2022 年度博士学位論文

記憶装置付きニューラルネットワークモデルによる構造化知識を用いた質問応答・対話

お茶の水女子大学大学院
人間文化創成科学研究科
理学専攻
村山 友理

指導教員 小林一郎 教授

2023 年 3 月

要約

本研究では，構造化知識を用いた質問応答及び対話に取り組んだ．自然言語表現と Linked Open Data 形式で記述された知識ベースを結びつけるために，自然言語文を知識ベース検索用のクエリ言語 SPARQL に変換することで知識への問い合わせを行う様々な研究が行われてきたが，これらにおいて自然言語文から生成したクエリが正しい答えを返すかは保証されていない．そこで，本研究では先行研究のアプローチを踏まえて，自然言語文から抽出したエンティティを知識側の語彙と対応付け，エンティティ間のリレーションを知識を繰り返し参照することで獲得することに重点を置いた SPARQL クエリの自動生成手法を提案する．KNP による述語項構造・係り受け解析と，データ主導による RDF トリプルの変形を用いた自然言語文から SPARQL クエリへの変換手法により，クエリが完成されれば答えを返すことが保証される．また，基本型，Degree 型，Count 型，List 型の 4 つの質問タイプを設定し，それぞれの質問タイプの自然言語質問文及び，より一般化した，それらを組み合わせた質問文に対してそれぞれ適切な SPARQL クエリを生成することができた．質問タイプを設定することにより，正確なクエリを容易に生成し，自然言語文が意図する質問に回答することを可能にした．

現在，自然言語処理では Transformer を用いた研究が主流であるが，いくつか問題点も指摘されている．その中の一つとして，Transformer の長期的な情報保存についての問題は Differentiable Neural Computer (DNC) が解決したが，別の問題として Transformer のような汎用言語モデルは，大規模テキストデータで事前学習を行うことで，さまざまな自然言語処理タスクを解くことができる一方で，知識利用や演算処理などの能力は大量のコーパス学習から暗黙的に得られるとされ，正しさは保証されないという問題がある．そこで，本研究の Research Question として，DNC をベースに Transformer と知識利用・演算処理を行うアーキテクチャを組み込むことで，知識と演算を必要とする自然言語処理タスクでの性能の向上を目指す．本研究の目的は，計算機の原理であるチューリングマシンをニューラルネットワークで模した DNC に，Transformer と知識利用・演算処理を行うアーキテクチャを組み込んだモデルを構築し，知識と演算

を必要とする自然言語処理タスクを扱えるか検証を行うことである．

提案モデルを構築する前に，DNC を日本語対話に適用する予備研究を行った．文脈を考慮した対話の研究は，言語として英語を扱ったものが大半であり，日本語を対象としたものは少ない．また，日本語による対話のデータセットはいくつか存在するが，明示的に文脈理解を問うようなデータセットではない．そこで，本研究では既存の英語のデータセットである bAbI データセットを参考に，喫茶店での注文の場面を想定した日本語注文対話データセットを作成し，課題の解決を試みる．作成したデータセットに対し DNC を適用することで，文脈を用いた日本語注文対話システムを構築する．実験を行なった結果，平均テスト誤り率として低い値を得ることができた．

文脈理解と知識活用は対話にとって重要であるが，これらの課題を扱った研究は未だあまり進められていない．また，対話のアーキテクチャとして Seq2Seq や，T5，BART といった sequence-to-sequence なモデル（相手の発話を入力として応答となる発話を生成するモデル）が主流になっている．一方，より自然で知的な対話を行うには文脈理解や知識活用が必要であるが，そのための長期間におけるデータ保存の能力には限界があると議論されてきた．そこで，文脈情報などの長期の情報を保持するために，DNC などの記憶装置付きニューラルネットワークモデルが提案されている．これらのモデルは記憶装置を付け加えたことにより従来のモデルに比べてより複雑な情報処理を行えるようになり，文脈を踏まえた対話においても高い精度を実現している．本研究では，DNC を基盤とした，文脈を捉えつつ，分散表現による構造化知識を用いた手法を提案する．一貫性があり，かつ大規模な知識を要求する質問応答で構成された CSQA データセットを用いた実験では，全体の評価事例における予測単語が不正解であった割合を表す誤り率によって提案モデルとオリジナルの DNC モデルを評価した結果，全体の結果は DNC より下がってしまったが，10 項目中 5 項目で結果が向上した．また，Dialog bAbI データセットを用いた実験では，DNC, rsDNC, DNC-MD に対して知識を扱うメモリを追加した提案モデルがオリジナルモデルを上回った．特に，各モデルは知識を要求するタスクで，それぞれおよそ 14%, 20%, 7%向上した．Movie Dialog データセットの実験では，rsDNC, DNC-DMS に対する提案モデルがオリジナルモデルより良い結果になった．

知識グラフを埋め込み空間内に表現し，論理計算を行う質問応答手法として Conditional Theorem Provers や Query2Box が提案されている．しかし，これらの手法では限られた計算のみしか扱うことができない．本研究では，DNC と，さらに DNC を改良した rsDNC と DNC-DMS に対して，質問応答タスクにおいて重要な要素である知

識利用と演算処理を新たに組み入れ，構造化知識に対する演算処理を含んだ質問文について正しい答えを生成するために任意の演算を用いて能力を向上させることを目指す．アメリカの地理に関する知識と演算処理を要求する GEO データセットとそれを拡張したデータセットにおいて，BERT をファインチューニングした結果より，DNC モデルをベースに BERT と知識を扱うメモリと演算を行うユニットを追加した提案手法の結果がすべて平均 top-1 accuracy と平均 top-10 accuracy の両方で向上した．さらに，rsDNC を改良した提案手法は GEO データセットにおける平均 top-1 accuracy で最も良い結果を達成し，総合的に DNC より高いスコアを得た．

キーワード：自然言語処理，構造化知識，深層学習

Abstract

Various approaches to connect natural language with knowledge have been studied. As the common problem for their studies, they aim to convert natural language into query language in order to access from natural language expression to knowledge represented in the formal form such as Linked Open Data. Based on the approaches of prior studies, in this study, we propose a method that generates SPARQL queries by iteratively referencing to knowledge compiled as Linked Open Data. In the experiment, our method was able to generate correct SPARQL queries for each question type.

Nowadays Transformer-based studies are commonplace in the natural language processing, yet there are some problems. The Differentiable Neural Computer (DNC) solved one of the problems that Transformer cannot store data over a long time. Another problem is that a general-purpose language model such as Transformer is able to solve various natural language processing tasks by pre-training the large scale text data, whereas the ability such as knowledge using and arithmetic processing is supposed to be obtained implicitly from training vast amount of corpora and the correctness is not guaranteed. Therefore, our research question is that we aim to improve the performance on natural language processing tasks requiring knowledge and the operation by incorporating architectures for knowledge and arithmetic processing, and Transformer into DNC. Our objective is to build a model incorporating Transformer and architectures for knowledge and arithmetic processing into DNC, a neural network modeling of the Turing machine which is the principle of the computer, and verify whether the model can deal with natural language processing tasks requiring knowledge and the operation.

Before building the proposed model, we conducted a preliminary study applying DNC to Japanese dialogue. There are many works on dialogue systems considering context using the neural network. In our work, to realize a robot cafeteria, we applied DNC to the process of ordering dialogue. We created a Japanese ordering dialogue dataset and conducted an experiment on it. As the result, we obtained the low mean test error rate.

It is essential for dialogue to understand context and use knowledge. However, researches dealing with these issues have not progressed much yet. In addition, nowadays sequence-to-sequence models such as Seq2Seq, T5 and BART are commonplace for dialogue architectures. On the other hand, it is necessary for more natural and intellectual dialogue to understand context and use knowledge. However, scientists have argued that such models are limited in their ability to store data over a long time. To retain the long-term information, neural network models with an external memory such as DNC have been proposed. These models learned to process more complex information compared to standard ones by adding the memory, and achieved a highly accurate performance in dialogue considering context. In our work, we enrich the DNC architecture and propose a method using both context and structured knowledge by the distributed representation. We conducted an experiment on the CSQA dataset composed of a series of coherently linked question answering that require a large scale knowledge graph. Even though the overall test error rate of our proposed method was lower than that of DNC, our method performed better at five out of ten items. In an experiment with the Dialog bAbI dataset, our improved DNC, rsDNC and DNC-MD outperformed their original models. In particular, each model obtained an improvement of approximately 14%, 20% and 7% respectively on a task requiring knowledge. In the Movie Dialog dataset, our improved rsDNC and DNC-DMS also yield better performance than their original models.

DNC, a neural network model with an addressable external memory, can solve algorithmic and question answering tasks. As improved versions of DNC, rsDNC and DNC-DMS have been proposed. However, how to integrate structured knowledge and calculation into these DNC models remains a challenging research question. We incorporate architectures for knowledge and calculation into such DNC models, i.e. DNC, rsDNC and DNC-DMS, to improve the ability to generate correct answers for questions with multi-hop reasoning and calculation over structured knowledge. Our improved rsDNC model achieved the best performance for the mean top-1 accuracy and our improved DNC-DMS model scored the highest for top-10 accuracy in GEO dataset. In addition, our model improving rsDNC outperformed the other models with the mean top-1 accuracy and the mean top-10 accuracy in augmented GEO dataset.

Keywords: Natural Language Processing, Linked Open Data, Deep Learning

目 次

図 目 次

表 目 次

第1章　序論

1.1　研究背景と目的

近年，様々な分野の膨大なデータベースを意味的に結びつけたデータ群が大規模知識 (Linked Open Data) として公開されている．自然言語処理の中で背景知識として用いることで，表層的な情報による推論だけでは難しい，常識などを含んだ自然言語理解に役立てることができる [Bosselut 19, Young 17, Miller 16]．本研究では，構造化知識を用いた自然言語処理に取り組み，まず初めに自然言語文から知識にアクセスする研究を行った．自然言語表現と Linked Open Data 形式で記述された知識ベースを結びつけるために，自然言語文を SPARQL クエリに変換することで知識への問い合わせを行う様々な研究が行われてきた．[Wang 07] では，自然言語文の構文解析結果に対して，知識を記述するための語彙や，WordNet [Miller 95] などの一般的な辞書からなる語彙，ユーザが定義する語彙などを適用することで RDF トリプルに変換する手法を提案している．[Zou 14] では，複数の RDF トリプルをつなぎ Semantic Query Graph なる知識グラフを構築し，断片的な知識を複数のトリプルからなる知識に変換することで自然言語文の解釈の精度を高めている．また，[鈴木 15] では，自然言語文を構文解析した結果を論理式に変換し，そこから SPARQL クエリを生成する手法を提案している．しかし，これらにおいて自然言語文から生成したクエリが正しい答えを返すかは保証されていない．そこで本研究の 1 つ目では，先行研究のアプローチを踏まえて，自然言語文から抽出したエンティティを知識側の語彙と対応付け，エンティティ間のリレーションを知識を繰り返し参照することで獲得することに重点を置いた SPARQL クエリの自動生成手法を提案する．提案手法により SPARQL クエリが完成されれば，答えを返すことが保証される．

本研究では次に，構造化知識をディープニューラルネットワークに取り入れる研究を行った．まず，記憶装置付きニューラルネットワークモデル Differentiable Neural Computer (DNC) [Graves 16] の背景を述べる．近年，ディープニューラルネットワークはコンピュータビジョンや自然言語処理といったさまざまなタスクの複雑なパター

ンマッチングにおいて顕著な発展を遂げてきた. しかし，グラフや木などのデータ構造の表現や変数の使用，長い系列に対する表現の操作といった能力には限界があるとされてきた.

Recurrent Neural Network (RNN) は系列データの長距離依存を捉えることができ，チューリング完全であることも知られている [Siegelmann 95] ため，抽象的な処理を行う能力があるとされるが,「理論的には」の話であり，現実には勾配消失問題 [Bengio 94] に苦しんだ. Long Short-Term Memory (LSTM) はこの問題について，RNN アーキテクチャにゲート機構を導入し，毎タイムステップのゲート値との要素ごとの積によって勾配を計算することで解決した [Hochreiter 97]. LSTM は大きく成功し，sequence-to-sequence model [Sutskever 14] と attention mechanism [Bahdanau 14, Luong 15] のおかげで RNN を上回るのに貢献した. Transformer [Vaswani 17] は，ほとんどすべて attention mechanism を基にしており，LSTM や伝統的なモデルをいろいろなタスクで打ち破ってきた. しかし，Transformer には長期的文脈情報を固定長の系列にエンコードするせいで context fragmentation problem [Dai 19] がある. この問題を解くために，過去の情報をメモリ内にキャッシュしておくさまざまな手法 [Dai 19, Rae 19, Martins 22] が提案されてきた. けれど未だ Transformer をベースとしたモデルは，先にディープニューラルネットワークの限界として挙げた課題への真の解法には至っていない.

一方，ノイマン型コンピュータでは，プログラムは単純な算術演算と論理演算を行う処理装置，制御命令文（順次実行，条件分岐，反復）を扱う制御装置，そして計算中にデータや命令文が読み書きされるメモリという 3 つの基礎的な仕組みによって実行される [Neumann 45]．このアーキテクチャは複雑なデータ構造を表現でき，アルゴリズムタスクを解くことができるように学習する. ノイマン型コンピュータはプロセッサ（すなわち処理装置と制御装置）による計算とメモリを分けているが，ニューラルネットワークは計算とメモリをネットワークの重みとして一緒にしている. 計算とメモリを明示的に分けたニューラルネットワークとして提案されたのが Neural Turing Machine (NTM) [Graves 14] である．NTM はテープの長さが無限であることによる有限状態チューリングマシンから類推された，読み書き可能な外部メモリを持つ．NTM 全体は微分可能なため，メモリへのアクセスのやり方を含めて end-to-end に学習できる. さらに，メモリアクセス機構を改良した Differentiable Neural Computer (DNC) [Graves 16] が提案され，路線図の最短路探索タスクや家系図の推測タスクなどの構造化データ上のアルゴリズムタスクを解いた. 前提付きの質問応答における実験では，入力系列はメモリに書き込まれ，答えを推測するのに必要な情報がメモリから読み出されること

から，変数を表現している. DNC は動的メモリアクセス機構により長い系列を学習することも可能にした.

DNC は Transformer の長期的な情報保存についての問題を解決したが，別の問題も指摘されている．Transformer のような汎用言語モデルは，大規模テキストデータで事前学習を行うことで，さまざまな自然言語処理タスクを解くことができる一方で，知識利用や演算処理などの能力は大量のコーパス学習から暗黙的に得られるとされ，正しさは保証されないという問題がある．そこで，本研究の Research Question として，DNC をベースに Transformer と知識利用・演算処理を行うアーキテクチャを組み込むことで，知識と演算を必要とする自然言語処理タスクでの性能の向上を目指す．本研究の目的は，計算機の原理であるチューリングマシンをニューラルネットワークで模した DNC に，Transformer と知識利用・演算処理を行うアーキテクチャを組み込んだモデルを構築し，知識と演算を必要とする自然言語処理タスクを扱えるか検証を行うことである.

提案モデルを構築する前に，DNC を日本語対話に適用する予備研究を行った．文脈を考慮した対話の研究は，言語として英語を扱ったものが大半であり，日本語を対象としたものは少ない．また，日本語による対話のデータセットはいくつか存在するが，明示的に文脈理解を問うようなデータセットではない．そこで，本研究では既存の英語のデータセットである bAbI データセット [Weston 15][1]を参考に，喫茶店での注文の場面を想定した日本語注文対話データセットを作成し，課題の解決を試みる．作成したデータセットに対し Differentiable Neural Computer (DNC) [Graves 16] を適用することで，文脈を用いた日本語注文対話システムを構築する.

文脈理解と知識活用は対話にとって重要であり，対話において知識を明示的に扱った研究として [Dinan 18] や [Young 17] が，また文脈と知識の両方を扱った研究として [Guo 18] などが挙げられるが，これらの課題を扱った研究は未だあまり進められていない．また，対話のアーキテクチャとして Seq2Seq [Sutskever 14] や，T5 [Raffel 19], BART [Lewis 19] といった sequence-to-sequence なモデル（相手の発話を入力として応答となる発話を生成するモデル）が主流になっている．一方，より自然で知的な対話を行うには文脈理解や知識活用が必要であるが，そのための長期間におけるデータ保存の能力には限界があると議論されてきた．そこで，文脈情報などの長期の情報を保持するために，Differentiable Neural Computer (DNC) [Graves 16] などの記憶装置付きニューラルネットワークモデルが提案されている．これらのモデルは記憶装置を付け

[1] http://www.thespermwhale.com/jaseweston/babi/tasks_1-20_v1-2.tar.gz

加えたことにより従来のモデルに比べてより複雑な情報処理を行えるようになり，文脈を踏まえた対話においても高い精度を実現している．

ここで，自然言語処理において知識を取り入れる際，形式的に記述された構造化知識は正しい結果を導くが，形式を重んじることから柔軟に利用することができないという問題がある．一方で，分散表現による知識表現では未知の知識項目に対しても柔軟に対応することができる．そこで，本研究では構造化知識を分散表現にして用いることで，正確に記述された知識をニューラルネットワークにおいて柔軟に活用する．知識を導入することの利点として，文脈情報からだけでは答えることができない知識を問う質問に対する応答，曖昧な質問の内容に対する正確な特定，そして未知の語彙に対する柔軟な対応が可能になることが挙げられる．

本研究の 2 つ目では，Differentiable Neural Computer (DNC) [Graves 16] を基盤とした，文脈を捉えつつ，分散表現による構造化知識を用いた手法を提案する．

知識グラフを埋め込み空間内に表現し，論理計算を行う質問応答手法として Conditional Theorem Provers [Minervini 20] や Query2Box [Ren 20] が提案されている．しかし，これらの手法では限られた計算のみしか扱うことができない．本研究の 3 つ目では，Differentiable Neural Computer (DNC) [Graves 16] と，さらに DNC を改良した rsDNC [Franke 18] と DNC-DMS [Csordás 19] に対して，質問応答タスクにおいて重要な要素である知識利用と演算処理を新たに組み入れ，構造化知識に対する演算処理を含んだ質問文について正しい答えを生成するために任意の演算を用いて能力を向上させることを目指す．

1.2　論文の構成

本論文の構成は次の通りである．第 1 章では，研究背景と目的を述べた．第 2 章では，記憶装置付きニューラルネットワークモデルを説明する．第 3 章では，自然言語文の SPARQL クエリ変換によるメニューオントロジーへのアクセス手法について述べる．第 4 章では，記憶装置付きニューラルネットワークモデルの文脈を考慮した日本語注文対話への適用の研究について述べる．続いて第 5 章では，記憶装置付きニューラルネットワークモデルによる文脈と構造化知識を用いた対話について述べる．第 6 章では，記憶装置付きニューラルネットワークモデルによる構造化知識と演算処理を用いた質問応答について述べる．最後に第 7 章で結論を記す．

第2章　記憶装置付きニューラルネットワークモデル

2.1　ノイマン型コンピュータと Differentiable Neural Computer

近年，ディープニューラルネットワークはコンピュータビジョンや自然言語処理といったさまざまなタスクの複雑なパターンマッチングにおいて顕著な発展を遂げてきた. しかし，グラフや木などのデータ構造の表現や変数の使用，長い系列に対する表現の操作といった能力には限界があるとされてきた.

Recurrent Neural Network (RNN) は系列データの長距離依存を捉えることができ，チューリング完全であることも知られている [Siegelmann 95] ため，抽象的な処理を行う能力があるとされるが,「理論的には」の話であり，現実には勾配消失問題 [Bengio 94] に苦しんだ. Long Short-Term Memory (LSTM) はこの問題について， RNN アーキテクチャにゲート機構を導入し，毎タイムステップのゲート値との要素ごとの積によって勾配を計算することで解決した [Hochreiter 97]. LSTM は大きく成功し，sequence-to-sequence model [Sutskever 14] と attention mechanism [Bahdanau 14, Luong 15] のおかげで RNN を上回るのに貢献した. Transformer [Vaswani 17] は，ほとんどすべて attention mechanism を基にしており，LSTM や伝統的なモデルをいろいろなタスクで打ち破ってきた. しかし，Transformer には長期的文脈情報を固定長の系列にエンコードするせいで context fragmentation problem [Dai 19] がある. この問題を解くために，過去の情報をメモリ内にキャッシュしておくさまざまな手法 [Dai 19, Rae 19, Martins 22] が提案されてきた. けれど未だ Transformer をベースとしたモデルは，先にディープニューラルネットワークの限界として挙げた課題への真の解法には至っていない.

一方，ノイマン型コンピュータでは，プログラムは単純な算術演算と論理演算を行う処理装置，制御命令文（順次実行，条件分岐，反復）を扱う制御装置，そして計算中にデータや命令文が読み書きされるメモリという 3 つの基礎的な仕組みによって実行される [Neumann 45]．ノイマン型コンピュータのアーキテクチャを図 2.1 に示す. このアーキテクチャは複雑なデータ構造を表現でき，アルゴリズムタスクを解くことが

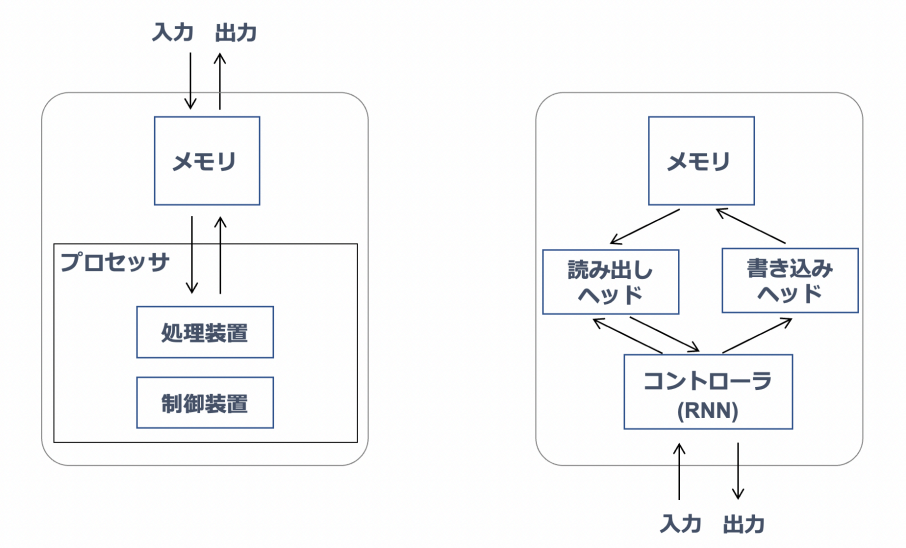


図 2.1: ノイマン型コンピュータ（左）と Neural Turing Machine（右）のアーキテクチャ

できるように学習する. ノイマン型コンピュータはプロセッサ（すなわち処理装置と制御装置）による計算とメモリを分けているが，ニューラルネットワークは計算とメモリをネットワークの重みとして一緒にしている. 計算とメモリを明示的に分けたニューラルネットワークとして提案されたのが Neural Turing Machine (NTM) [Graves 14] である．NTM はテープの長さが無限であることによる有限状態チューリングマシンから類推された，読み書き可能な外部メモリを持つ．NTM のアーキテクチャを図 2.1 に示す. NTM 全体は微分可能なため，メモリへのアクセスのやり方を含めて end-to-end に学習できる. さらに，メモリアクセス機構を改良した Differentiable Neural Computer (DNC) [Graves 16] が提案され，路線図の最短路探索タスクや家系図の推測タスクなどの構造化データ上のアルゴリズムタスクを解いた. 前提付きの質問応答における実験では，入力系列はメモリに書き込まれ，答えを推測するのに必要な情報がメモリから読み出されることから，変数を表現している. DNC は動的メモリアクセス機構により長い系列を学習することも可能にした.

2.2　Differentiable Neural Computer

最初に，DNC [Graves 16] の全体図を図 2.2 に示し，簡単に概要を説明する.

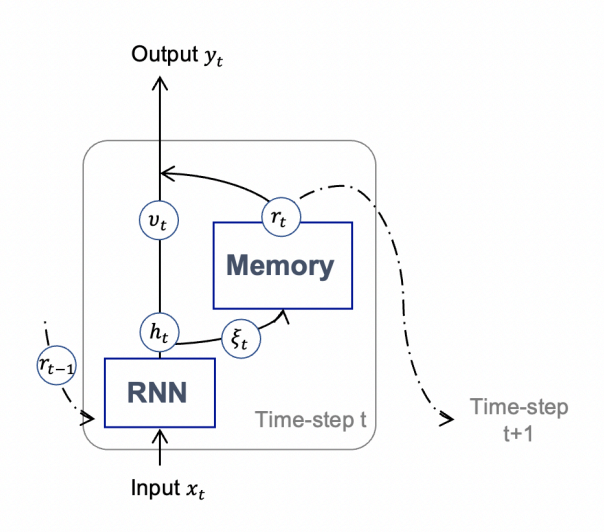


図 2.2: Differentiable Neural Computer の全体図

各タイムステップ t で行う処理は以下の通りである：

1. コントローラ (RNN) は，毎時刻につき，入力 $\boldsymbol{x}_t$ と前時刻でメモリから読み出した情報 $\boldsymbol{r}_{t-1}$ を合わせて受け取り，$\boldsymbol{h}_t$ を出力する．
2. $\boldsymbol{h}_t$ の線形変換により，出力 $\boldsymbol{v}_t = W_y \boldsymbol{h}_t$ と，メモリを制御するためのパラメータを格納したベクトル $\boldsymbol{\xi}_t = W_\xi \boldsymbol{h}_t$ に分ける．
3. $\boldsymbol{\xi}_t$ によってメモリへの書き込みが行われ，メモリの状態が更新される．
4. 最後にメモリからの読み出しが行われ，読み出された情報 $\boldsymbol{r}_t$ は RNN の出力 $\boldsymbol{v}_t$ に $\boldsymbol{y}_t = \boldsymbol{v}_t + W_r \boldsymbol{r}_t$ のように足されて出力 $\boldsymbol{y}_t$ が計算されるのと同時に，次時刻の RNN への入力に回される．

以上の処理を繰り返すことにより，メモリへの読み書き操作を取り入れたニューラルネットワークが実現される．

2.2.1　メモリの構造

メモリは N×W の行列である．N はメモリスロットの総数（アドレスの総数），W はスロットの長さ（格納する数値ベクトルの次数）であり，どちらも固定とする．メモリの状態は刻々と更新されていき，タイムステップ t でのメモリを表す行列を M_t と表記する．

2.2.2　コントローラ

各タイムステップ t につき，コントローラはデータセットから入力 $\boldsymbol{x}_t \in \mathbb{R}^X$ と前タイムステップのメモリ $M_{t-1} \in \mathbb{R}^{N\times W}$ から読み出された R 個のベクトル $\boldsymbol{r}^1_{t-1}, ..., \boldsymbol{r}^R_{t-1}$ を受け取り，$\boldsymbol{y}_t \in \mathbb{R}^Y$ を出力する．そして現タイムステップのメモリの制御方法を定義した interface vector$\boldsymbol{\xi}_t$ を出力する．表記を簡略化するために，入力と読み出しベクトルを結合したものを $\boldsymbol{\chi}_t = [\boldsymbol{x}_t; \boldsymbol{r}^1_{t-1}; ...; \boldsymbol{r}^R_{t-1}]$ とする．LSTM の変数は以下の通りである．

$$
\begin{aligned}
\boldsymbol{i}^l_t &= \sigma(W^l_i[\boldsymbol{\chi}_t; \boldsymbol{h}^l_{t-1}; \boldsymbol{h}^{l-1}_t] + \boldsymbol{b}^l_i) \\
\boldsymbol{f}^l_t &= \sigma(W^l_f[\boldsymbol{\chi}_t; \boldsymbol{h}^l_{t-1}; \boldsymbol{h}^{l-1}_t] + \boldsymbol{b}^l_f) \\
\boldsymbol{s}^l_t &= \boldsymbol{f}^l_t \boldsymbol{s}^l_{t-1} + \boldsymbol{i}^l_t tanh(W^l_s[\boldsymbol{\chi}_t; \boldsymbol{h}^l_{t-1}; \boldsymbol{h}^{l-1}_t] + \boldsymbol{b}^l_s) \\
\boldsymbol{o}^l_t &= \sigma(W^l_o[\boldsymbol{\chi}_t; \boldsymbol{h}^l_{t-1}; \boldsymbol{h}^{l-1}_t] + \boldsymbol{b}^l_o) \\
\boldsymbol{h}^l_t &= \boldsymbol{o}^l_t tanh(\boldsymbol{s}^l_t)
\end{aligned}
$$

l はレイヤのインデックス，$\sigma(x) = 1/(1 + exp(-x))$ はロジスティックシグモイド関数，$\boldsymbol{h}^l_t, \boldsymbol{i}^l_t, \boldsymbol{f}^l_t, \boldsymbol{s}^l_t, \boldsymbol{o}^l_t$ はそれぞれ時刻 t におけるレイヤ l の隠れベクトル，入力ゲートベクトル，忘却ゲートベクトル，状態ベクトル，そして出力ゲートベクトルである．すべての t に対して $\boldsymbol{h}^0_t = \boldsymbol{0}$，すべての l について $\boldsymbol{h}^l_0 = \boldsymbol{s}^l_0 = \boldsymbol{0}$ である．W は重み行列，$\boldsymbol{b}$ はバイアスを表す.

各タイムステップにおいて，コントローラは以下のように定義される output vector $\boldsymbol{v}_t$ と interface vector $\boldsymbol{\xi}_t \in \mathbb{R}^{(W\times R)+3W+5R+3}$ を出力する．

$$
\begin{aligned}
\boldsymbol{v}_t &= W_y[\boldsymbol{h}^1_t; ...; \boldsymbol{h}^L_t] \\
\boldsymbol{\xi}_t &= W_\xi[\boldsymbol{h}^1_t; ...; \boldsymbol{h}^L_t]
\end{aligned}
$$

コントローラが再帰的だとすると，その出力は現タイムステップまでの入力の完全な履歴 $(\boldsymbol{\chi}_1, ..., \boldsymbol{\chi}_t)$ の関数である．したがって，コントローラが行う操作は以下のようにまとめられる．

$$(\boldsymbol{v}_t, \boldsymbol{\xi}_t) = \mathcal{N}([\boldsymbol{\chi}_1; ...; \boldsymbol{\chi}_t]; \boldsymbol{\theta})$$

ここで，$\boldsymbol{\theta}$ はネットワークの重みである．最後に，read vector を結合したものと $RW\times Y$ の重み行列 W_r を掛け合わせたベクトルに $\boldsymbol{v}_t$ を足し，output vector $\boldsymbol{y}_t$ を得る．

$$\boldsymbol{y}_t = \boldsymbol{v}_t + W_r[\boldsymbol{r}^1_t; ...; \boldsymbol{r}^R_t]$$

このようにすることで直前に読み出された情報を用いて出力を調整することができる．

2.2.3　Interface パラメータ

メモリに対して制御を行う前に，interface vector $\boldsymbol{\xi}_t$ は以下のように細かく分割される．

$$\boldsymbol{\xi}_t = [\boldsymbol{j}_t^{r,1}; ...; \hat{\beta}_t^{r,R}; \boldsymbol{k}_t^W; \hat{\beta}_t^W; \hat{\boldsymbol{e}}_t; \boldsymbol{v}_t; \hat{f}_t^1; ...; \hat{f}_t^R; \hat{g}_t^a; \hat{g}_t^W; \hat{\boldsymbol{\pi}}_t^1; ...; \hat{\boldsymbol{\pi}}_t^R]$$

個々の要素は正しい定義域内に含まれるよう様々な関数により処理される．ロジスティックシグモイド関数は $[0, 1]$，oneplus 関数は $[1, \infty)$ に制約をかけるために用いられる．

$$oneplus(x) = 1 + log(1 + e^x)$$

Softmax 関数はベクトルを $\mathcal{S}_N$ に制限するのに用いられる．

$$\mathcal{S}_N = \left\{ \boldsymbol{\alpha} \in \mathbb{R}^N : \boldsymbol{\alpha}_i \in [0, 1], \sum_{i=1}^{N} \boldsymbol{\alpha}_i = 1 \right\}$$

また，ˆ記号はさらにスケール変換をかけることを意味する．1 ステップでメモリからの読み出しは複数回，書き込みは 1 回のみ行われる．R はメモリからの読み出し回数である．処理を行った後，以下のようなスカラ及びベクトルの集合を得る．

- R 個の read key$\{\boldsymbol{k}_t^{r,i} \in \mathbb{R}^W; 1 \leq i \leq R\}$;
- R 個の read strength$\{\beta_t^{r,i} = oneplus(\hat{\beta}_t^{r,i}) \in [1, \infty); 1 \leq i \leq R\}$;
- write key$\boldsymbol{k}_t^W \in \mathbb{R}^W$;
- write strength$\beta_t^W = oneplus(\hat{\beta}_t^W) \in [1, \infty)$;
- erase vector$\boldsymbol{e}_t = \sigma(\hat{\boldsymbol{e}}_t) \in [0, 1]^W$;
- write vector$\boldsymbol{v}_t \in \mathbb{R}^W$;
- R 個の free gate$\{f_t^i = \sigma(\hat{f}_t^i) \in [0, 1]; 1 \leq i \leq R\}$;
- allocation gate$g_t^a = \sigma(\hat{g}_t^a) \in [0, 1]$;
- write gate$g_t^w = \sigma(\hat{g}_t^w) \in [0, 1]$;

- R 個の read mode$\{\boldsymbol{\pi}_t^i = softmax(\hat{\boldsymbol{\pi}}_t^i) \in \mathcal{S}_3; 1 \leq i \leq R\}$;

これらの用途について次に説明する．

2.2.4　メモリの読み書き

メモリの読み書きを行うには，読み書き対象となるメモリスロットのアドレスを指定する必要がある．この操作を微分可能な計算で表現するには，どのアドレスを重点的に読み出す/書き込むかの重み付けを行えば良い．これらの重み，read/write weighting は要素の総和がほぼ 1 である非負の数のベクトルである．

$$\Delta_N = \left\{ \boldsymbol{\alpha} \in \mathbb{R}^N : \boldsymbol{\alpha}_i \in [0,1], \sum_{i=1}^{N} \boldsymbol{\alpha}_i \leq 1 \right\}$$

read/write weighting をどのように求めるかが重要であり，これらを求めてしまえば読み出し/書き込みの演算自体は単純である．読み出し操作では，R 個の read weighting $\{\boldsymbol{w}_t^{r,1}, ..., \boldsymbol{w}_t^{r,R} \in \Delta_N\}$ によってアドレスの内容の重み付き和を計算し，read vector $\{\boldsymbol{r}_t^1, ..., \boldsymbol{r}_t^R\}$ を以下のように定義する．

$$\boldsymbol{r}_t^i = M_t^\top \boldsymbol{w}_t^{r,i}$$

read vector は次時刻のコントローラへの入力に追加される．書き込み操作では，write weighting $\boldsymbol{w}_t^W \in \Delta_N$ により，write vector $\boldsymbol{v}_t \in \mathbb{R}^W$（書き込みたいデータのベクトル）と erase vector $\boldsymbol{e}_t \in [0,1]^W$（スロット内のデータをどのようなパターンで消去するかを表したベクトル）を接続し，以下のようにメモリ行列 M_{t-1} に対して消去・書き込みを行い更新する．

$$M_t = M_{t-1} \circ (E - \boldsymbol{w}_t^W \boldsymbol{e}_t^\top) + \boldsymbol{w}_t^W \boldsymbol{v}_t^\top$$

$\circ$ は要素ごとの積，E はすべて 1 の N×W 行列を表す．

次に read/write weighting の計算について詳細に述べる．

2.2.5　メモリの番地付け

write weighting の更新

書き込み先のアドレスの選択には 2 つの方法があり，write weighting は 2 つの要素から構成される：

1. Content-based addressing: interface vector を通して入力された key をもとに書き込み先スロットを選択する．
2. Dynamic memory allocation: 前タイムステップまでの読み出し状況をもとに使用済みの情報が残っているスロットを選択する．

Content-based addressing

write key$\boldsymbol{k}_t^W \in \mathbb{R}^W$ とメモリ $M \in \mathbb{R}^{N\times W}$ の各スロットに格納されているデータを照合し，類似度を計算する．

$$C(M, \boldsymbol{k}, \beta)[i] = \frac{exp\{\mathcal{D}(\boldsymbol{k}, M[i,\cdot])\beta\}}{\sum_j exp\{\mathcal{D}(\boldsymbol{k}, M[j,\cdot])\beta\}}$$

write strength $\beta_t^W \in [1, \infty)$ は weighting のピークの鋭さを調整するパラメータであり，$\beta \to \infty$ の極限で C は鋭いピークがひとつだけ立つ．$\mathcal{D}$ はコサイン類似度である．

$$\mathcal{D}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u}\cdot\boldsymbol{v}}{|\boldsymbol{u}||\boldsymbol{v}|}$$

weighting $C(M, \boldsymbol{k}, \beta) \in \mathcal{S}_N$ はメモリアドレスに対する正規化確率分布を定義する．この方法による write content weighting は以下のようになる．

$$\boldsymbol{c}_t^W = C(M_{t-1}, \boldsymbol{k}_t^W, \beta_t^W)$$

retention vector ψ_t の構成

コントローラが必要に応じてメモリを開放，確保できるように，使用可能なメモリスロットのリストを構築する．前タイムステップで読み出しを行ったメモリスロットを開放して良いかどうかのフラグ（実際には 0~1 の連続値なので重み）を free gate $f_t^i (i = 1, ..., R)$ とする．「$f_t^i \boldsymbol{w}_{t-1}^{r,i} \simeq 1 \Leftrightarrow f_t^i \simeq 1$ かつ $\boldsymbol{w}_{t-1}^{r,i} \simeq 1$」ならば読み込み済みか

つ開放可なので上書き可である.「$f_t^i \boldsymbol{w}_{t-1}^{r,i} \simeq 0 \Leftrightarrow f_t^i \simeq 0$ または $\boldsymbol{w}_{t-1}^{r,i} \simeq 0$」ならば前タイムステップで読み込まれていない，または，読み込み済みだったとしても開放フラグが立っていないので上書き不可となる．R 回分の使用中（上書き不可）フラグである memory retention vector $\boldsymbol{\psi}_t \in [0,1]^N$ は free gate を用いて以下のように定義される．

$$\boldsymbol{\psi}_t = \prod_{i=1}^{R} (\mathbf{1} - f_t^i \boldsymbol{w}_{t-1}^{r,i})$$

R 回のうち 1 度でも上書き可になったメモリスロットは上書き可 $\boldsymbol{\psi}_t \simeq 0$ となる．一方，使用中（上書き不可）$\boldsymbol{\psi}_t \simeq 1$ となるのは，R 回全てについて上書き不可だったときである．

usage vector $\boldsymbol{u}_t$ の構成

各メモリスロットの使用中度合いを表す重み memory usage vector $\boldsymbol{u}_t \in [0,1]^N$ を以下の更新式で定義する．

$$\boldsymbol{u}_t = (\boldsymbol{u}_{t-1} + \boldsymbol{w}_{t-1}^W - \boldsymbol{u}_{t-1} \circ \boldsymbol{w}_{t-1}^W) \circ \boldsymbol{\psi}_t$$

$\circ$ は要素ごとの積である．$(\cdot)$ 内第 3 項は $\boldsymbol{u}_t$ の各要素が 1 を超えないように調整するための補正項である．$\boldsymbol{u}_t = 1$ に達すると書き込みによる重みの更新は停止し，もし $\boldsymbol{u}_t > 1$ となっても更新により値を減少させる．直感的には，free gate により保持されている ($\boldsymbol{\psi}_t[i] \approx 1$)，すでに使用中若しくは書き込まれたばかりのどちらかだった場合，スロットは使用中となる．スロットへのあらゆる書き込みはその使用度を上げ（最大で 1 になり），free gate が用いられたときのみ使用度は下がる（最小で 0 となる）．したがって，$\boldsymbol{u}_t$ の要素の範囲は $[0,1]$ に制約される．

allocation weighting の構成

$\boldsymbol{u}_t$ の要素を値が小さい順に並べた時のメモリアドレスのインデックスのベクトルを $\boldsymbol{\phi}_t \in \mathbb{Z}^N$ とする．$\boldsymbol{\phi}_t[1]$ は最も使用度が低いアドレスのインデックスとなる．ここで，Δ_N に含まれる重みベクトルは総和が 1 以下であり，つまり重みがないということがどのアドレスにもアクセスしない null 操作を明示的に担う．書き込みを行うための新

しいアドレスを与える allocation weighting $\boldsymbol{a}_t \in \Delta_N$ を以下で定義する．

$$\boldsymbol{a}_t[\phi_t[j]] = (1 - \boldsymbol{u}_t[\phi_t[j]]) \prod_{i=1}^{j-1} \boldsymbol{u}_t[\phi_t[i]]$$

$1 - \boldsymbol{u}_t$ なので使用済み（書き込み可）の度合いを表す重みである．すべての使用度が 1 であると，$\boldsymbol{a}_t = \boldsymbol{0}$ となり，コントローラはスロットの開放が行われない限りメモリを確保することができない．

Write weighting

コントローラは新しく確保されたスロット，または内容によって選ばれたスロットに書き込み，あるいは書き込みはまったく行わないということもできる．allocation weighting $\boldsymbol{a}_t$ と write content weighting $\boldsymbol{c}_t^W \in \mathcal{S}_N$ を組み合わせて write weighting $\boldsymbol{w}_t^W \in \Delta_N$ は以下のように定義される．

$$\boldsymbol{w}_t^W = g_t^w \left[g_t^a \boldsymbol{a}_t + (1 - g_t^a) \boldsymbol{c}_t^W \right] \tag{2.1}$$

allocation gate $g_t^a \in [0, 1]$ は使用済みフラグあるいは key のどちらに基づいて書き込みを行うかを制御するゲートである．write gate $g_t^w \in [0, 1]$ はそもそも書き込みを行うかを制御し，メモリを不必要な上書きから保護するのに用いられる．

read weighting の更新

読み出し先のアドレスの選択にも 2 つの方法がある：

1. Content-based addressing: interface vector を通して入力された key をもとに読み出し先スロットを選択する．
2. Temporal memory linkage: 書き込み順に従ってスロットを選択する．

Content-based addressing

write weighting のときと同様に，各読み出しヘッド i に対して read key$\{\boldsymbol{k}_t^{r,i}; 1 \le i \le R\}$ と read strength$\{\beta_t^{r,i}; 1 \le i \le R\}$ を用いて類似度が計算される．この方法による read

content weighting は以下である．

$$\boldsymbol{c}_t^{r,i} = C(M_t, \boldsymbol{k}_t^{r,i}, \beta_t^{r,i})$$

precedence weighting の構成

前述のメモリを確保する方法では，メモリスロットに対して書き込みが行われた順番に関する情報は一切保存されていない．しかし，このような情報を保持しておくことが有用な場合も多くある．例えば，命令のシーケンスが順番通りに記録され，引き出されなければならないときなどが挙げられる．したがって，メモリスロットが連続して上書きされていくのを追跡するために temporal link matrix $L_t \in [0,1]^{N\times N}$ を用いる．L_t を定義するために，成分 $\boldsymbol{p}_t[i]$ がスロット i が最後に書き込まれたという程度を表す precedence weighting $\boldsymbol{p}_t \in \Delta_N$ を以下の更新式で定義する．

$$\begin{aligned}\boldsymbol{p}_0 &= \mathbf{0} \\ \boldsymbol{p}_t &= \left(1 - \sum_i \boldsymbol{w}_t^W[i]\right)\boldsymbol{p}_{t-1} + \boldsymbol{w}_t^W\end{aligned}$$

$\boldsymbol{w}_t^W$ は式 (3.1) で定義した write weighting である．「強く」書き込みが行われた場合は，$\sum \boldsymbol{w}_t^W \simeq 1$ となるので，前タイムステップの情報 $\boldsymbol{p}_{t-1}$ は消去され，現タイムステップでどこに書き込まれたかが保存される．書き込みが行われなかった場合は，最後に行われた書き込みの重みが保持され続ける．

temporal link matrix の構成

メモリスロットへの書き込み順を表す L_t を構成する．$L_t[i, j]$ はメモリ M_t においてスロット i はスロット j の後に書き込まれたという程度を表す．スロットが上書きされる度に，リンク行列はそのスロットに出入りする古いリンクを消去するよう更新される．そして最後に書き込みが行われたスロットからの新しいリンクが追加される．L_t

の更新式を以下で与える．

$$
\begin{aligned}
L_0[i,j] &= 0 \quad \forall i,j \\
L_t[i,i] &= 0 \quad \forall i \\
L_t[i,j] &= (1 - \boldsymbol{w}_t^W[i] - \boldsymbol{w}_t^W[j])L_{t-1}[i,j] + \boldsymbol{w}_t^W[i]\boldsymbol{p}_{t-1}[j]
\end{aligned}
$$

i は現タイムステップ，j は前タイムステップである．あるスロットからそれ自身への遷移をどのように表現するかは明確ではないため，自分自身へのリンクは除外される（行列の対角成分は常に 0 とする）．L_t の行と列はそれぞれ特定のメモリスロットへの，及び特定のメモリスロットからの一時的なリンクの重みを表している．

forward/backward weighting の構成

メモリへの書き込み順を考慮した forward/backward weighting $\{\boldsymbol{f}_t^i \in \Delta_N; 1 \leq i \leq R / \boldsymbol{b}_t^i \in \Delta_N; 1 \leq i \leq R\}$ を以下で構成する．

$$
\begin{aligned}
\boldsymbol{f}_t^i &= L_t \boldsymbol{w}_{t-1}^{r,i} \\
\boldsymbol{b}_t^i &= L_t^\top \boldsymbol{w}_{t-1}^{r,i}
\end{aligned}
$$

$\boldsymbol{w}_{t-1}^{r,i}$ は前タイムステップからの i 番目の read weighting である．

Read weighting

read mode vector $\boldsymbol{\pi}_t^i \in \mathcal{S}_3$ により，backward weighting $\boldsymbol{b}_t^i$ と forward weighting $\boldsymbol{f}_t^i$，そして read content weighting $\boldsymbol{c}_t^{r,i}$ を合わせて，read weighting $\boldsymbol{w}_t^{r,i} \in \mathcal{S}_3$ を以下のように定義する．

$$
\boldsymbol{w}_t^{r,i} = \boldsymbol{\pi}_t^i[1]\boldsymbol{b}_t^i + \boldsymbol{\pi}_t^i[2]\boldsymbol{c}_t^{r,i} + \boldsymbol{\pi}_t^i[3]\boldsymbol{f}_t^i
$$

読み出しモードにおいて $\boldsymbol{\pi}_t^i[2]$ が強ければ，$\boldsymbol{k}_t^{r,i}$ を用いた内容による検索に戻る．$\boldsymbol{\pi}_t^i[3]$ であれば，read key は無視し，読み出しヘッドは書き込まれた順にメモリスロットを指していく．$\boldsymbol{\pi}_t^i[1]$ であれば，読み出しヘッドは逆順で動いていく．

2.3　関連研究

Memory Networks [Weston 14] は，巨大メモリとメモリへの入出力操作を行う学習構成要素からなるメモリ付きニューラルネットワークである．メモリに文脈情報などを格納しておき，そこから質問に関する情報を連鎖的に読み出し利用することにより，文脈を考慮した質問応答を可能にしている．Memory Networks では，メモリに対して hard attention を用いており，それは微分不可能なので，学習データをアノテーションし，質問に関する情報の選択について別途学習させる必要がある．End-To-End Memory Networks [Sukhbaatar 15] は，Memory Networks の質問に関する情報の選択において，微分可能な Soft Attention を用いたモデルである．それを用いることにより，パラメータを end-to-end に学習することができ，学習データをアノテーションする必要がなくなるため，より多様なデータを扱うことができる．Dynamic Memory Networks [Kumar 15] は，End-To-End Memory Networks に対して入力文や質問文，メモリ内部のベクトルのエンコードに GRU を用いたモデルである．

第3章　自然言語文のSPARQLクエリ変換によるメニューオントロジーへのアクセス

3.1　研究背景と目的

近年，様々な分野の膨大なデータベースを意味的に結びつけたデータ群が大規模知識 (Linked Open Data) として公開されている．自然言語処理の中で背景知識として用いることで，表層的な情報による推論だけでは難しい，常識などを含んだ自然言語理解に役立つことができる．自然言語表現と Linked Open Data 形式で記述された知識ベースを結びつけるために，自然言語文を SPARQL クエリに変換することで知識への問い合わせを行う様々な研究が行われてきた．[Wang 07] では，自然言語文の構文解析結果に対して，知識を記述するための語彙や，WordNet [Miller 95] などの一般的な辞書からなる語彙，ユーザが定義する語彙などを適用することで RDF トリプルに変換する手法を提案している．[Zou 14] では，複数の RDF トリプルをつなぎ Semantic Query Graph なる知識グラフを構築し，断片的な知識を複数のトリプルからなる知識に変換することで自然言語文の解釈の精度を高めている．また，[鈴木 15] では，自然言語文を構文解析した結果を論理式に変換し，そこから SPARQL クエリを生成する手法を提案している．

しかし，これらにおいて自然言語文から生成したクエリが正しい答えを返すかは保証されていない．そこで本研究では，先行研究のアプローチを踏まえて，自然言語文から抽出したエンティティを知識側の語彙と対応付け，エンティティ間のリレーションを知識を繰り返し参照することで獲得することに重点を置いた SPARQL クエリの自動生成手法を提案する．提案手法により SPARQL クエリが完成されれば，答えを返すことが保証される．

3.2　関連技術

3.2.1　セマンティック Web

セマンティック Web は，現在のドキュメント中心の Web からデータ中心の Web を目指す構想であり，意味的なつながりを含めて Web データを構造的に表現することにより機械処理を可能にし，膨大な知識を獲得することでコンピュータの知能を大幅に向上させることが期待されている．Web の発明者ティム・バーナーズ＝リーにより提唱され，今日までにリンクトデータや RDF，SPARQL などの多くの技術が開発されてきた．

リンクトデータ

リンクトデータはセマンティック Web において最も重要な要素である．従来の Web ではドキュメント間をハイパーリンクによりつないでいたのに対し，リンクトデータではもの，抽象物といったすべてのリソースについてその関係性が意味的にリンクされる．

ティム・バーナーズ＝リーによるリンクトデータに関する基本原則[1]は次のように言及されている：

1. 事物の名前に URI を用いる．
2. HTTP プロトコルで URI へアクセスして参照できる．
3. URI へアクセスしたとき，RDF や SPARQL などの標準技術を使って有益な情報を得ることができる．
4. 他の URI へのリンクを含むことで，さらに他の事物を発見できる．

Linked Open Data (LOD，大規模知識) は，上述したリンクトデータ基本原則に基づき Web 上に公開されているオープンライセンスのリンクトデータが拡張され，さまざまな分野の機械可読なデータが互いに外部リンクでつながれることにより，意味的に結合された膨大な知識群として実現されたものである．

[1] https://www.w3.org/DesignIssues/LinkedData.html

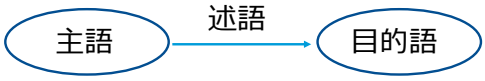


図 3.1: RDF トリプルの基本要素

RDF

RDF (Resource Description Framework) はリソースに関する属性や関係を表現する枠組みである．RDF グラフは，リソース間の意味的なリンク関係を記述するネットワークを構築し，主語，述語，目的語の三つ組（RDF トリプルと呼ぶ）による言明から構成される．主語は URI が表すリソース，目的語は URI が表すリソースまたはリテラルが表すデータ値，述語は URI が表す主語と目的語の関係としてそれぞれ記述される．RDF トリプルは図 3.1 のようにグラフ構造で図示され，主語と目的語はノードで，述語は二つのノード間のエッジで表される．

SPARQL

SPARQL (SPARQL Protocol And RDF Query Language) は，RDF グラフに問い合わせるための質問言語である．検索したい RDF データの条件をクエリで表現して問い合わせると，SPARQL エンジンはクエリ文により RDF データを検索し答えを出力する．SPARQL のクエリには四つの形式がある：変数リストのデータテーブルを出力する SELECT 文，RDF データを構成する CONSTRUCT 文，指定リソースの RDF データを出力する DESCRIBE 文，RDF データに対する条件判定を行う ASK 文．クエリ内の記述において，変数は “?x” のようにクエスチョンマークの後ろに文字列を付けて表現される．URI とリテラルは変数に代入される定数に相当する．

主語，述語，目的語に変数を含んだ RDF トリプルの拡張を RDF トリプルパターンという．SPARQL クエリは RDF トリプルパターンによって検索条件を記述する．クエリの実行では，トリプルパターン内の変数に検索条件を満たす定数を代入する．条件に一致するすべての組み合わせを検索し，つまり RDF グラフから部分グラフを検索する．例えば，`SELECT ?x WHERE {<http://ja.dbpedia.org/resource/日本> <http://ja.dbpedia.org/property/首都> ?x .}` というクエリを DBpedia Japanese[2]で検索すると`<http://ja.dbpedia.org/resource/東京都>`が答えとして返ってくる．

[2] `http://ja.dbpedia.org`

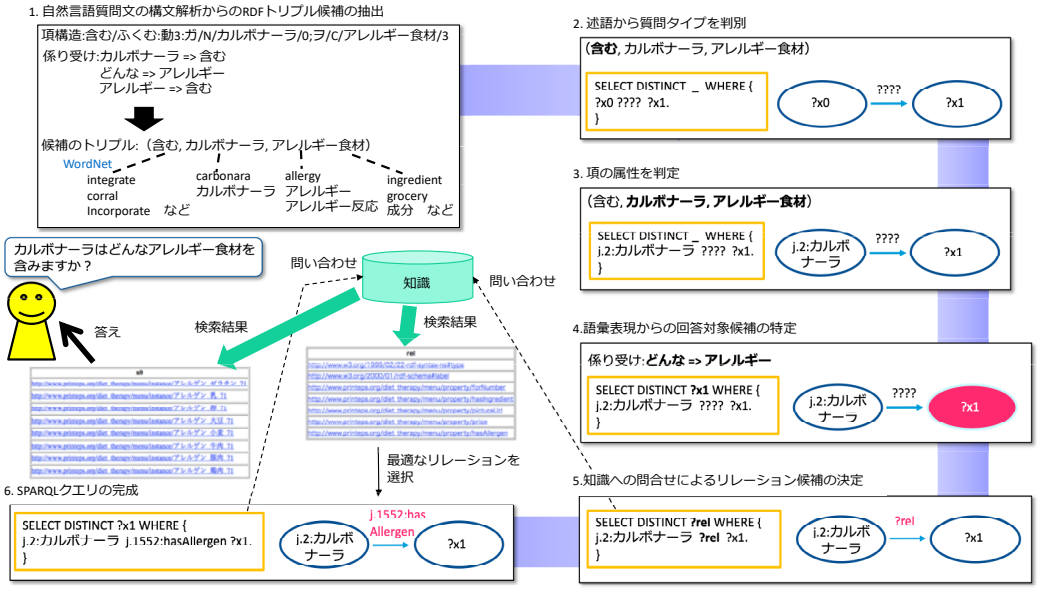


図 3.2: 自然言語文から SPARQL クエリへの変換手法の全体図

3.2.2　述語項構造

述語項構造とは，テキスト中に出現した述語とその項から構成される構造のことである．述語は，動詞述語（例「太郎はお酒を飲む.」），形容詞述語（例「太郎は背が高い.」），および名詞述語（例「太郎は学生だ.」）の 3 つを本研究では述語とする．項は，述語が表す動作や状態への参与者として不可欠な要素を指す．例えば,「昨日，太郎が車を運転した.」という文では，述語「運転した」に対して「太郎が」はガ格の項,「車を」はヲ格の項である．これを RDF トリプルで表すと，項はエンティティに，述語はリレーションに対応し,（太郎，運転，車）となる．

3.3　提案手法

喫茶店での注文の場面を想定し，ユーザの発話とメニューオントロジーを結びつけるために自然言語文を SPARQL クエリへ変換する手法を提案する．提案手法の全体図を図 3.2 に示す．

入力として自然言語質問文を受け取り，6 つの処理を経て SPARQL クエリを自動生成する手続きについて以下に示す．

表 3.1: 質問タイプの分類

型	述語	例文と候補のトリプル	SPARQLクエリのテンプレート	RDFグラフのテンプレート

基本型	含む
	使う
		カルボナーラはどんなアレルギー食材を含みますか？
		（含む, カルボナーラ, アレルギー食材）
		
		トマトを使ったパスタはありますか？
		（使う, パスタ, トマト）
			SELECT DISTINCT _ WHERE {
			?x0 ???? ?x1.
			}
				

Degree型	低い 高い	塩分の低い/高いパスタは何ですか？ （低い/高い, パスタ, 塩分）	SELECT DISTINCT _ WHERE { ?x0 ???? ?x1. ?x1 j.1552:amount ?amount. } ORDER BY ASC/DSEC(?amount) LIMIT 3	

Count型	----	ハンバーグは何種類ありますか？
		（有る, ハンバーグ, 何種類）
			SELECT DISTINCT (COUNT(?x0) as ?count)
			WHERE {
			?x0 rdf:type :クラス名.
			}
				

List型	----	どんなパスタがありますか？
		（有る, パスタ, ----）
		※三つ組なし
			SELECT DISTINCT ?x0 WHERE {
			?x0 rdf:type :クラス名.
			}	

1. **自然言語質問文の構文解析からのトリプル候補の抽出**

   京都大学黒橋河原研究室で開発された日本語構文・格・照応解析システム KNP[3] を用いて，自然言語質問文を解析し，語彙の依存関係および述語項構造を抽出する．述語項構造の解析結果の内，述語に対して項を 2 つ持つものから候補の RDF トリプル（以下，トリプルと呼ぶ）として（述語，項 1，項 2）を抽出する．トリプルがとる各単語について WordNet [Miller 95] を用いてその類義語を調べる．これは [Wang 07] の手法で用いているように，自然言語質問文中の表層表現や類義語がデータ中のトリプルで記述されている語彙になっている可能性があり，そのような知識に対して柔軟なアクセスができるように配慮するためである．

2. **述語による質問タイプの判別**

   候補のトリプルの述語に基づき，質問タイプを判別する．本研究で設定した述語の種類ごとの質問タイプの分類を表 3.1 に示す．

   「含む」,「使う」などの述語は基本型としてテンプレートで {`?x0 ???? ?x1.`} を与える.「低い」のような程度を問い合わせる述語は Degree 型とし，テンプレートで {`?x0 ???? ?x1. ?x1 j.1552:amount ?amount.`} を与える．本研究では知識のドメインを料理のメニューに限定しているため，程度に対する問い合わせは「量」への問い合わせであるとする．`ORDER BY ASC(?amount) LIMIT 3` により，量によって答えを昇順に並べ上位 3 つを返すように設定している．候補のトリプル中に「何種類」という単語が含まれる場合，Count 型とし，

[3] `http://nlp.ist.i.kyoto-u.ac.jp/index.php?KNP`

テンプレート {?x0 rdf:type ：クラス名.} と項目を数える COUNT 関数を与える.「どんなパスタがありますか？」の場合は，KNP を用いて三つ組の情報を得られないため，提案手法のステップ 2，3，5 を除いて SPARQL クエリを生成する．ある単語が「どんな」と「がありますか」の 2 つのキーワードに挟まれた場合，該当する項目をすべて回答する List 型に分類されるとする．

この段階では，項 1，項 2 はまだ未確定なのでそれぞれ?x0,?x1 としておく．例えば，述語が「含む」であれば基本型と判定されることから，テンプレートとして {?x0 ???? ?x1.} が与えられる．

3. **項の属性判定**
   候補のトリプルの項に注目して，その属性がクラス，インスタンス，リテラルのいずれであるかを判定する．クラスとインスタンスの判定は，オントロジーに沿って事前に作成した辞書を用いて行い，単語が辞書に存在しない場合はリテラルと判断する．クラスと判定された場合はトリプルパターン {?x rdf:type ：クラス名 } を追加する．インスタンスだった場合は ‘?x’ を ‘：インスタンス名’ に変換する．リテラルの場合は変数のままにしておく．

4. **語彙表現からの回答対象候補の特定**
   質問文中の語彙表現によって質問の答えになる回答対象候補を特定する．候補を特定する語彙表現として,「どんな」,「はありますか」,「は何ですか」を採用する．質問文中に「どんな」が現れた場合はその直後に来る単語を,「はありますか」と「は何ですか」の場合は直前にくる単語を回答対象候補とし，それに対応する変数を SELECT DISTINCT 後の変数リストに加える．

5. **知識への問合せによるリレーション候補の決定**
   上記の処理によって作成した SPARQL クエリの ????の部分を?rel に，SELECT DISTINCT 後の変数を一旦?rel にしてリレーションの候補を知識に問い合わせることにより決定する．

6. **SPARQL クエリの完成**
   リレーションの候補の検索結果において，各リレーション候補と，ステップ 1 で WordNet を使って調べておいたトリプルの各単語の類義語すべての文字列のバイグラムを取り（例えば，{ha,as,sA,Al,ll,le,er,rg,ge,en} と {al,ll,le,er,rg,gy}），Jaccard 係数によって類似度を求め，定めた閾値を

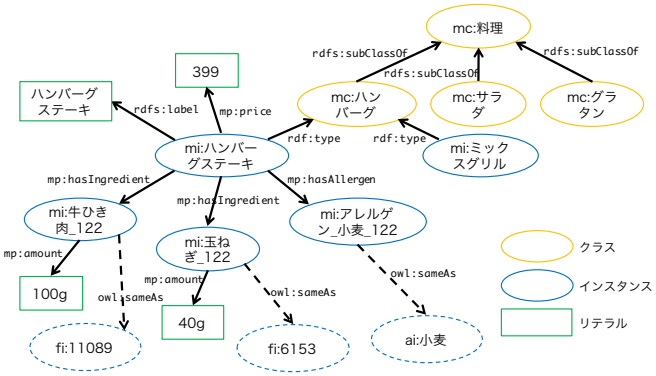


図 3.3: メニューオントロジーの一部

超えた候補を最適なリレーションと決定する．閾値以上の候補が複数存在する場合は，Zipf の法則に従いより希少な方を最適なリレーションとして選択する．以上の手続きにより入力された自然言語質問文の内容を反映した SPARQL クエリを完成させる．

3.4　実験

喫茶店におけるメニューに関して，設定した 4 つの質問タイプに基づく自然言語質問文を入力として実験を行なった．提案手法によって生成された SPARQL クエリを，クエリを通して知識から得られた回答の妥当性によって評価する．

3.4.1　実験設定

自然言語質問文から SPARQL クエリを生成する際に参照する知識として小規模のメニューオントロジーを構築した．図 3.3 にメニューオントロジーの一部を示す．SPARQL 環境として，Virtuoso (version 07.20.3214), SPARQL 1.1 を使用した．

3.4.2　結果と考察

表 3.1 の例文に挙げた「カルボナーラはどんなアレルギー食材を含みますか？」「トマトを使ったパスタはありますか？」「塩分の低いパスタは何ですか？」「ハンバーグは何種類ありますか？」「どんなハンバーグがありますか？」の 5 文に対して実験を行った．また,「塩分が低くて，トマトを使ったパスタはありますか？」のように Degree 型

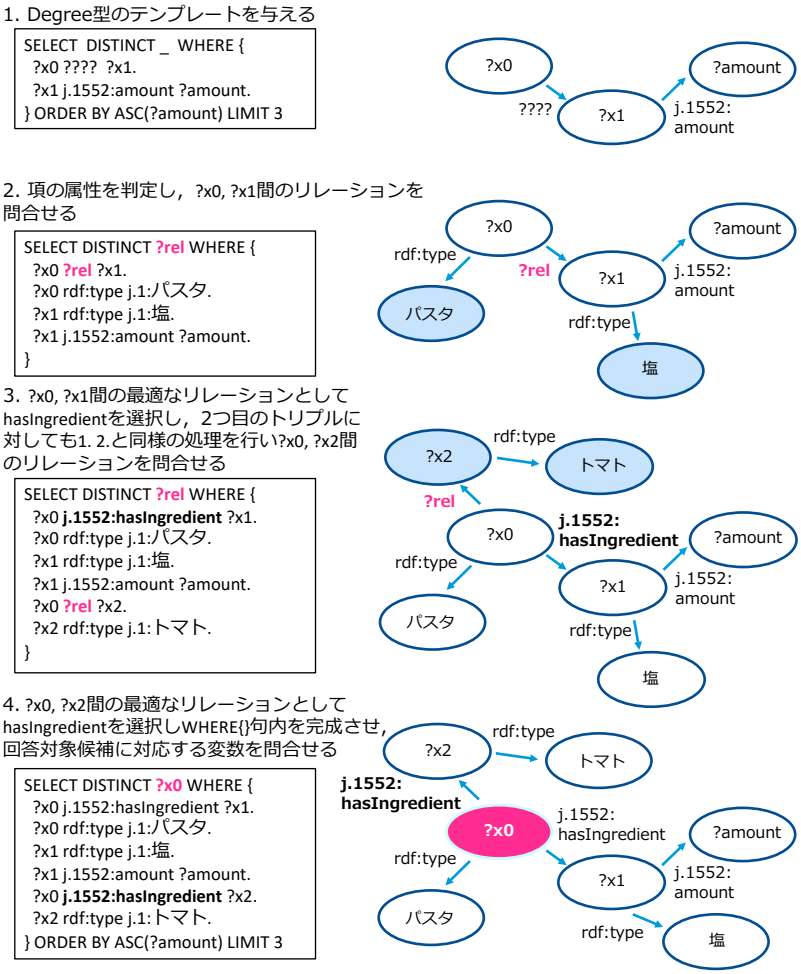


図 3.4: 例「塩分が低くて，トマトを使ったパスタはありますか？」の処理の流れ

と基本型を組み合わせた，述語を 2 つ含む文に対しても実験を行った．この具体的な処理の流れを図 3.4 に示す．

まず，KNP を用いて（低い，パスタ，塩分），（使う，パスタ，トマト）の 2 つの候補のトリプルを抽出する．1 つ目のトリプル（低い，パスタ，塩分）に対し，提案手法のステップ 2 により Degree 型のテンプレートを与える．ステップ 3 により項の属性を判定し，`?x0`，`?x1` 間のリレーションを知識に問い合わせ最適なものとして `hasIngredient` を選択する．次に，2 つ目のトリプル（使う，パスタ，トマト）に対しても同様の処理を行いクエリの `WHERE {}` 句内を完成させる．最後に，ステップ 4 により特定した回答対象候補に対応する変数を `SELECT DISTINCT` 後の変数リストに加え，求める SPARQL クエリを生成する．

表 3.2: 各質問タイプにおける知識抽出結果

型	自然言語質問文	生成された SPARQL クエリ	知識から得られた回答

基本型	カルボナーラはどんなアレルギー食材を含みますか？	SELECT DISTINCT ?x1 WHERE {
		j.2:カルボナーラ j.1552:hasAllergen ?x1.
		}
			アレルゲン_ゼラチン_71，アレルゲン_小麦_71
			アレルゲン_乳_71，アレルゲン_牛肉_71
			アレルゲン_卵_71，アレルゲン_豚肉_71
			アレルゲン_大豆_71，アレルゲン_鶏肉_71

	トマトを使ったパスタはありますか？	SELECT DISTINCT ?x0 WHERE {
		?x0 j.1552:hasIngredient ?x1.
		?x0 rdf:type j.1:パスタ.
		?x1 rdf:type j.1:トマト.
		}
			トマトクリームスパゲッティ
			ミートソースボロニア風
			半熟卵のミートソースボロニア風

Degree 型	塩分の低いパスタは何ですか？	SELECT DISTINCT ?x0 WHERE { ?x0 j.1552:hasIngredient ?x1. ?x0 rdf:type j.1:パスタ. ?x1 rdf:type j.1:塩. ?x1 j.1552:amount ?amount. } ORDER BY ASC(?amount) LIMIT 3	パルマ風スパゲッティ ミートソースボロニア風 半熟卵のミートソースボロニア風

Count 型	ハンバーグは何種類ありますか？	SELECT DISTINCT (COUNT(?x0) as ?count)
		WHERE {
		?x0 rdf:type j.1:ハンバーグ.
		}
			6

List 型	どんなハンバーグがありますか？	SELECT DISTINCT ?x0 WHERE {
		?x0 rdf:type j.1:ハンバーグ.
		}
			イタリアンハンバーグ
			ハンバーグステーキ
			ミックスグリル
			焼肉とハンバーグの盛合せ
			煮込みハンバーグ
			野菜ソースのハンバーグ

Degree 型 + 基本型	塩分が低くて，トマトを使ったパスタはありますか？	SELECT DISTINCT ?x0 WHERE { ?x0 j.1552:hasIngredient ?x1. ?x0 rdf:type j.1:パスタ. ?x1 rdf:type j.1:塩. ?x1 j.1552:amount ?amount. ?x0 j.1552:hasIngredient ?x2. ?x2 rdf:type j.1:トマト. } ORDER BY ASC(?amount) LIMIT 3	ミートソースボロニア風 半熟卵のミートソースボロニア風 トマトクリームスパゲッティ

表 3.2 に示すように，各質問タイプにおいてそれぞれ質問文に答えるのに有用な知識を得ることができ，したがって提案手法が適切な SPARQL クエリを生成したことが確かめられた．

3.5　まとめ

KNP による述語項構造・係り受け解析と，データ主導による RDF トリプルの変形を用いた，自然言語文から SPARQL クエリへ変換する手法を提案した．エンティティを知識側の語彙と対応付け，エンティティ間のリレーションを知識から獲得することによって，SPARQL クエリが完成されれば答えは保証される．また，基本型，Degree 型，Count 型，List 型の 4 つの質問タイプを設定し，それぞれの質問タイプの自然言語質問文及び，より一般化した，それらを組み合わせた質問文に対してそれぞれ適切な SPARQL クエリを生成することができた．質問タイプを設定することにより，正確なクエリを容易に生成し，自然言語文が意図する質問に回答することを可能にした．

問題点として，自然言語表現と知識側の語彙を対応づける難しさが挙げられる．そこで，形式的に記述された知識に対してより柔軟にアプローチするために，ニューラルネットワークを用いる手法を検討する．

第4章　記憶装置付きニューラルネットワークモデルの文脈を考慮した日本語注文対話への適用

4.1　研究背景と目的

文脈を考慮した対話の研究は，言語として英語を扱ったものが大半であり，日本語を対象としたものは少ない．また，日本語による対話のデータセットはいくつか存在するが，明示的に文脈理解を問うようなデータセットではない．そこで，本研究では既存の英語のデータセットである bAbI データセット [Weston 15][1]を参考に，喫茶店での注文の場面を想定した日本語注文対話データセットを作成し，課題の解決を試みる．作成したデータセットに対し Differentiable Neural Computer (DNC) [Graves 16] を適用することで，文脈を用いた日本語注文対話システムを構築する．

4.2　文脈を考慮した日本語注文対話データセットの作成

喫茶店での注文の場面を想定し，bAbI データセット [Weston 15] の 20 タスクの内 10 タスクを参考に日本語の注文対話データセットを作成する．bAbI データセットとは人間の行う推論を真似た学習ができるように人工的に作られたデータセットであり，タスク毎に “John is in the playground. John picked up the football. Bob went to the kitchen. Where is the football? A:playground” のようなショートストーリーとそれについての質問と答えを含んだデータが 20 種類用意されている．

表 4.1 に作成した注文対話データセットの例を示す．尚，例では A，B，C とアルファベット順に並んでいるが，この順番はランダムにしている．このデータセットに含まれる質問は文脈を理解していなければ答えられないようになっており，これを用いることで日本語における文脈を踏まえた推論の学習を行うことができる．

[1] http://www.thespermwhale.com/jaseweston/babi/tasks_1-20_v1-2.tar.gz

表 4.1: 作成した注文対話データセットの例

Single Supporting Fact（一文参照） A : ハムサンド B : クロワッサン C : カルボナーラ A の注文は? ハムサンド B の注文は? クロワッサン C の注文は? カルボナーラ	**Counting（数え上げ）** A : グレープフルーツジュース B : グレープフルーツジュース C : レモンティー グレープフルーツジュースの注文数は? 2 レモンティーの注文数は? 1
Two Supporting Facts（二文参照） A : ハムサンド B : クロワッサン C : カルボナーラ A : アップルパイ B : バナナブレッド C : チョコレートケーキ A の注文は? ハムサンド, アップルパイ B の注文は? クロワッサン, バナナブレッド C の注文は? カルボナーラ, チョコレートケーキ	**Lists（列挙）** A : ボロネーゼをシェアで A : ハムサンド B : クロワッサン C : カルボナーラ A の注文は? ハムサンド, ボロネーゼのシェア B の注文は? クロワッサン, ボロネーゼのシェア C の注文は? カルボナーラ, ボロネーゼのシェア 全員の注文は? ハムサンド, クロワッサン, カルボナーラ, ボロネーゼ
Three Supporting Facts（三文参照） A : ハムサンド B : クロワッサン C : カルボナーラ A : アップルパイ B : バナナブレッド C : チョコレートケーキ A : アップルジュース B : グレープフルーツジュース C : カプチーノ A の注文は? ハムサンド, アップルパイ, アップルジュース B の注文は? クロワッサン, バナナブレッド, グレープフルーツジュース C の注文は? カルボナーラ, チョコレートケーキ, カプチーノ	**Simple Negation（否定）** A : ハムサンド B : クロワッサン C : カルボナーラ B : クロワッサンではなくてハンバーガーにします A の注文は? ハムサンド B の注文は? ハンバーガー C の注文は? カルボナーラ
Conjunction（接続詞） A : ハムサンドとアップルパイとアップルジュース B : クロワッサンとグレープフルーツジュース C : カルボナーラ A の注文は? ハムサンド, アップルパイ, アップルジュース B の注文は? クロワッサン, グレープフルーツジュース C の注文は? カルボナーラ	**Basic Coreference（基本的共参照）** A : グラタン B : 私もそれで C : ハムサンド A の注文は? グラタン B の注文は? グラタン C の注文は? ハムサンド
Yes No Questions（イエスノー質問） A : アップルパイ B : バナナブレッド C : チョコレートケーキ A の注文はアップルパイですか? はい B の注文はチョコレートケーキですか? いいえ C の注文はバナナブレッドですか? いいえ	**Compound Coreference（複合的共参照）** A : グラタンとアイスティー B : 私もそれらで C : ハムサンドとアップルジュース A の注文は? グラタン, アイスティー B の注文は? グラタン, アイスティー C の注文は? ハムサンド, アップルジュース

- **Single Supporting Fact（一文参照）**

  bAbI データセットにおいて “ Mary went to the bathroom. John moved to the hallway. Mary travelled to the office. Where is Mary? A:office” のように 1 文を見れば質問に対して答えられるようなタスクとなっており，本データセットでは「〜の注文は?」という質問に対して該当する客の発言 1 文を見れば答えを出せるタスクとして用意した.

- **Two Supporting Facts （二文参照）と Three Supporting Facts（三文参照）**

  Single Supporting Fact と同様に，それぞれ 2 文，3 文を参照すれば答えられるタスクとなっている.

- **Conjunction（接続詞）**
  bAbI データセットでは接続詞の “and” を扱っており，本データセットでもそれに対応して「と」を扱うようにした．

- **Yes No Questions（イエスノー質問）**
  質問に対して，はい/いいえで答えるタスクである．

- **Counting（数え上げ）**
  bAbI データセットでは “ Daniel picked up the football. Daniel dropped the football. Daniel got the milk. Daniel took the apple. How many objects is Daniel holding? A: two” のように人物が物を拾ったり落としたりして最終的に持っている物の数を答えるタスクだが，本研究では注文の場面を想定しているため，料理の注文数を答えるタスクとした．

- **Lists（列挙）**
  Counting タスクのように人物が物を拾ったり落としたりし，最終的に持っている物のリストを答えるタスクであるが，本データセットでは「〜の注文は?」という質問に加えて「全員の注文は?」といった質問にも答えるようにした．

- **Simple Negation（否定）**
  否定表現を扱うタスクであり，本データセットでは注文の場面で想定されうる注文の変更を扱うこととした．

- **Basic Coreference（基本的共参照）**
  “ Daniel was in the kitchen. Then he went to the studio. Sandra was in the office. Where is Daniel? A:studio” において代名詞 he が Daniel を指していることが分からないと答えられないようなタスクであり，注文対話においても「私もそれで」の「それ」が何を指すかを問うタスクとした．

- **Compound Coreference（複合的共参照）**
  Basic Coreference における代名詞が複数の人物や物を意味する場合に，きちんと指し示すことができるかを試すタスクである．

本データセットでは，Counting タスク 408 個，Basic Coreference タスク 132 個，Compound Coreference タスク 264 個，他はそれぞれ 1,320 個作成し，総数 10,044 個の注文対話データセットを構築した．

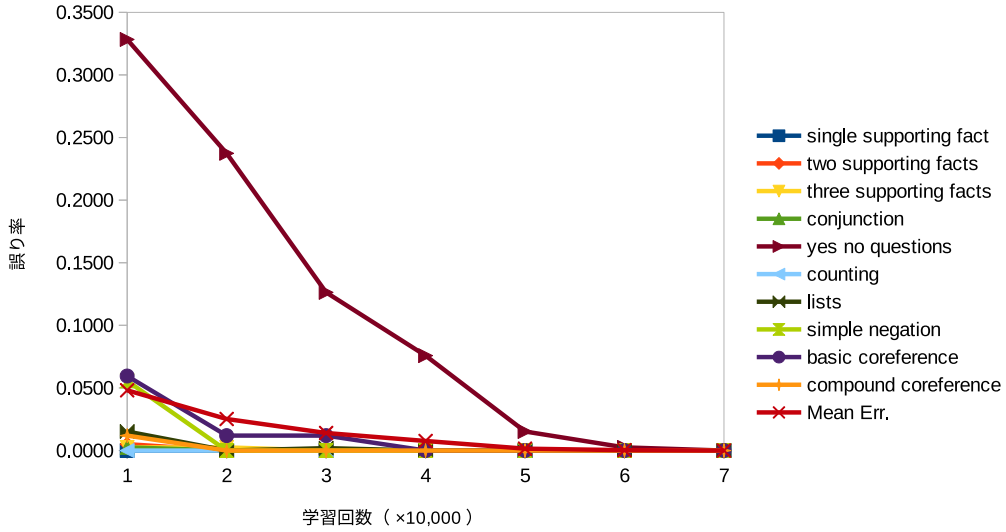


図 4.1: 実験結果

4.3　予備実験

作成した注文対話データセットに対して DNC [Graves 16] を適用して予備実験を行なった.

4.3.1　実験設定

モデルのハイパーパラメータは主に [Graves 16] に基づく；隠れ層サイズ 256 の 1 層 LSTM [Hochreiter 97]，バッチサイズ 1，学習率 1×10^{-4}，メモリ次元 256×64，読み出しヘッド数 4，書き込みヘッド数 1，モメンタム 0.9 の RMSProp オプティマイザ [Tieleman 12]．7 万回学習を行ない，全体の評価事例における予測単語が不正解であった割合を表す誤り率によってモデルを評価した.

4.3.2　結果

実験結果を図 4.1 に示す．Yes No Questions タスクが最も誤り率が高かったが，学習が進むにつれて値は低くなっていった．次に誤り率が高かったのが Basic Coreference タスクであり，これも学習回数を重ねる毎に徐々に下がっていった．また，Simple Negation タスクは，1 万回学習を行なった時点での誤り率はやや高かったが，2 万回以降 0 になった．平均誤り率は 7 万回での評価において 0 になるまで緩やかに減少していった.

4.3.3　考察

Yes No Questions タスクは質問に対して，本データセットにおいて唯一はい/いいえのどちらかを答えるものであるが，間違えた例の全てにおいて正解が「はい」のところを「いいえ」と誤答するなど，正解とは逆のものを答えてしまっていた．Basic Coreference タスクについては，データセットの数が最も少ないタスクであり，データセット数が学習に影響したものと考えられる．Simple Negation タスクは，間違えた例の全てが正解が「ボロネーゼ」の場合であり，誤答として「ハンバーガー」と「ミート」が多く見られ，メインの料理の中で間違えていた．今回作成したデータセットは語彙数も 72 と少なく，かなり単純なものだったため，学習の難易度は比較的低かったと思われる．

4.4　まとめ

本研究では，喫茶店における注文対話の処理にニューラルネットワークモデルを適用した．日本語の注文対話データセットを作成し，DNC を用いて実験を行なった結果，平均テスト誤り率として低い値を得ることができた．今回，bAbI データセットを参考に注文対話を作成したが，今後の課題として，実際の注文対話のコーパスを収集して多様な注文対話に対応できるようにしていきたいと考えている．また，実際の喫茶店で想定される，料理に関する質問などは文脈だけから答えることは難しいため，知識を導入することによってより複雑な対話理解に取り組んでいきたい．

第5章　記憶装置付きニューラルネットワークモデルによる文脈と構造化知識を用いた対話

5.1　研究背景と目的

文脈理解と知識活用は対話にとって重要であり，対話において知識を明示的に扱った研究として [Dinan 18] や [Young 17] が，また文脈と知識の両方を扱った研究として [Guo 18] などが挙げられるが，これらの課題を扱った研究は未だあまり進められていない．また，対話のアーキテクチャとして Seq2Seq [Sutskever 14] や，T5 [Raffel 19], BART [Lewis 19] といった sequence-to-sequence なモデル（相手の発話を入力として応答となる発話を生成するモデル）が主流になっている．一方，より自然で知的な対話を行うには文脈理解や知識活用が必要であるが，そのための長期間におけるデータ保存の能力には限界があると議論されてきた．そこで，文脈情報などの長期の情報を保持するために，Differentiable Neural Computer (DNC) [Graves 16] などの記憶装置付きニューラルネットワークモデルが提案されている．これらのモデルは記憶装置を付け加えたことにより従来のモデルに比べてより複雑な情報処理を行えるようになり，文脈を踏まえた対話においても高い精度を実現している．

ここで，自然言語処理において知識を取り入れる際，形式的に記述された構造化知識は正しい結果を導くが，形式を重んじることから柔軟に利用することができないという問題がある．一方で，分散表現による知識表現では未知の知識項目に対しても柔軟に対応することができる．そこで，本研究では構造化知識を分散表現にして用いることで，正確に記述された知識をニューラルネットワークにおいて柔軟に活用する．知識を導入することの利点として，文脈情報からだけでは答えることができない知識を問う質問に対する応答，曖昧な質問の内容に対する正確な特定，そして未知の語彙に対する柔軟な対応が可能になることが挙げられる．

本研究では，Differentiable Neural Computer (DNC) [Graves 16] を基盤とした，文脈

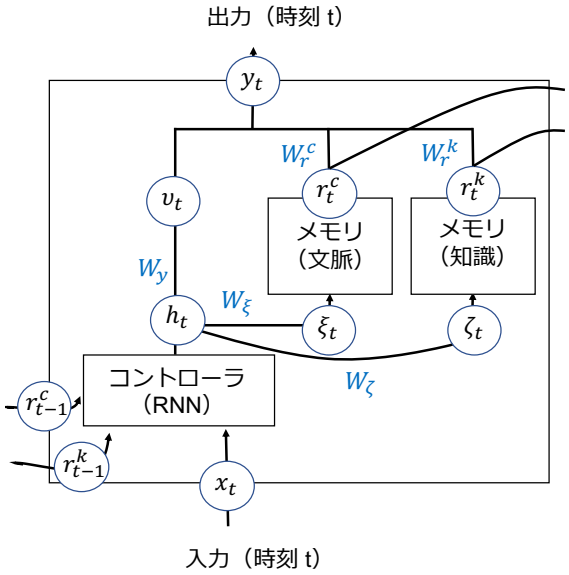


図 5.1: 知識メモリを持つ提案モデルの全体図

を捉えつつ，分散表現による構造化知識を用いた手法を提案する．

5.2　提案手法

DNC [Graves 16] を拡張し知識を格納するメモリを追加することで，文脈と知識の両方を用いた応答の生成を目指す．提案モデルの全体図を図 5.1 に示す．

知識メモリには構造化知識をあらかじめ格納したメモリを用い，書き換え操作は行わず，読み出し操作のみを行う．

1 タイムステップで行う処理の流れは以下の通りである．

1. コントローラ (RNN) は入力 $\boldsymbol{x}_t$ と，前タイムステップで文脈メモリ $M_{t-1}^c \in \mathbb{R}^{N\times W}$ から読み出した R 個のベクトルのセット $\boldsymbol{r}_{t-1}^c = [\boldsymbol{r}_{t-1}^{c,1}; ...; \boldsymbol{r}_{t-1}^{c,R}]$ ($\boldsymbol{r}_{t-1}^c$ は $\boldsymbol{r}_{t-1}^{c,1}, ..., \boldsymbol{r}_{t-1}^{c,R}$ の結合) に加えて，前タイムステップで知識メモリ $M_{t-1}^k \in \mathbb{R}^{N\times W}$ から読み出した R 個のベクトルのセット $\boldsymbol{r}_{t-1}^k = [\boldsymbol{r}_{t-1}^{k,1}; ...; \boldsymbol{r}_{t-1}^{k,R}]$ を受け取る．それから隠れベクトル $\boldsymbol{h}_t$ を出力する．

2. $\boldsymbol{h}_t$ の線形変換により，出力 $\boldsymbol{v}_t = W_y\boldsymbol{h}_t$ と，現タイムステップにおける文脈メモリを制御するためのパラメータを格納したベクトル $\boldsymbol{\xi}_t = W_\xi\boldsymbol{h}_t$，現タイムステップにおける知識メモリを制御するためのベクトル $\boldsymbol{\zeta}_t = W_\zeta\boldsymbol{h}_t$ を得る．

3. $\boldsymbol{\xi}_t$ によって文脈メモリへの書き込みが行われ，メモリの状態が更新される．知識メモリへの書き込みは行われない．

4. 最後に，現タイムステップで文脈メモリから読み出したベクトルを結合した $\boldsymbol{r}_t^c$ と W_r^c をかけて得られるベクトルと，現タイムステップで知識メモリから読み出したベクトルを結合した $\boldsymbol{r}_t^k$ と W_r^k をかけて得られるベクトルに $\boldsymbol{v}_t$ を足し，出力 $\boldsymbol{y}_t$ を計算する．

$$\boldsymbol{y}_t = \boldsymbol{v}_t + W_r^c \boldsymbol{r}_t^c + W_r^k \boldsymbol{r}_t^k$$

read vector $\boldsymbol{r}_t^c$ と $\boldsymbol{r}_t^k$ は次タイムステップの RNN への入力に追加される．

以上の処理を繰り返すことにより，二つのメモリへの読み書き操作を行う．

DNC との主な違いは，コントローラへの入力に前時刻で知識メモリから読み出した情報 $\boldsymbol{r}_{t-1}^k$ を，コントローラからの出力に知識メモリから読み出された情報 $\boldsymbol{r}_t^k$ を新たに与える点である．モデルは文脈メモリと知識メモリからの読み出し情報を継承していくため、multi-hop reasoning を実行する．

5.2.1　知識メモリの構築

本研究では知識として構造化知識，つまり，主語，述語，目的語の三つ組で表現されたデータ（例えば,「日本の首都は東京である.」は三つ組 (日本，首都，東京) で表される）を扱う．DNC のモデルを用いて，三つ組の全要素を受け取った上で三つ組の内 2 つを入力され，残りの 1 つを出力する（例えば，“日本”，“首都”，“東京”，“日本”，“首都” のような入力に対して “東京” を返す）ように学習し終わった後のメモリを提案モデルの中で知識メモリとして利用する．

5.3　実験

5.3.1　実験設定

すべてのモデルのハイパーパラメータは主に [Graves 16] に基づく；隠れ層サイズ 256 の 1 層 LSTM [Hochreiter 97]，バッチサイズ 32，学習率 1×10^{-4}，文脈メモリ次元 256×64，知識メモリ次元 512×128，読み出しヘッド数 4，書き込みヘッド数 1，モメンタム 0.9 の RMSProp オプティマイザ [Tieleman 12]．[Franke 18] に従い rsDNC [Franke 18] のドロップアウト率は 10%とした．知識ベースの単語の埋め込みには TransE [Bordes 13]

表 5.1: CSQA データセットの対話例

USER	SYSTEM
Can you tell me which cities border Verderio Inferiore ?	Cornate d'Adda, Bernareggio, Robbiate
And which cities flank that one ?	Did you mean Robbiate ?
No, I meant Cornate d'Adda.	Bottanuco, Busnago, Trezzo sull'Adda
Which ones share border with Robbiate and are not part of Azerbaijan ?	Trezzo sull'Adda, Busnago, Bottanuco
Which city is the capital of min number of french administrative divisions ?	Riga
Is Rot am See located adjacent to that one ?	No
How many political territories have strategic relation with around 22 other countries ?	11
Which are the nations connected with Robbiate and are not located in Italy ?	The answer count is 11. Do you want to see all possibilities ?
No, show only a few of them.	Papua New Guinea, Jamaica, Antigua and Barbuda, Austria, New Zealand, The Bahamas, Gabon
How many countries are diplomatically related to Italy ?	74
And how many of them also with Australia ?	56
Which countries have lesser number of political relations with others than Australia ?	Denmark, Canada, Grenada

を使用した．知識ベースの語彙ではないが，対話に現れる “the” や “what” のような単語には [Saha 18] を参考に GloVe embeddings [Pennington 14] を使用した．各単語の埋め込みベクトルの次元は 200 である．学習中に Validation 結果の 10 エポック連続下降を 5 回繰り返したら，Early stopping とした．ランダムな初期化の下で各モデルを 3 回実行し，平均の結果を報告する．

5.3.2　CSQA データセット

Complex Sequential Question Answering (CSQA) データセット [Saha 18] は一貫性があり，かつ大規模な知識を要求する質問とその答えで構成された会話のデータセットであり，以下の 4 つの能力を問うものとなっている．

1. 複雑な自然言語質問文の解析
2. 発話における共参照と省略を解決するために文脈の利用
3. 曖昧な質問の明確化
4. 質問に答えるために知識グラフの関係するサブグラフの検索

表 5.1 に CSQA データセットの対話例を示す．約 20 万対話（合計 160 万ターン）から成り，知識として Wikidata[1]が用いられている．今回の実験では，Wikidata に対して

[1] https://www.wikidata.org/wiki/Wikidata:Main_Page

表 5.2: 各質問タイプにおける誤り率

質問タイプ	DNC	提案手法
All	**57.18%**	57.74%
Simple Question (Direct)	70.26%	**69.98%**
Simple Question (Coreferenced)	69.82%	**69.14%**
Simple Question (Ellipsis)	**78.23%**	78.57%
Logical Reasoning (All)	66.72%	**65.60%**
Quantitative Reasoning (All)	**61.31%**	64.16%
Comparative Reasoning (All)	**50.76%**	52.95%
Clarification	29.11%	**28.45%**
Verification (Boolean) (All)	19.20%	**15.85%**
Quantitative Reasoning (Count) (All)	**43.07%**	47.06%
Comparative Reasoning (Count) (All)	**83.36%**	85.64%

Saha らと同じ前処理を行なった上で 10 回以上出現する単語のみを含んだトリプルを抽出した結果，トリプルの総数は 9,453,367，エンティティとリレーションの語彙数はそれぞれ 920,391，328 となった．

まず，オリジナルの DNC モデルに対して Wikidata で学習し，知識をあらかじめ格納した知識メモリを構築する．そして，その知識メモリを用いて提案モデルの実験を行なった．また，提案モデルとの比較のため，Wikidata は用いず会話のデータセットのみを用いて DNC モデルで実験を行い，知識メモリを追加することの有効性を検証した．

知識メモリには 50 万回学習したメモリを用い，DNC モデルと提案モデルについては両方 10 万回学習を行なった．

結果

全体の評価事例における予測単語が不正解であった割合を表す誤り率によって評価した．各質問タイプにおける誤り率の結果を表 5.2 に示す．

Simple Question は知識グラフ内の 1 つのトリプルのみで答えられる質問（例えば，“Which city is the capital of Japan?”）である．“Q: Which rivers flow through Tokyo? A: Sumida, Meguro, Kanda, ...” のような複数の正解を持つ，つまり，複数のトリプルが関係している質問も複数トリプルによる複合的な推論を必要とする訳ではないので Simple Question とする．また，文脈に依存するか否かにより，Direct と Indirect（共

参照と省略）に分かれる．Direct に比べて Coreferenced はあまり変わらなかったが，Ellipsis では結果が下がった．Direct と Coreferenced で提案モデルの方が良い結果になった．

Simple Question に対して推論を必要とするような複雑な質問として，Logical Reasoning，Quantitative Reasoning，Comparative Reasoning の三つがある．Logical Reasoning は知識グラフ内の複数のトリプルに対して論理的推論を必要とする質問（例えば，“Which rivers flow through Tokyo and Saitama?”）である．この質問に答えるにはトリプル (river, flows through, Tokyo) に現れる river の集合とトリプル (river, flows through, Saitama) に含まれる river の集合の作成が必要であり，質問の最終的な答えはこれら 2 つの集合の共通集合である．

Quantitative Reasoning は max，min，count などの標準的な集合関数を含んだ推論を必要とする質問（例えば，“Which river flows through maximum number of prefectures?”）である．Quantitative Reasoning (Count) は “How many rivers flow through Tokyo?” のような数字を答えさせる質問である．

Comparative Reasoning は特定のリレーションに基づくエンティティ間の比較を必要とする質問（例えば，“Which prefectures have more number of rivers than Tokyo?”）である．知識グラフ内の複数のトリプルに対する推論が求められ，count，sort や more/less といった演算子の学習が不可欠である．Comparative Reasoning (Count) は “How many prefectures have more number of rivers than Tokyo?” のような比較の後に数え上げが必要な質問である．

Simple Question に比べて Comparative Reasoning (Count) 以外の結果は上がったが，Comparative Reasoning (Count) の結果が全質問タイプ中で最も悪かった．比較して数え上げるというのがうまくモデル化できていないと思われる．Logical Reasoning のみ提案モデルの方が結果が良くなった．

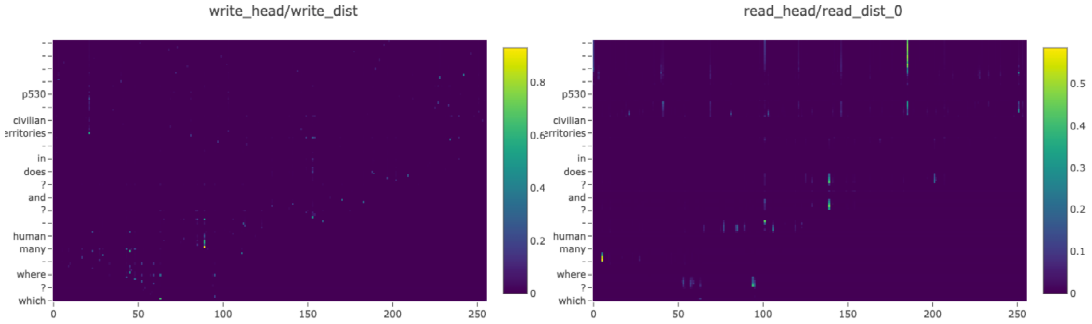
Verification (Boolean) は Yes/No によって答えられる確認をする質問（例えば，“Does Sumida flow through Tokyo?”）である．他の質問タイプに比べてかなり誤り率は低く，Yes/No のみで答えるため比較的簡単であったと思われる．

Clarification は例えば，直前に “Q: Which rivers flow through Tokyo? A: Sumida, Meguro, Kanda, ...” といったやり取りがあり，次に “And Does it also flow through Saitama?” のような共参照を含むが参照先のエンティティが 1 つに定まらない場合に “Did you mean Sumida?” のように質問の意図を明確化するために正しく聞き返すことができるかを問うタスクである．他とは異なり，エンティティと知識グラフにない単語の系列からな

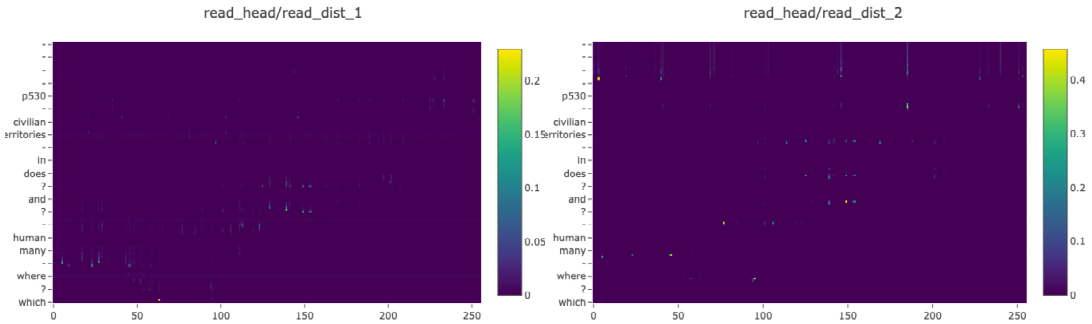
る自然言語の応答を生成しなければならない．知識グラフを用いなくても答えられるためか誤り率は低く，提案モデルの方が良い値となった．

考察

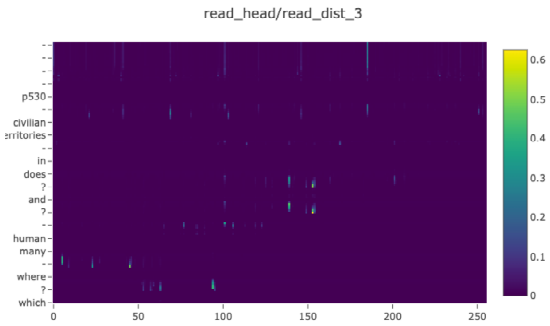
図 5.2 に DNC モデルの各タイムステップにおけるメモリからの読み/書きのアテンションの重みを可視化した結果，図 5.3 に提案モデルの各タイムステップにおける文脈メモリからの読み/書きのアテンションの重みを可視化した結果，図 5.4 に提案モデルの各タイムステップにおける知識メモリからの読み出しのアテンションの重みを可視化した結果を示す．



(a) 書き込みヘッド　　**(b)** 読み出しヘッド 0

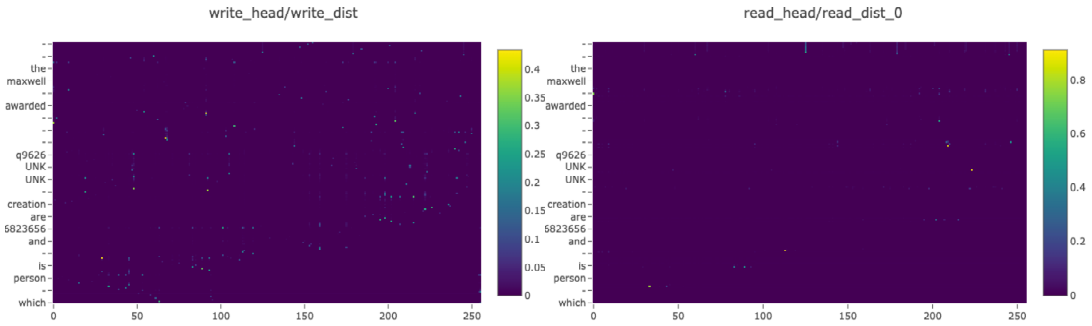


(c) 読み出しヘッド 1　　**(d)** 読み出しヘッド 2

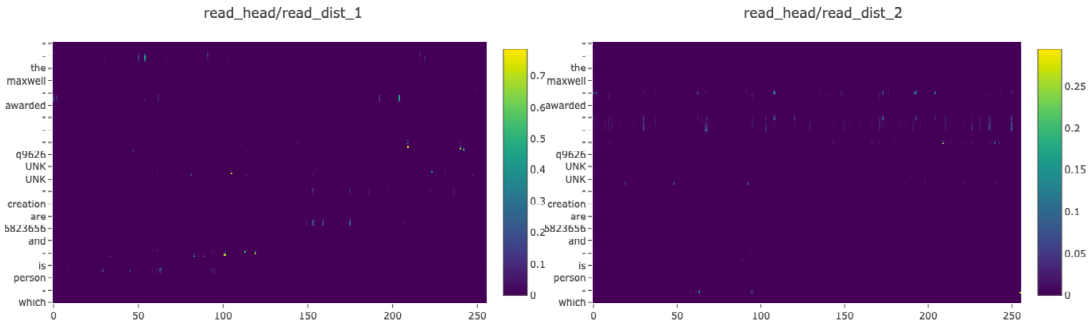


(e) 読み出しヘッド 3

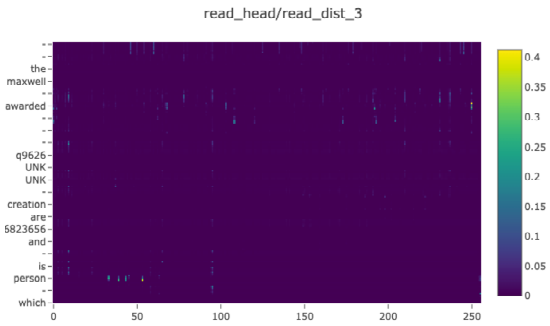
図 5.2: DNC モデルの各タイムステップにおけるメモリに対する読み出し/書き込みのアテンションの重みの可視化結果



(a) 書き込みヘッド　　**(b)** 読み出しヘッド 0

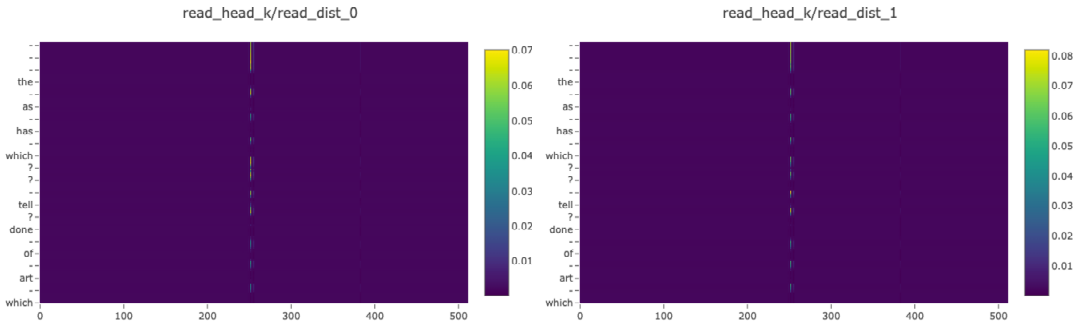


(c) 読み出しヘッド 1　　**(d)** 読み出しヘッド 2



(e) 読み出しヘッド 3

図 5.3: 提案モデルの各タイムステップにおける文脈メモリに対する読み出し/書き込みのアテンションの重みの可視化結果



(a) 読み出しヘッド 0　　**(b)** 読み出しヘッド 1

(c) 読み出しヘッド 2　　**(d)** 読み出しヘッド 3

図 5.4: 提案モデルの各タイムステップにおける知識メモリからの読み出しのアテンションの重みの可視化結果

縦軸は各タイムステップ，つまり各単語に対応し，横軸はメモリのスロット数である．図 5.2 ではアテンションが強くかかっている部分が所々あるものの，書き込み操作と読み出し操作の間にあまり相関は見られなかった．図 5.3 についても同様の結果となった．図 5.4 では特定のスロットのみにアテンションが当たっていた．

提案モデルのテストデータにおける応答予測の一例を図 5.5 に示す．

5.3.3　Dialog bAbI データセット

Dialog bAbI データセット [Bordes 16] はレストラン予約という設定におけるタスク指向型対話 6 タスクのセットである．その中から，Task 1-4 をつなげた Task 5 (Full dialogs) を使用し，さらにスペシャルトークン `api_call` で始まる文を本研究では除去した．訓練，開発，テストセットはそれぞれ 1,000 例である．訓練セットと開発

表 5.3: Dialog bAbI データセットにおける平均 per-response accuracy

Task	DNC	DNC +KM	rsDNC	rsDNC +KM	DNC-DMS	DNC-DMS +KM	DNC-MD	DNC-MD +KM
Full dialogs	83.04%	**86.52%**	87.77%	**92.46%**	**84.48%**	83.83%	82.50%	**84.30%**
Full dialogs (OOV)	73.06%	**74.60%**	**76.11%**	75.90%	**73.57%**	71.83%	72.33%	**72.52%**

表 5.4: Dialog bAbI データセットにおける詳細結果．“w/o KB facts” は知識ベースのファクトなしで答えられるタスクを表し，“w/ KB facts” は質問に答えるのに知識ベースのファクトが必要なタスクを表す．“w/ KB facts (OOV)” の結果はすべて 0.00%だったため省略した．

Task	DNC	DNC +KM	rsDNC	rsDNC +KM	DNC-DMS	DNC-DMS +KM	DNC-MD	DNC-MD +KM
w/o KB facts	99.99%	99.99%	100.00%	100.00%	100.00%	100.00%	99.99%	**100.00%**
w/ KB facts	29.53%	**43.98%**	49.14%	**68.66%**	**35.49%**	32.78%	27.28%	**34.73%**
w/o KB facts (OOV)	95.98%	**98.01%**	**99.99%**	99.72%	**96.65%**	94.37%	95.03%	**95.27%**

セットで現れなかったエンティティを含む Out-Of-Vocabulary (OOV) テストセットも 1,000 例用意されている．知識ベースは (`resto_seoul_cheap_korean_1stars`, `R_cuisine`, `korean`) のようなレストランのドメインにおける 8,400 ファクトを持つ．エンティティ数は 3,635，リレーション数は 7，対話の語彙サイズは 2,043 である．

結果

表 5.3 に Dialog bAbI データセットにおける全てのモデルのテスト 3 実行分の平均 per-response accuracy を示す．DNC はオリジナルの DNC モデルであり，rsDNC は robust and scalable DNC [Franke 18] であり，DNC-DMS [Csordás 19] は 3 つの修正 (i.e. de-allocation mechanisms, masked content based addressing, and sharpness enhancement) を持ち，そして DNC-MD (DNC-DMS の一種) は masking と de-allocation modifications のみである．“+KM” モデルは知識メモリユニットを追加した提案モデルである．

DNC+KM はオリジナルの DNC を Full dialogs と OOV タスクの両方で上回った．rsDNC+KM は全体で最も高い結果を達成した．DNC-DMS+KM のスコアは DNC-DMS のスコアより低かった．[Csordás 19] で QA タスクにおいては DNC-DMS より DNC-MD の方が結果が良かったと報告されているため，DNC-MD モデルの実験も行い，DNC-MD+KM の値が DNC-MD より高いことが分かった．これは temporal memory linkage を使わない知識メモリの性質によるものだと推察する．知識メモリを作成するときの知識ベーストリプルの順番と対話中の単語の順番は無関係なので，sharpness enhancement がうまく機能しなかったと考えられる．

表 5.4 に KB facts なしで答えられるタスクと，質問に答えるのに KB facts を必要とするタスクに分けた詳細な結果を示す．w/o KB facts タスクは 12,351 文，w/ KB facts タスクは 3,912 文をテストセットに含む．知識を要求しない一般的な対話を扱う w/o KB facts タスクと w/o KB facts (OOV) タスクについては，どちらも総じて高い結果となった．知識を要求する対話を扱う w/ KB facts タスクについては，DNC が 29.53%で，DNC+KM が 43.98%であったため，提案モデルの方が 14.45%向上した．また，rsDNC は 68.66%，rsDNC+KM は 49.14%だったため，提案モデルは 19.52%の結果の向上が見られた．さらに，DNC-MD は 27.28%，DNC-MD+KM は 34.73%であり，提案モデルが 7.45%向上した．

rsDNC+KM に注目し，モデルがメモリに何を書き込み，メモリから何を読み出すのかを調査するために，タイムステップ毎のメモリに対する読み出し/書き込みのアテンションの重みの結果を可視化する．図 5.6，図 5.7，図 5.8 は表 5.5 に示す，モデルの出力がすべて正解となった成功例についてのアテンションの重みを可視化している．図 5.6 において，横軸は文脈メモリにおける番地を表し，縦軸は入力データ，つまりタイムステップ毎のユーザ発話あるいはモデル出力を表す．モデルは入力系列を受け付ける間は何も返さず，入力がないタイミングに応答を生成する．例えば，入力系列が “can, you, make, a, restaurant, reservation, in, paris, -, -, -” のとき出力系列は “-, -, -, -, -, -, -, -, -, i’m, on, it” である（‘-’ はパディングワード）．図 5.6 は表 5.5 の例のターン 2 からターン 6 における文脈メモリへの書き込みのアテンションの重みを示す．“paris” のような知識ベースの単語に強いアテンションがあり，“actually” や “instead” などのレストランの条件を変更するのに使われる単語にもアテンションがかかっているのが興味深い．図 5.7 は表 5.5 の例のターン 8 からターン 14 における文脈メモリからの読み出しのアテンションの重みを示す．モデルのメモリは 4 つの読み出しヘッドを使用しており，その内の 1 つを示す．モデルがレストラン情報を出力するとき “indian” が強く書き込まれたスロットにアテンションがあり，正解はインド料理のレストランなので，“indian” という読み出した情報が有効だったと思われる．図 5.8 は知識メモリからの読み出しのアテンションの重みを示す．モデルがレストラン情報を答える前にアテンションがかかっている様子が見られたが，文脈メモリに比べて明瞭な特徴は見つからなかった．

表 5.6 に示すようにモデルが w/ KB facts タスクを全部間違えたときの結果が悪い例も検証した．図 5.9 は表 5.6 のターン 2 からターン 8 における文脈メモリへの書き込みのアテンションの重みの結果の一部を示す．“moderate” や “british” といった知識ベー

表 5.5: Dialog bAbI データセットでの rsDNC+KM の出力がすべて正解となった成功例

	ユーザの発話	rsDNC+KM の出力	正解
1	good morning	hello what can i help you with today ✓	hello what can i help you with today
2	can you make a restaurant reservation in paris for six with indian cuisine	i'm on it ✓	i'm on it
3	<silence>	which price range are looking for ✓	which price range are looking for
4	i am looking for a expensive restaurant	ok let me look into some options for you ✓	ok let me look into some options for you
5	actually i would prefer in madrid	sure is there anything else to update ✓	sure is there anything else to update
6	instead could it be in a moderate price range	sure is there anything else to update ✓	sure is there anything else to update
7	no	ok let me look into some options for you ✓	ok let me look into some options for you
8	<silence>	what do you think of this option: resto_madrid_moderate_indian_7stars ✓	what do you think of this option: resto_madrid_moderate_indian_7stars
9	no i don't like that	sure let me find an other option for you ✓	sure let me find an other option for you
10	<silence>	what do you think of this option: resto_madrid_moderate_indian_5stars✓	what do you think of this option: resto_madrid_moderate_indian_5stars
11	no i don't like that	sure let me find an other option for you ✓	sure let me find an other option for you
12	<silence>	what do you think of this option: resto_madrid_moderate_indian_2stars ✓	what do you think of this option: resto_madrid_moderate_indian_2stars
13	let's do it	great let me do the reservation ✓	great let me do the reservation
14	may i have the phone number of the restaurant	here it is resto_madrid_moderate_indian_2stars_phone ✓	here it is resto_madrid_moderate_indian_2stars_phone
15	you rock	is there anything i can help you with✓	is there anything i can help you with
16	no thanks	you're welcome ✓	you're welcome

スの単語に強いアテンションがあった．図 5.10 は表 5.6 のターン 10 からターン 14 における文脈メモリからの読み出しのアテンションの重みの結果の一部を示す．モデルがレストラン情報を出力するときに “british” が書き込まれたスロットと “with” が書き込まれたスロットにアテンションがあった．“with” という不必要な情報がモデルの出力に負の影響かもしれない．図 5.11 は知識メモリからの読み出しのアテンションの重みを示す．モデルがレストラン情報を答える前にアテンションがかかっており，良例における知識メモリからの読み出しのアテンションの重みと似た挙動を示した．

5.3.4　Movie Dialog データセット

Movie Dialog データセット [Dodge 16] は映画のトピックに関する 4 つの対話タスクのセットである．質問応答タスクと推薦タスクを組み合わせた Task 3 (QA+Recommendation Dialog) を使用した．対話は 3 ターン構成である：最初のターンは推薦を要求する，e.g. “I'm looking for a Brian De Palma movie. Response: Blow Out”．2 番目のターンでは，ユーザはモデルの前の応答についてのファクトイド型質問を尋ねる， e.g. “Who does that star? Response: John Travolta, John Lithgow, Nancy Allen, Dennis Franz”．3 番目のターンでは，ユーザは別の推薦を要求し，彼らの趣向について追加情報を与える，e.g. “I prefer Robert De Niro movies. Can you suggest an alternative? Response: Hi Mom!”.

表 5.6: Dialog bAbI データセットでの rsDNC+KM の出力が “w/ KB facts” タスクですべて不正解だったときの結果が悪い例

	ユーザの発話	rsDNC+KM の出力	正解
1	good morning	hello what can i help you with today ✓	hello what can i help you with today
2	i’d like to book a table in a moderate price range	i’m on it ✓	i’m on it
3	<silence>	any preference on a type of cuisine ✓	any preference on a type of cuisine
4	with british food	where should it be ✓	where should it be
5	madrid please	how many people would be in your party ✓	how many people would be in your party
6	for four people please	ok let me look into some options for you ✓	ok let me look into some options for you
7	instead could it be in a expensive price range	sure is there anything else to update ✓	sure is there anything else to update
8	actually i would prefer for six	sure is there anything else to update ✓	sure is there anything else to update
9	no	ok let me look into some options for you ✓	ok let me look into some options for you
10	<silence>	what do you think of this option: resto_london_expensive_british_4stars ✗	what do you think of this option: resto_madrid_expensive_british_8stars
11	no this does not work for me	sure let me find an other option for you ✓	sure let me find an other option for you
12	<silence>	what do you think of this option: resto_paris_expensive_british_4stars ✗	what do you think of this option: resto_madrid_expensive_british_6stars
13	i love that	great let me do the reservation ✓	great let me do the reservation
14	do you have its phone number	here it is resto_madrid_moderate_british_5stars_phone ✗	here it is resto_madrid_expensive_british_6stars_phone
15	thanks	is there anything i can help you with✓	is there anything i can help you with
16	no thank you	you’re welcome ✓	you’re welcome

表 5.7: Movie Dialog データセットにおける平均 hits@k．“k=1” は hits@1，“k=10” は hits@10 を意味する．

Task	DNC	DNC +KM	rsDNC	rsDNC +KM	DNC-DMS	DNC-DMS +KM	DNC-MD	DNC-MD +KM
Whole test set (k=1)	**18.74%**	18.54%	18.26%	**18.54%**	18.57%	**18.61%**	**18.76%**	18.58%
Whole test set (k=10)	**37.72%**	37.49%	35.72%	**36.13%**	37.53%	**37.88%**	**38.38%**	37.33%

データセットは訓練用に 100 万対話例，開発とテスト用に 1 万ずつ含む．それらの内，計算資源の制約により訓練用に 100k，開発用に 4,907，そしてテスト用に 4,766 を使用した．Movie Dialog データセットの知識ベースは Open Movie Database (OMDb)[2]と MovieLens dataset[3]から構築されている．減らした対話データ中に出現するエンティティを持つトリプルのみを元の知識ベースから抽出し，126,999 トリプルを得た．エンティティ数は 37,055，リレーション数は 10，対話の語彙サイズは 26,314 である．

結果

表 5.7 に Movie Dialog データセットにおける全てのモデルのテスト 3 実行分の平均 hits@1 及び hits@10 を示す．hits@1 と hits@10 のどちらも我々の DNC+KM と

[2]http://beforethecode.com/projects/omdb/download.aspx.
[3]http://grouplens.org/datasets/movielens/

表 5.8: Movie Dialog データセットにおける詳細結果．“k=1” は hits@1，“k=10” は hits@10 を意味する．

Task	DNC	DNC +KM	rsDNC	rsDNC +KM	DNC-DMS	DNC-DMS +KM	DNC-MD	DNC-MD +KM
Response 1 (Recs) (k=1)	**22.07%**	21.97%	14.46%	**14.56%**	21.49%	**21.95%**	**21.87%**	21.64%
Response 2 (QA) (k=1)	**7.18%**	6.80%	8.97%	**9.28%**	**7.07%**	6.82%	**7.27%**	6.87%
Response 3 (Similar) (k=1)	38.16%	**38.23%**	40.99%	**41.39%**	38.35%	**38.50%**	38.30%	**38.64%**
Response 1 (Recs) (k=10)	50.25%	**50.52%**	37.20%	**37.97%**	49.05%	**50.28%**	**50.43%**	49.36%
Response 2 (QA) (k=10)	**18.84%**	18.26%	20.62%	**20.72%**	18.79%	**18.88%**	**19.89%**	18.52%
Response 3 (Similar) (k=10)	**61.69%**	61.62%	64.28%	**64.90%**	**62.31%**	62.23%	**62.08%**	61.68%

DNC-MD+KM はオリジナルモデルより結果が悪かったが，我々の rsDNC+KM と DNC-DMS+KM はオリジナルモデルより結果が良かった．表 5.8 により詳細な分析のための結果を示す．それぞれのタスクの出力は知識ベースのエンティティのリストである．Response 1 (Recs) は対話の最初のターンであり，推薦を要求する．Response 1 のテストセットに含まれるエンティティ数は 5,421 である．Response 2 (QA) は 2 番目のターンであり，モデルは前のターンから文脈を考えてファクトイド型質問に答える必要がある．Response 2 は 1 つ以上のエンティティを答えることをしばしば求められるため，テストセットに 9,867 エンティティを持つ．Response 3 (Similar) は 3 番目のターンであり，モデルは彼らの趣向に関するユーザの追加情報を与えられて別の推薦を提供する．4,939 エンティティがテストセットに含まれる．Response 3 タスクでは知識メモリを持つ我々のモデルの hits@1 スコアがどの DNC モデルでも高かった．rsDNC モデルにおいては，rsDNC+KM の hits@1 と hits@10 はどちらも 3 タスクすべてで向上した．rsDNC+KM のスコアが Response 2 と Response 3 タスクで他のモデルを上回ったにもかかわらず，Response 1 タスクの結果はかなり低かった．Response 1 タスクは “Gentlemen of fortune, Revanche, Eternal sunshine of the spotless mind, Prometheus, Fanny and Alexander, The hurt locker, and 127 hours are films I really like. I’m looking for a Brian De Palma movie.” のような長い入力文を扱う必要があるため，rsDNC モデルは長い系列の処理に難があると考えられる．

5.4　まとめ

DNC を拡張し，文脈を捉えつつ構造化知識を活用した対話のモデルを提案した．CSQA データセットを用いて提案モデルとオリジナルの DNC モデルの実験を行ない，誤り率を比較した結果，10 項目中 5 項目で結果は上がったが，全体の結果は DNC モデルより下がってしまった．原因の一つとして，知識メモリのサイズが約 900 万のト

リプルを記録するには 512 × 128 ではまだ小さすぎたことが考えられる．今回の実験では学習イテレーション数が 10 万回だったので，学習回数を増やして結果をまた考察したい．今後の課題として，提案モデルでは論理や集合といった関数が正しくモデル化されていないので，自然言語文に対して正確なパージングを行なった上でこれらの関数を end-to-end で設計する方法を考えていきたい．3 つの DNC モデル，vanilla DNC, rsDNC, DNC-DMS に知識メモリを追加した．背景知識を必要とする対話タスクにおける提案手法の効果について分析した．提案モデル，DNC+KM, rsDNC+KM, DNC-MD+KM が the (6) dialog bAbI tasks dataset の full dialog tasks でオリジナルモデルを上回った．特に，各モデルは KB ファクトを要求するタスクで，それぞれおよそ 14%, 20%, 7%向上した．Movie Dialog dataset では，rsDNC+KM, DNC-DMS+KM がオリジナルモデルより良い結果になった．

Which party does unk belong to? - - -
q29468 [q9626]
[]
Which sex does that person possess? - - -
q6581097 [q6581097]
[]
Is unk a that sex? - - -
yes [yes]
[]
Which languages can unk read and were used originally for the creation of q26823656? - - -
q1860 [q1860]
[]
What are considered to be the original language of creation of q26823656 or unk? - - -
q1860 [q1860]
q1321 []
Is that language a p361 unk? - - -
yes [no]
[]
Does unk and q192 serve as a chairman for q9626? - - -
no [no]
and [and]
yes [yes]
respectively [respectively]
[]
Whose first name are q1914312? - - -
UNK [q313972]
UNK [q215040]
[kalu]
[peter]
[q215040]
[davies]
[]
Which military decoration was that person awarded with? - - -
did [did]
you [you]
mean [mean]
UNK [q6796118]
? [?]
[]
No, i meant maxwell davenpolt taylor. Could you tell me the answer for that? - - -
UNK [q17231624]
[]

図 5.5: 提案モデルのテストデータにおける応答予測の一例．[] 内の単語は正解を表す．予測した単語と正解が一致していれば緑字，異なれば赤字で表示される．

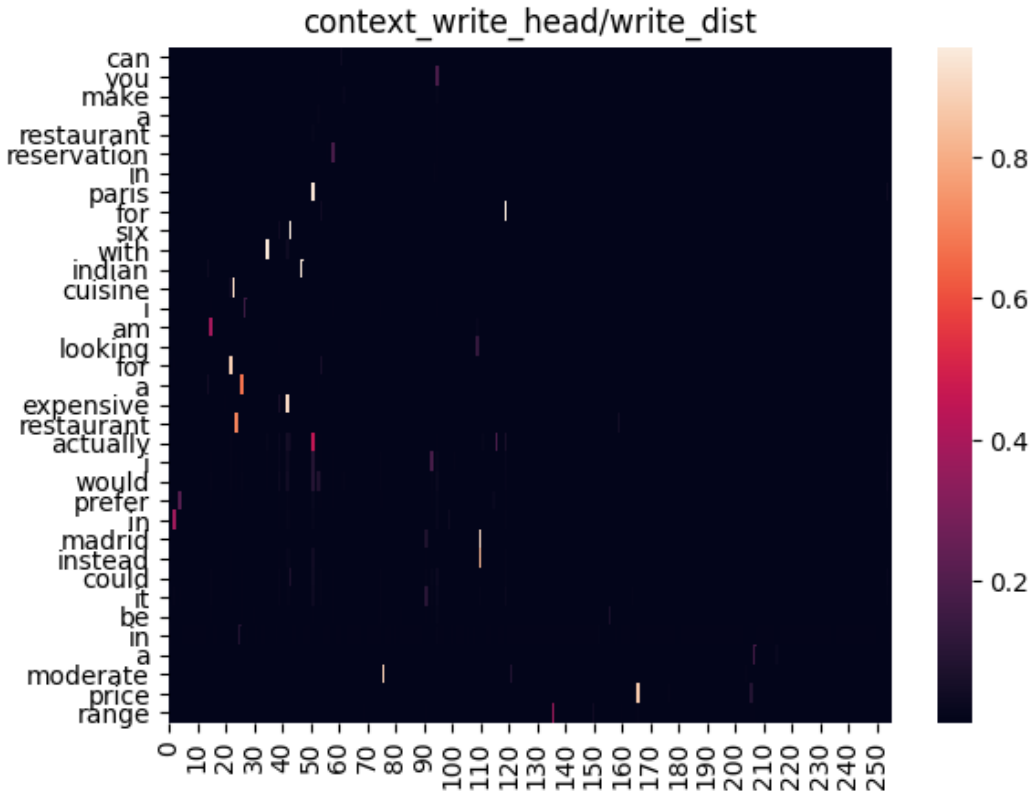


図 5.6: Dialog bAbI データセットでの rsDNC+KM の成功例における文脈メモリへの書き込みのアテンションの重みの可視化結果．横軸は文脈メモリにおける番地を表し，縦軸はタイムステップ毎の入力データあるいはモデル出力を表す．

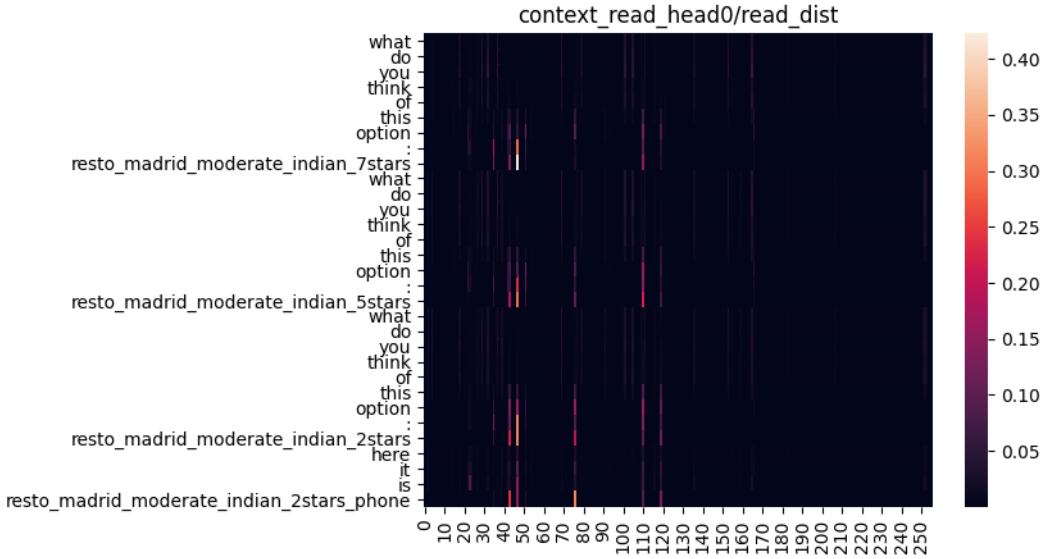


図 5.7: Dialog bAbI データセットでの rsDNC+KM の成功例における文脈メモリからの読み出しのアテンションの重みの可視化結果

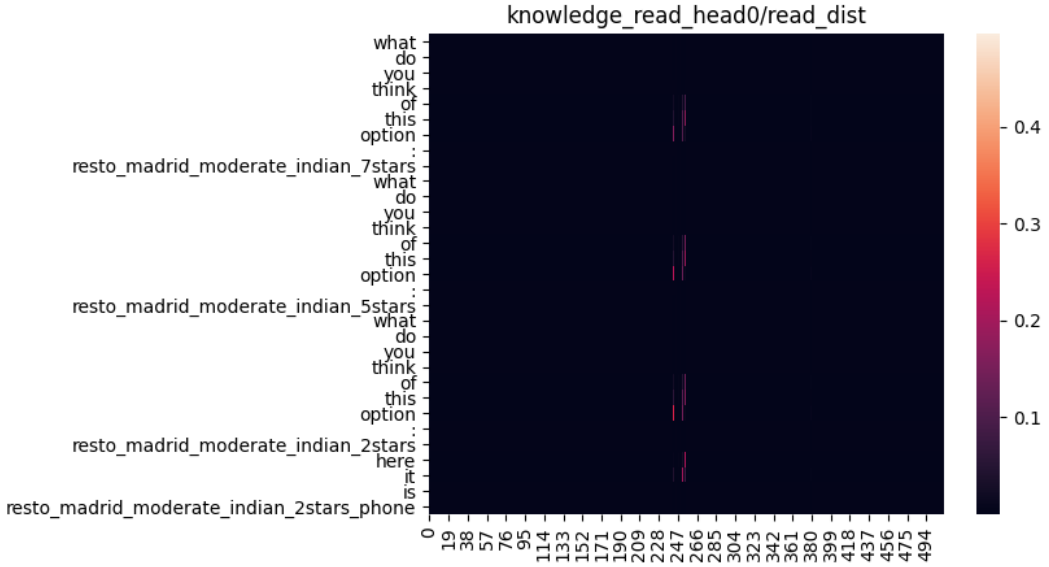


図 5.8: Dialog bAbI データセットでの rsDNC+KM の成功例における知識メモリからの読み出しのアテンションの重みの可視化結果

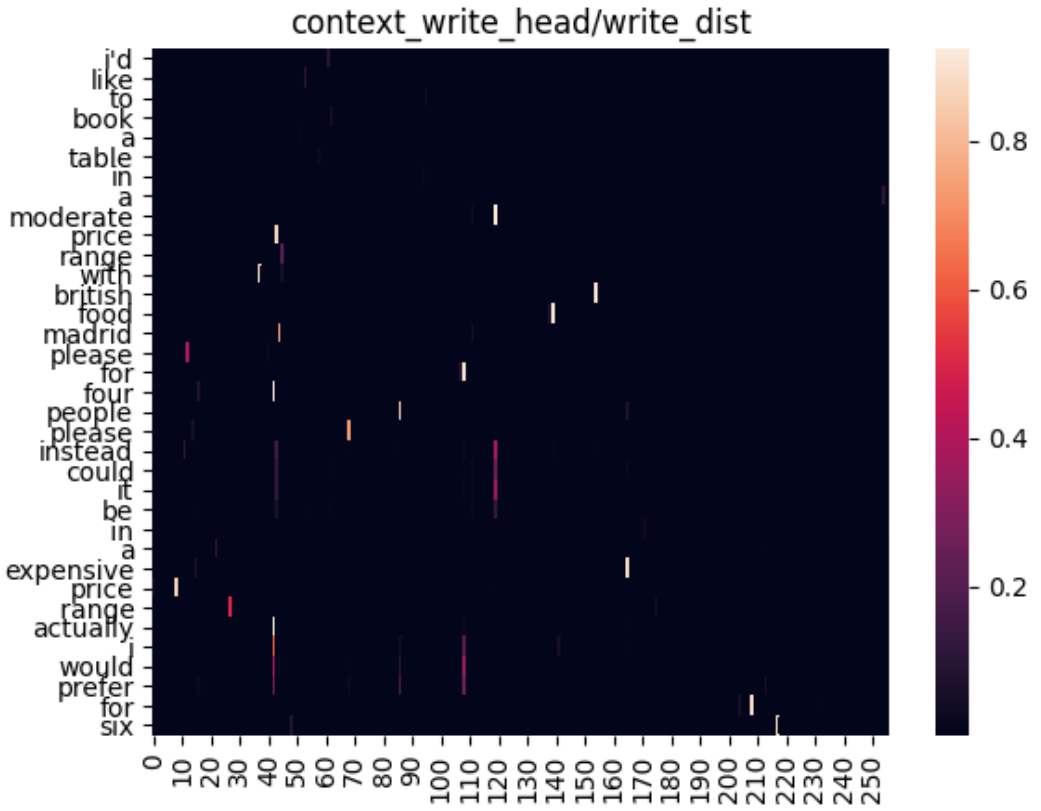


図 5.9: Dialog bAbI データセットでの rsDNC+KM の結果が悪い例における文脈メモリへの書き込みのアテンションの重みの可視化結果

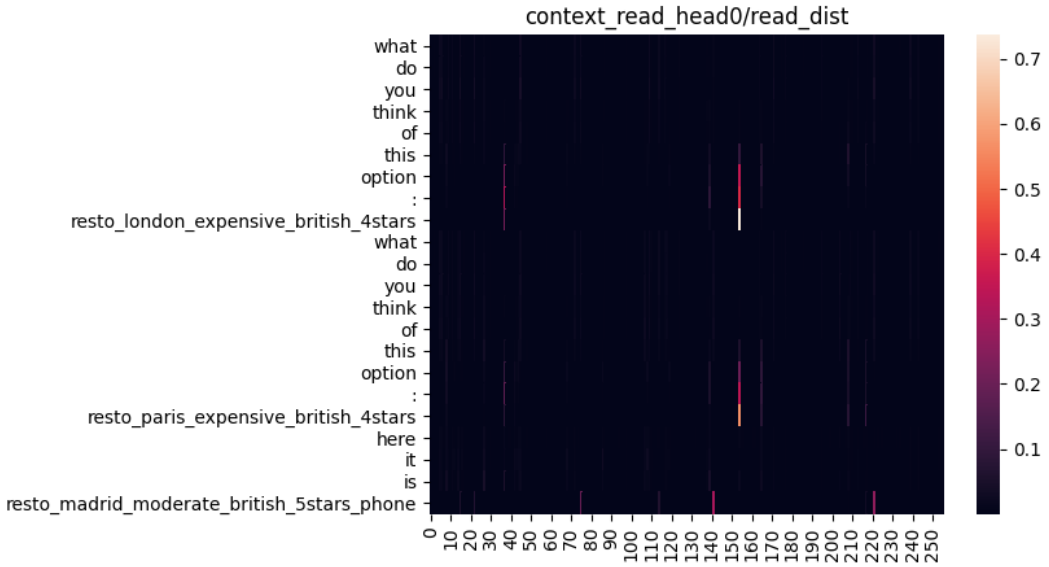


図 5.10: Dialog bAbI データセットでの rsDNC+KM の結果が悪い例における文脈メモリからの読み出しのアテンションの重みの可視化結果

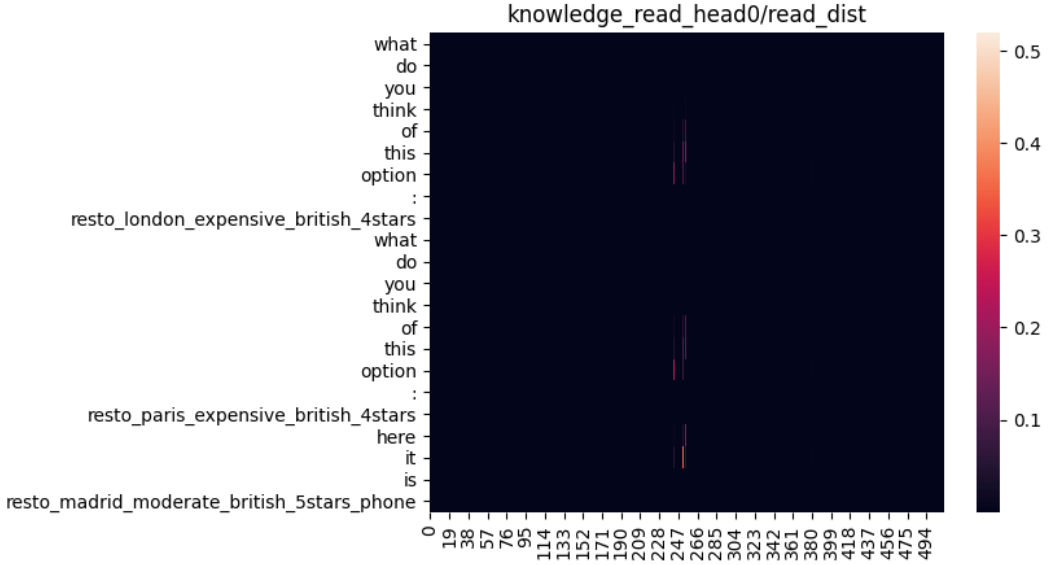


図 5.11: Dialog bAbI データセットでの rsDNC+KM の結果が悪い例における知識メモリからの読み出しのアテンションの重みの可視化結果

第6章　記憶装置付きニューラルネットワークモデルによる構造化知識と演算処理を用いた質問応答

6.1　研究背景と目的

知識グラフを埋め込み空間内に表現し，論理計算を行う質問応答手法として Conditional Theorem Provers [Minervini 20] や Query2Box [Ren 20] が提案されている．しかし，これらの手法では限られた計算のみしか扱うことができない．本研究では，Differentiable Neural Computer (DNC) [Graves 16] と，さらに DNC を改良した rsDNC [Franke 18] と DNC-DMS [Csordás 19] に対して，質問応答タスクにおいて重要な要素である知識利用と演算処理を新たに組み入れ，構造化知識に対する演算処理を含んだ質問文について正しい答えを生成するために任意の演算を用いて能力を向上させることを目指す．

6.2　提案手法

3 つのモデル，DNC, rsDNC, DNC-DMS を構造化知識を保存するための新たなメモリアーキテクチャとプロセッサアーキテクチャを追加することで拡張する．

6.2.1　プロセッサ

図 6.1 は DNC に基づく知識メモリとプロセッサを持つ提案モデルの全体図を示す．提案モデルは control unit (RNN) と 2 つのメモリユニットとプロセッサユニットから成る．

各タイムステップ t で行う処理は以下の通りである：

1. コントローラ (RNN) は入力 $\boldsymbol{x}_t$ と，前タイムステップで文脈メモリ $M^c_{t-1} \in \mathbb{R}^{N\times W}$ から読み出した R 個のベクトルのセット $\boldsymbol{r}^c_{t-1} = [\boldsymbol{r}^{c,1}_{t-1}; ...; \boldsymbol{r}^{c,R}_{t-1}]$ ($\boldsymbol{r}^c_{t-1}$ は $\boldsymbol{r}^{c,1}_{t-1}, ..., \boldsymbol{r}^{c,R}_{t-1}$

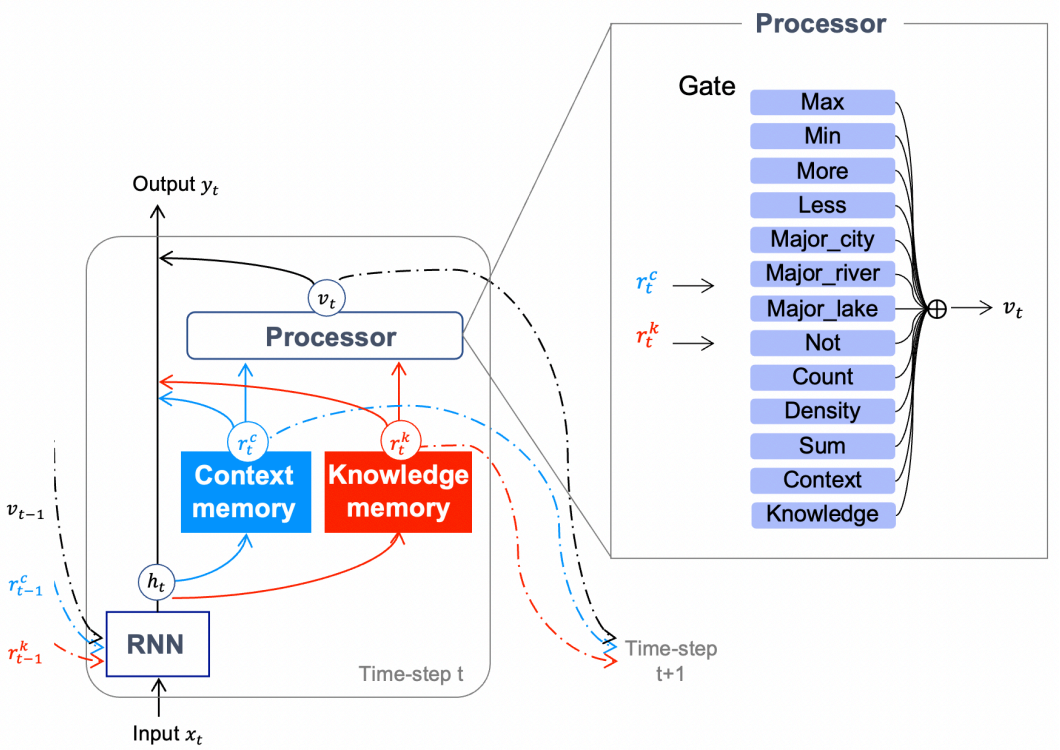


図 6.1: 知識メモリとプロセッサを持つ提案モデルの全体図

の結合) に加えて，前タイムステップで知識メモリ $M^k_{t-1} \in \mathbb{R}^{N\times W}$ から読み出した R 個のベクトルのセット $\boldsymbol{r}^k_{t-1} = [\boldsymbol{r}^{k,1}_{t-1}; ...; \boldsymbol{r}^{k,R}_{t-1}]$，そして前タイムステップのプロセッサの演算結果のベクトル $\boldsymbol{v}_{t-1}$ を受け取る．それから隠れベクトル $\boldsymbol{h}_t$ を出力する．

2. $\boldsymbol{h}_t$ の線形変換により，出力 $\boldsymbol{v}_t = W_y\boldsymbol{h}_t$ と，現タイムステップにおける文脈メモリを制御するためのパラメータを格納したベクトル $\boldsymbol{\xi}_t = W_\xi\boldsymbol{h}_t$，現タイムステップにおける知識メモリを制御するためのベクトル $\boldsymbol{\zeta}_t = W_\zeta\boldsymbol{h}_t$，そして現タイムステップのプロセッサに用いられるゲートベクトル $\boldsymbol{g}_t = W_g\boldsymbol{h}_t$ を得る．

3. $\boldsymbol{\xi}_t$ によって文脈メモリへの書き込みが行われ，メモリの状態が更新される．知識メモリへの書き込みは行われない．

4. プロセッサでの演算処理は $\boldsymbol{g}_t$ と，現タイムステップで文脈メモリから読み出したベクトルを結合した $\boldsymbol{r}^c_t$，現タイムステップで知識メモリから読み出したベクトルを結合した $\boldsymbol{r}^k_t$ を用いて行われる．

5. 最後に，$\boldsymbol{r}_t^c$ と W_r^c をかけて得られるベクトルと，$\boldsymbol{r}_t^k$ と W_r^k をかけて得られるベクトル，及び現タイムステップのプロセッサからのベクトル $\boldsymbol{v}_t$ と W_v をかけて得られるベクトルに $\boldsymbol{v}_t$ を足し，出力 $\boldsymbol{y}_t$ を計算する．

$$\boldsymbol{y}_t = \boldsymbol{v}_t + W_r^c \boldsymbol{r}_t^c + W_r^k \boldsymbol{r}_t^k + W_v \boldsymbol{v}_t$$

read vector $\boldsymbol{r}_t^c$ と $\boldsymbol{r}_t^k$，value vector $\boldsymbol{v}_t$ は次タイムステップの RNN への入力に追加される．

プロセッサ処理

プロセッサユニットにおいて単純な算術演算と論理演算を行うために，13 の演算を設定した:`Max`, `Min`, `More`, `Less`, `Major_city`, `Major_river`, `Major_lake`, `Not`, `Count`, `Density`, `Sum`, `Context`, `Knowledge`. 文脈メモリからの read vector $\boldsymbol{r}^c$ と知識メモリからの read vector $\boldsymbol{r}^k$ は，オリジナルの語彙トークンのリスト，「文脈リスト」と「知識リスト」にそれぞれ変換される．

Superlatives　: `Max` 演算は知識リストを受け取り，その最大値を返す．`Min` 演算は同様に知識リストを受け取り，その最小値を返す．

Comparatives　: `More` 演算は文脈リストと知識リストの両方を受け取り，知識リストの最初の数より大きい数を文脈リスト中から返す．`Less` 演算は同様に文脈リストと知識リストの両方を受け取り，知識リストの最初の数より小さい数を文脈リスト中から返す．`Major_city`, `Major_river`, `Major_lake` 演算は知識リストを受け取り，それぞれ “150,000”, “750”, “5,000” より大きい数を返す．

Negation　: `Not` 演算は文脈リストと知識リストの両方を受け取り，文脈リストから知識リストを引いた差を返す．

Calculation　: `Count` 演算は知識リストを受け取り，その要素数を返す．`Density` 演算は文脈リストと知識リストの両方を受け取り，要素毎に知識リストで割られた文脈リストを返す．`Sum` 演算は知識リストを受け取り，その要素の和を返す．

No operation　: `Context` 演算は文脈リストを受け取り，それ自身を返し，`Knowledge` 演算は知識リストを受け取り，同様にそれ自身を返す．これらの 2 つの演算は上記の演算がどれも適当でない場合のために用意した．

13 の演算の出力は gate vector $\boldsymbol{g}$ のソフトマックス出力と結合し，value vector $\boldsymbol{v}$ を構築する．

6.3　実験

6.3.1　実験設定

すべてのモデルのハイパーパラメータは主に [Graves 16] に基づく；隠れ層サイズ 256 の 1 層 LSTM [Hochreiter 97]，バッチサイズ 2，学習率 1×10^{-4}，メモリ次元 256×64，読み出しヘッド数 4，書き込みヘッド数 1，モメンタム 0.9 の RMSProp オプティマイザ [Tieleman 12]．[Franke 18] に従い rsDNC [Franke 18] のドロップアウト率は 10%とした．HuggingFace[1] bert-base-uncased model を隠れ次元 768 の BERT [Devlin 19] encoder に使用した．数字は [Geva 20] を参考に 1 桁ずつ分割した．5 分割交差検証 [Stone 74] で 5 エポックずつモデルを学習させた．ランダムな初期化の下で各モデルを 3 回実行し，平均の結果を報告する．

知識ベースを用いて知識メモリを構築するために，3 つのオリジナルモデルをメモリ次元 256×64 で，5 分割交差検証で 10 エポックずつ学習させた．他の設定は前述の通りである．DNC, rsDNC, DNC-DMS の top-10 accuracy はそれぞれ 78.90%, 78.39%, 79.63%だった．知識メモリ構築に用いる事前学習モデルと提案手法に用いる学習モデルは 同じ種類である．つまり，DNC を用いて学習した知識メモリが，DNC をベースとした提案モデルで使用される．

6.3.2　データセット

GEO データセット [Zelle 96][2] アメリカの地理に関する 880 の質問文と Prolog 形式のファクトのデータベースを含む．質問文の語彙サイズは 280 である．質問文は superlatives, comparatives, negation, count, density, sum といった calculation を含む. GEO データセットは質問文に対する答えを含んでいなかったため，答えを人手でアノテー

[1] https://github.com/huggingface/transformers
[2] https://cs.stanford.edu/ pliang/software/

表 6.1: 知識ベースの一部

type	alabama	state
capital	alabama	montgomery
population	alabama	3894.0e+3
area	alabama	51.7e+3
located	birmingham	alabama
len	mississippi	3778
traverse	mississippi	minnesota
next to	alabama	tennessee
high point	alabama	cheaha mountain
elevation	cheaha mountain	734
low point	alabama	gulf of mexico

ションを行った. 答えはデータベースからの地理エンティティのリストである. 答えの長さは 0 から 386 エンティティである. 例えば，“Which rivers flow through Alaska ?” という質問は答えがなく，別の質問 “Give me the cities in the U.S. ?” はかなり多くの答えを持つ. 600 サンプルを学習に，280 サンプルをテストに用いた.

GEO のデータベースから RDF 形式の三つ組を作成し，表 6.1 に示す. 1 列目はリレーション，2 列目はサブジェクトエンティティ，3 列目はオブジェクトエンティティである. 三つ組 (`type`, `alabama`, `state`) は `alabama` の `type` は `state` であると意味する. `type` リレーションに対するオブジェクトエンティティとして，5 つのエンティティ：`state`, `city`, `river`, `mountain`, `lake` を我々の KB に追加した. この KB は地理ドメイン内の 11 リレーション，1,275 エンティティ，2,250 トリプルを含む.

さらに, 新たな 500 QA ペアを人手で作成し GEO データセットを拡張した. この拡張したデータセットを “GEO 1380” と呼ぶ. 1,000 サンプルを学習に，380 サンプルをテストに用いた.

6.3.3　結果

表 6.2 に，GEO データセットと GEO 1380 における全てのモデルのテスト 3 実行分の平均 top-1 accuracy (Acc@1) と top-10 accuracy (Acc@10) を示す. 明快化のために，以下の記法を用いる：DNC はオリジナルの DNC モデル [Graves 16]，rsDNC は [Franke 18] により提案された robust and scalable DNC モデル，DNC-DMS [Csordás 19] は DNC

表 6.2: GEO データセットと GEO 1380 における平均 Acc@k

	GEO		GEO 1380	
	Acc@1	Acc@10	Acc@1	Acc@10
BERT (base) [Devlin 19]	18.44	39.73	21.79	45.04
DNC [Graves 16] + BERT	20.93	44.98	24.37	51.16
rsDNC [Franke 18] + BERT	20.76	44.86	26.04	53.41
DNC-DMS [Csordás 19] + BERT	20.20	44.46	25.37	52.53
DNC + BERT + Knowledge Memory	19.53	43.58	25.39	52.82
rsDNC + BERT + Knowledge Memory	20.69	43.98	**26.08**	**53.87**
DNC-DMS + BERT + Knowledge Memory	20.88	**45.40**	25.10	52.23
DNC + BERT + Knowledge Memory + Processor	20.03	43.82	23.76	51.46
rsDNC + BERT + Knowledge Memory + Processor	**21.20**	45.16	25.79	53.28
DNC-DMS + BERT + Knowledge Memory + Processor	19.51	43.58	22.82	50.69

に対して 3 つの修正 (i.e. de-allocation mechanisms, masked content based addressing, sharpness enhancement) を持つ. “+KM” 表記付きのモデルは知識メモリユニットを追加した我々の提案モデルである. “+KM+P” 表記付きのモデルは知識メモリユニットとプロセッサユニットを追加した我々の 2 つ目の提案モデルである.

GEO データセットにおいて，rsDNC+KM+P は Acc@1 で最も良い結果を達成し，DNC-DMS+KM は Acc@10 で最も高くスコアした. GEO 1380 データセットでは，rsDNC+KM は他のモデルを上回った. QA タスクに特化した rsDNC を基にしたモデルは良い結果になる傾向がある. すべてのスコアが GEO 1380 で上がったため，より大きなデータセットはモデルの精度向上に有効である.

表 6.3 に，GEO 1380 データセットにおける全てのモデルの各質問タイプのテスト 3 実行分の平均 Acc@10 を示す.

Simple ：1Hop は “What states border Texas?” のように質問文が 1 つのリレーションを含むことを意味する. 例えば，“What states border states that border states that border states that border Texas?” は 4 つのリレーションを含むため，この例は 4Hop である. 表 **??**に示すように，1Hop は GEO と GEO 1380 の両方の 30%以上を占めるため，このタイプで高くスコアすることが全体の良い結果につながる. 4Hop のポイントが 2Hop のポイントより高いのは，4Hop の方が 2Hop より難しいにも関わらず奇妙に思えるかもしれない. これは 4Hop のサンプル数が少ないせいだと考えられ，マルチホップサンプルを拡張する必要がある.

Superlatives　: Argmax は “What is the largest state?” のように質問文が “largest”, “highest”, “longest” などといった単語を含むことを意味する. Argmax と Argmin において，rsDNC+KM+P が最も高くスコアした. superlatives の他のタイプでは，rsDNC+KM+P はトップスコアに届かなかったが，それでもベストなモデルと比べて遜色ない.

Comparatives　: このタイプは “What states high point are higher than that of Colorado?” のように 3 つの表現：More, Less, Major を扱う. ホップ数が増えると，スコアは下がる.

Negation　: Not は “What state has no rivers?” のように質問文が否定表現を含むことを意味する. 我々の rsDNC+KM は他のモデルを上回った.

Calculation　: このタイプは “How many states are in the USA?” のように 3 つの演算：Count, Density, Sum を扱う. Count と Sum のポイントは他のタイプと比べて比較的高い一方で，Density の結果はとても低い.

Compound　: このタイプは “How many states have a higher point than the highest point of the state with the largest capital city in the U.S.?” のように上記の 4 タイプ：Superlatives, Comparatives, Negation, Calculation を複合した質問文を扱う. このタイプの質問文数は小さい（ほとんどが 1）ため，極端な結果，100.0%が見られる. DNC+KM+P は他と比べても良い結果となった.

6.4　まとめ

3 つの DNC モデル，vanilla DNC, rsDNC, DNC-DMS に知識メモリアーキテクチャとプロセッサアーキテクチャを追加し，実験を行い，背景知識や単純な算術演算と論理演算を必要とする質問応答タスクに対して効果を分析した. 提案した rsDNC+KM+P は GEO データセットにおいて平均 Acc@1 で最も良い結果を達成し，我々の DNC-DMS+KM は平均 Acc@10 で最も高くスコアした. さらに，提案モデル rsDNC+KM は GEO 1380 データセットにおいて平均 Acc@1 と平均 Acc@10 で他のモデルを上回った.

今後の課題では，提案モデルに順次実行，条件分岐，反復といった制御命令を処理するアーキテクチャを加えて改良したい.

表 6.3: GEO 1380 データセットにおける各質問タイプの平均 Acc@10

	BERT (base)	DNC +b	rsDNC +b	DNC-dms +b	DNC +b+km	rsDNC +b+km	DNC-dms +b+km	DNC +b+km+p	rsDNC +b+km+p	DNC-dms +b+km+p
Simple										
1Hop	69.68	74.83	75.98	76.29	77.03	**79.23**	76.93	78.46	77.85	74.59
2Hop	18.95	23.60	25.67	**26.53**	23.77	24.29	23.94	21.96	24.29	22.74
4Hop	17.86	46.43	60.71	59.52	**66.67**	51.19	**66.67**	50.0	52.38	47.62
Superlatives										
Argmax	65.50	67.44	67.83	68.60	68.99	68.22	67.83	67.44	**69.38**	67.05
Argmax-1Hop	81.60	85.28	85.28	85.06	**86.15**	85.93	85.50	81.82	85.71	84.42
Argmax-2Hop	70.41	75.85	**78.91**	76.19	75.51	76.53	75.85	72.45	77.21	77.21
Argmin	66.67	79.63	82.10	79.01	77.16	82.10	79.01	75.93	**83.33**	80.86
Argmin-1Hop	72.11	88.44	89.12	89.80	**92.52**	88.44	89.12	85.71	90.48	83.67
Argmin-2Hop	35.29	43.79	50.33	**56.21**	50.98	47.71	48.37	41.18	54.90	32.68
Comparatives										
Major	59.32	70.62	84.18	77.40	80.23	**86.44**	68.36	79.66	77.40	72.88
Major-1Hop	32.84	42.69	**49.26**	40.72	42.86	46.96	40.23	36.95	45.65	39.90
Major-2Hop	14.29	21.43	22.79	21.43	20.07	21.77	**23.13**	22.45	20.75	21.09
Negation										
Not-1Hop	36.62	52.27	63.64	60.61	60.86	**65.15**	56.31	54.55	60.35	56.06
Calculation										
Count	97.53	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Count-1Hop	80.0	96.0	96.0	85.33	**97.33**	92.0	90.67	88.0	94.67	93.33
Count-2Hop	50.0	**100.0**	**100.0**	66.67	**100.0**	**100.0**	83.33	**100.0**	**100.0**	**100.0**
Density	6.17	8.11	7.68	7.62	8.59	**9.32**	8.05	8.35	8.23	8.41
Sum-1Hop	81.25	81.25	81.25	81.25	81.25	81.25	81.25	81.25	81.25	81.25
Compound										
Argmax-Argmax	83.33	83.33	**100.0**	**100.0**	83.33	83.33	**100.0**	**100.0**	83.33	**100.0**
Argmax-Argmax-3Hop	80.95	**85.71**	**85.71**	**85.71**	**85.71**	80.95	80.95	**85.71**	**85.71**	80.95
Argmax-Argmax-Major-1Hop	83.33	83.33	**100.0**	83.33	83.33	**100.0**	**100.0**	**100.0**	83.33	**100.0**
Argmax-Argmin-1Hop	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Argmax-Argmin-2Hop	50.0	50.0	50.0	50.0	50.0	**66.67**	50.0	50.0	50.0	**66.67**
Argmax-Argmin-Density	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Argmax-Major	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Argmax-Major-1Hop	**100.0**	95.83	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Argmax-Major-2Hop	47.62	33.33	38.10	38.10	28.57	33.33	38.10	**57.14**	33.33	42.86
Argmax-Not-1Hop	66.67	77.78	77.78	66.67	66.67	**88.89**	66.67	66.67	66.67	77.78
Argmax-Density	87.50	83.33	91.67	91.67	95.83	75.0	87.50	**100.0**	87.50	87.50
Argmax-Density-1Hop	73.81	78.57	80.95	76.19	**85.71**	73.81	76.19	**85.71**	78.57	78.57
Argmin-Argmax-1Hop	**100.0**	83.33	83.33	83.33	50.0	83.33	50.0	50.0	83.33	83.33
Argmin-Density	87.50	91.67	**100.0**	95.83	95.83	91.67	95.83	**100.0**	91.67	87.50
Not-Major-1Hop	21.37	58.97	**74.36**	65.81	56.41	41.88	65.81	56.41	69.23	48.72
Count-Argmax	66.67	**100.0**	88.89	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Count-Argmax-2Hop	50.0	83.33	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Count-More	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Count-More-Argmax-Argmax-2Hop	55.56	88.89	66.67	88.89	77.78	77.78	88.89	**100.0**	66.67	88.89
Count-Less-2Hop	50.0	33.33	33.33	50.0	**66.67**	50.0	50.0	50.0	50.0	50.0
Cout-Not	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Density-Argmax	54.39	57.89	57.89	57.89	59.65	60.53	61.40	**63.16**	59.65	59.65
Density-Argmax-1Hop	10.79	13.97	12.38	12.38	13.02	**14.60**	**14.60**	3.81	12.06	11.43
Density-Argmin	83.33	83.33	83.33	83.33	83.33	83.33	83.33	83.33	83.33	83.33
All	45.04	51.16	53.41	52.53	52.82	**53.87**	52.23	51.46	53.28	50.69

表 6.4: 各質問タイプ数

		GEO	GEO 1380
Simple	1Hop	94	145
	2Hop	10	11
	4Hop	1	1
Superlatives	Argmax	20	27
	Argmax-1Hop	43	49
	Argmax-2Hop	21	24
	Argmin	10	16
	Argmin-1Hop	13	15
	Argmin-2Hop	4	5
Comparatives	Major	3	5
	Major-1Hop	8	12
	Major-2Hop	2	3
Negation	Not-1Hop	2	3
Calculation	Count	8	11
	Count-1Hop	6	11
	Count-2Hop	1	1
	Density	3	4
	Sum-1Hop	2	2
Compound	Argmax-Argmax	1	1
	Argmax-Argmax-3Hop	1	1
	Argmax-Argmax-Major-1Hop	1	1
	Argmax-Argmin-1Hop	1	2
	Argmax-Argmin-2Hop	1	1
	Argmax-Argmin-Density	1	1
	Argmax-Major	1	1
	Argmax-Major-1Hop	3	3
	Argmax-Major-2Hop	1	1
	Argmax-Not-1Hop	1	1
	Argmax-Density	3	3
	Argmax-Density-1Hop	3	3
	Argmin-Argmax-1Hop	1	1
	Argmin-Density	4	4
	Not-Major-1Hop	0	1
	Count-Argmax	1	1
	Count-Argmax-2Hop	1	1
	Count-More	0	1
	Count-More-Argmax-Argmax-2Hop	1	1
	Count-Less-2Hop	1	1
	Count-Not	1	1
	Density-Argmax	1	2
	Density-Argmax-1Hop	1	1
	Density-Argmin	0	1
	All	280	380

第7章　結論

本研究では，データ主導によるクエリの生成および，記憶装置付きニューラルネットワークモデルによる文脈と構造化知識を用いた対話システムの構築ならびに質問応答を行った．計算機の原理であるチューリングマシンをニューラルネットワークで模したDNCに，Transformerと知識利用・演算処理を行うアーキテクチャを組み込んだモデルを構築し，知識と演算を必要とする自然言語処理タスクを扱えるか検証を行った．

第2章では，記憶装置付きニューラルネットワークモデルについて述べた．

第3章では，自然言語のSPARQLクエリ変換によるメニューオントロジーへのアクセス手法について述べた．KNPによる述語項構造・係り受け解析と，データ主導によるRDFトリプルの変形を用いた，自然言語からSPARQLクエリへ変換する手法を提案した．エンティティを知識側の語彙と対応付け，エンティティ間のリレーションを知識から獲得することによって，SPARQLクエリが完成されれば答えは保証される．また，基本型，Degree型，Count型，List型の4つの質問タイプを設定し，それぞれの質問タイプの自然言語質問文及び，より一般化した，それらを組み合わせた質問文に対してそれぞれ適切なSPARQLクエリを生成することができた．質問タイプを設定することにより，正確なクエリを容易に生成し，自然言語文が意図する質問に回答することを可能にした．

第4章では，文脈を考慮した日本語の注文対話データセットを作成し，ニューラルネットワークモデルを適用して実験を行なった結果，平均テスト誤り率として低い値を得ることができた．

第5章では，また，DNCを拡張し，文脈を捉えつつ構造化知識を活用した対話のモデルを提案した．CSQAデータセットを用いて提案モデルとオリジナルのDNCモデルの実験を行ない，誤り率を比較した結果，10項目中5項目で結果は上がったが，全体の結果はDNCモデルより下がってしまった．原因の一つとして，知識メモリのサイズが約900万のトリプルを記録するには512×128ではまだ小さすぎたことが考えられる．今回の実験では学習イテレーション数が10万回だったので，学習回数を増やして結果をまた考察したい．3つのDNCモデル，vanilla DNC, rsDNC, DNC-DMSに

知識メモリを追加した．背景知識を必要とする対話タスクにおける提案手法の効果について分析した．提案モデル，DNC+KM, rsDNC+KM, DNC-MD+KM が the (6) dialog bAbI tasks dataset の full dialog tasks でオリジナルモデルを上回った．特に，各モデルは KB ファクトを要求するタスクで，それぞれおよそ 14%, 20%, 7%向上した．Movie Dialog dataset では，rsDNC+KM, DNC-DMS+KM がオリジナルモデルより良い結果になった．

第 6 章では，3 つの DNC モデル，vanilla DNC, rsDNC, DNC-DMS に知識メモリアーキテクチャとプロセッサアーキテクチャを追加し，実験を行い，背景知識や単純な算術演算と論理演算を必要とする質問応答タスクに対して効果を分析した. アメリカの地理に関する知識と演算処理を要求する GEO データセットとそれを拡張したデータセットにおいて，BERT をファインチューニングした結果より，DNC モデルをベースに BERT と知識を扱うメモリと演算を行うユニットを追加した提案手法の結果がすべて平均 top-1 accuracy と平均 top-10 accuracy の両方で向上した．さらに，rsDNC を改良した提案手法は GEO データセットにおける平均 top-1 accuracy で最も良い結果を達成し，総合的に DNC より高いスコアを得た.

以上の研究により，構造化知識を質問応答や対話に取り入れることの有用性ならびに記憶装置付きニューラルネットワークモデルに BERT と知識と演算を扱うアーキテクチャを追加した手法の結果が BERT のみを用いた結果よりも向上することが確かめられた.

謝辞

本研究を進めるにあたり，指導教官として常日頃より熱心にご指導くださり，温かく支え続けてくださった本学基幹研究院自然科学系の小林一郎教授に深謝いたします．また，審査をご担当くださいました本学基幹研究院自然科学系の小口正人教授，戸次大介准教授，浅井健一教授，伊藤貴之教授に厚く感謝申し上げます．

金子晃名誉教授には，計算機の使用に関してご指導いただき，実験環境の整備にご尽力いただきました．Lis Kanashiro Pereira 特任講師には，研究について的確なご指導をいただきました．心より感謝いたします．

慶應義塾大学の山口高平教授，ならびに森田武史准教授には，貴重なデータを提供していただくとともに数多くのご教授をいただき，成蹊大学の中野有紀子教授には，大変有益なご助言をいただきました．深く感謝いたします．

最後に，今まで研究以外の面においても支えてくださった小林研究室のメンバー，友人，そしていつもそばで温かく見守ってくれた家族に心から感謝の意を表します．

関連図書

[Bahdanau 14] Bahdanau, D., Cho, K., and Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate (2014)

[Bengio 94] Bengio, Y., Simard, P., and Frasconi, P.: Learning Long-Term Dependencies with Gradient Descent is Difficult, *Trans. Neur. Netw.*, Vol. 5, No. 2, p. 157–166 (1994)

[Bordes 13] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O.: Translating Embeddings for Modeling Multi-relational Data, in Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. eds., *Advances in Neural Information Processing Systems 26*, pp. 2787–2795, Curran Associates, Inc. (2013)

[Bordes 16] Bordes, A. and Weston, J.: Learning End-to-End Goal-Oriented Dialog, *CoRR*, Vol. abs/1605.07683, (2016)

[Bosselut 19] Bosselut, A., Rashkin, H., Sap, M., Malaviya, C., Çelikyilmaz, A., and Choi, Y.: COMET: Commonsense Transformers for Automatic Knowledge Graph Construction, *CoRR*, Vol. abs/1906.05317, (2019)

[Csordás 19] Csordás, R. and Schmidhuber, J.: Improving Differentiable Neural Computers Through Memory Masking, De-allocation, and Link Distribution Sharpness Control, *CoRR*, Vol. abs/1904.10278, (2019)

[Dai 19] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R.: Transformer-XL: Attentive Language Models beyond a Fixed-Length Context, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy (2019), Association for Computational Linguistics

[Devlin 19] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K.: 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), pp. 4171–4186, Minneapolis, Minnesota (2019), Association for Computational Linguistics

[Dinan 18] Dinan, E., Roller, S., Shuster, K., Fan, A., Auli, M., and Weston, J.: Wizard of Wikipedia: Knowledge-Powered Conversational agents, *CoRR*, Vol. abs/1811.01241, (2018)

[Dodge 16] Dodge, J., Gane, A., Zhang, X., Bordes, A., Chopra, S., Miller, A. H., Szlam, A., and Weston, J.: Evaluating prerequisite qualities for learning end-to-end dialog systems, Vol. abs/1511.06931, (2016)

[Franke 18] Franke, J., Niehues, J., and Waibel, A.: Robust and Scalable Differentiable Neural Computer for Question Answering, *CoRR*, Vol. abs/1807.02658, (2018)

[Geva 20] Geva, M., Gupta, A., and Berant, J.: Injecting Numerical Reasoning Skills into Language Models, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 946–958, Online (2020), Association for Computational Linguistics

[Graves 14] Graves, A., Wayne, G., and Danihelka, I.: Neural Turing Machines (2014), cite arxiv:1410.5401

[Graves 16] Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., and Hassabis, D.: Hybrid computing using a neural network with dynamic external memory, *Nature*, Vol. 538, No. 7626, pp. 471–476 (2016)

[Guo 18] Guo, D., Tang, D., Duan, N., Zhou, M., and Yin, J.: Dialog-to-Action: Conversational Question Answering Over a Large-Scale Knowledge Base, in Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. eds., *Advances in Neural Information Processing Systems 31*, pp. 2942–2951, Curran Associates, Inc. (2018)

[Hochreiter 97] Hochreiter, S. and Schmidhuber, J.: Long short-term memory, *Neural computation*, Vol. 9, No. 8, pp. 1735–1780 (1997)

[Kumar 15] Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Gulrajani, I., and Socher, R.: Ask Me Anything: Dynamic Memory Networks for Natural Language Processing, *CoRR*, Vol. abs/1506.07285, (2015)

[Lewis 19] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L.: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, *CoRR*, Vol. abs/1910.13461, (2019)

[Luong 15] Luong, T., Pham, H., and Manning, C. D.: Effective Approaches to Attention-based Neural Machine Translation, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, Lisbon, Portugal (2015), Association for Computational Linguistics

[Martins 22] Martins, P. H., Marinho, Z., and Martins, A.: ∞-former: Infinite Memory Transformer, in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5468–5485, Dublin, Ireland (2022), Association for Computational Linguistics

[Miller 95] Miller, G. A.: WordNet: A Lexical Database for English, *COMMUNICATIONS OF THE ACM*, Vol. 38, pp. 39–41 (1995)

[Miller 16] Miller, A. H., Fisch, A., Dodge, J., Karimi, A., Bordes, A., and Weston, J.: Key-Value Memory Networks for Directly Reading Documents, *CoRR*, Vol. abs/1606.03126, (2016)

[Minervini 20] Minervini, P., Riedel, S., Stenetorp, P., Grefenstette, E., and Rocktäschel, T.: Learning Reasoning Strategies in End-to-End Differentiable Proving, *CoRR*, Vol. abs/2007.06477, (2020)

[Neumann 45] Neumann, J. v.: First Draft of a Report on the EDVAC, Technical report (1945)

[Pennington 14] Pennington, J., Socher, R., and Manning, C. D.: Glove: Global Vectors for Word Representation., in *EMNLP*, Vol. 14, pp. 1532–1543 (2014)

[Rae 19] Rae, J. W., Potapenko, A., Jayakumar, S. M., and Lillicrap, T. P.: Compressive Transformers for Long-Range Sequence Modelling, *CoRR*, Vol. abs/1911.05507, (2019)

[Raffel 19] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J.: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, *CoRR*, Vol. abs/1910.10683, (2019)

[Ren 20] Ren, H., Hu, W., and Leskovec, J.: Query2box: Reasoning over Knowledge Graphs in Vector Space using Box Embeddings, *CoRR*, Vol. abs/2002.05969, (2020)

[Saha 18] Saha, A., Pahuja, V., Khapra, M. M., Sankaranarayanan, K., and Chandar, S.: Complex Sequential Question Answering: Towards Learning to Converse Over Linked Question Answer Pairs with a Knowledge Graph (2018)

[Siegelmann 95] Siegelmann, H. and Sontag, E.: On the Computational Power of Neural Nets, *J. Comput. Syst. Sci.*, Vol. 50, No. 1, p. 132–150 (1995)

[Stone 74] Stone, M.: Cross-validatory choice and assessment of statistical predictions, *Roy. Stat. Soc.*, Vol. 36, pp. 111–147 (1974)

[Sukhbaatar 15] Sukhbaatar, S., szlam, a., Weston, J., and Fergus, R.: End-To-End Memory Networks, in Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. eds., *Advances in Neural Information Processing Systems 28*, pp. 2440–2448, Curran Associates, Inc. (2015)

[Sutskever 14] Sutskever, I., Vinyals, O., and Le, Q. V.: Sequence to Sequence Learning with Neural Networks, *CoRR*, Vol. abs/1409.3215, (2014)

[Tieleman 12] Tieleman, T. and Hinton, G.: Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude (2012)

[Vaswani 17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I.: Attention is All you Need, in Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. eds., *Advances in Neural Information Processing Systems*, Vol. 30, Curran Associates, Inc. (2017)

[Wang 07] Wang, C., Xiong, M., Zhou, Q., and Yu, Y.: PANTO: A Portable Natural Language Interface to Ontologies., in Franconi, E., Kifer, M., and May, W. eds., *ESWC*, Vol. 4519 of *Lecture Notes in Computer Science*, pp. 473–487, Springer (2007)

[Weston 14] Weston, J., Chopra, S., and Bordes, A.: Memory Networks (2014), cite arxiv:1410.3916

[Weston 15] Weston, J., Bordes, A., Chopra, S., Rush, A. M., Merriënboer, van B., Joulin, A., and Mikolov, T.: Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks (2015), cite arxiv:1502.05698

[Young 17] Young, T., Cambria, E., Chaturvedi, I., Huang, M., Zhou, H., and Biswas, S.: Augmenting End-to-End Dialog Systems with Commonsense Knowledge, *CoRR*, Vol. abs/1709.05453, (2017), Withdrawn.

[Zelle 96] Zelle, J. M. and Mooney, R. J.: Learning to Parse Database Queries Using Inductive Logic Programming, in *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, p. 1050–1055, AAAI Press (1996)

[Zou 14] Zou, L., Huang, R., Wang, H., Yu, J. X., He, W., and Zhao, D.: Natural language question answering over RDF: a graph data driven approach, in *SIGMOD Conference* (2014)

[鈴木 15] 鈴木遼司, 三輪誠, 佐々木裕 F 交通オントロジーを対象とした質問文の SPARQL クエリ変換 (2015)

付 録A　研究業績

原著論文

- Murayama, Y., Kobayashi, I.: Towards Question Answering with Multi-hop Reasoning and Calculation over Knowledge using a Neural Network Model with External Memories. Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII) , 2022.

国際会議発表

- Murayama, Y., Kobayashi, I., Morita, T., Nakano, Y., and Yamaguchi, T.: Development of an Access Method to Menu Ontology Data through Converting Natural Language into SPARQL Queries. The 8th Joint International Semantic Technology Conference (JIST2018) Workshop, 2018.

- Murayama, Y., Pereira,L.K., and Kobayashi, I.: Dialogue over Context and Structured Knowledge using a Neural Network Model with External Memories. The 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing (AACL- IJCNLP2020) Knowledgeable NLP: the First Workshop on Integrating Structured Knowledge and Neural Networks for NLP, 2020.

- Murayama, Y., Kobayashi, I.: Towards Question Answering with Multi-hop Reasoning over Knowledge using a Neural Network Model with External Memories. Joint 12th International Conference on Soft Computing and Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCISISIS), 2022.

国内学会発表

- 村山友理，小林一郎，森田武史，中野有紀子，山口高平，自然言語の SPARQL クエリ変換に基づく大規模知識へのアクセス手法の基礎的検討，言語処理学会第 24 回年次大会，2018 年 3 月.

- 村山友理，小林一郎，森田武史，中野有紀子，山口高平，自然言語の SPARQL クエリ変換に基づく構造化知識へのアクセス手法の開発，人工知能学会全国大会 (第 32 回)，2018 年 6 月.
- 村山友理，小林一郎，森田武史，中野有紀子，山口高平，文脈を考慮した対話のニューラルネットワークモデルの日本語注文対話への適用，人工知能学会全国大会 (第 33 回)，2019 年 6 月.
- 村山友理，小林一郎，記憶装置付きニューラルネットワークモデルによる文脈と知識を用いた対話システムの構築，NLP 若手の会（YANS）第 14 回シンポジウム（2019），2019 年 8 月.
- 村山友理，小林一郎，記憶装置付きニューラルネットワークモデルによる文脈と構造化知識を用いた対話への取り組み，言語処理学会第 26 回年次大会，2020 年 3 月.
- 村山友理，小林一郎，記憶装置付きニューラルネットワークモデルによる文脈と構造化知識を用いた対話，人工知能学会全国大会 (第 34 回)，2020 年 6 月.
- 村山友理，小林一郎，テンソル表現による知識を用いて論理計算を行う質問応答に向けて，NLP 若手の会（YANS）第 16 回シンポジウム（2021），2021 年 8 月.
- 村山友理，小林一郎，記憶装置付きニューラルネットワークモデルによる構造化知識と演算処理を用いた質問応答，言語処理学会第 29 回年次大会，2023 年 3 月.（発表予定）