Neural DTS に対する型検査アルゴリズムの実装の試み

飯沼瑞稀　高橋優太　田上青空　戸次大介
お茶の水女子大学
{g1920504,takahashi.yuta,tagami.sora,bekki}@is.ocha.ac.jp

概要

現代的な記号推論システムである依存型理論の応用の一つに，自然言語の意味論がある．依存型理論による自然言語意味論は含意関係認識タスクに応用されており，その含意関係認識システムの基礎には型検査アルゴリズムがある．こうした型検査アルゴリズムは，型理論のような記号推論からなるシステムに与えられてきたものであるが，近年，依存型理論による自然言語意味論にはニューラルネットを埋め込んだシステムが提案されている．本研究では第一に，そうしたシステムの一つである Neural DTS に対して型検査アルゴリズムを定式化する．第二に，WordNet から抽出したデータセットを用いて，埋め込まれたニューラルネットの精度を型検査の出力を参照して調べる．

1　はじめに

記号推論は，ニューラルネットによる推論・予測としばしば対比されるが，そうした記号推論を行うための現代的なシステムの一つとして型理論 (type theory) と呼ばれる論理システムがある．型理論はもともと，数学の証明を厳密に表現すること，言い換えれば数学を形式化することを目的として考案された [1]．その一方で，命題と型の同型性および証明とプログラムの同型性を示すカリー・ハワード対応 (Curry-Howard correspondence) を背景に，現代的な型理論はそれが考案された当初からプログラミング言語とも見なされている．

型理論は計算機科学と論理学の共通部分として研究されてきたが，型理論の中でも特に依存型理論 (dependent type theory) は，一般化量化子や照応といった自然言語の意味論に早くから応用されてきた [2]．上述のように，依存型理論は論理システムともプログラミング言語とも見なすことができる．そのため，自然言語の文がもつ複雑な意味構造を解釈できる論理システムとして依存型理論を用いるだけでなく，その計算論的性質もまた自然言語の意味論へと応用されてきた．

こうして依存型理論は自然言語の意味論における理論的選択肢の一つとなり，現在における主要なアプローチとしては，例えば依存型意味論 (Dependent Type Semantics, DTS) [3], the formal semantics using Modern Type Theories [4], Type Theory with Records (TTR) [5] が挙げられる．さらに，依存型理論による自然言語意味論に対する実装も与えられ，こうした実装は，含意関係認識タスクへの手法を提供している [6, 7, 8, 9, 10]．ここでの実装において重要になるのは，型検査アルゴリズム (type checking algorithm) という，プログラム t および型 A が与えられたときに t が型 A をもつかどうかを判定するアルゴリズムである．カリー・ハワード対応を踏まえると，このアルゴリズムは，証明 t と命題 A が与えられたときに t が A の証明であるかどうかを判定するアルゴリズムともいえる．上述の含意関係認識の手法においては含意判定の場合にその証明が構成されるため，含意関係認識システムがつくった証明が，求めている命題の証明になっているかどうか検査する必要がある．この点で，型検査アルゴリズムは含意関係認識システムの基礎になっている．

以上のように，型検査アルゴリズムは，依存型理論を用いた自然言語意味論の基盤であるといえる．従来このアルゴリズムは，もっぱら記号推論システムであるような型理論に対して与えられてきた．その一方で，依存型理論を用いた自然言語意味論には，ニューラルネットを埋め込んだシステムが考案されている [11, 12]．その中でも，[13, 14] において基本構想が説明された Neural DTS は，自然言語の複雑な意味現象を説明できる従来の DTS の記号推論を維持しつつ，「犬である」といった述語をニューラル分類器に置き換えることによって，ニューラルネットによる予測を併用する．

本研究では Neural DTS の実装に向けた一歩として，まず，DTS の語彙が含む述語を多層パーセプト

ロン (MLP) による分類器で置き換え，得られたシステムに対する型検査アルゴリズムを実装する．前者および後者の実装にはともに Haskell を用い，特に MLP の実装には PyTorch の Haskell バインディングである hasktorch を用いる．次に，WordNet から抽出したデータセットを用いて，型検査アルゴリズムに埋め込むニューラルネットの精度を型検査の出力を参照して調べる．

2 先行研究

Type theory with records (TTR) は，DTS と同じく依存型理論による自然言語意味論のフレームワークであるが，TTR の中にニューラルネットを埋め込んだシステムがすでに考案されている [11, 12]．Cooper [11] は，外界の出来事 (events) や対象を分類するためにエージェントが学習するような型を TTR が提供できることを示すために，TTR に属する型からニューラルネット上の出来事の型への写像を与えている．さらに，ここでの写像に対する Python による実装も与えられている．その一方で，こうしたニューラルネット上の出来事の型とニューラル分類器の接続は示唆にとどまっており，本研究が扱うような学習済み分類器を TTR へ埋め込むことは行われていない．

Larsson [12] は，知覚データに関する学習可能な分類器が埋め込まれた TTR を提案している．ここでの知覚データに関する分類は，DTS における命題 $P_i(e_j)$ における分類に対応しており，このバージョンの TTR は Neural DTS と共通点を多くもつ．しかし，[12] の目的は，こうした TTR の実装というよりはそれの自然言語の意味論への応用であり，型検査アルゴリズムをどう定式化するかという問題については論じられていない．

3 提案手法

本研究が提案する手法は，ニューラル分類器をパラメータとする型検査アルゴリズムを定義しておき，型検査を行う際には，学習済みの分類器をそのアルゴリズムに渡すというものである．以下ではまず，ニューラル分類器をパラメータとする型検査アルゴリズムを定義する (§ 3.1)．次に，本研究の型検査アルゴリズムに埋め込まれる MLP を hasktorch により実装する (§ 3.2)．そして，以上を組み合わせて，MLP による分類器を呼び出すことのできる，Neural DTS の型検査アルゴリズムを定義する (§ 3.3)．

3.1 ニューラル分類器をパラメータとする型検査アルゴリズム

ニューラル分類器をパラメータとする型検査アルゴリズムを定義するために，[15] において Haskell によって実装された依存型理論の型検査アルゴリズムを，以下の 2 点で改訂する．簡潔さのために，以下では DTS の語彙に含まれる述語（すなわち 1 項関係）を例にとって説明するが，§ 4 で見るように 2 項関係（および任意の n 項関係）をとることもできることに注意されたい．

(i) DTS の語彙に含まれる述語 P_i および名前 e_j から成る型 $P_i(e_j)$ へとアルゴリズムを拡張する．
(ii) 型検査アルゴリズムを与える関数 typeChk に，ニューラル分類器をパラメータとして与えて再定義する

まず (i) について説明する．DTS の語彙には述語と名前が含まれ，名前の型は **entity** であり，述語の型は **entity** → type である．DTS では命題の型は type として表されるため，述語の型は，名前から命題への関数ということになる．そして，Neural DTS に対する型検査には，述語を名前に適用して得られる命題 $P_i(e_j)$ についての型検査が新たなケースとして加わる．例えば，述語が **dog** で名前が john のとき，得られる命題は **dog**(john) となり，これは「John は犬である」という命題に対応する．Neural DTS の型検査に $P_i(e_j)$ といった命題のケースを加えるのは，典型的にはこうした命題がニューラル分類器による予測の表現になるからである．

命題 $P_i(e_j)$ についての型検査では，大まかに言えば次のような判定がなされる．文脈 Γ および命題 $P_i(e_j)$ が与えられたとき，Γ のもとで証拠 w_{ij} が $P_i(e_j)$ の証明であるかどうか（つまり，Γ のもとで w_{ij} が型 $P_i(e_j)$ をもつかどうか）が判定される．ただし，w_{ij} は命題 $P_i(e_j)$ のために用意する定項である．以下では，Γ のもとで証拠 w_{ij} が $P_i(e_j)$ の証明であるかどうかという問いを次のように記号化する：

$$\Gamma \vdash w_{ij} : P_i(e_j)\ ?$$

次に (ii) について説明する．ニューラル分類器によって述語を置き換えるという Neural DTS の基本構想に基づき，命題 $P_i(e_j)$ についての型検査は，述語 P_i と名前 e_j の one-hot 表現をそれぞれニューラル分類器に入力した結果を用いて行う．結果となる出力が閾値を超えれば名前 e_j は述語 P_i をみたす

といえるから，命題 $P_i(e_j)$ は証明 w_{ij} をもつとみなし，$\Gamma \vdash w_{ij} : P_i(e_j)$? という問いには YES と答える．閾値を超えなければ e_j は P_i をみたさないということであるから，命題 $P_i(e_j)$ はそもそも証明をもたず，それゆえこの問いには NO と答える．

以上の説明が示すように，本研究が提案する Neural DTS の型検査では，その途中でニューラル分類器が呼び出される．このことを可能にするため，もともとの型検査関数 `typeChk` にニューラル分類器をパラメータとして与える．

```
typeChk :: (Tensor -> Tensor -> Tensor) ->
             Int ->
             Context -> TermChk -> Type ->
             Result ()
```

ここでの分類器は，述語の one-hot 表現と名前の one-hot 表現を入力としてとる分類器であるため，パラメータの型は以下のようになっている：

$$\texttt{Tensor} -> \texttt{Tensor} -> \texttt{Tensor}$$

そして，上でインフォーマルに述べた，型検査 $\Gamma \vdash w_{ij} : P_i(e_j)$? に答えを与えるためのアルゴリズムを，`typeChk` のパターンマッチングによる定義の中で新たに設けるケースにて実装する．

このようにして，[15] の型検査アルゴリズムを命題 $P_i(e_j)$ のケースまで拡張しつつ，型検査関数 `typeChk` にニューラル分類器のパラメータを加えることで，MLP を埋め込める型検査アルゴリズムを定義する．

3.2 hasktorch による MLP の実装

§ 3.1 で定義した型検査アルゴリズムと組み合わせるために，単純な MLP によるニューラル分類器を実装する．ここでの実装には，hasktorch の Github リポジトリにある，MLP による排他的論理和の実装を参考にしている [16]．

本研究において実装する MLP は以下の標準的な構造をもつ：

埋め込み層　述語の one-hot 表現および名前の one-hot 表現をそれぞれ線形層でベクトル空間へ埋め込み，得られた分散表現を結合する．

隠れ層　結合した分散表現を入力としてとる線形層を中間層として一つ用意する．

出力層　分類器の出力とするために次元を調整した線形層を出力層とする．

活性化関数としてはシグモイド関数を用いる．また，損失関数としてはバイナリ交差エントロピー誤差を用いる．

以上の構造をもつ MLP を，微分可能プログラムを提供する hasktorch の `Parameterized` 型クラスを用いて実装する．また，次節において説明する，この MLP による分類器の学習には，初期化関数を提供する `Randomizable` 型クラスを用いている．

3.3 型検査アルゴリズムへの MLP の埋め込み

本研究による提案手法の最後のステップとして，§ 3.1 で定義した型検査関数 `typeChk` に，§ 3.2 で構成した MLP を埋め込むことで，目標となる型検査アルゴリズムを与える関数 `neuralTC` を定義する．この関数の型は以下のようになる：

```
neuralTC :: Context -> TermChk -> Type -> IO ()
```

関数 `neuralTC` は，文脈 Γ，プログラム t そして型 A が与えられると，Γ のもとで t が A を型としてもつかどうかを判定する．この点は，依存型理論に対する通常の型検査アルゴリズムと変わらないが，型検査の途中でニューラル分類器を呼び出して，その出力を用いて型検査 $\Gamma \vdash w_{ij} : P_i(e_j)$? を判定する点で通常のものとは異なっている．

文脈 Γ，プログラム t そして型 A を入力として `neuralTC` に与えると，その計算は次のように進む．

学習フェーズ　以下の手順を 1 イテレーションとする．

1. データセットの読み込み：数値のリストとして表現されたデータセットから，指定された数だけランダムに要素を選び出す．次に，それをテンソルに変換してミニバッチを構成し，教師データおよび訓練データを作成する．
2. モデルの更新：訓練データに対する出力となる予測値と，教師データの正解ラベルの間の誤差を計算し，hasktorch が提供する関数 `runStep` により分類器モデルを更新する．本研究における分類器は命題 $P_i(e_j)$ が成り立つかどうかを判定するものであるため，ここでの分類は 2 値分類となり，正解ラベルは 1 もしくは 0 である．

指定された回数の学習を繰り返したら，変数 `trained` に学習済みの分類器モデルを格納する．

型検査フェーズ　この段階ではまず，先のフェーズにおいて得られた学習済みモデル `trained` を，型 `Tensor -> Tensor -> Tensor` をもつニュー

ラル分類器 cls に変換する．そして，入力として与えられている文脈 Γ，プログラム t および型 A とともにこの cls を，§ 3.1 において定義した関数 typeChk に与えて，型検査を行う．

以上が，本研究において実装する Neural DTS のための型検査アルゴリズム neuralTC の概略である．

4 実験

WordNet のデータベースを用いて，2 つの entity の間にある関係が成立するかどうかを，それ以外のデータから予測する実験を行った．これは例えば「猫には尻尾がある」といった既知の事実に対する常識的推論や，関係ネットワークにおけるリンク予測として見ることができる．データセットとして，WordNet から抽出した 40,943 個の entity と 18 個の relation に関する 141,442 個の学習用関係トリプレット，5,000 個のテスト用関係トリプレット，5,000 個の検証用関係トリプレットを用いた．このデータセットは [17] において作成されたものである[1)]．例えば，(drill, has part, chuck) などのトリプレットが学習データに含まれており，これらを正解データとして使用した．不正解データとして，正解データと同じ数だけ，正解データに含まれない関係トリプレットをランダムに作成した．実験として，作成した neuralTC がトリプレット上の関係を正しく予測できるかという 2 値分類を行った．typeChk では，

1. スコアが閾値以上で，かつ型検査が正常な場合：2 つの entity が関係を満たしている
2. スコアが閾値未満で，かつ型検査が正常な場合：関係を満たしていない
3. 型の不整合

の 3 種の返り値があり，このうち 1. を正解の予測データ，2. を不正解の予測データとして扱った．学習反復数 50, 2 値分類の閾値が 0.7 のとき，accuracy = 0.7536, precision = 0.8064282, recall = 0.6674, f1 = 0.73035675 となった．学習反復数を増やしたり，学習パラメータを最適化させたりすることで，より精度は上がると考えられる．

5 おわりに

本研究ではまず，Neural DTS に対する型検査アルゴリズムを定式化した．次に，このアルゴリズムの中で呼び出されるニューラル分類器となる MLP を実装した．今後は，学習反復数の増加やパラメータの最適化によって精度の向上を図るとともに，parser と接続し，DTS の基本述語の集合を入力としてニューラル分類器の学習を行えるようにすることで，Neural DTS の実装を目指す．

謝辞

本研究は，JST CREST JPMJCR20D2 の支援を受けたものである．

参考文献

[1] Per Martin-Löf. An intuitionistic theory of types. In G. Sambin and Jan M. Smith, editors, **Twenty-five years of constructive type theory**, Vol. 36 of **Oxford Logic Guides**, pp. 127–172. Clarendon Press, 1998.

[2] G. Sundholm. Constructive generalized quantifiers. **Synthese**, Vol. 79, pp. 1–12, 1989.

[3] Daisuke Bekki and Koji Mineshima. Context-passing and underspecification in dependent type semantics. In Stergios Chatzikyriakidis and Zhaohui Luo, editors, **Modern Perspectives in Type-Theoretical Semantics**, pp. 11–41. Springer International Publishing, Cham, 2017.

[4] Stergios Chatzikyriakidis and Zhaohui Luo. On the interpretation of common nouns: Types versus predicates. In Stergios Chatzikyriakidis and Zhaohui Luo, editors, **Modern Perspectives in Type-Theoretical Semantics**, pp. 43–70. Springer International Publishing, Cham, 2017.

[5] Robin Cooper. Adapting type theory with records for natural language semantics. In Stergios Chatzikyriakidis and Zhaohui Luo, editors, **Modern Perspectives in Type-Theoretical Semantics**, pp. 71–94. Springer International Publishing, Cham, 2017.

[6] Stergios Chatzikyriakidis and Zhaohui Luo. Natural language inference in coq. **J. Log. Lang. Inf.**, Vol. 23, No. 4, pp. 441–480, 2014.

[7] Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. Higher-order logical inference with compositional semantics. In L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, and Y. Marton, editors, **Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015**, pp. 2055–2061. The Association for Computational Linguistics, 2015.

[8] Stergios Chatzikyriakidis and Zhaohui Luo. Proof assistants for natural language semantics. In M. Amblard, P. de Groote, S. Pogodalla, and C. Retoré, editors, **Logical Aspects of Computational Linguistics. Celebrating 20 Years of LACL (1996-2016) - 9th International Conference, LACL 2016, Nancy, France, December 5-7, 2016, Proceedings**, Vol. 10054 of **Lecture Notes in Computer Science**, pp. 85–98, 2016.

[9] Koji Mineshima, Ribeka Tanaka, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. Building compositional semantics and higher-order inference sys-

1) [17] は次の French ANR grant の支援を受けたものである：https://www.hds.utc.fr/everest/doku.php?id=en:smemlj12

tem for a wide-coverage japanese CCG parser. In J. Su, X. Carreras, and K. Duh, editors, **Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016**, pp. 2236–2242. The Association for Computational Linguistics, 2016.

[10] Stergios Chatzikyriakidis and Jean-Philippe Bernardy. A wide-coverage symbolic natural language inference system. In Mareike Hartmann and Barbara Plank, editors, **Proceedings of the 22nd Nordic Conference on Computational Linguistics, NoDaLiDa 2019, Turku, Finland, September 30 - October 2, 2019**, pp. 298–303. Linköping University Electronic Press, 2019.

[11] Robin Cooper. Representing types as neural events. **J. Log. Lang. Inf.**, Vol. 28, No. 2, pp. 131–155, 2019.

[12] Staffan Larsson. Discrete and probabilistic classifier-based semantics. In **Proceedings of the Probability and Meaning Conference (PaM 2020)**, pp. 62–68, Gothenburg, June 2020. Association for Computational Linguistics.

[13] Daisuke Bekki, Ribeka Tanaka, and Yuta Takahashi. Learning knowledge with neural DTS. In **Proceedings of the 3rd Natural Logic Meets Machine Learning Workshop (NALOMA III)**, pp. 17–25, Galway, Ireland, August 2022. Association for Computational Linguistics.

[14] Daisuke Bekki, Ribeka Tanaka, and Yuta Takahashi. Integrating Deep Neural Networks with Dependent Type Semantics. In Roussanka Loukanova, Peter LeFanu Lumsdaine, and Reinhard Muskens, editors, **Logic and Algorithms in Computational Linguistics 2021 (LACompLing2021)**. Springer Cham, to appear.

[15] Andres Löh, Conor McBride, and Wouter Swierstra. A tutorial implementation of a dependently typed lambda calculus. **Fundam. Informaticae**, Vol. 102, No. 2, pp. 177–207, 2010.

[16] hasktorch: MLP implementation of logical XOR. Accessed Jan. 11, 2023. https://github.com/hasktorch/hasktorch/tree/master/examples/xor-mlp.

[17] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data - Application to word-sense disambiguation. **Mach. Learn.**, Vol. 94, No. 2, pp. 233–259, 2014.