# Neural DTS: A hybrid NLI system combining two procedural approaches

**Mizuki Iinuma** **Sora Tagami** **Yuta Takahashi** **Daisuke Bekki**
Ochanomizu University
{iinuma.mizuki, tagami.sora, takahashi.yuta, bekki}@is.ocha.ac.jp

## 1 Introduction

According to Marr (1982)'s three-level methodology in cognitive science, it is claimed that an information processing system should be understood in the order of the computational level, the procedural (in other words, representational and algorithmic) level, and the hardware-implementational level. From this perspective, on the one hand, the study of truth-conditional semantics in theoretical linguistics, insofar as it targets the meaning of a sentence, can be regarded as being concerned with the computational level with some exceptions such as van Benthem's semantic automata. Then, the proof-theoretic semantics, which emerged in the mid-1990s as an alternative to the truth-conditional semantics, can also be regarded as a progress from the computational level to the procedural level in the sense that its conception of truth is defined via *proof constructions* and therefore is implementable, given the recent progress of theorem provers for higher-order logics (Chatzikyriakidis and Luo, 2014, 2016; Daido and Bekki, 2020).

On the other hand, in recent years, end-to-end neural NLI systems based on large language models (Lan et al., 2020; Raffel et al., 2020; He et al., 2021) achieved a huge success. From the viewpoint of Marr's methodology, those systems satisfy one of the requirements of the computational theory by specifying the inputs and outputs in training data, although it is arguable that end-to-end neural NLI systems can only partially satisfy the requirements because the functions to be computed by those systems are required to be stated explicitly. Then, at the procedural level, they just automatically compute the values of parameters by fine-tuning the model. This methodology is in sharp contrast with that of proof-theoretic semantics, in which the understanding of the procedural level is pursued via hypothesis verification.

These two approaches complement each other in terms of their performances (precision vs. coverage, depth vs. speed, and their explanatory nature). Therefore, it is natural to seek a hybrid semantic system integrating the two approaches, not only from the technical point of view but also from the methodological point of view, comparing the two approaches to the procedural level of semantics. Neural DTS (Bekki et al., 2023) is one such hybrid approach for NLI, incorporating a deep neural network within dependent type semantics (DTS), a proof-theoretic semantics based on dependent type theory (Bekki, 2014; Bekki and Mineshima, 2017). Neural DTS uses neural classifiers to determine the truth of atomic predicates. Then, it uses inference rules of DTS, which are natural deduction rules for dependent type theory, to construct a proof for complex propositions.

Although a learning algorithm for Neural DTS is depicted in Bekki et al. (2022), an actual implementation of Neural DTS has been a remaining issue. In particular, it was not obvious how the execution of a type checking algorithm and the feed-forward neural classifiers co-exist while neural classifiers can be trained by data.

In this work, we provide an implementation of Neural DTS by combining the implementations of neural classifiers and that of DTS's type checking algorithm so that it can process the interaction of neural classifiers' predictions and type-logical inferences, including negations, conjunctions, and disjunctions. Our implemantation of type checking algorithm is based on Löh et al. (2010). Also, we implemented three different neural classifiers and compared their performances: multi-layer perceptrons (MLPs), neural tensor network (NTN) as introduced in Socher et al. (2013), and NTN as introduced in Ding et al. (2015).

We will demonstrate and evaluate the behavior of this implementation by using complex propositions consisting of binary relations. Overall, Neural DTS is a feasible NLI system which shows mixed advantages of two procedural approaches: proof-theoretic semantics and neural NLI systems.

## 2 Dependent Type Semantics

DTS is a semantics of natural language based on the Dependent Type Theory that can define types depending on terms. In DTS, meaning is indicated by the types of Dependent Type Theory. A sentence is true if there exists a term with the type of the semantic representation of the sentence. By using various types such as $\Pi$ type and $\Sigma$ type, even complex linguistic

phenomena such as quantifiers and presuppositional binding can be systematically understood. However, in the current DTS, even semantically similar entities, such as a dog and a puppy, are treated as separate entities, so that the fact that they are similar entities cannot be used for inference unless these concepts are related by explicit knowledge. Also, representations are not learnable, so they must be introduced as axioms in inference. Neural DTS addresses these points by incorporating a classifier.

## 3 Inference using type checking algorithm

A type checking algorithm for Neural DTS infers whether a relational proposition is true or not. In the process of such an inference, a type checking algorithm uses the result of prediction by the trained classifier in Neural DTS.

### 3.1 Classifier with MLP and NTN

As a comparative experiment, we implemented an MLP and two versions of an NTN, the original (Socher et al., 2013) and Ding et al. (2015) which provide the two different ways of representing binary relations. These networks are implemented using Hasktorch[1], a Haskell interface for deep learning.

**MLP** The MLP in this study is formulated as

$f(c_1, r, c_2) = \sigma(W_3(\sigma(W_2(\sigma(W_1(c_1 \oplus r \oplus c_2) + b_1)) + b_2)) + b_3)$,

where $c_1, c_2$ are class embeddings, $r$ is a relational embedding, and $c_1 \oplus r \oplus c_2$ is the concatenation of these embeddings. As usual, $W_i$ and $b_i$ are a weight and a bias for each $i \in \{1, 2, 3\}$, and $\sigma$ is the sigmoid function as an activation function.

**NTN** Socher's NTN is formulated by using the bilinear tensor product $c_1^{\mathsf{T}} W_r^{[1:k]} c_2$ as follows:

$$g(c_1, r, c_2) = U_r^{\mathsf{T}} \mathtt{tanh}\left(c_1^{\mathsf{T}} W_r^{[1:k]} c_2 + V_r \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b_r\right),$$

where $V_r \in R^{k \times 2d}$, $U_r \in R^k$, and $b_r \in R^k$.

### 3.2 Type checking algorithm in Neural DTS

A type checking algorithm takes a context $\Gamma$, a term $t$, and a type $A$ as inputs and then determines whether $t$ is of type $A$ in $\Gamma$ or not, in other words, whether $\Gamma \vdash t : A$ holds or not. By the so-called *Curry-Howard correspondence*, a type corresponds to a proposition, and a term of type $A$ corresponds to a proof of the proposition $A$. This means that we can check whether a witness term $w^+_{nmk}$ (resp. $w^-_{nmk}$) is a proof of a proposition $c_n r_m c_k$ (resp. the negation $\neg(c_n r_m c_k)$) by checking whether $w^+_{nmk}$ is of type $c_n r_m c_k$ (resp. $w^-_{nmk}$ is of type $\neg(c_n r_m c_k)$), where

- $w^+_{nmk}$ is of type $c_n r_m c_k$ if the prediction by the classifier $h(c_n, r_m, c_k)$ is YES with $h \in \{f, g\}$,
- $w^-_{nmk}$ is of type $\neg(c_n r_m c_k)$ if the prediction by the classifier $h(c_n, r_m, c_k)$ is NO.

In other words, we reduce the typability of a witness term to the prediction by a neural classifier $h(c_n, r_m, c_k)$ explained above. Our algorithm calls a neural classifier by the type checking of a witness term, and predicts whether $c_n r_m c_k$ holds or not by using the results of feeding the relation $r_m$ and classes $c_n, c_k$ into the classifier.

Then, we combine the type checking of a witness term with the usual type checking algorithm in dependent type theory, so that the algorithm provides an inference procedure for the set of propositions composed from relational propositions and their negations by conjunction and disjunction. The type checking of a witness term is thus the part of our algorithm in which a neural network is incorporated. For instance, suppose that the prediction of the classifier $g(c_i, r_j, c_k)$ is YES while the prediction of the classifier $g(c_l, r_m, c_n)$ is NO, namely, $g(c_i, r_j, c_k) \geq \mathsf{threshold}$ and $g(c_l, r_m, c_n) < \mathsf{threshold}$ hold. In this case, the procedure to infer the conjunction $c_i r_j c_k \wedge \neg(c_l r_m c_n)$ is illustrated as follows:

$$\frac{\dfrac{g(c_i, r_j, c_k) \geq \mathsf{threshold}}{\vdash w^+_{ijk} : c_i r_j c_k} \quad \dfrac{g(c_l, r_m, c_n) < \mathsf{threshold}}{\vdash w^-_{lmn} : \neg(c_l r_m c_n)}}{\vdash (w^+_{ijk}, w^-_{lmn}) : c_i r_j c_k \wedge \neg(c_l r_m c_n)}$$

## 4 Experiment

To demonstrate the advantage of a hybrid approach, we conducted an experiment to make inferences on a set of propositions by the above type checking algorithm. This set consists of propositions in the one of the forms $c_n r_m c_k$, $\neg(c_n r_m c_k)$, $(c_n r_m c_k) \wedge (c_{n'} r_{m'} c_{k'})$, or $(c_n r_m c_k) \vee (c_{n'} r_{m'} c_{k'})$. We use the Yago4 dataset[2] to learn and infer relational propositions between classes. It consists of 13,000,788 entities and 110 relations, and each entity is assigned to one or more of the 249 applicable classes. For example, in the triplet `(Windows_10, copyrightHolder, Microsoft)`, the Windows_10's class is `[CreativeWork, Product, Thing]`, and the Microsoft's class is `[Organization, Thing]`. In this case, we can associate the classes of triplet and entity, e.g., `[CreativeWork, copyrightHolder, Organization]`. The triples thus created without duplication are used as positive data, and the same number of triplets not included in the positive data

[1] https://github.com/hasktorch/

[2] https://yago-knowledge.org/downloads/yago-4

are randomly generated and used as negative data. There are 12,066 training triplets, 4,022 testing triplets, and 4,022 validation triplets. We used 1,000 training epochs with 0.5 as the binary classification threshold and a learning rate of $5e^{-2}$.

Table 1: Evaluate values by classifier

| Value | Accuracy | Precision | Recall | f1 |
|---|---|---|---|---|
| MLP | 0.9614 | 0.9517 | 0.9722 | 0.9618 |
| Socher's NTN | 0.9801 | 0.9710 | 0.9898 | 0.9803 |
| Ding's NTN | 0.4953 | 0.4961 | 0.6054 | 0.5454 |

Table 1 shows the evaluate values of inferences of atomic relational proposition for each classifier. Compared with MLP, Socher's NTN was more accurate, while Ding's NTN was less accurate. This accuracy affects the accuracy of the overall inference.

The following triplets are examples of successful inference. `[Photograph,copyrightHolder,Person]` is a valid relation triplet and was correctly inferred to be valid in the experiment. `[Chapter,copyrightHolder,Pond]` is an invalid triplet, which is also correctly inferred not to be valid. These were not included in training dataset, but were knowledge acquired by learning. Our type checking algorithm has an embedded neural network, so it can perform type checking on unknown atomic propositions. On the other hand, some of failure cases are as follows. `[MusicPlaylist, children, Person]` is an invalid relation triplet, but it has been inferred to be valid. Conversely, `[Book, hasPart, Chapter]` is a valid triplet, but it has been inferred that it does not hold. Thus, the accuracy of the classifier is crucial because it can fail to acquire knowledge. We then confirmed that we were able to connect to symbolic reasoning in negation, conjunction, and disjunction using the reasoning results of these atomic propositions.

## 5 Conclusion

We provided an implementation of Neural DTS as a type checking algorithm in which MLP and NTN are incorpolated as neural classifiers. While MLP and NTN make inferences on relations between two classes in our algorithm, logical inferences via type checking are performed on complex propositions such as negations, conjunctions and disjunctions. Due to the logical nature of Neural DTS, inferences on the complex propositions are always correct when the truth of relations are correctly predicted. Unlike most logical systems, in addition, neural classifiers can make correct predictions even on unknown relationships. Thus, Neural DTS shows mixed advantages of proof-theoretic semantics and neural NLI systems as their hybrid.

In future work, we plan to conduct an experiment among entities as well. In addition, the symbolic inference procedure can only handle negation, conjunction, and disjunction at the moment, but as mentioned above, if we add the types such as $\Pi$ type and $\Sigma$ type to the procedure, we will be able to handle a certain range of natural language sentences in this framework. We plan to extend our procedure to these types in the future to enhance the advantages of DTS as semantics. This type checking and neural network piping in Neural DTS is significant for the future implementation of Neural DTS.

## References


Daisuke Bekki. 2014. Representing anaphora with dependent types. In *Logical Aspects of Computational Linguistics - 8th International Conference, LACL 2014, Toulouse, France, June 18-20, 2014. Proceedings*, volume 8535 of *Lecture Notes in Computer Science*, pages 14–29. Springer.

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

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

Daisuke Bekki, Ribeka Tanaka, and Yuta Takahashi. 2023. 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)*, Studies in Computational Intelligence. Springer Cham.

Stergios Chatzikyriakidis and Zhaohui Luo. 2014. Natural language inference in coq. *J. Log. Lang. Inf.*, 23(4):441–480.

Stergios Chatzikyriakidis and Zhaohui Luo. 2016. Proof assistants for natural language semantics. In *Logical Aspects of Computational Linguistics. Celebrating 20 Years of LACL (1996-2016) - 9th International Conference, LACL 2016, Nancy, France, December 5-7, 2016, Proceedings*, volume 10054 of *Lecture Notes in Computer Science*, pages 85–98.

Hinari Daido and Daisuke Bekki. 2020. Development of an automated theorem prover for the fragment of dts. In *the 17th International Workshop on Logic and Engineering of Natural Language Semantics (LENLS17)*.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. *Research Center for Social Computing and Information Retrieval Harbin Institute of Technology, China*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DEBERTA: Decoding-Enhanced BERT with disentangled attention. In *International Conference of Learning Representations (ICLR2021)*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for self-supervised learning of language representations. In *International Conference of Learning Representations (ICLR2020)*.

Andres Löh, Conor McBride, and Wouter Swierstra. 2010. A tutorial implementation of a dependently typed lambda calculus. *Fundam. Informaticae*, 102(2):177–207.

David Marr. 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., New York, NY.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.