FactList2* は，関数 *convert_trainning_data*()を使用して，マッチングの類推モデルを学習するための学習データを作成するために使用される．（図 3.7　26 行目）例えば式(3.18)の項に対してマッチング不可能な事実(3.20)が得られたとき，式(3.21)の学習データを作成する．学習データは *TrainingData* に格納される．

$$male(tom). \tag{3.20}$$

$$male(X) \ = \ male(tom). \quad true. \tag{3.21}$$

関数 *convert_test_data*()を用いて， *FactList1* と *FactList2* を結合し，マッチングの類推モデルをテストするためのテストデータを作成する．（図 3.7　27 行目）例えば式(3.18)の項に対してマッチング可能な事実(3.22)が得られたとき，式(3.23)のテストデータを作成する．学習データは *TestData* に格納される．

$$female(mary). \tag{3.22}$$

$$male(X) \ = \ female(mary). \quad true. \tag{3.23}$$

TrainingData を使用し，関数 *build_model*()によってマッチングの類推モデルを生成する．（図 3.7　28 行目）その後， *TestData* をマッチングの類推モデル（関数 *matching_analogy_model*()）に入力し，類推結果を得る．（図 3.7　30-32 行目）尚，関数 *build_model*()は，実行される都度，最初からモデルの学習を実行する．これは、類推対象の項をモデルに与えないようにする必要があるためである．

例えば，未知の単語“female”を含むテストデータが，学習済みのマッチングの類推モデルに入力すると，モデルは単語分散表現で表現される類似性によって“female”とマッチング可能な結果を出力する．結果は *UnificationResultList* に保存される． 関数 *extract_high_correct_rate*()によって， *UnificationResultList* から”true”を出力する割合が最も高い項から順に抽出され，抽出結果は ConvertAtomList に格納される．（図 3.7　35 行目）

Algorithm 3.2: 仮説検証アルゴリズム
Input: rules written in Prolog *HypothesisList*
Output : rules written in Prolog *RuleList*
1: *RuleList* ← []
2: **for** *i* = 0 **to** size(*HypothesisList*) **do**
3: *UnifiedRuleList* ← unify(*KnowledgeBase, HypothesisList*[*i*])
4: *Head* ← extract_head(*HypothesisList*[*i*])
5: *UnifiedHeadList* ← unify(*KnowledgeBase, Head*)
6: **if** size(*UnifiedRuleList*) / size(*UnifiedHeadList*) >= γ
7: append(*HypothesisList, HypothesisList*[*i*])
8: **return** *RuleList*

図 3.8 仮説検証アルゴリズム

関数 *extract_high_correct_rate*()の第二引数のパラメータ β は、抽出する項の数を意味する．

最後に，*RuleResult* と *ConvertAtomList* を使用して，関数 *convert_atom*()により仮説を生成する．（図 3.7 35 行目）具体的には，*RuleResult* の本体の項を *ConvertAtomList* の項に置き換えることで，仮説を生成する．例えば式(3.17)の規則の類推結果から式(3.24)，式(3.25)，式(3.26)，式(3.27)の仮説を得る．

$$father(X,Y) :- female(X), parent(X,Y). \tag{3.24}$$

$$father(X,Y) :- male(X), parent(X,Y). \tag{3.25}$$

$$father(X,Y) :- female(X), spouse(X,Y). \tag{3.26}$$

$$father(X,Y) :- male\ (X), spouse(X,Y). \tag{3.27}$$

(2) 評価器

評価器は類推エンジンから *HypothesisList* を受け取ると，知識ベースと照らして仮説検証を行う．図 3.8 に評価器が仮説検証を行うアルゴリズムを示す．

初めに，評価器は知識ベースを参照して仮説ごとに関数 *unifiy*()で単一化を行い，仮説と単一化可能な規則を収集する．（図 3.8 3 行目）関数 *unifiy*()は Prolog 処理系の単一化と同じ処理を行う．例えば式(3.27)の仮説の場合，単一化可能な規則として式(3.28)を得る．

$$father(tom, bob) :- male(tom), parent(tom, bob). \quad (3.28)$$

続いて，関数 *extract_head*()によって仮説の頭部と単一化可能な事実を収集する．（図 3.8 4-5 行目）例えば式(3.26)の仮説の場合，単一化可能な事実として式(3.29)を得る．

$$father(tom, bob). \quad (3.29)$$

最後に，仮説全体が知識ベースと単一化可能な規則の数と仮説の頭部が単一化可能な事実の数の比率を計算する．比率が γ 以上の場合は仮説を採択し，それ以下の場合は棄却する．（図 3.8 6-7 行目）本論文では，$\gamma = 1.0$ としている．

3.5 評価実験

Prolog で記述された 3 種類の知識ベースを利用してモデルの学習及び類推システムの構築を行い，その評価実験を行った．ここではまず 3 種類のデータを用いた実験結果を示した後，考察を述べる．

3.5.1 Kinsources を用いた実験

Kinsources [74] を用いてモデルの学習及び類推システムの構築を行った．Kinsources は血縁関係を表すデータの集合体である．知識ベースは Prolog で記述され，5,887 のアトムと 10 種類の述語から成る．

表 3.1　Kinsources の学習用データセットとモデルの正答率

	規則の類推モデル			
データセット	A-1	A-2		
Training Set	26,709	25,199		
Validation Set	3,215	3,196		
Test Set	3,325	3,804		
正答率	0.8842	0.9008		
	マッチングの類推モデル			
データセット	B-1	B-2	B-3	B-4
Training Set	66,971	67,103	29,906	38,863
Validation Set	8,402	8,412	3,685	4,815
Test Set	8,363	8,383	3,705	4,877
正答率	1.0000	1.0000	1.0000	1.0000

表 3.2　Kinsources における規則の類推結果

データセット	類推のTest Set	頭部の形式	本体の形式	一致率（α=1）
A-1	1,215	mother(X,Y).	male(X),parent(X,Y).	0.6724
	1,180	daughter(X,Y).	male(X),parent(Y,X).	0.4712
	689	wife(X,Y).	male(X),spouse(X,Y).	0.3149
A-2	1,272	father(X,Y).	female(X),parent(X,Y).	0.5460
	1,341	son(X,Y).	female(X),parent(Y,X).	0.4459
	712	husband(X,Y).	female(X),parent(X,Y).	0.2725

規則の類推モデルとマッチングの類推モデルの学習に用いたデータセット及び各データセットを用いて作成したモデルの正答率を表 3.1 に示す．データセット A-1 とデータセット A-2 は規則の類推モデルを作成するためのものであり，知識ベースに含まれる規則をもとに作成した．データセット A-1 とデータセット A-2 は頭部に 3 種類の述語を持つ規則から成る． データセット B-1，データセット B-2，データセット B-3，データセット B-4 はマッチングの類推モデルを作成するためのものであり，知識ベースに含まれる事実をもとに作成した．データセット B-1，データセット B-2，データセット B-3，データセット B-4 は 3 種類の述語を含む質問から成る．正答率はモデルの出力がテストデータと完全一致する割合であり，式（2.1）によって計算される．

表 3.1 に示した規則の類推モデルを使用して，規則を類推した結果を表 3.2 に示す．表 3.1 に示したマッチングの類推モデルを使用して，マッチングの類推を行った結果を表 3.3 に示す．

表 3.3　Kinsources におけるマッチングの類推結果

データセット	類推の Test Set	質問の形式	一致率（β=3）
B-1	1,184	female(X)=male(X).	1.0000
B-2	1,184	male(X)=female(X).	1.0000
B-3	3,905	parent(X,Y)=spouse(X,Y).	1.0000
B-4	3,905	spouse(X,Y)=parent(X,Y).	1.0000

表 3.4　Kinsources における類推結果

データセット	隠れた規則の数	発見した規則	発見率
A-1	3	mother(X,Y):-female(X),parent(X,Y). daughter(X,Y):-female(X),parent(Y,X). wife(X,Y):-female(X),spouse(X,Y).	1.0000
A-2	3	father(X,Y):-male(X),parent(X,Y). son(X,Y):-male(X),parent(Y,X). husband(X,Y):-male(X),spouse(X,Y).	1.0000

表 3.3 の一致率は，モデルからの出力が真である割合を意味する．表 3.4 は、3.4.2 項で説明した類推システムにデータセット A-1 および A-2 を組み込んだ結果を示している．類推エンジンのパラメータは $\alpha = 1.0$，$\beta = 3.0$ に設定した．表 3.4 の“隠れた規則”の数は、データセットから削除した規則の数を表している．“隠れた規則”は，類推システムによる類推の対象となる．表 3.4 の“発見した規則”は，提案システムによって実際に発見された規則になる．表 3.4 の発見率は，“隠れた規則”の内，発見された規則の割合である．発見率が高いほどシステムが規則を発見する能力が高いと言える．

3.5.2　IMDb を用いた実験

IMDb [108] を用いてモデルの学習及び類推システムの構築を行った．IMDb は映画やテレビ関連のデータベースである．知識ベースは Prolog で記述され，20,658 のアトムと 14 種類の述語から成る．

表 3.5　IMDb の学習用データセットとモデルの正答率

	規則の類推モデル			
データセット	A'-1	A'-2		
Training Set	90,904	77,103		
Validation Set	11,373	9,653		
Test Set	11,373	9,632		
正答率	0.9489	0.9499		
	マッチングの類推モデル			
データセット	B'-1	B'-2	B'-3	B'-4
Training Set	88,868	91,229	90,821	46,650
Validation Set	11,147	11,404	11,361	5,815
Test Set	11,147	11,417	11,372	5,742
正答率	1.0000	1.0000	1.0000	1.0000

表 3.6　IMDb における規則の類推結果

データセット	類推のTest Set	頭部の形式	本体の形式	一致率（α=1）
A'-1	2,076	production(X,Y).	person(X),movie(Y),produce(X,Y).	0.3801
A'-2	3,784	producer(X,Y).	company(X),movie(Y),produce(X,Y).	0.3554

表 3.7　IMDb におけるマッチングの類推結果

データセット	類推の Test Set	質問の形式	一致率（β=3）
B'-1	243	person(X)=company(X).	0.8313
	280	person(X)=movie(X).	0.0000
B'-2	243	company(X)=person(X).	0.0000
	280	company(X)=movie(X).	0.0000
B'-3	78	movie(X)=person(X).	0.1538
	85	movie(X)=company(X).	0.0000
B'-4	1,176	produce(X,Y)=distribution(X,Y).	0.9932
	1,220	produce(X,Y)=sfx(X,Y).	0.0328
	1,654	produce(X,Y)=acting(X,Y).	0.0097

規則の類推モデルとマッチングの類推モデルの学習に用いたデータセット及び各データセットを用いて作成したモデルの正答率を表 3.5 に示す．データセット A'-1 とデータセット A'-2 は頭部に 5 種類の述語を持つ規則から成る． データセット B'-1，データセット B'-2，データセット B'-3，データセット B'-4 はマッチングの類推モデルを作成するためのものであり，知識ベースに含まれる事実をもとに作成した．データセット B'-1，データセット B'-2，データセッ

表 3.8　IMDb における類推結果

データセット	隠れた規則の数	発見した規則	発見率
A’-1	1	production(X,Y):-company(X),movie(Y),produce(X,Y).	1.0000
A’-2	1	-	0.0000

表 3.9　clutrr の学習用データセットとモデルの正答率

規則の類推モデル		
データセット	A’’-1	A’’-2
Training Set	90,904	77,103
Validation Set	11,373	9,653
Test Set	11,373	9,632
正答率	0.9489	0.9499
マッチングの類推モデル		
データセット	B’’-1	B’’-2
Training Set	88,868	91,229
Validation Set	11,147	11,404
Test Set	11,147	11,417
正答率	1.0000	1.0000

ト B’-3，データセット B’-4 は 8 種類の述語を含む質問から成る．

表 3.5 に示した規則の類推モデルを使用して，規則を類推した結果を表 3.6 に示す．表 3.5 に示したマッチングの類推モデルを使用して，マッチングの類推を行った結果を表 3.7 に示す．データセット A’-1 とデータセット A’-2 について， 3.4.2 項で述べた類推システムで類推を行った結果を表 3.8 に示す．類推エンジンのパラメータは $\alpha = 1.0$， $\beta = 3.0$ に設定した．表 3.8 の”隠れた規則”の数，“発見した規則”，および発見率は、表 3.4 と同じ意味である．

3.5.3　CLUTRR を用いた実験

推論をテストするためのベンチマーク用データセットの生成器である clutrr [109] を使用して，モデルの学習及び類推システムの構築を行った．本章における研究で使用するデータセットは，task1 で relation length 2 として生成した．clutrr により生成したデータセットから Prolog で記述した知識ベースを構築した．尚，規則の頭部には現れず，本体にのみ出現するアトムは変数 Y に置き換えた．知識ベースは 42 のアトムと 16 種類の述語から成る．

表 3.10　clutrr における規則の類推結果

データセット	類推のTest Set	頭部の形式	本体の形式	一致率（α=3）
A''-1	415	mother(X,Z).	sister(Y,Z),father(X,Y).	0.1229
			son(Y,Z),grandfather(X,Y).	0.0964
			brother(Y,Z),father(X,Y).	0.0892
	569	sister(X,Z).	daughter(Y,Z),uncle(X,Y).	0.1406
			father(Y,Z),son(X,Y).	0.1002
			son(Y,Z),uncle(X,Y).	0.0545
	348	daughter(X,Z).	wife(Y,Z),son(X,Y).	0.1236
			son(Y,Z),brother(X,Y).	0.1178
			daughter(Y,Z),brother(X,Y).	0.0891
	360	grandmother(X,Z).	brother(Y,Z),grandfather(X,Y).	0.1667
			sister(Y,Z),grandfather(X,Y).	0.1417
			father(Y,Z),father(X,Y).	0.0361
	376	granddaughter(X,Z).	wife(Y,Z),grandson(X,Y).	0.1729
			husband(Y,Z),grandson(X,Y).	0.1649
	279	aunt(X,Z).	father(Y,Z),son(X,Y).	0.1111
			father(Y,Z),brother(X,Y).	0.0609
			father(Y,Z),father(X,Y).	0.0502
	249	niece(X,Z).	brother(Y,Z),son(X,Y).	0.1245
			daughter(Y,Z),brother(X,Y).	0.0683
			son(Y,Z),brother(X,Y).	0.0402
A''-2	430	father(X,Z).	father(Y,Z),mother(X,Y).	0.0651
			son(Y,Z),grandmother(X,Y).	0.0628
			father(Y,Z),sister(X,Y).	0.0419
	603	brother(X,Z).	brother(Y,Z),sister(X,Y).	0.1244
			father(Y,Z),daughter(X,Y).	0.0896
			sister(Y,Z),sister(X,Y).	0.0564
	329	son(X,Z).	son(Y,Z),sister(X,Y).	0.2158
			daughter(Y,Z),sister(X,Y).	0.0669
	338	grandfather(X,Z).	father(Y,Z),mother(X,Y).	0.2041
			father(Y,Z),sister(X,Y).	0.0740
			mother(Y,Z),mother(X,Y).	0.0266
	338	grandson(X,Z).	grandson(Y,Z),sister(X,Y).	0.1940
			husband(Y,Z),granddaughter(X,Y).	0.1511
	246	uncle(X,Z).	father(Y,Z),sister(X,Y).	0.1870
			father(Y,Z),mother(X,Y).	0.1138
			father(Y,Z),daughter(X,Y).	0.0285
	256	nephew(X,Z).	son(Y,Z),sister(X,Y).	0.3672

表 3.11　clutrr におけるマッチングの類推結果

データセット	類推のTest Set	頭部の形式	本体の形式	一致率（α=3）
A''-1	415	mother(X,Z).	sister(Y,Z),father(X,Y).	0.1229
			son(Y,Z),grandfather(X,Y).	0.0964
			brother(Y,Z),father(X,Y).	0.0892
	569	sister(X,Z).	daughter(Y,Z),uncle(X,Y).	0.1406
			father(Y,Z),son(X,Y).	0.1002
			son(Y,Z),uncle(X,Y).	0.0545
	348	daughter(X,Z).	wife(Y,Z),son(X,Y).	0.1236
			son(Y,Z),brother(X,Y).	0.1178
			daughter(Y,Z),brother(X,Y).	0.0891
	360	grandmother(X,Z).	brother(Y,Z),grandfather(X,Y).	0.1667
			sister(Y,Z),grandfather(X,Y).	0.1417
			father(Y,Z),father(X,Y).	0.0361
	376	granddaughter(X,Z).	wife(Y,Z),grandson(X,Y).	0.1729
			husband(Y,Z),grandson(X,Y).	0.1649
	279	aunt(X,Z).	father(Y,Z),son(X,Y).	0.1111
			father(Y,Z),brother(X,Y).	0.0609
			father(Y,Z),father(X,Y).	0.0502
	249	niece(X,Z).	brother(Y,Z),son(X,Y).	0.1245
			daughter(Y,Z),brother(X,Y).	0.0683
			son(Y,Z),brother(X,Y).	0.0402
A''-2	430	father(X,Z).	father(Y,Z),mother(X,Y).	0.0651
			son(Y,Z),grandmother(X,Y).	0.0628
			father(Y,Z),sister(X,Y).	0.0419
	603	brother(X,Z).	brother(Y,Z),sister(X,Y).	0.1244
			father(Y,Z),daughter(X,Y).	0.0896
			sister(Y,Z),sister(X,Y).	0.0564
	329	son(X,Z).	son(Y,Z),sister(X,Y).	0.2158
			daughter(Y,Z),sister(X,Y).	0.0669
	338	grandfather(X,Z).	father(Y,Z),mother(X,Y).	0.2041
			father(Y,Z),sister(X,Y).	0.0740
			mother(Y,Z),mother(X,Y).	0.0266
	338	grandson(X,Z).	grandson(Y,Z),sister(X,Y).	0.1940
			husband(Y,Z),granddaughter(X,Y).	0.1511
	246	uncle(X,Z).	father(Y,Z),sister(X,Y).	0.1870
			father(Y,Z),mother(X,Y).	0.1138
			father(Y,Z),daughter(X,Y).	0.0285
	256	nephew(X,Z).	son(Y,Z),sister(X,Y).	0.3672

表 3.12　clutrr における類推結果

データセット	隠れた規則の数	発見した規則	発見率
A’’-1	28	mother(X,Z):-daughter(Y,Z),grandmother(X,Y). mother(X,Z):-son(Y,Z),grandmother(X,Y). mother(X,Z):-brother(Y,Z),mother(X,Y). mother(X,Z):-sister(Y,Z),mother(X,Y). sister(X,Z):-daughter(Y,Z),aunt(X,Y). sister(X,Z):-son(Y,Z),aunt(X,Y). sister(X,Z):-father(Y,Z),daughter(X,Y). sister(X,Z):-mother(Y,Z),daughter(X,Y). daughter(X,Z):-wife(Y,Z),daughter(X,Y). grandmother(X,Z):-sister(Y,Z),grandmother(X,Y). grandmother(X,Z):-mother(Y,Z),mother(X,Y). grandmother(X,Z):-father(Y,Z),mother(X,Y). grandmother(X,Z):-brother(Y,Z),grandmother(X,Y). granddaughter(X,Z):-wife(Y,Z),granddaughter(X,Y). granddaughter(X,Z):-husband(Y,Z),granddaughter(X,Y). niece(X,Z):-brother(Y,Z),daughter(X,Y).	0.5714
A’’-2	28	brother(X,Z):-father(Y,Z),son(X,Y). brother(X,Z):-mother(Y,Z),son(X,Y). grandfather(X,Z):-father(Y,Z),father(X,Y). grandfather(X,Z):-mother(Y,Z),father(X,Y). grandson(X,Z):-husband(Y,Z),grandson(X,Y). grandson(X,Z):-wife(Y,Z),grandson(X,Y).	0.2143

規則の類推モデルとマッチングの類推モデルの学習に用いたデータセット及び各データセットを用いて作成したモデルの正答率を表 3.9 に示す．データセット A”-1 とデータセット A”-2 は頭部に 7 種類の述語を持つ規則から成る．データセット B”-1 およびデータセット B”-2 はマッチングの類推モデルを作成するためのものであり，知識ベースに含まれる事実をもとに作成した．データセット B”-1 とデータセット B”-2 はどちらも 7 種類の述語を含む質問から成る．

表 3.9 に示した規則の類推モデルを使用して，規則を類推した結果を表 3.10 に示す．表 3.9 に示したマッチングの類推モデルを使用して，マッチングの類推を行った結果を表 3.11 に示す．データセット A”-1 とデータセット A”-2 について， 3.4.2 項で述べた類推システムで類推を行った結果を表 3.12 に示す．類推エンジンのパラメータは α = 3.0， β = 3.0 に設定した．表 3.12 の”隠れた規則”の数，“発見した規則”，および発見率は、表 3.4 と同じ意味である．

表 3.13　計算複雑度と検索効率の結果

知識ベース	手法	データセット	計算複雑度（生成された仮説の数）	検索効率
Kinsources	ベースライン	A-1	256	0.0117
		A-2	256	0.0117
	提案手法	A-1	12	0.2500
		A-2	12	0.2500
IMDb	ベースライン	A’-1	1,089	0.0009
		A’-2	1,089	0.0009
	提案手法	A’-1	8	0.1250
		A’-2	4	N/A
clutrr	ベースライン	A’’-1	4,096	0.0068
		A’’-2	4,096	0.0068
	提案手法	A’’-1	162	0.0988
		A’’-2	183	0.0328

3.5.4　計算複雑度と検索効率の比較

ベースラインと提案手法の計算複雑度と検索効率の比較を行った．生成された仮説の数を計算複雑度とし，仮説の数毎の発見した規則の数を検索効率とする．

ベースラインでは，例えば式（3.30）に示したテンプレートを既知の規則から作成し，総当たりでテンプレートの匿名変数にアトムを当てはめることで未知の規則を検索した．尚，式（3.30）の“_”は匿名変数を意味する．

$$\_(X,Y) :- \_(X), \_(X,Y). \tag{3.30}$$

ベースラインによって生成された仮説の数は，以下の式を使用して計算した． ここで，式（3.31）の R_1 は知識ベースから生成されたテンプレートの数，T はテンプレートに含まれる項の数，k はテンプレートの項と同じ数の引数を持つ知識ベース内の項の数を意味する．

$$n_b = \sum_{i=1}^{R_1} \prod_{j=1}^{T_i} k_{ij} \tag{3.31}$$

提案手法によって生成された仮説の数は，式（3.32）を使用して計算した． R_2 は提案システムに入力された事実の数， α（実験では 1 または 3），β（実験では 3）は類推エンジンのパラメータ，T（実験では 2 または 3）は類推結果の規則に含まれる項の数を意味する．

$$n_p = \sum_{i=1}^{R_2} \alpha_i \cdot \beta_i^{T_i} \tag{3.32}$$

表 3.13 に計算複雑度と検索効率についてベースラインと提案手法を比較した結果を示す．知識ベースは 3.5.1 項の Kinsource，3.5.2 項の IMDb，3.5.3 項の clutrr を用いた．

また，ベースラインと提案手法の検索効率に差があるか分析を行った．分析には表 3.13 の内，提案手法の検索効率が算出不可能であったデータセット A’-2 以外を用いた．ベースラインの検索効率の平均は 0.0076，標準偏差は 0.0040 であった．一方，提案手法の検索効率の平均は 0.1513，標準偏差は 0.0860 であった．ウィルコクソン符号付順位和検定の結果，ベースラインと提案手法の検索効率には，有意な差が見られた （p=0.042）．ここで有意水準は $p<0.05$ とした．

3.5.5 考察

Kinsources を用いた実験では，2 つのデータセットにおいてそれぞれ 3 つ全ての規則を抽出することができた．具体的には“father”の規則から“mother”の規則，“son”の規則から“daughter”の規則，“husband”の規則から“wife”の規則を類推することができた．同様に“mother”の規則から“father”の規則，“daughter”の規則から“son”の規則，“wife”の規則から”husband”の規則を類推することができた．

IMDb を用いた実験では，1 つのデータセットで規則を抽出することができた．一方でもう 1 つのデータセットでは規則を抽出することができなかった．具体的には“production”の規則から“producer”の規則を類推することはできたが，“producer”の規則から“production”の規則を類推することができなかった．これは“company(X) = person(X).”に関するマッチングをうまく類推できなかったことが原因と考えられる．

clutrr を用いた実験では，2 つのデータセットから 22 の規則を抽出することができた．この実験の結果から，データセットに同じ頭部を持つ異なる複数の規則が含まれている場合でも，

提案システムにより類推を行うことができることが示された．例えば，“mother(X,Z）:-daughter(Y,Z),grandmother(X,Y)”と“mother(X,Z):-son(Y,Z),grandmother(X,Y). ”は同じ頭部を持つ異なる規則である．提案システムは全てのルールを検出することはできなかったが，データセットA”-1では規則の発見率が57.14%，データセットA”-2では21.43%であった．発見した規則を利用して，提案システムによる繰り返し類推する行うことも可能である．

ベースラインと提案手法を比較した結果，提案手法により生成された仮説の数は，どのデータセットにおいてもベースラインより少なかった．これにより，ベースライン比べて提案手法は計算複雑度が低いといえる．更に，提案手法の検索効率は，規則が見つからなかったデータセットを除いて，ベースラインの検索効率よりも高い結果であった．知識ベースのサイズが大きければ大きいほど，提案手法はベースラインと比較して，計算複雑度と検索効率が有利であると考えられる．

本提案による類推システムを用いることにより，3種類の知識ベースから成る6つのデータセットから未知の規則を抽出することができた．一方，3つのデータセットにおいては規則を抽出することはできなかった．規則の抽出の成否は，どれだけうまく規則の類推とマッチングの類推ができたかに依存している．すなわち，モデルでターゲット領域の問題とソース領域の問題の類似性をうまく表現することができれば，多くの規則を抽出することが可能になると考えられる．本提案のモデルは単語埋め込みベクトルによる単語分散表現で問題の類似性を捉えている．このため，単語分散表現を改良することにより，類推システムの性能を向上できる可能性がある．

3.6　まとめ

本章では深層学習を用いて一階述語論理に基づいた類推を行うシステムを提案した．オブジェクト間の関係性の類似及びオブジェクト間の類似に着目し，深層学習を用いた記号処理を学習することで，類推を実現した．実験結果から，本システムは既学習の規則から未学習の規則を類推して効率よく抽出可能なことを示すことができた．

一階述語論理を対象として深層学習による類推を可能としたのは，本システムが初めてである．また，本システムに用いたモデルは，単語分散表現を利用することにより高いロバスト性も備えているため，Web上の大量のデータからも類推を行うことが可能と考えられる．

第４章 深層学習によるファジィ記号処理を用いた知識コミュニケーションが可能なエージェントの構築

４.１　序論

1960 年代に行われていた適応制御理論の研究 [110] を発端とし，強化学習の研究が取り組まれるようになった． 1990 年代に入ると Q 学習 [111] などの開発にともない，強化学習の研究が大きく発展した．2000 年代に入り深層学習の研究が活発になると，Q 学習における行動価値関数をニューラルネットワークに置き換えた深層強化学習 [112] が登場した．これにより，強化学習の性能は飛躍的に向上し，ゲームや自動運転といった分野で実用化が進んでいる．

深層強化学習によって開発された囲碁プログラム AlphaGo [113] は，これまで実現が困難と考えられていた人間のプロ棋士への勝利を初めて実現した．一方で，AlphaGo が指した手の理由を人が解釈することは困難といった課題が浮き彫りになり，強化学習モデルの説明の必要性が認識されるようになった．更には，強化学習モデルに人が意図した動作を反映して制御することや人が環境の状態を伝えて行動の提案を受け取ることは困難という課題もある．

このような背景から，近年ブラックボックスである機械学習モデルを人が解釈できるようにホワイトボックス化する説明可能な AI の研究 [114-118] が盛んに行われるようになった．特に強化学習に焦点をおいた説明可能な強化学習の研究 [119-128] も取り組まれている．

他方で，1970 年から 80 年代にかけて，エキスパートシステム [6] をはじめとする記号処理による AI の研究が盛んに行われた．記号処理は，知識表現に記号を用いているため人が理解しやすいという利点がある一方，知識ベースの構築を人手で厳密に行う必要があった．これにより，システム構築のコストが高くなることから，次第に記号処理の研究は下火となっていった．しかしながら，近年 Web 上の大量のデータを用いることができるようになり，深層学習

によるニューラルネットワークの学習能力の向上にともない，深層学習を用いた記号処理の研究 [24-26,33,42-44,129] が行われるようになった．

深層学習を用いて作られたエージェントは人が中身を理解することは困難であり，人が環境の状態を伝えてエージェントから行動の提案を受け取ることは難しい．そこで，本章では学習済みの強化学習モデルから記号表現を用いた深層学習モデルを再現し，人が解釈しやすく，表現しやすい知識表現を用いてコミュニケーションが可能なエージェントを構築する方法を提案する．知識表現には記号表現を用いることにより，エキスパートシステムのように人が内容を理解しやすくなり，人が規則を記述することもできる．通常，人は連続値で表現されている状態を即座に解釈することは困難であり，言語により状態を曖昧に表現する．本提案では，深層強化学習モデルの入出力をファジィ化し，記号表現を用いたモデルに置き換えることで，個々人の感性に合わせた表現でエージェントとコミュニケーションをすることや，人がエージェントに規則を追加することが可能となる．ファジー表現は個々人によって異なるため，強化学習を行う前に事前に学習データをシンボル化するのではなく，学習済みの強化学習モデルを再現した方が学習コストは低くなると考える．更に，深層学習を使用した記号処理は高い学習能力があるため，深層強化学習モデルを再現できると考えられる．

本章における研究では深層学習による記号処理を用いることで，

1) 人の解釈が困難な出力を行う学習済みの強化学習モデルから個々人が理解しやすい記号表現を入出力するモデルを再現して適用
2) 人が表現しやすい知識表現を用いて人が意図を伝えることが可能

という従来研究にはない特長を持ったエージェントを構築する．本提案手法は，学習済みの強化学習モデルから人とコミュニケーション可能なエージェントを構築する初めての事例である．本成果により，強化学習により獲得したエージェントによる人へのコーチングといったアプリケーションの実用化に貢献することができると考える．具体的には車の運転やビデオゲームなどを，個々人の感性にパーソナライズされた表現を使ってコーチングするアプリケーションが考えられる．

本章ではまず，4.2節で本章における研究と関連する研究について述べる．4.3節で強化学習モデルの入出力を記号で表現する手法について述べ，4.4 節では，強化学習モデルから記号表現されたモデルの生成方法と生成したモデルを組み込んだエージェントを提案する．4.4 節では，提案手法の実験結果について報告する．尚，本章における研究は [130] の研究をベースとして行った．

4.2 関連研究

4.2.1 説明可能な強化学習

説明可能な強化学習は，その目的によって，人がモデルの内部構造やパラメータを理解することを目指す研究 [119-121] と，人がモデルの判断結果を理解することを目指す研究 [122-128] に分類することができる．また，モデルの生成方法によって，自分自身の説明が可能なモデルを生成する研究 [120,122] と、あるモデルを説明するために別なモデルを生成する研究 [121,123-128] に分類することもできる．

これらの分類の中で，人がモデルの判断結果を理解することを目的とし，そのモデルを説明するために別なモデルを生成する研究 [123-128] が行われている．その中で，Structural Causal Model を使用した研究 [127] や Soft Decision Trees を使用した研究 [128] がある．しかしながらこれらの研究の何れも，説明を行うためのモデルを生成しており，生成したモデルを強化学習モデルの代替として使用することは困難である．

4.2.2 ニューラルネットワークによる記号推論

深層学習が登場する以前より，多層ニューラルネットワークを用いて記号処理を学習し，推論を行う研究が行われてきた．命題論理を対象とした推論 [34-36] や一階述語論理を対象とした推論 [37-41] が行われたが，当時のハードウェア能力と多層ニューラルネットワークの学習能力の限界により，実用的なシステムの構築には至らなかった．

深層学習登場後は，グラフニューラルネットワークを用いて，一階述語論理に基づく演繹推論や帰納推論の研究 [24-25,42-44,129] が行われた．その後，フィードフォワードネットワークを用いた研究 [26] や再帰型ニューラルネットワークを用いた研究 [33,79] も行われるようになった．

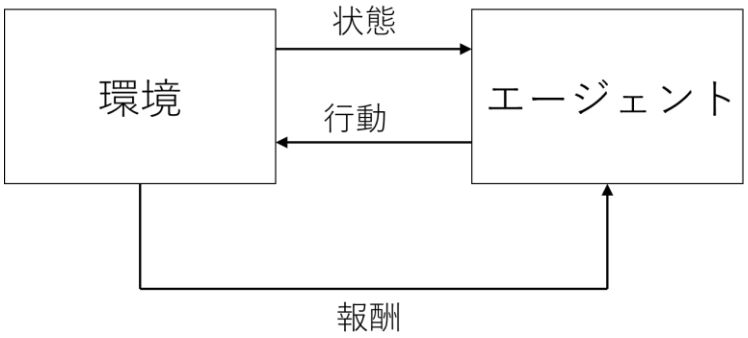


図 4.1　強化学習の概念図

4.2.1　Advice-Taking モデル

これまで自然言語のインターフェースを介して人からエージェントへのアドバイスを可能とするモデルを作成する研究 [131-133] が行われてきた．これらの研究では人がエージェントにアドバイスを与えることによって，モデルの学習時間の短縮や性能向上を目的としている．一方で，学習済みのモデルを使用してエージェントがユーザに対してアドバイスを与えることを可能とする研究は行われていない．

4.3　強化学習モデルの記号表現

本論文では，強化学習モデルの入出力をファジィ化によって記号表現し，これを使用して記号表現を使用したモデルの生成を行う．記号表現には，一階述語論理のサブセットである Prolog [60] を用いる．本章では強化学習の入出力を Prolog で表現する方法について述べる。

4.3.1　強化学習モデルの入出力

図 4.1 に強化学習の概念図を示す．エージェントは環境に対して行動を起こし，環境は行動に応じた状態と報酬をエージェントに与える．エージェントは報酬に応じて行動の方策を修正する．これを繰り返すことで学習が行われる．学習済みの強化学習モデルの入力は環境から与えられる状態であり，出力はエージェントの行動となる．

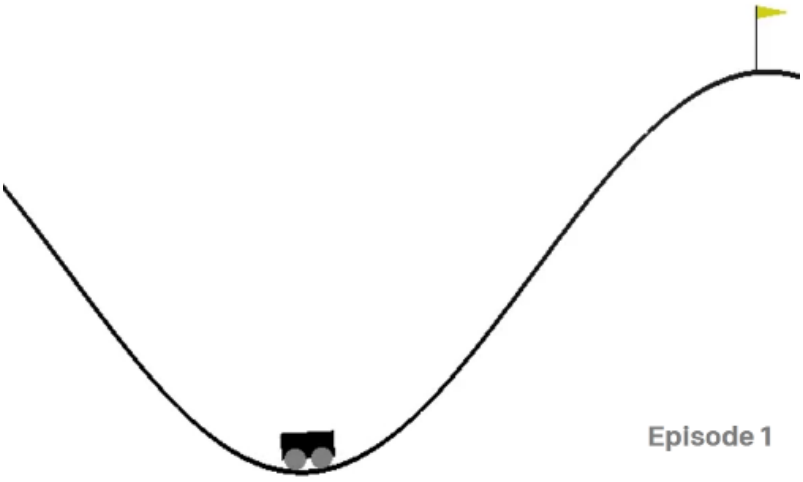


図 4.2　MountainCar-v0 の環境

表 4.1　MountainCar-v0 の行動と状態

行動	0：左へ加速する 1：加速しない 2：右へ加速する
状態	クルマの位置：-1.2～0.6 クルマの速度：-0.07～0.07

図 4.2 は OpenAI Gym [134] が提供する強化学習のシミュレーション環境の一つである MountainCar-v0 の環境を描画した図である．MountainCar-v0 はクルマを山の頂上に移動させることを目的とする．表 4.1 に MountainCar-v0 が取り得る行動と状態を示す．MountainCar-v0 の場合，学習済みの強化学習モデルへの入力はクルマの位置とクルマの速度の 2 種類の連続値となる．出力は行動であり，0，1，2 の離散値となる．

4.3.2　強化学習モデルの入出力の記号化

4.3.1 節で述べた強化学習モデルの入出力を記号化するため，Prolog で表現をする．強化学習は，次の状態が現在の状態と行動にのみ依存する性質であるマルコフ性を仮定している．このため，ある時点の行動は，ある時点の状態をモデルが解釈した結果であると説明をすることができる．これを Prolog の規則にすると頭部は行動，規則の本体は状態の連言で表される．

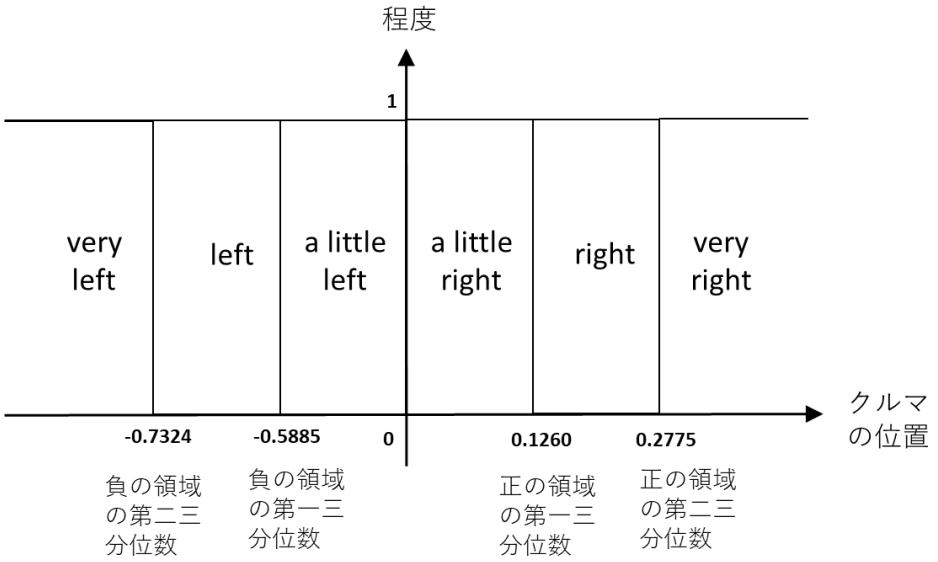


図 4.3　クリスプ型のメンバーシップ関数の例

MountainCar-v0 を Prolog の規則にすると式(4.1)のようになる．action は行動，position はクルマの位置，speed はクルマの速度を表す．変数 X は試行の回数，変数 A は行動の種別，変数 B はクルマの位置の程度を示す言語学的変数，変数 C はクルマの速度の程度を示す言語学的変数となる．

$$action(X, A) :- position(X, B), speed(X, C). \tag{4.1}$$

action は離散値であるため，変数 A は“push_left”，“stay”，“push_right”の 3 種類の値をとる．クルマの位置とクルマの速度は連続値であり，それぞれ言語学的変数 B と言語学的変数 C へと変換する．

連続値の言語学的変数への変換はファジィ理論のメンバーシップ関数を用いる．図 4.3 は変数 B をクリスプ型のメンバーシップ関数で表した例である．クルマの位置が負の場合はクルマは左に位置し，正の場合は右に位置する．クルマが左に位置する場合，変数 B は“very left”，“left”，“a little left”の 3 種類の値をとる．クリスプ集合の境界は観測された負のクルマの位置の分位数とする．クルマの位置が，0 と-0.5885（第一三分位数）の間の場合は変数 B は“a little left”，-0.5885（第一三分位数）と-0.7324（第二三分位数）の間の場合は“left”，-0.7324（第二三分位数）より小さい場合は“very left”となる．クルマが右に位置する場合は“very right”，“right”，“a little right”の 3 種類の値をとる．クリスプ集合の境界は観測さ

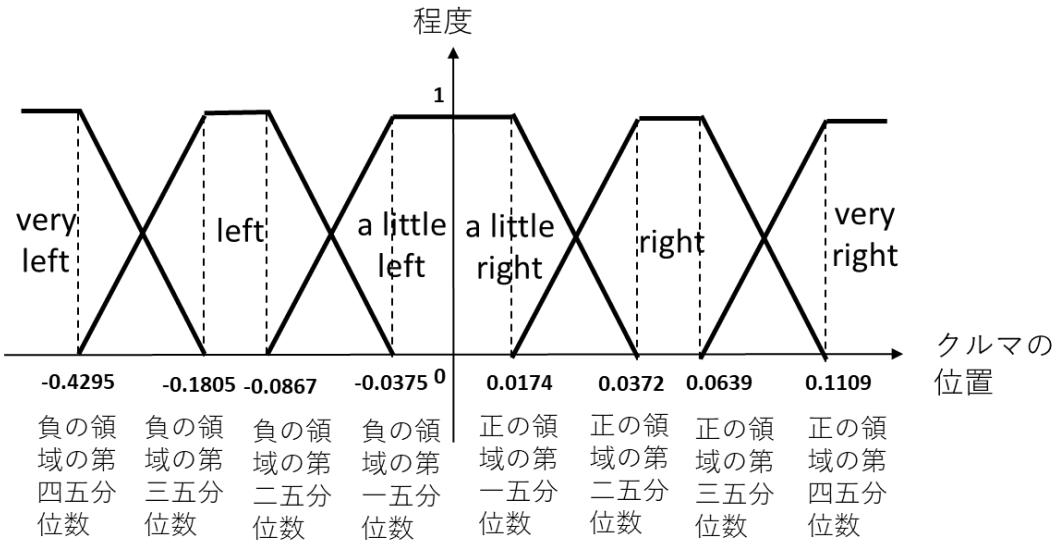


図 4.4　台形型のメンバーシップ関数の例

れた正のクルマの位置の分位数とする．クルマの位置が，0 と 0.0082（第一三分位数）の間の場合は変数 B は“a little right”，0.0082（第一三分位数）と 0.0369（第二三分位数）の間の場合は”right”，0.0369（第二三分位数）より大きい場合は“very right”となる．

一方，図 4.4 は変数 B を台形型のメンバーシップ関数で表した例である．クルマが左に位置する場合，変数 B は“very left”，“left”，“a little left”の 3 種類の値をとる．台形集合の頂点は観測された負のクルマの位置の分位数とする．クルマの位置が，0 と-0.0375（第一五分位数）の間の場合は変数 B は“a little left”となる．-0.0375（第一五分位数）と-0.0867（第二五分位数）の間の場合は程度の大きさに応じた確率に従い，“a little left”もしくは“left”となる．-0.0867（第二五分位数）と-0.1805（第三五分位数）の間の場合は”left”となる．-0.1805（第三五分位数）と-0.4295（第四五分位数）の間の場合は程度の大きさに応じた確率に従い，“left”もしくは“very left”となる．-0.4295（第四五分位数）より小さい場合は”very left”となる．クルマが右に位置する場合は“very right”，“right”，“a little right”の 3 種類の値をとる．台形集合の頂点は観測された正のクルマの位置の分位数とする．クルマの位置が，0 と 0.0174（第一五分位数）の間の場合は変数 B は“a little right”となる．0.0174（第一五分位数）と 0.0372（第二五分位数）の間の場合は程度の大きさに応じた確率に従い，“a little right”もしくは“right”となる．0.0372（第二五分位数）と 0.0639（第三五分位数）の間の場合は“right”となる．0.0639（第三五分位数）と 0.1109（第四五分位数）の間の場合は程度の大きさに応じた確率に従い，“right”もしくは“very right”となる．0.1109

（第四五分位数）より大きい場合は“very right”となる．

変数 C についても変数 B と同様にメンバーシップ関数を用いて表すことができる．クルマの速度が負の場合はクルマは左に移動し，正の場合は右に移動する．変数 C の言語学的変数を正負それぞれ 3 種類で表現する場合，クルマが左に移動するときは“fast to the left”，“to the left”，“slowly to the left”の値をとり，クルマが右に移動するときは”fast to the right”，” to the right”，“slowly to the right”の値をとる．

式(4.2)に強化学習モデルの入出力に基づく規則の例を示す．10 回目の試行でクルマの位置が大きく左で，クルマの速度はゆっくり左に移動している場合，クルマを右へ押すことを表している．

$$action(10, psuh\_right) : -position(10, very\_left), speed(10, slowly\_to\_the\_left). \quad (4.2)$$

4.4 提案する人とコミュニケーション可能なエージェント

本節では，4.3 節で示した強化学習モデルから得られた記号表現をニューラルネットワークで学習する方法について述べる．更に，記号表現を用いたモデルを組み込んだエージェントについて説明する．

4.4.1 記号表現を用いたモデルの再現

本節では，記号表現を用いたモデルの生成方法について述べる．記号表現を用いたモデルの生成には，学習済みの強化学習モデルへの入出力を記号で表現したデータを用いる．ここで，連続値である状態から変換を行う言語学的変数は，正負それぞれについて 5 種類までの値をとることとする．印象評価実験の手法である Semantic Differential （SD）法 [135-136] では形容詞対を 5 段階もしくは 7 段階の尺度で表している．また，リッカート尺度 [137] では度合いを 5 段階の尺度とすることが多い．そこで，本章における研究では，人が判別しやすい度合いの尺度は 5 段階までと仮定した．例えば，MountainCar-v0 の正のクルマの位置を 5 種類の値で表現すると，“非常に小さく右”，“小さく右”，“右”，“大きく右”，“非常に大きく右”となる．

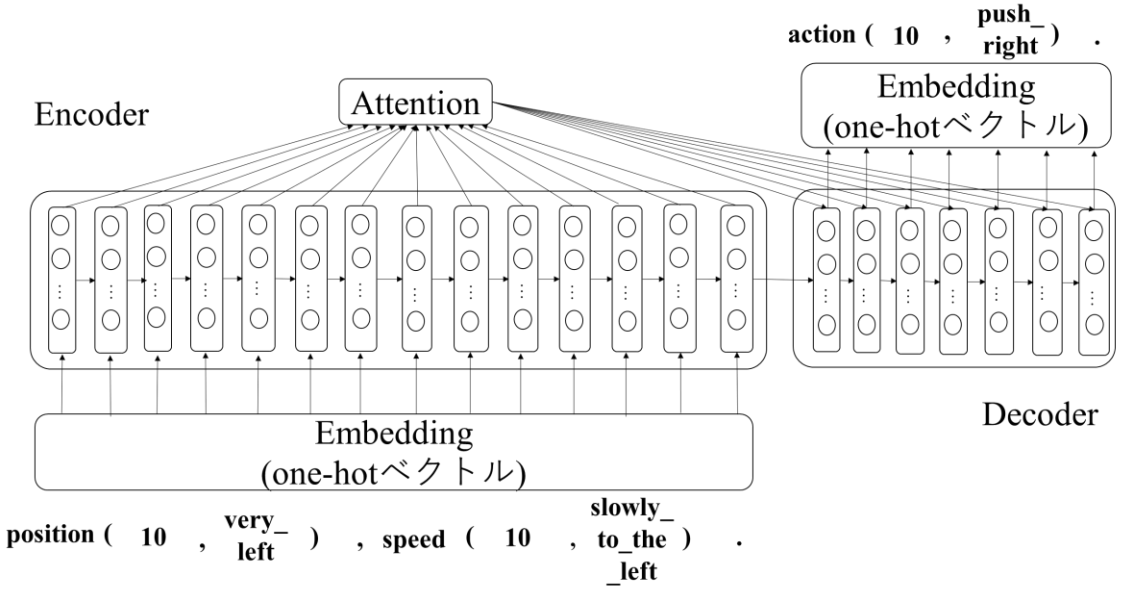


図 4.5　提案する記号表現を用いたモデル

記号表現をしたデータの学習は，[33] で提案されている再帰型ニューラルネットワークをベースとする．[33] では再帰型ニューラルネットワークと Transformer [46] を比較し，再帰型ニューラルネットワークの方が性能が良いことを確認している．このため，提案システムでも再帰型ニューラルネットワークを使用する．[33] では Prolog の規則の頭部を入力すると，Prologの規則の本体を出力するように再帰型ニューラルネットワークを学習していた．本章における研究では，強化学習モデルの入力から出力結果を推論するため，Prologの規則の本体を入力すると Prolog の規則の頭部を出力するように学習を行う．[33] では未知語を扱うために単語分散表現 [61-63] を用いて入力ベクトルを Embedding していた．しかしながら本章における研究で用いる強化学習の環境は既知の事象で成立する仮定のため，本提案では単語分散表現は使用しない.

本章における研究で提案する再帰型ニューラルネットワークを図 4.5 に示す．初めに，規則本体の単語列は入力用 Embedding 層に渡され，one-hot でベクトル化される．

続いて，入力用 Embedding 層で生成されたベクトルは Attention 付き Seq2Seq [54] へと渡される．Attention は，入力したデータのどの部分を注目するか推測することにより，Seq2Seq のパフォーマンスを向上させる．Attention 付き Seq2Seq は Encoder，Decoder，Attention から構成される．Encoder は入力系列を受け取ると圧縮したベクトルを出力する．Attention は出力系列のコンテキストに基づいて，入力系列内の各単語に与える注意の度合い

を計算する．圧縮ベクトルには注意の度合いによる重みが加えられる．Decoder は Encoder と Attention からベクトルを受け取ると出力系列を生成する．

Encoder と Decoder には Long Short Term Memory（LSTM） [59] を用いる．Encoder には過去の情報と同様に未来の情報も扱える Bi-LSTM [69] を適用する．Bi-LSTM は 3 層から成り，出力層は 128 次元である．Decoder には短期記憶を引き継がない Stateless LSTM を適用する．Stateless LSTM は出力層が 128 次元であり，活性化関数に Maxout [70] を用いる．

最後に，Attention 付き Seq2Seq の出力は出力用 Embedding 層に渡され，one-hot で Embedding されて規則頭部の単語列として出力される．

本論文中の学習に関しては，Dropout 率は 0.1，バッチサイズを 128 とし，20 エポックの学習を行った．最適化には Adam [71] を用い，$\alpha = 0.001$，$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\varepsilon = 1.0 \times 10^{-8}$ とした．

式(4.2)の規則を例にすると，式(4.3)を入力したときに式(4.4)を出力するようにモデルの学習が行われる．

$$position(10, very\_left), speed(10, slowly\_to\_the\_left). \tag{4.3}$$

$$action(10, psuh\_right). \tag{4.4}$$

4.4.2 人とコミュニケーション可能なエージェント

本節では，4.4.1 項で生成した記号表現を用いたモデルを組み込んだ人とコミュニケーション可能なエージェントについて述べる．人とコミュニケーション可能なエージェントは，人がファジィ表現を用いることで個々人の感性に合わせた表現でエージェントに環境の状態を伝え，行動のアドバイスを得ることを可能とする．また，人が記述した規則を与えることで，学習済みのモデルに対して人の意図を反映させることも可能とする．

図 4.6 は記号表現を用いたモデルを使用した人とコミュニケーション可能なエージェントの概念図である．図中の Environment は強化学習のシミュレーション環境である．Agent は提

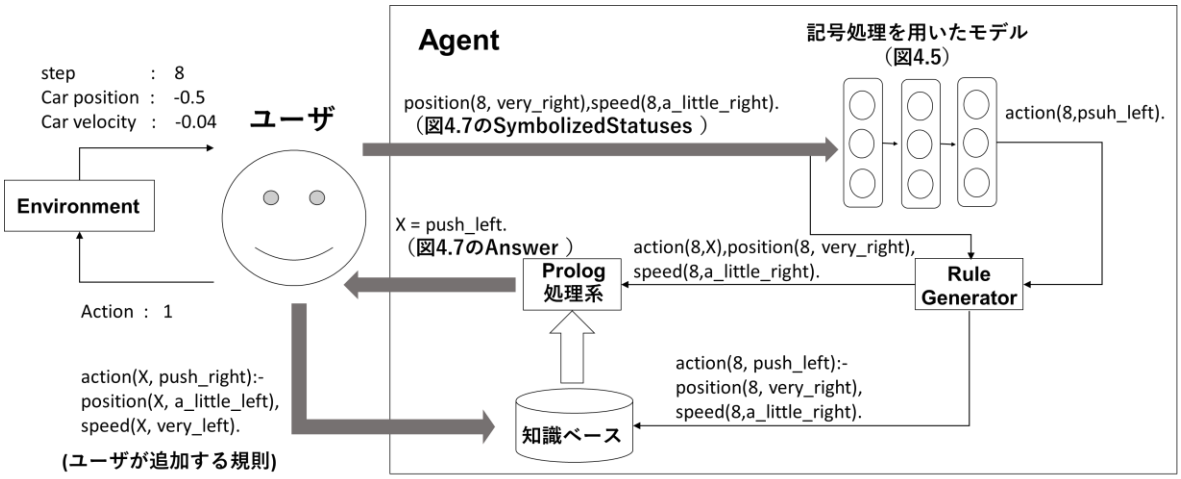


図 4.6　人とコミュニケーション可能なエージェントの概念図

Algorithm 4.1: Agentのアルゴリズム
Input: Statuses represented by linguistic variables *SymbolizedStatuses*
Output: An action represented by linguistic variables *Answer*
1: *SymbolizedAction* ← transfer(*SymbolizedStatuses*)
2: *Rule, Question* ← generate_rule(*SymbolizedAction, SymbolizedStatuses*)
3: add_knowledge_base(*Rule*)
4: *Answer* ← prolog_processing(*KnowledgeBase, Question*)
5: **return** *Answer*

図 4.7　Agent のアルゴリズム

案する人とコミュニケーション可能なエージェントである． ユーザは Agent にファジィ表現を用いた Environment の状態を伝えると，Agent から行動のアドバイスを受け取ることができる．また，ユーザは Agent に規則を追加することもできる．ユーザにより追加された規則は，Agent 内部のモデルにより生成された規則よりも優先して適用される．

図 4.7 は Agent のアルゴリズムである．ここでは Environment は MountainCar-v0 として説明をする．ユーザは，Environment の状態をファジィを用いた記号で表現する．Agent はユーザから記号化された状態（図 4.6 中 SymbolizedStatuses，図 4.7 中 1 行目）を受け取ると，学習済みの記号表現を用いたモデルに入力し，記号化された行動が出力される．（図 4.6 中 Symbolic Processing Model，図 4.7 中 1 行目）次に記号化された状態と記号化された行動から質問文と規則を生成する．（図 4.6 中 Rule Generator， 図 4.7 中 2 行目）記号化された状態が式(4.5)，記号化された行動が式(4.6)の場合，生成される規則は式(4.7)，質問は式(4.8)のようになる．質問は記号化された状態と記号化された行動の連言とする．

$$position(8, very\_right), speed(8, a\_little\_right). \tag{4.5}$$

$$action(8, push\_left). \tag{4.6}$$

$$action(8, push\_left) :- position(8, very\_right), speed(8, a\_little\_right). \tag{4.7}$$

$$action(8, X), position(8, very\_right), speed(8, a\_little\_right). \tag{4.8}$$

生成された規則は知識ベースに追記する．（図 4.6 中 知識ベース， 図 4.7 中 3 行目）知識ベースには式(4.9)のような規則をユーザが追加しておくことができる．

$$action(X, push\_right) :- position(X, a\_little\_left), speed(X, very\_left). \tag{4.9}$$

続いて，Prolog 処理系は知識ベースを読み込み，生成された質問に回答する．（図 4.6 中 知識ベース， 図 4.7 中 4 行目）ここで，Agent により生成された規則はユーザが知識ベースに格納した規則の後ろに追記されているため，質問に対する解が複数存在する場合でも，ユーザが知識ベースに格納した規則の結果が一番目に出力される．例えばエージェントからの回答は式(4.10)のようになる．

$$X = push\_left. \tag{4.10}$$

ユーザはエージェント回答に基づいて Environment に対する行動をとることができる．

4.5 評価実験

2 種類の強化学習のシミュレーション環境を用いてモデルの学習及びエージェント構築を行い，評価実験を行った．本章では，実験結果を示した後，考察を述べる．尚，付録 A に人の曖昧な言語表現に基づいてメンバーシップ関数を作成し，深層強化学習の再現を行った実験結果を示す．

表 4.2　MountainCar-v0 の学習に要したエピソード数

強化学習アルゴリズム	エピソード数
DQN	7,800
DDQN	3,200

表 4.3　MountainCar-v0 の記号表現モデルのデータセット

	DQN	DDQN
Training Set	124,753	99,937
Validation Set	16,219	12,436
Test Set	15,846	12,613

4.5.1　MountainCar-v0 を用いた実験

MountainCar-v0 を強化学習したモデルから記号表現を用いたモデルを再現し，エージェントを構築した．強化学習アルゴリズムには，Deep Q Network（DQN）[112] と Double Deep Q Network（DDQN）[138] の 2 種類を用いた．これにより，異なる強化学習アルゴリズムでも再現可能なことを確認する．DQN，DDQN ともに 1 エピソードの平均報酬が 160 を超えるまでモデルの学習を行った．1 エピソードあたりの試行は 200 までとする．表 4.3 に MountainCar-v0 の学習に要したエピソード数を示す．

表 4.4 に，表 4.3 で学習した強化学習モデルから生成したデータセットを示す．学習済みの強化学習モデルを使用して試行を繰り返すことで試行の回数毎の状態と行動を取得した．状態のクルマの位置，状態のクルマの速度の言語学的変数の数を変化せたメンバーシップ関数を用いて学習データの記号化を行い，データセットを構築した．メンバーシップ関数には 4.3.1 項で述べたクリスプ型と台形型の 2 種類を使用した．連続値で表される状態は，4.3.2 項で説明した方法で言語学変数に変換した．表 4.4 のデータセットを用いて記号表現を用いたモデルを生成し，一致率を算出した．記号表現を用いたモデルとベースラインの一致率を比較した結果を表 4.5 に示す．一致率は，記号表現を用いたモデルにテストデータの状態を入力し，得られた行動を数値化した結果がテストデータの行動と正確に一致した割合を表す．ベースラインはデータセットの生成に用いた学習済みの強化学習モデルとした．

表 4.4　MountainCar-v0 におけるモデルの一致率

強化学習アルゴリズム	メンバーシップ関数	言語学的変数の数	ベースラインの一致率	記号表現モデルの一致率
DQN	クリスプ型	正の領域：1 負の領域：1	0.0000	0.8565
		正の領域：2 負の領域：2	0.0000	0.9231
		正の領域：3 負の領域：3	0.0000	0.9598
		正の領域：4 負の領域：4	0.0000	0.9636
		正の領域：5 負の領域：5	0.0000	**0.9773**
	台形型	正の領域：2 負の領域：2	0.0000	0.8942
		正の領域：3 負の領域：3	0.0000	0.9300
		正の領域：4 負の領域：4	0.0000	0.9539
		正の領域：5 負の領域：5	0.0000	**0.9665**
DDQN	クリスプ型	正の領域：1 負の領域：1	0.0000	0.9148
		正の領域：2 負の領域：2	0.0000	0.9390
		正の領域：3 負の領域：3	0.0000	0.9726
		正の領域：4 負の領域：4	0.0000	0.9742
		正の領域：5 負の領域：5	0.0000	**0.9743**
	台形型	正の領域：2 負の領域：2	0.0000	0.9244
		正の領域：3 負の領域：3	0.0000	0.9386
		正の領域：4 負の領域：4	0.0000	0.9432
		正の領域：5 負の領域：5	0.0000	**0.9481**

4.5.2　CartPole-v1 を用いた実験

CartPole-v1 を強化学習したモデルから記号表現を用いたモデルを再現し，エージェントを

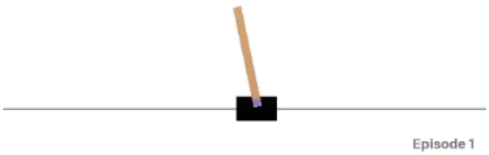


図 4.8　CartPole-v1 の環境

表 4.5　CartPole-v1 の行動と状態

行動	0：カートを左方向に押す 1：カートを右方向に押す
状態	カートの水平位置　　：-2.4～2.4 カートの水平速度　　：-Inf～Inf ポールと垂直方向の角度：-41.8°　～41.8° ポールの角速度　　：-Inf～Inf

表 4.6　CartPole-v1 の学習に要したエピソード数

強化学習アルゴリズム	エピソード数
DQN	2,187
DDQN	820

表 4.7　CartPole-v1 の記号表現モデルのデータセット

	DQN	DDQN
Training Set	136,386	141,269
Validation Set	16,919	17,564
Test Set	17,190	17,933

構築した．図 4.8 は CartPole-v1 の環境を描画した図である．CartPole-v1 はカートを移動させてバランスを取り，棒が倒れないようにすることを目的とする．表 4.5 に CartPole-v1 が取り得る行動と状態を示す．強化学習アルゴリズムには，DQN と DDQN の 2 種類を用いた．これにより，異なる強化学習アルゴリズムでも再現可能なことを確認する．DQN，DDQN ともに 1 エピソードの平均報酬が 200 を超えるまでモデルの学習を行った．1 エピソードあたりの試行は 200 までとする．表 4.6 に CartPole-v1 の学習に要したエピソード数を示す．

表 4.7 に，表 4.6 で学習した強化学習モデルから生成したデータセットを示す．学習済みの

表 4.8　CartPole-v1 におけるモデルの一致率

強化学習アルゴリズム	メンバーシップ関数	言語学的変数の数	ベースラインの一致率	記号表現モデルの一致率
DQN	クリスプ型	正の領域：1 負の領域：1	0.0000	0.8593
		正の領域：2 負の領域：2	0.0000	0.8850
		正の領域：3 負の領域：3	0.0000	**0.8934**
		正の領域：4 負の領域：4	0.0000	0.8921
		正の領域：5 負の領域：5	0.0000	0.8901
	台形型	正の領域：2 負の領域：2	0.0000	0.8697
		正の領域：3 負の領域：3	0.0000	0.8700
		正の領域：4 負の領域：4	0.0000	**0.8789**
		正の領域：5 負の領域：5	0.0000	0.8753
DDQN	クリスプ型	正の領域：1 負の領域：1	0.0000	0.9002
		正の領域：2 負の領域：2	0.0000	0.8991
		正の領域：3 負の領域：3	0.0000	0.9063
		正の領域：4 負の領域：4	0.0000	0.9048
		正の領域：5 負の領域：5	0.0000	**0.9123**
	台形型	正の領域：2 負の領域：2	0.0000	0.9012
		正の領域：3 負の領域：3	0.0000	0.9001
		正の領域：4 負の領域：4	0.0000	0.9026
		正の領域：5 負の領域：5	0.0000	**0.9080**

モデルを使用して試行を繰り返すことで試行の回数毎の状態と行動を取得した．4 種類ある状態の言語学的変数の数を変化せたメンバーシップ関数を用いて学習データの記号化を行い，データセットを構築した．メンバーシップ関数には 4.3.1 項で述べたクリスプ型と台形型の 2 種類を使用した．連続値で表される状態は，4.3.2 項で説明した方法で言語学変数に変換した．

表 4.7 のデータセットを用いて記号表現を用いたモデルを生成し，一致率を算出した．記号表現を用いたモデルとベースラインの一致率を比較した結果を表 4.8 に示す．ベースラインはデータセットの生成に用いた学習済みの強化学習モデルとした．

4.5.3 ベースラインと記号表現モデルの一致率比較

ベースラインと記号表現モデルで，一致率に差があるか分析を行った．分析には表 4.4，表 4.8 の一致率を用いた．ベースラインの一致率の平均は 0.0000，標準偏差は 0.0000 であった．一方，記号表現モデルの一致率の平均は 0.9483，標準偏差は 0.0194 であった．ウィルコクソン符号付順位和検定の結果，ベースラインと記号表現モデルの一致率には，有意な差が見られた （$p=1.7\times10^{-7}$）．ここでの有意水準は $p<0.05$ とした．

4.5.4 考察

2.5.3 項の結果より，ベースラインよりも，記号表現モデルの方が，パフォーマンスが良かった．ベースラインは入力に連続変数を受け付け，言語学的変数は受け付けないため，一致率は全て 0.0000 となった．

MountainCar-v0 を用いた実験では，DQN で学習したモデルにクリスプ型のメンバーシップ関数を用いて記号表現モデルを再現した場合，言語学的変数の数が正負ともに 5 のときに一致率が 0.9773 で最大となった．台形型のメンバーシップ関数を用いた場合は，言語学的変数の数が正負ともに 5 のときに一致率が 0.9665 で最大となった．DDQN で学習したモデルにクリスプ型のメンバーシップ関数を用いて記号表現モデルを再現した場合は，言語学的変数の数が正負ともに 5 のときに一致率が 0.9743 で最大となった．台形型のメンバーシップ関数を用いた場合は，言語学的変数の数が正負ともに 5 のときに一致率が 0.9481 で最大となった．

CartPole-v1 を用いた実験では，DQN で学習したモデルにクリスプ型のメンバーシップ関数を用いて記号表現モデルを再現した場合，言語学的変数の数が正負ともに 3 のときに一致率が 0.8934 で最大となった．台形型のメンバーシップ関数を用いた場合は，言語学的変数の数が正負ともに 4 のときに一致率が 0.8789 で最大となった．DDQN で学習したモデルにクリスプ型のメンバーシップ関数を用いて記号表現モデルを再現した場合は，言語学的変数の数が正負ともに 5 のときに一致率が 0.9123 で最大となった．台形型のメンバーシップ関数を用いた

場合は，言語学的変数の数が正負ともに 5 のときに一致率が 0.9080 で最大となった．

MountainCar-v0 と CartPole-v1 の実験結果において，強化学習アルゴリズムは DQN と DDQN の両方で高い一致率を示した．尚，データセットはランダム化されているため，再現されたモデルは強化学習モデルと同じマルコフ性を持っている．

MountainCar-v0 と CartPole-v1 の両方の実験結果で，言語学的変数の数が増加するにつれて一致率も上昇する傾向があった．これは，言語学的変数の数が多いほど表現可能な規則の数が増えるためと考えられる．

また，MountainCar-v0 と CartPole-v1 の実験結果ともに，クリスプ型メンバーシップ関数と台形型メンバーシップ関数両方とも高い一致率となった．人が環境の状態を観察して記号表現をする場合，曖昧さが発生するため，台形型メンバーシップ関数を用いるのが実態に即していると考えられる．台形型メンバーシップ関数の場合，言語学的変数を確率的に選択しているにも関わらず，高い一致率となった．

4.6 まとめ

本章では学習済みの強化学習モデルから記号表現を用いたモデルを再現し，それを利用して，人とコミュニケーションが可能なエージェントを提案した．人が解釈しやすく，表現しやすい知識表現に変換するために，強化学習モデルの入出力をファジィ化し，環境の状態とエージェントの行動を一階述語論理の規則として記述した．実験の結果から，規則を再帰型ニューラルネットワークで学習することで，記号表現を用いたモデルを高い精度で獲得することができた．これにより，人が個々人の感性に合わせた表現でエージェントに環境の状態を伝えることや，人がエージェントに規則を追加することが可能なエージェントを実現した．

本提案手法は，学習済みの強化学習モデルから人が解釈しやすく，表現しやすい知識表現を用いてコミュニケーションを行うことが可能なエージェントを構築した初めての事例である．本成果により，エージェントによる人へのコーチングといったアプリケーションへの応用に貢献ができると考えられる．

第５章 結論

５.１　本研究のまとめ

本論文では深層学習を用いた一階述語論理による推論について述べた．本研究では，従来の人工知能による記号推論と深層学習の手法を融合することにより，ロバスト性と解釈可能性の両方を備える推論を行うことを可能とした．

第 2 章では，深層学習を用いて一階述語論理による演繹推論を行い，知識ベースから質問応答システムを構築する手法を提案した．実験の結果，提案する記号処理は従来手法に比べて，(1)記号処理について高い表現力を実現し，(2)未学習の記号が含まれていた場合でも解を導くことを可能とし，(3)小規模なデータからも学習が可能な特長を備えていることが示された．特に(2)は単語分散表現の利用により，従来の人工知能研究の大きな課題であった未学習データへの対応を実現した．また，実際に Prolog で記述された知識ベースから記号処理の学習を行い，質問応答システムを構築した．これは，従来手法により実現することは非常に困難である．以上より提案システムは未学習データを扱うことが可能であり，Web 上の大量なデータから知識ベースを構築するための潜在的な能力があることも示した．

第 3 章では，深層学習を用いて一階述語論理による類推を行う手法を提案した．提案システムは単語分散表現と再帰型ニューラルネットワークを組み合わせたモデルとなっている．学習済みの本モデルに未学習データを入力すると，既学習データの中から未学習データに類似した結果が得られることを利用して類推を実現する．実験の結果，3 種類の知識ベースにおいて既学習の規則から未学習の規則を類推することができた．本提案手法は深層学習を用いて一階述語論理による類推を行う初めての事例である．

第 4 章では，深層強化学習モデルをファジィ表現と一階述語論理を組み合わせた記号表現モデルに置き換えることで，人とコミュニケーションを可能とする手法を提案した．本提案では，環境の状態とエージェントの行動をファジィ化し，一階述語論理の規則として表現したものを学習することにより，記号表現を用いたモデルを生成する．深層強化学習モデルを記号表現を

用いたモデルに置き換えることで，人が個々人の感性に合わせた表現でエージェントに環境の状態を伝えることや，人がエージェントに規則を追加することを可能とする．実験の結果，強化学習のシミュレーション環境 2 種類について，学習済みの深層強化学習モデルを高い精度で記号表現を用いたモデルで再現することができた．再現したモデルを使用して，これまで実現できていなかった人とコミュニケーション可能なエージェントを構築した．本提案手法は，学習済みの強化学習モデルから人が表現しやすい知識表現を用いたコミュミニケーションが可能なエージェントを構築する初めての事例である．

5.2 今後の課題

最後に，今後の課題について述べる．

2 章では深層学習による記号処理を用いた知識ベースからの質問応答システム構築について述べた．本章における研究では，マッチング，導出及びリスト処理といった記号処理を深層学習により実現した．今後の課題としては，深層学習による単一化やバックトレースといった処理を実施することや，ガベージコレクション等のメモリ管理の実現，単語分散表現に文脈情報を利用可能な BERT を適用することがあげられる．更に，Generalized Phrase Structure Grammar（GPSG）[139] や Head-driven Phrase Structure Grammar（HPSG）[140]，Lexical Function Grammar（LCG）[141] といった形式文法の素性構造に対する単一化を深層学習で実現し，意味解析を行うことも課題としてあげられる．

3 章では深層学習による記号処理を用いた類推について述べた．本章における研究では，オブジェクト間の関係性の類似及びオブジェクト間の類似に着目することで，一階述語論理の規則の類推を深層学習により実現した．今後の課題としては，問題解決の目標の類似性に着目したプラグマティックな類推の実現があげられる．また，単語分散表現の改良を行うことで類推システムの性能を向上させることも課題としてあげられる．

4 章では深層学習によるファジィ記号処理を用いた知識コミュニケーションが可能なエージェントの構築について述べた．本章における研究では深層強化学習モデルの入出力をファジィ化し，記号表現を用いたモデルに置き換えることで，エージェントが個々人の感性に合わせて表現した規則を伝えることや，人がエージェントに規則を追加することが可能とした．今後の課題としては，2 章のように単語分散表現を用いることで述語にスケーラビリティを持たせる

ことや，自然言語による対話インターフェースにより人とエージェントのコミュニケーションを可能とすることがあげられる．また，強化学習のシミュレーション環境の行動が連続値をとる場合や状態が画像となる場合に対応することも課題としてあげられる．

参考文献

[1] G. Boole, *The Mathematical Analysis of Logic: Being an Essay Towards a Calculus of Deductive Reasoning*. Cambridge, UK: Cambridge University Press, 2009, p.92.

[2] F. Gottlob, *Begriffsschrift: eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Paris, FR: Hachette Livre Bnf, 2012, p.105.

[3] K. Gödel, "Über die Vollständigkeit des Logikkalküls", doctoral dissertation, University of Vienna, 1929.

[4] J. McCarthy, "Programs with Common Sense," in *Semantic Information Processing*, 1968, pp. 403-418.

[5] A. Colmerauer and P. Roussel, "The birth of Prolog," *ACM SIGPLAN Notices*, vol. 28, no. 3, pp. 37-52, Mar. 1993, doi: 10.1145/155360.155362.

[6] D. C. Luckham and N. Suzuki, "Automatic program verification V: verification-oriented proof rules for arrays, records and pointers," Stanford University, Stanford, CA, USA, Technical Report, 1976.

[7] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, *Applications of Artificial Intelligence for Organic Chemistry: The Dendral Project*. New York, NY, USA: McGraw-Hill, 1980, p. 203.

[8] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81-106, Mar. 1986, doi: 10.1007/BF00116251.

[9] S. Muggleton, "Inductive logic programming," *New Generation Computing*, vol. 8, no. 4, pp. 295-318, Feb. 1991, doi: 10.1007/BF03037089.

[10] G. Brewka, *Nonmonotonic Reasoning: Logical Foundations of Commonsense*. Cambridge, UK: Cambridge University Press, 1991, p.184.

[11] J. Doyle, "The ins and outs of reason maintenance," in *Proceedings of the Eighth international joint conference on Artificial intelligence - Volume 1*, San Francisco, CA, USA, Aug. 1983, pp. 349-351.

[12] A. C. Kakas, R. A. Kowalski, and F. Toni, “Abductive Logic Programming,” *Journal of Logic and Computation*, vol. 2, no. 6, pp. 719-770, Dec. 1992, doi: 10.1093/logcom/2.6.719.

[13] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115-133, Dec. 1943, doi: 10.1007/BF02478259.

[14] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychological Review*, vol. 65, no. 6, pp. 386-408, 1958, doi: 10.1037/h0042519.

[15] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. Expanded Edition, MA, USA: The MIT Press, 1987, p. 312.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533-536, Oct. 1986, doi: 10.1038/323533a0.

[17] G. E. Hinton and R. R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” *Science*, Jul. 2006, doi: 10.1126/science.1127647.

[18] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Computation*, vol. 18, no. 7, pp. 1527-1554, Jul. 2006, doi: 10.1162/neco.2006.18.7.1527.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, Red Hook, NY, USA, Dec. 2012, pp. 1097-1105.

[20] G. E. Hinton, “Preface to the Special Issue on Connectionist Symbol Processing,” *Artificial Intelligence*, vol. 46, no. 1-2, pp. 1-4, 1990, doi: 10.1016/0004-3702(90)90002-h.

[21] D. S. Touretzky, “BoltzCONS: Dynamic symbol structures in a connectionist network,” *Artificial Intelligence*, vol. 46, no. 1, pp. 5-46, Nov. 1990, doi: 10.1016/0004-3702(90)90003-I.

[22] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, “Modeling Relational Data with Graph Convolutional Networks,” 2017, *arXiv:1703.06103*. [Online]. Available: http://arxiv.org/abs/1703.06103

[23] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, “Complex embeddings for simple link prediction,” in *Proceedings of the 33rd International Conference on International*

Conference on Machine Learning - Volume 48, New York, NY, USA, Jun. 2016, pp. 2071-2080.

[24] T. Rocktäschel and S. Riedel, "End-to-end differentiable proving," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, Dec. 2017, pp. 3791-3803.

[25] P. Minervini, M. Bosnjak, T. Rocktäschel, and S. Riedel, "Towards Neural Theorem Proving at Scale," 2018, *arXiv:1807.08204*. [Online]. Available: http://arxiv.org/abs/1807.08204

[26] H. Dong, J. Mao, T. Lin, C. Wang, L. Li, and D. Zhou, "Neural logic machines," in *Proceedings of the International Conference on Learning Representations*, New Orleans, LA, USA, 2019, pp. 1-22

[27] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton, "Quantitative evaluation of passage retrieval algorithms for question answering," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, Jul. 2003, pp. 41-47. doi: 10.1145/860435.860445.

[28] R. Sequiera G. Baruah, Z. Tu, S. Mohammed, J. Rao, H. Zhang, and J. Lin,"Exploring the Effectiveness of Convolutional Neural Networks for Answer Selection in End-to-End Question Answering," 2017, *arXiv:1707.07804*. [Online]. Available: http://arxiv.org/abs/1707.07804

[29] N. T. Thomas, "An e-business chatbot using AIML and LSA," in *Proceedings of 2016 International Conference on Advances in Computing, Communications and Informatics*, Sep. 2016, pp. 2740-2742. doi: 10.1109/ICACCI.2016.7732476.

[30] L. Cui, S. Huang, F. Wei, C. Tan, C. Duan, and M. Zhou, "SuperAgent: A Customer Service Chatbot for E-commerce Websites," in *Proceedings of 55th Annual Meeting of the Association for Computational Linguistics-System Demonstrations*, Vancouver, Canada, Jul. 2017, pp. 97-102.

[31] H. Bhargava and D. Power, "Decision support systems and Web technologies: A status report," in *Proceedings of* Americas' Conference on Information Systems, Boston, MA, USA, Dec. 2001, p. 46.

[32] M. S. Kohn et al., "IBM's Health Analytics and Clinical Decision Support," *Yearbook of Medical Informatics*, vol. 9, pp. 154-162, Aug. 2014, doi: 10.15265/IY-2014-0002.

[33] H. Honda and M. Hagiwara, “Question Answering Systems With Deep Learning-Based Symbolic Processing,” *IEEE Access*, vol. 7, pp. 152368-152378, 2019, doi: 10.1109/ACCESS.2019.2948081.

[34] J. W. Shavlik and G. G. Towell, “An Approach to Combining Explanation-based and Neural Learning Algorithms,” *Connection Science*, vol. 1, no. 3, pp. 231-253, Jan. 1989, doi: 10.1080/09540098908915640.

[35] G. G. Towell and J. W. Shavlik, “Knowledge-based artificial neural networks,” *Artificial Intelligence*, vol. 70, no. 1, pp. 119-165, Oct. 1994, doi: 10.1016/0004-3702(94)90105-8.

[36] A. S. Avila Garcez and G. Zaverucha, “The Connectionist Inductive Learning and Logic Programming System,” *Applied Intelligence*, vol. 11, no. 1, pp. 59-77, Jul. 1999, doi: 10.1023/A:1008328630915.

[37] L. Shastri, “Neurally motivated constraints on the working memory capacity of a production system for parallel processing: Implications of a connectionist model based on temporal synchrony,” in *Proceedings of 14th Annual Conference of the Cognitive Science Society*, Bloomington, IN, USA: Psychology Press, vol. 14, Jul./Aug. 1992, p. 159.

[38] L. Ding, “Neural Prolog-the concepts, construction and mechanism,” in *Proceedings of 1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, Oct. 1995, vol. 4, pp. 3603-3608 vol.4. doi: 10.1109/ICSMC.1995.538347.

[39] M. V. M. França, G. Zaverucha, and A. S. d’Avila Garcez, “Fast relational learning using bottom clause propositionalization with artificial neural networks,” *Machine Learning*, vol. 94, no. 1, pp. 81-104, Jan. 2014, doi: 10.1007/s10994-013-5392-1.

[40] E. Komendantskaya, “Unification neural networks: unification by error-correction learning,” *Logic Journal of the IGPL*, vol. 19, no. 6, pp. 821-847, Dec. 2011, doi: 10.1093/jigpal/jzq012.

[41] S. Hölldobler, “A structured connectionist unification algorithm,” in *Proceedings of the eighth National conference on Artificial intelligence - Volume 1*, Boston, Massachusetts, Jul. 1990, pp. 587-593.

[42] G. Sourek, V. Aschenbrenner, F. Zelezny, S. Schockaert, and O. Kuzelka, “Lifted Relational Neural Networks: Efficient Learning of Latent Relational Structures,” *Journal of Artificial Intelligence Research*, vol. 62, pp. 69-100, May 2018, doi: 10.1613/jair.1.11203.

[43] W. W. Cohen, “TensorLog: A Differentiable Deductive Database,” 2016, *arXiv:1605.06523*. [Online]. Available: http://arxiv.org/abs/1605.06523

[44] L. Serafini and A.S. d’Avila Garcez, “Logic tensor networks: Deep learning and logical reasoning from data and knowledge,” in *Proceedings of 11th International Workshop Neural-Symbolic Learning and Reasoning*, New York, NY, USA, 2016, pp. 1-12.

[45] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei,“Language Models are Few-Shot Learners,”2020, *arXiv:2005.14165*. [Online]. Available: http://arxiv.org/abs/2005.14165

[46] A. Vaswani N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez,L. Kaiser, and I. Polosukhi, “Attention is all you need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, Dec. 2017, pp. 6000-6010.

[47] D. Mostafa, G. Stephan, V. Oriol, U. Jakob, and K. Lukasz, “Universal Transformers”, in *Proceedings of the International Conference on Learning Representations*, New Orleans, LA, USA, May. 2019.

[48] M. Nye, M.H. Tessler, J.B. Tenenbaum, and B.M. Lake, “Improving Coherence and Consistency in Neural Sequence Models with Dual-System, Neuro-Symbolic Reasoning”, in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, Dec. 2021.

[49] T. Lai, T. Bui, S. Li, and N. Lipka, “A Simple End-to-End Question Answering Model for Product Information,” in *Proceedings of the First Workshop on Economics and Natural Language Processing*, Melbourne, Australia, Jul. 2018, pp. 38-43. doi: 10.18653/v1/W18-3105.

[50] B. Peng, Z. Lu, H. Li, and K.-F. Wong, “Towards Neural Network-based Reasoning,” 2015, *arXiv:1508.05508*. [Online]. Available: http://arxiv.org/abs/1508.05508

[51] D. Weissenborn, “Separating Answers from Queries for Neural Reading Comprehension,” 2016, *arXiv:1607.03316*. [Online]. Available: http://arxiv.org/abs/1607.03316

[52] Y. Shen, P. Huang, J. Gao, and W. Chen, “Reasonet: Learning to stop reading in machine comprehension,” in *Proceedings of 23rd ACM SIGKDD International Conference Knowledge Discovery Data Mining,* Barcelona, Spain, Aug. 2017, pp. 1047-1055.

[53] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, Cambridge, MA, USA, Dec. 2014, pp. 3104-3112.

[54] D. Bahdanau, K. Cho, and Y. Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate,” in *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA, Jun. 2015, pp. 1-15.

[55] L. Dong and M. Lapata, “Language to Logical Form with Neural Attention,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, Aug. 2016, pp. 33-43. doi: 10.18653/v1/P16-1004.

[56] P. Yin and G. Neubig, “TRANX: A Transition-based Neural Abstract Syntax Parser for Semantic Parsing and Code Generation,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Brussels, Belgium, Nov. 2018, pp. 7-12. doi: 10.18653/v1/D18-2002.

[57] V. L. Shiv and C. Quirk, “Novel positional encodings to enable tree-based transformers,” in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA: Curran Associates Inc., 2019, pp. 12081-12091.

[58] D. Dalal and B. V. Galbraith, “Evaluating Sequence-to-Sequence Learning Models for If-Then Program Synthesis,” 2020, *arXiv:2002.03485*. [Online]. Available: http://arxiv.org/abs/2002.03485

[59] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[60] I. Bratko, *Prolog Programming for Artificial Intelligence*. 2nd Edition. Reading, MA, USA: Addison-Wesley, 1990, p. 597.

[61] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in *Proceedings of the International Conference on Learning Representations*, Scottsdale, AZ, USA, 2013, pp. 1-12.

[62] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, Lake Tahoe, NV, USA, 2013, pp. 3111-3119.

[63] T. Mikolov, W. Yih, and G. Zweig, "Linguistic Regularities in Continuous Space Word Representations," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, Jun. 2013, pp. 746-751.

[64] F. Gray, "Pulse code communication," U.S. Patent 2 632 058 A, Mar. 17, 1953.

[65] H. Kanayama, Y. Miyao, and J. Prager, "Answering Yes/no questions via question inversion," in *Proceedings of the 24th International Conference on Computational Linguistics*, Mumbai, India, Dec. 2012, pp. 1377-1392.

[66] D. Ravichandran and E. Hovy, "Learning surface text patterns for a Question Answering system," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, USA, Jul. 2002, pp. 41-47. doi: 10.3115/1073083.1073092.

[67] R. Higashinaka and H. Isozaki, "Corpus-based Question Answering for Why-questions," in *Proceedings of International Joint Conference on Natural Language Processing*, Hyderabad, India, 2008, pp. 418-425.

[68] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179-211, Apr. 1990, doi: 10.1016/0364-0213(90)90002-E.

[69] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," in *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673-2681, Nov. 1997, doi: 10.1109/78.650093.

[70] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout Networks," in *Proceedings of the 30th International Conference on Machine Learning*, May 2013, pp. 1319-1327.

[71] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2017, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[72] A. Maas, A. Hannu, and A. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of 30th International Conference on Machine Learning*, Atlanta, GA, USA, 2013, p. 3.

[73] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, Jun. 2019, pp. 4171-4186. doi: 10.18653/v1/N19-1423.

[74] Kinsources: A Collaborative Web Platform for Kinship Data Sharing. Accessed: May 19, 2018. [Online]. Available: https://www.kinsources.net/

[75] L. R. Tang and R. J. Mooney, "Automated Construction of Database Interfaces: Intergrating Statistical and Relational Learning for Semantic Parsing," in *Proceedings of 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, Hong Kong, China, Oct. 2000, pp. 133-141. doi: 10.3115/1117794.1117811.

[76] Y. Bekkers, B. Caner, O. Ridoux, and L. Ungaro, "MALI: a memory with a real-time garbage collector for implementing logic programming languages," in *Proceedings of the 3rd Symposium on Logic Programming*, Salt Lake City, UT, USA, Sep. 1986, pp. 258-264.

[77] K. Appleby, M. Carlsson, S. Haridi, and D. Sahlin, "Garbarge collection for Prolog based on WAM," *Communications of the ACM*, vol. 31, no. 6, pp. 719-741, Jun. 1988, doi: 10.1145/62959.62968.

[78] Y. Bekkers, O. Ridoux, and L.Ungaro, "Dynamic Memory Management for Sequential Logic Programming Languages," in *Proceedings of Memory Management: International Workshop*, St. Malo, France, Sep. 1992, pp. 82-102.

[79] N. Cingillioglu and A. Russo, "DeepLogic: Towards End-to-End Differentiable Logical Reasoning," 2018, *arXiv:1805.07433*. [Online]. Available: http://arxiv.org/abs/1805.07433

[80] S. Reed, Y. Zhang, Y. Zhang, and H. Lee, "Deep visual analogy-making," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, Cambridge, MA, USA, Dec. 2015, pp. 1252-1260.

[81] F. Hill, A. Santoro, D. Barrett, A. Morcos, T. Lillicrap, "Learning to Make Analogies by Contrasting Abstract Relational Structure," in *Proceedings of the International Conference on Learning Representations*, New Orleans, LA, USA, 2019, pp. 1-18.

[82] M. Hagiwara and Y. Anzai, “Analogical reasoning by neural network,” in *Proceedings of IJCNN-91-Seattle International Joint Conference on Neural Networks*, Seattle, WA, USA, Jul. 1991, p. 995. doi: 10.1109/IJCNN.1991.155664.

[83] Y. Salu, “A neural network for analogical reasoning,” in *Proceedings of 1994 IEEE International Conference on Neural Networks*, Orlando, FL, USA, Jun. 1994, pp. 4772-4777. doi: 10.1109/ICNN.1994.375047.

[84] R. Kumaraswamy, P. Odom, K. Kersting, D. Leake, and S. Natarajan, “Transfer Learning via Relational Type Matching,” in *Proceedings of 2015 IEEE International Conference on Data Mining*, Nov. 2015, pp. 811-816. doi: 10.1109/ICDM.2015.138.

[85] L. Mihalkova and R. Mooney, “Transfer learning by mapping with minimal target data,” in *Proceedings of AAAI Workshop Transfer Learning Complex Tasks*, Chicago, IL, USA, 2008, pp. 1-6.

[86] H. Honda and M. Hagiwara, “Analogical Reasoning With Deep Learning-Based Symbolic Processing,” *IEEE Access*, vol. 9, pp. 121859-121870, 2021, doi: 10.1109/ACCESS.2021.3109443.

[87] J. G. Carbonell, “A computational model of analogical problem solving,” in *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 1*, San Francisco, CA, USA, Aug. 1981, pp. 147-152.

[88] D. Gentner, “Structure-mapping: A theoretical framework for analogy,” *Cognitive Science*, vol. 7, no. 2, pp. 155-170, Apr. 1983, doi: 10.1016/S0364-0213(83)80009-3.

[89] B. Falkenhainer, K. D. Forbus, and D. Gentner, “The structure-mapping engine: Algorithm and examples,” *Artificial Intelligence*, vol. 41, no. 1, pp. 1-63, Nov. 1989, doi: 10.1016/0004-3702(89)90077-5.

[90] K. J. Holyoak and P. Thagard, “Analogical mapping by constraint satisfaction,” *Cognitive Science*, vol. 13, no. 3, pp. 295-355, 1989, doi: 10.1207/s15516709cog1303_1.

[91] D. Hofstadter. *Fluid concepts and creative analogies: Computer Models of the Fundamental Mechanisms of Thought*. NY, USA: Basic Books, 1996, p. 528.

[92] J. E. Hummel and K. J. Holyoak, “Distributed representations of structure: A theory of analogical access and mapping,” *Psychological Review*, vol. 104, no. 3, pp. 427-466, 1997, doi: 10.1037/0033-295X.104.3.427.

[93] L. B. Larkey and B. C. Love, "CAB: Connectionist Analogy Builder," *Cognitive Science*, vol. 27, no. 5, pp. 781-794, 2003, doi: 10.1016/S0364-0213(03)00066-1.

[94] D. Gentner and K. D. Forbus, "Computational models of analogy," *Wiley Interdiscip Rev Cogn Sci*, vol. 2, no. 3, pp. 266-276, May 2011, doi: 10.1002/wcs.105.

[95] K. J. Holyoak, "The Pragmatics of Analogical Transfer," in *Psychology of Learning and Motivation*, vol. 19, G. H. Bower, Ed. Academic Press, 1985, pp. 59-87. doi: 10.1016/S0079-7421(08)60524-1.

[96] B. A. Spellman and K. J. Holyoak, "Pragmatics in analogical mapping," *Cogn Psychol*, vol. 31, no. 3, pp. 307-346, Dec. 1996, doi: 10.1006/cogp.1996.0019.

[97] H. Liu, Y. Wu, and Y. Yang, "Analogical Inference for Multi-relational Embeddings," in *Proceedings of the 34th International Conference on Machine Learning*, Jul. 2017, pp. 2168-2178.

[98] Y. Wu, X. Liu, Y. Feng, Z. Wang, R. Yan, and D. Zhao, "Relation-aware entity alignment for heterogeneous knowledge graphs," in *Proceedings of 28th International Joint Conferences on Artificial Intelligence*, Aug. 2019, pp. 5278-5284.

[99] H. Wang and Q. Yang, "Transfer Learning by Structural Analogy," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, no. 1, Art. no. 1, Aug. 2011.

[100] T. Hinrichs and K. D. Forbus, "Transfer Learning through Analogy in Games," *AI Magazine*, vol. 32, no. 1, Art. no. 1, Mar. 2011, doi: 10.1609/aimag.v32i1.2332.

[101] R. Kumaraswamy, N. Ramanan, P. Odom, and S. Natarajan, "Interactive Transfer Learning in Relational Domains," *Künstl Intell*, vol. 34, no. 2, pp. 181-192, Jun. 2020, doi: 10.1007/s13218-020-00659-6.

[102] H. Larochelle, D. Erhan, and Y. Bengio, "Zero-data learning of new tasks," in *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, Chicago, Illinois, Jul. 2008, pp. 646-651.

[103] M. Palatucci, D. Pomerleau, G. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, Dec. 2009, pp. 1410-1418.

[104] R. Socher, M. Ganjoo, C. D. Manning, and A. Y. Ng, "Zero-shot learning through cross-modal transfer," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, Red Hook, NY, USA, Dec. 2013, pp. 935-943.

[105] S. Reed, Z. Akata, H. Lee, and B. Schiele, "Learning Deep Representations of Fine-Grained Visual Descriptions," in *Proceedings of 2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp. 49-58. doi: 10.1109/CVPR.2016.13.

[106] Z. Xie, W. Cao, X. Wang, Z. Ming, J. Zhang, and J. Zhang, "A biologically inspired feature enhancement framework for zero-shot learning," in *Proceedings of 7th IEEE International Conference on Cyber Security and Cloud Computing /6th IEEE International Conference on Edge Computing*, Aug. 2020, pp. 120-125.

[107] Z. Xie, W. Cao, and Z. Ming, "A further study on biologically inspired feature enhancement in zero-shot learning," *International Journal of Machine Learning and Cybernetics*, vol. 12, Jan. 2021, doi: 10.1007/s13042-020-01170-y.

[108] IMDb: Internet Movie Database. Accessed: Nov. 29, 2019. [Online]. Available: http://klog.dinfo.unifi.it/data/imdb/ext.pl.gz

[109] K. Sinha, S. Sodhani, J. Dong, J. Pineau, and W. L. Hamilton, "CLUTRR: A Diagnostic Benchmark for Inductive Reasoning from Text," 2019, *arXiv:1908.06177*. [Online]. Available: http://arxiv.org/abs/1908.06177

[110] W. B and H. M. E, "Adaptive switching circuits.," *IRE Wescon Conv Rec*, vol. 4, no. 4, pp. 96-101, 1960.

[111] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279-292, May 1992, doi: 10.1007/BF00992698.

[112] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," 2013, *arXiv:1312.5602*. [Online]. Available: http://arxiv.org/abs/1312.5602

[113] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D.

Hassabis, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484-489, Jan. 2016, doi: 10.1038/nature16961.

[114] Z. C. Lipton, “The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery.,” *Queue*, vol. 16, no. 3, pp. 31-57, Jun. 2018, doi: 10.1145/3236386.3241340.

[115] F. Doshi-Velez, and B. Kim, “Towards a rigorous science of interpretable machine learning,” 2017, *arXiv:1702.08608*. [Online]. Available: http://arxiv.org/abs/1702.08608

[116] A. A. Freitas, “Comprehensible classification models: a position paper,” *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 1, pp. 1-10, Mar. 2014, doi: 10.1145/2594473.2594475.

[117] G. Montavon, W. Samek, and K.-R. Müller, “Methods for interpreting and understanding deep neural networks,” *Digital Signal Processing*, vol. 73, pp. 1-15, Feb. 2018, doi: 10.1016/j.dsp.2017.10.011.

[118] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera,“Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI,” *Information Fusion*, vol. 58, pp. 82-115, Jun. 2020, doi: 10.1016/j.inffus.2019.12.012.

[119] A. Verma, V. Murali, R. Singh, P. Kohli, and S. Chaudhuri, “Programmatically Interpretable Reinforcement Learning,” in *Proceedings of the 35th International Conference on Machine Learning*, Jul. 2018, pp. 5045-5054.

[120] D. Hein, A. Hentschel, T. Runkler, and S. Udluft, “Particle swarm optimization for generating interpretable fuzzy reinforcement learning policies,” *Engineering Applications of Artificial Intelligence*, vol. 65, pp. 87-98, 636424128000000000, doi: 10.1016/j.engappai.2017.07.005.

[121] G. Liu, O. Schulte, W. Zhu, and Q. Li, “Toward Interpretable Deep Reinforcement Learning with Linear Model U-Trees,” in *Proceedings of European Conference on Machine Learning / European Conference on Principles of Data Mining and Knowledge Discovery*, Dublin, Ireland, Sep. 2018, pp. 414-429. doi: 10.1007/978-3-030-10928-8_25.

[122] T. Shu, C. Xiong, and R. Socher, “Hierarchical and interpretable skill acquisition in multi-task reinforcement learning,” 2017, *arXiv:1712.07294*. [Online]. Available: http://arxiv.org/abs/1712.07294

[123] P. Sequeira and M. Gervasio, “Interestingness elements for explainable reinforcement learning: Understanding agents, capabilities and limitations,” 2019, *arXiv:1912.09007*. [Online]. Available: http://arxiv.org/abs/1912.09007

[124] Y. Fukuchi, M. Osawa, H. Yamakawa, and M. Imai, “Autonomous self-explanation of behavior for interactive reinforcement learning agents,” in *Proceedings of the 5th International Conference on Human Agent Interaction*, Oct. 2017, pp. 97-101. doi: 10.1145/3125739.3125746.

[125] J. H. Lee, “Complementary reinforcement learning towards explainable agents,” 2019, *arXiv:1901.00188*. [Online]. Available: http://arxiv.org/abs/1901.00188

[126] J. Waa, J. Diggelen, K. Bosch, M. Neerincx, “Contrastive explanations for reinforcement learning in terms of expected consequences,” in *Proceedings of the IJCAI 2018 Workshop on Explainable Artificial Intelligence,* Stockholm, Sweden Jul. 2018.

[127] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, “Explainable reinforcement learning through a causal lens,” 2019, *arXiv:1905.10958*. [Online]. Available: http://arxiv.org/abs/1905.10958

[128] Y. Coppens, K. Efthymiadis, T. Lenaerts, and A. Nowe, “Distilling Deep Reinforcement Learning Policies in Soft Decision Trees,” in *Proceedings of the IJCAI 2019 Workshop on Explainable Artificial Intelligence*, Macau, China, Aug. 2019, pp. 1-6.

[129] P. Minervini, S. Riedel, P. Stenetorp, E. Grefenstette, and T. Rocktäschel, “Learning Reasoning Strategies in End-to-End Differentiable Proving,” in *Proceedings of the 37th International Conference on Machine Learning*, Nov. 2020, pp. 6938-6949.

[130] H. Honda and M. Hagiwara, (in press). “Deep-Learning-Based Fuzzy Symbolic Processing with Agents Capable of Knowledge Communication,” In *Proceedings of 14th International Conference on Agents and Artificial Intelligence*, Feb. 2022.

[131] G. Kuhlmann, P. Stone, R. Mooney, and J. Shavlik, “Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer,” In *Proceedings of the AAAI-2004 Workshop on Supervisory Control of Learning and Adaptive Systems*, Jul. 2004.

[132] D. K. Misra, J. Sung, K. Lee, and A. Saxena, “Tell me dave: Context-sensitive grounding of natural language to manipulation instructions,” *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 281-300, 2016.

[133] J. Kim, T. Misu, Y.-T. Chen, A. Tawari, and J. Canny, "Grounding Human-To-Vehicle Advice for Self-Driving Vehicles," in *Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2019, pp. 10583-10591. doi: 10.1109/CVPR.2019.01084.

[134] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016, *arXiv:1606.01540*. [Online]. Available: http://arxiv.org/abs/1606.01540

[135] C. E. Osgood, G. Suci, and P. Tannenbaum, *The measurement of meaning*, IL, USA: University of Illinois Press, 1957, p.360.

[136] C. E. Osgood, W. H. May, and M. S. Miron, *Cross-Cultural Universals of Affective Meaning*. IL, USA: University of Illinois Press, 1975, p. 486.

[137] R. Likert, "A technique for the measurement of attitudes," *Archives of Psychology*, vol. 22 140, pp. 55-55, 1932.

[138] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-Learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Phoenix, Arizona, Feb. 2016, pp. 2094-2100.

[139] G. Gazdar,E. Klein,G. K. Pullum, and I. A. Sag, *Generalized Phrase Structure Grammar*. MA, USA: Harvard University Press, 1985, p. 320.

[140] C. Pollard and I. A. Sag, *Information-based Syntax and Semantics*. CA, USA: CSLI Publications, 1987, p. 233.

[141] J. Bresnan, A. Asudeh, I. Toivonen, and S. Wechsler, *Lexical Functional Syntax*. 2nd Edition, NJ, USA: Wiley-Blackwell, 2013, p. 536.

[142] H. Honda and M. Hagiwara, "Reproduction of Deep Reinforcement Learning Model with Fuzzy Symbolic Representation," In *Proceedings of the 22nd International Symposium on Advanced Intelligent Systems*, Cheongju, Korea, Dec. 2021.

謝辞

本研究は，著者が慶應義塾大学大学院理工学研究科後期博士課程在学中に，同大学理工学部萩原将文教授の指導のもとに行われたものです．本研究は研究のすゝめ奨学金並びに KLL 後期博士課程研究助成金の支援を受けています．

主指導教員である萩原将文先生には，親身にご指導いただき，大変お世話になりました．萩原先生には，研究課題の発見の仕方，課題への取り組み方など数多くのことを学ばせていただきました．萩原先生のご指導により，今後の研究者としての礎を築くことができたと思います．また，自由に研究ができる環境を提供いただき，非常に楽しく研究に取り組むことができました．本当にありがとうございました．

ご多忙な中，副査を引き受けていただいた今井倫太先生，杉浦孔明先生，櫻井彰人先生に深く感謝申し上げます．

今井先生には，本研究の今後の展開に向けたアドバイスを多くいただきました．いただいたアドバイスを元に今後も本研究を発展させていきたいと思います．本当にありがとうございました．

杉浦先生には，本研究に関するアドバイスだけではなく，研究に対する取り組み方や姿勢についても親身にご指導いただきました．今後も引き続き研究者として研鑽に努めていきたいと思います。本当にありがとうございました．

櫻井先生には，社会人での博士課程進学のご相談に乗っていただき，進学を後押しいただきました．また，櫻井先生に学部在学中にご指導いただいた「ニューラルネットワークによる文法学習」に関する卒業論文と修士在学中にご指導いただいた「論理プログラミングによる構文解析」に関する修士論文が本研究の基礎となっています．本研究はこれまで慶應義塾大学で学んだことの集大成とすることができました．本当にありがとうございました．

最後に，社会人博士課程進学をご承諾くださった勤務先の本田技研工業株式会社並びに，博士課程進学に理解を示し支えてくれた家族に感謝致します．

付録

A. 人の曖昧な言語表現に基づいた強化学習モデルの再現

A.1 人の曖昧な言語表現に関する実験

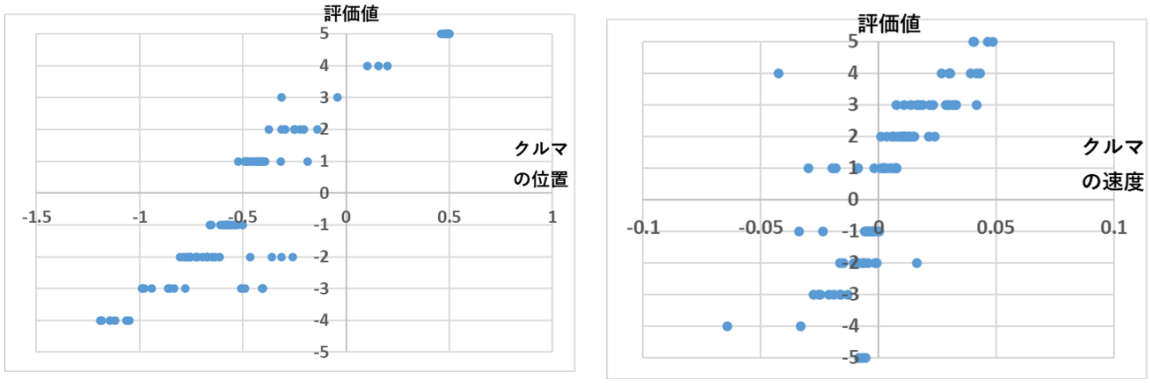
OpenAI Gym [134] が提供する強化学習のシミュレーション環境の一つである MountainCar-v0 を用いて，シミュレーション環境の状態を人が言語表現をした場合の曖昧さについて評価実験を行った．本付録における研究は [142] の研究をベースとして行った．MountainCar-v0 の状態はクルマの位置（-1.2～0.6 の範囲）とクルマの速度（-0.07～0.07 の範囲）の 2 種類の連続値である．行動はクルマの行動であり，0（左へ加速），1（加速しない），2（右へ加速）の離散値となる．

評価実験の実施にあたり，double deep Q network （DDQN）[138] による学習中におけるシミュレーション環境の動画をエピソード毎に保存した．学習が完了するまでの全 2,398 エピソードの中からランダムに 100 エピソードの動画を選択し，被験者に提示する．被験者は，動画が終了するタイミングでのクルマの位置とクルマの速度をそれぞれ-5～5 の 10 段階（言語的変数はとても大きく左，大きく左，左，小さく左，とても小さく左，とても小さく右，小さく右，右，大きく右，とても大きく右に対応する.）で評価する．

図 A.1 に被験者 1 名による評価結果を示す．結果から各評価値の多くは一定範囲に収まっているものの，被験者の誤りなどによる外れ値も含まれていることがわかる．

A.2 メンバーシップ関数の作成

A.1 節の実験の結果，各評価値の範囲は前後の評価値と重なっている部分があることから，



(a) クルマの位置　　(b) クルマの速度

図 A.1　人 の曖昧な言語表現に関する実験の結果

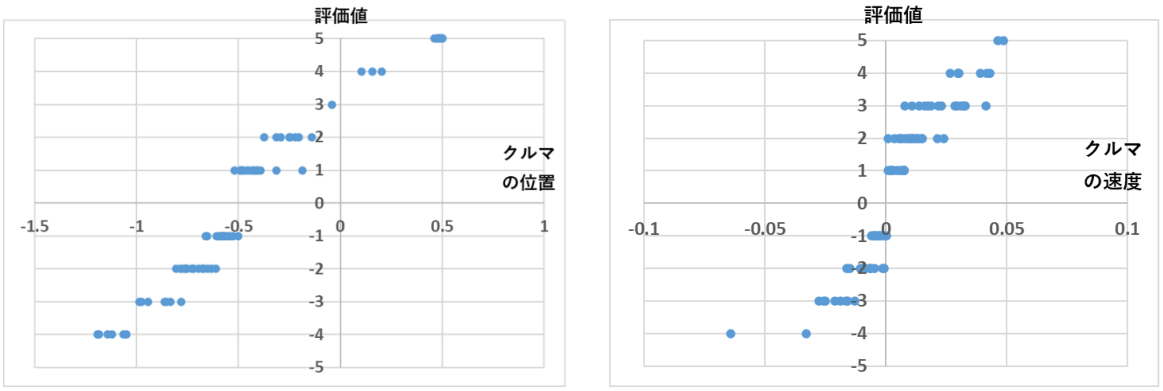
Algorithm A.1 : スクリーニングアルゴリズム

Input: *EvaluationResults*
Output: *ScreenedResults*
DeletedFlag ← *True*
while *DeletedFlag* **do**
  DeletedFlag ← *False*
  for *ScoreNum* = 0 **to** size(*EvaluationResults*) **do**
    for *ValueNum* = 0 **to** size(*EvaluationResults*[*ScoreNum*]) **do**
      if *ScoreNum* > *0*
        if *EvaluationResults*[*ScoreNum*][*ValueNum*] > max(*EvaluationResults*[*ScoreNum*-1])
          delete(*EvaluationResults*[*ScoreNum*][*ValueNum*])
          DeletedFlag ← *True*
      if *ScoreNum* < size(*EvaluationResults*) -1
        if *EvaluationResults*[*ScoreNum*][*ValueNum*] < min(*EvaluationResults*[*ScoreNum*+1])
          delete(*EvaluationResults*[*ScoreNum*][*ValueNum*])
          DeletedFlag ← *True*
for *ScoreNum* = 0 **to** size(*EvaluationResults*)-2 **do**
  for *ValueNum* = 0 **to** size(*EvaluationResults*[*ScoreNum*]) **do**
    if *EvaluationResults*[*ScoreNum*][*ValueNum*] > min(*EvaluationResults*[*ScoreNum*+1]) and
        EvaluationResults[*ScoreNum*][*ValueNum*] > min(*EvaluationResults*[*ScoreNum*+*2*])
      delete(*EvaluationResults*[*ScoreNum*][*ValueNum*])
ScreenedResults ← *EvaluationResults*
return *ScreenedResults*

図 A.2　スクリーニングアルゴリズム

台形型のメンバーシップ関数を作成するのが適していると考えられる．しかしながら，主観評価結果は外れ値を含み，3 種類以上の評価値で値が重なっているものも存在する．そこで，主観評価結果からメンバーシップ関数の作成を可能にするために，データのスクリーニングを行う．

図 A.2 にスクリーニングのアルゴリズムを示す．本アルゴリズムに評価値別の値のリスト *EvaluationResults* を入力すると，スクリーニングされた *ScreenedResults* を出力する． 本アルゴリズムでは，まず初めに，評価値ごとの値を探索していき，自身より 1 小さい評価値の最



(a) クルマの位置　　(b) クルマの速度

図 A.3　スクリーニングの結果

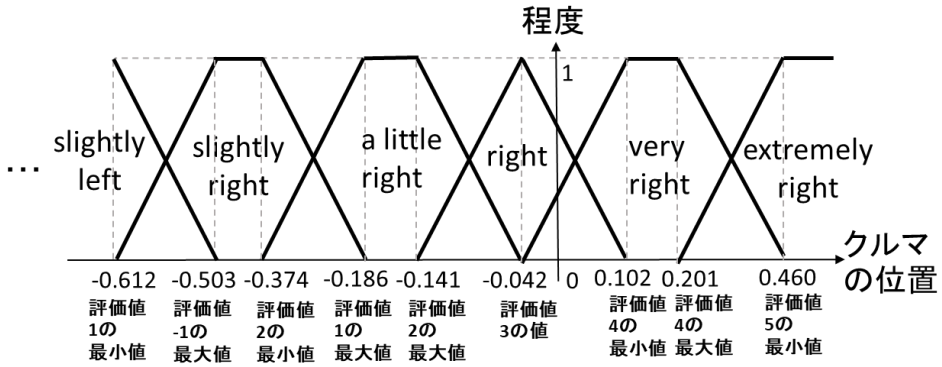


図 A.4　クルマの位置のメンバーシップ関数

大値よりも大きい値，及び自身より 1 大きい評価値の最小値より小さい値を繰り返し削除する．(図 A.2 2-13 行目）続いて，再度評価値ごとの値を探索し，自身より 1 小さい評価値の値の範囲及び自身より 1 大きい評価値の値の範囲の両方に含まれる値を削除する．(図 A.2 14-17 行目）図 A.1 を図 A.2 のアルゴリズムでスクリーニングした結果を図 A.3 に示す．

図 A.3(a)の一部をメンバーシップ関数に変換したものを図 A.4 に示す．メンバーシップ関数を用いて状態の値から言語的変数を導く際は，程度の大きさに応じた確率に従って言語的変数を決定する．

表 A.1　データセット及び実験の結果

Training Set	100,000
Validation Set	12,500
Test Set	12,500
一致率	0.9716

A.3　人の曖昧な言語表現に基づく深層強化学習モデルの再現実験

A.2 節で作成したメンバーシップ関数を用いて，4.4.1 項の手法により MountainCar-v0 の深層強化学習モデルを再現した．深層強化学習モデルの再現に使用したデータセット及び実験結果を表 A.1 に示す．データセットは，DDQN により 1 エピソードの平均報酬が 160 を超えるまで学習を行った深層強化学習モデルを繰り返し試行することで生成した．データセットの記号化は 4.3.2 項に示した方法で行った．メンバーシップ関数の作成には A.1 節の評価実験の結果を A. 2 節の方法でスクリーニングしたものを使用した．また，データセットはランダム化した上で，Training Set，Validation Set，Test Set に分割した．

前述のデータセットを用いて実験をした結果，一致率は 0.9716 となった．一致率は，再現モデルにテストデータの状態を入力し，得られた行動がテストデータの行動と正確に一致した割合を表す．実験の結果，学習済みの深層強化学習モデルを高い精度でファジィ化した記号表現を用いた深層学習モデルで再現することができた．