Hybrid Approach for lightblue

Sora Tagami
Researcher, Ochanomizu University

August 1, 2025 | ESSLLI 2025

About me

Sora Tagami

Researcher at Bekki Lab, Ochanomizu University

- Research interest: Neural network × Symbolic Hybrid approaches

Full time Software Engineer at Google Japan

- Working for user generated content feature in Google Maps

Why a Hybrid Approach?

Neural Network based approach

Achieved state-of-the-art results in many NLP tasks, becoming a dominant force, especially with the rise of Large Language Models.

Symbolic based approach

Rooted in formal linguistic theories developed since the mid-20th century.

Why a Hybrid Approach?

Reliability
Due to their foundation in sound formal logic, symbolic systems inherently offer a high degree of reliability: if such a system returns a conclusion as 'true', that conclusion is unfailingly guaranteed to be logically true within its defined framework. This means a symbolic system will never assert something false as true.

Neural Network based approach

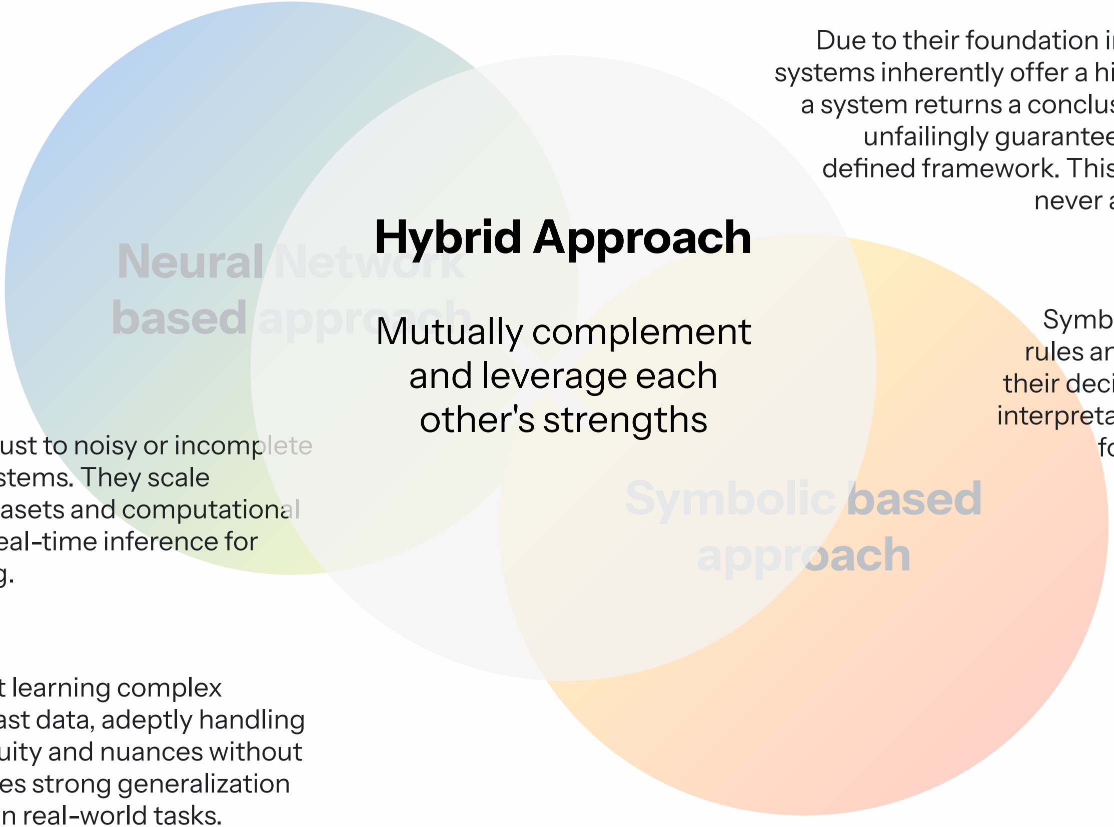
Symbolic based approach

Explainability
Symbolic systems use well-defined rules and explicit knowledge, making their decision-making transparent and interpretable. This traceability is crucial for applications requiring trust, auditability, or debugging.

Scalability
Neural networks are robust to noisy or incomplete data, unlike symbolic systems. They scale efficiently with large datasets and computational power, enabling quick, real-time inference for massive data processing.

Generalizability
Neural networks excel at learning complex patterns directly from vast data, adeptly handling natural language ambiguity and nuances without explicit rules. This enables strong generalization and high performance on real-world tasks.

Why a Hybrid Approach?



Reliability

Due to their foundation in sound formal logic, symbolic systems inherently offer a high degree of reliability: if such a system returns a conclusion as 'true', that conclusion is unfailingly guaranteed to be logically true within its defined framework. This means a symbolic system will never assert something false as true.

Explainability

Symbolic systems use well-defined rules and explicit knowledge, making their decision-making transparent and interpretable. This traceability is crucial for applications requiring trust, auditability, or debugging.

Scalability

Neural networks are robust to noisy or incomplete data, unlike symbolic systems. They scale efficiently with large datasets and computational power, enabling quick, real-time inference for massive data processing.

Generalizability

Neural networks excel at learning complex patterns directly from vast data, adeptly handling natural language ambiguity and nuances without explicit rules. This enables strong generalization and high performance on real-world tasks.

Summary of Hybrid Approach for lightblue

01

Hybrid Approach for Syntax (CCG)

Review recent neural-based research for CCG.

Slide 07

02

Hybrid Approach for Semantics (DTS)

Discuss our current work on integrating neural networks with DTS.

Slide 27

03

hasktorch

Introduce the machine learning framework we are adopting.

Slide 46

Hybrid Approach for Syntax (CCG)

CCG Syntactic Parsing

CCG parser takes 2 steps:

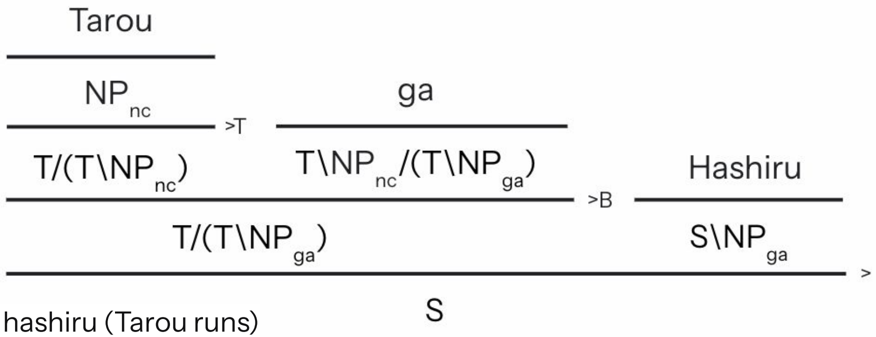
Supertagging

Assigns a lexical category to each word in a sentence.

Parsing

Combines categories using CCG rules to build a derivation tree.

Tarou : NP_{nc}
ga : $T\backslash NP_{nc}/(T\backslash NP_{ga})$
Hasiru : $S\backslash NP_{ga}$



Example: Tarou ga hashiru (Tarou runs)

CCG Syntactic Parsing with Neural Network Suppertagging

- With the evolution of neural networks since the mid-2010s, neural CCG parsing has rapidly developed.
- **Supertagging is almost parsing** (Bangalore and Joshi, 1999).
    - CCG uses **over 1,000 categories**.
    - Due to its importance, many neural methods were developed for supertagging, significantly improving accuracy.

CCG Syntactic Parsing with Neural Network Suppertagging

By the evolution of Neural Network, CCG parsing with Neural Network has been also developing since mid of 2010s.

CCG has **over 1,000 categories** - which has much syntactic information

- It is said that Supertagging is almost parsing (Bangalore and Joshi, 1999).
- Because of its importance, even as neural networks began to develop, many methods utilizing neural networks for Supertagging were created, which significantly contributed to improvements in its accuracy.

Lewis (2021) shows how effective the neural supertagger is:

PARSER	S-TAGGER	P	R	F	CAT	COV.
C&C	Maxent	–	–	85.3	–	99.1
C&C (*w/gold pos*)	Maxent	88.1	86.4	87.2	94.2	99.1
Java C&C (*w/gold pos*)	Maxent	88.0	87.3	87.7	94.3	100.0
Java C&C	Transformer	91.9	91.5	91.7	96.3	100.0

Table 2: Parser accuracy of (Java) C&C on Sec. 00 with different supertaggers.

Early Non-Neural Baselines:
C&C (Clark and Curran, 2004) : A chart-based parser implemented in C++
Java C&C (Clark et al., 2015) : A Java re-implementation with improvements
Maxtent (Clark, 2002) : A maximum entropy supertagger.

P/R/F: Precision, Recall, and F-score over CCG dependencies.
CAT: Lexical category accuracy.

CCG Syntactic Parsing with Neural Network Suppertagging

Lewis and Steedman (2014)
Feed-forward network with embeddings improved CCG supertagging

Xu et al. (2015)
RNNs capture sequence context for improved CCG supertagging

Lewis et al. (2016)
Bi-LSTMs capture long-range dependencies for enhanced supertagging accuracy

Vaswani et al. (2016)
Bi-LSTM-LM uses language model for supertag interactions

Clark et al. (2018)
Bi-LSTM encoder improved by Cross-View Training (CVT)

CCG Syntactic Parsing with Neural Network Suppertagging

Lewis and Steedman (2014a)
Feed-forward network with embeddings improved CCG supertagging

Xu et al. (2015)
RNNs capture sequence context for improved CCG supertagging

Lewis et al. (2016)
Bi-LSTMs capture long-range dependencies for enhanced supertagging accuracy

Vaswani et al. (2016)
Bi-LSTM-LM uses language model for supertag interactions

Clark et al. (2018)
Bi-LSTM encoder improved by Cross-View Training (CVT)

Kogkalidis et al. (2019)
Transformer-based model constructs supertag types from primitives, boosting generalization

Bhargava and Penn (2020)
LSTM predicts supertags as **sequences of CCG primitives,** generating novel categories

Liu et al. (2021)
Bi-LSTM generates CCG categories by decomposing them into atomic tag sequences

Prange et al. (2021)
RoBERTa encoder + TreeRNN/AddrMLP generates supertags as trees

Kogkalidis and Moortgat (2023)
A-GCN
Geometry-aware, graph convolutions enhance constructive supertagging via output structure.

CCG Syntactic Parsing with Neural Network Suppertagging

Lewis and Steedman (2014a)
Feed-forward network with embeddings improved CCG supertagging

Lewis et al. (2016)
Bi-LSTMs capture long-range dependencies for enhanced supertagging accuracy

Vaswani et al. (2016)
Bi-LSTM-LM uses language model for supertag interactions

Tian et al. (2020)
BERT + A-GCN enhances supertagging via chunk-based graphs

Yamaki et al. (2023)
Holographic embeddings enable recursive composition for improved supertagging and parsing.

Zhao and Penn(2024)
LLMs (Llama2) for supertagging boosted by repeating input

Xu et al. (2015)
RNNs capture sequence context for improved CCG supertagging

Clark et al. (2018)
Bi-LSTM encoder improved by Cross-View Training (CVT)

Kogkalidis et al. (2019)
Transformer-based model constructs supertag types from primitives, boosting generalization

Bhargava and Penn (2020)
LSTM predicts supertags as **sequences of CCG primitives**, generating novel categories

Liu et al. (2021)
Bi-LSTM generates CCG categories by decomposing them into atomic tag sequences

Prange et al. (2021)
RoBERTa encoder + TreeRNN/AddrMLP generates supertags as trees

Kogkalidis and Moortgat (2023)
A-GCN
Geometry-aware, graph convolutions enhance constructive supertagging via output structure.

CCG Syntactic Parsing with Neural Network Suppertagging

Suppertagging Model	Parser	Supertag accuracy	Labeled F1 score
Lewis and Steedman (2014a)	C&C	-	86.11
Xu et al. (2015)	C&C	93.00	87.68
Lewis et al. (2016)	C&C	94.7	88.1
Vaswani et al. (2016)	C&C	94.5	88.32
Clark et al. (2018)	-	96.1	-
Kogkalidis et al. (2019)			
Bhargava and Penn (2020)	C&C	96.00	90.9
Tian et al.(2020)	EasyCCG	96.2	90.58
Liu et al. (2021)	C&C	96.05	90.87
Prange et al. (2021)	C&C	96.22	90.91
Yamaki et al. (2023)	C&C	**96.6**	**92.12**
Kogkalidis and Moortgat (2023)	-	**96.29**	-
Zhao and Penn (2024)	C&C	**96.64**	**92.05**

- Supertagging accuracy has steadily increased with the introduction of new neural models.
- Recent models achieve over **96%** supertagging accuracy on the CCGbank test set.
- Improvements in supertagging directly translate to higher F1 scores for parsing, with recent results exceeding **92%**

TestData: CCGbank (Hockenmaier and Steedman, 2007)
C&C parser: Clark, 2015
Easy CCG: Lewis and Steedman, 2014b

CCG Syntactic Parsing with Neural Network Parsing

- Beyond supertagging, the parsing process itself can be enhanced with neural networks.
- This process requires three key components:
  - **Parsing algorithm**: determines the order in which the categories are put together
  - **Parsing model**: scores each possible analysis
  - **Search algorithm**: efficiently finds the highest-scoring analysis

CCG Syntactic Parsing with Neural Network Parsing

This process requires three key components

- **Parsing algorithm**: determines the order in which the categories are put together
- **Parsing model**: scores each possible analysis
- **Search algorithm**: efficiently finds the highest-scoring analysis

Parsing algorithms

- **Chart-based**
  - Uses dynamic programming to store all partial parse results in a "chart," avoiding redundant computations.
  - Was the first algorithm successfully applied to wide-coverage CCG parsing
  - Hockenmaier and Steedman (2002), Clark and Curran (2004)
- **Shift-reduce**
  - Use a stack to build a parse tree. It works by shifting input symbols onto the stack or reducing a recognized sequence on the stack into a non-terminal symbol according to grammar rules.
  - Zhang and Clark (2011), Xu et al. (2014), Ambati et al. (2016)

CCG Syntactic Parsing with Neural Network Parsing

This process requires three key components:

- **Parsing algorithm**: determines the order in which the categories are put together
- **Parsing model**: scores each possible analysis
- **Search algorithm**: efficiently finds the highest-scoring analysis

Parsing model

- Based on **lexicalised Probabilistic Context-Free Grammars (PCFGs)**
    - Early models that used relative frequency counts to estimate parameters.
    - Hockenmaier and Steedman (2002), Collins (1997)
- **Discriminative, feature-based models**
    - Superseded PCFGs by applying maxent models, similar to those used in tagging.
    - Clark and Curran (2004), Riezler et al. (2002)
- Alternative estimation methods based on **the structured perceptron framework**
    - Provided a simpler estimation technique and was also successfully applied to CCG.
    - Clark and Curran (2007), Collins and Roark (2004)

CCG Syntactic Parsing with Neural Network Parsing

This process requires three key components:

- **Parsing algorithm**: determines the order in which the categories are put together
- **Parsing model**: scores each possible analysis
- **Search algorithm**: efficiently finds the highest-scoring analysis

Search algorithm

- **Dynamic Programming**
  - Used by early chart-based parsers to find the optimal parse.
  - Hockenmaier and Steedman (2002), Clark and Curran (2004)
- **Heuristic Beam search**
  - Often used by faster shift-reduce parsers, performing well even with small beam widths.
  - Zhang and Clark (2011)
- **A* search**
  - Uses a heuristic function to guide the search, guaranteeing optimality with an admissible heuristic.
  - Lee et al (2016)

CCG Syntactic Parsing with Neural Network Parsing

This process requires three key components:

- **Parsing algorithm**: determines the order in which the categories are put together
- **Parsing model:** scores each possible analysis
- **Search algorithm**: efficiently finds the highest-scoring analysis

Recent hybrid research

- focuses on replacing the parsing model with a neural network.
- learns to score the actions or derivations proposed by the parsing algorithm.
- adopts parsing and search algorithm varies across different models.

CCG Syntactic Parsing with Neural Network Parsing

Xu (Wenduan) (2016)

LSTM Shift-Reduce CCG Parsing: An **LSTM shift-reduce parser** linearizes parsing history. The LSTMs choose actions, and the model is globally optimized for F1 score.

Lewis et al. (2016)

A Bi-directional LSTM supertagger informs an **A* parser** to find the optimal supertag sequence for a complete parse, without an explicit bi-lexical model

Yoshikawa et al. (2017)

An **A* parser** factors tree probability into **bi-LSTM** supertags and bilexical dependencies

Clark (2021)

A **Transformer** supertagger feeds the Java C&C chart parser. The Transformer also provides span scores for the parser's nodes.

Ambati et al. (2016)

A neural network-based shift-reduce parser uses a feed-forward NN to score actions. It supports both **greedy** and **beam-search**

Lee et al. (2016)

Combines local model with **Tree-LSTM/Bi-LSTM** global model for parse structure. Uses A* decoding with optimality guarantees

Stanojević and Steedman (2020)

This fully **incremental transition-based parser** (based on **RNNG**) uses a global unnormalized model trained with **beam-search** optimization to address identified biases

Yamaki et al. (2023)

A **RoBERTa encoder** with **holographic embeddings** composes phrase-level representations for **span-based parsing** and **supertagging**

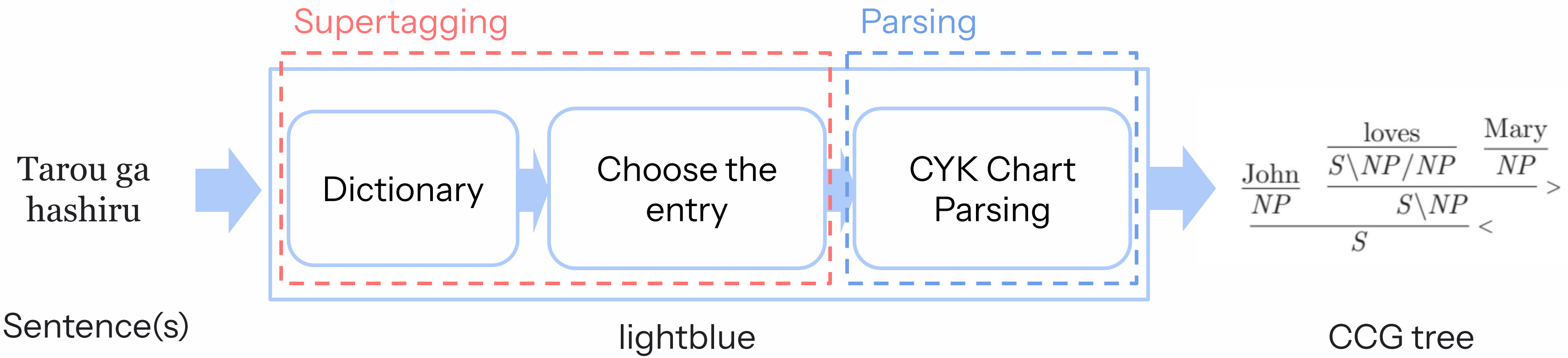
CCG Syntactic Parsing with Neural Network Parsing

Suppertagging Model	Supertag Accuracy	Labeled F1 score
Ambati et al. (2016)	92.03	83.33
Xu (Wenduan) (2016)	94.6	87.8
Lee et al. (2016)	–	88.7
Yoshikawa et al. (2017)	–	88.8
Stanojević and Steedman (2020)	95.6	90.6
Clark (2021)	96.5	**92.9**
Yamaki et al. (2023)	96.60	**92.67**

- The Clark (2021) model, which combines a Transformer with a chart parser, currently achieves the state-of-the-art F1 score of **92.9.**
- Yamaki et al. (2023)'s supertagger combined with the classic C&C parser achieves a very close score of **92.12**.
- Incorporating a neural parser provides a performance boost, but the largest gains come from a high-quality neural supertagger.

CCG Syntactic Parsing in the lightblue

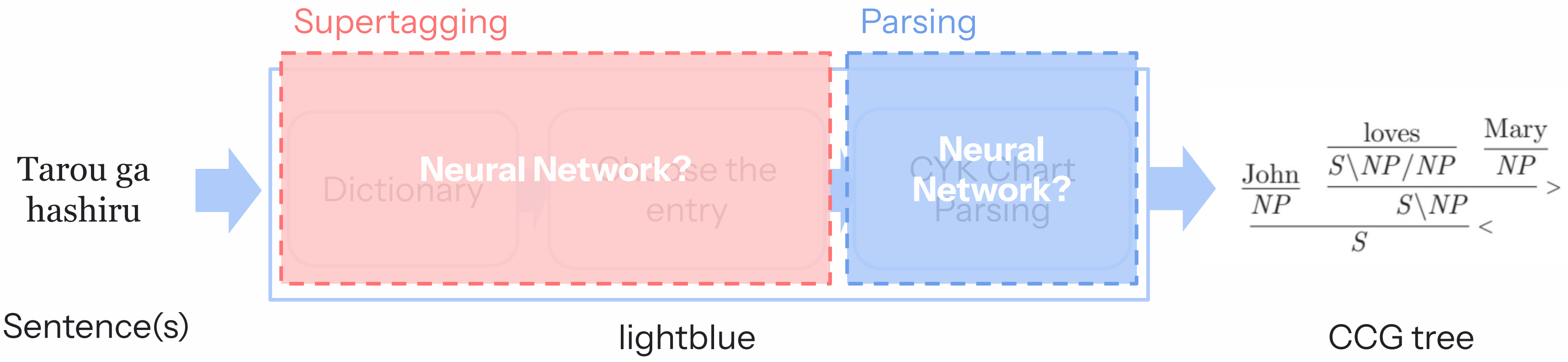
- Supertagging relies on a manually defined lexicon.
  - Juman dictionary (open word)
  - Additional lexical items extracted from Bekki 2010 (closed word)
- lightblue adopts **CYK chart parsing** as a parsing algorithm



Neural-based CCG Syntactic Parsing in the lightblue

How to integrate with Neural network?

- Option 1: Replace the dictionary-based supertagging with a neural CCG supertagger.
- Option 2: Replace the CYK parser with a neural CCG parser.
- Option 3: Replace both components with an end-to-end neural model.

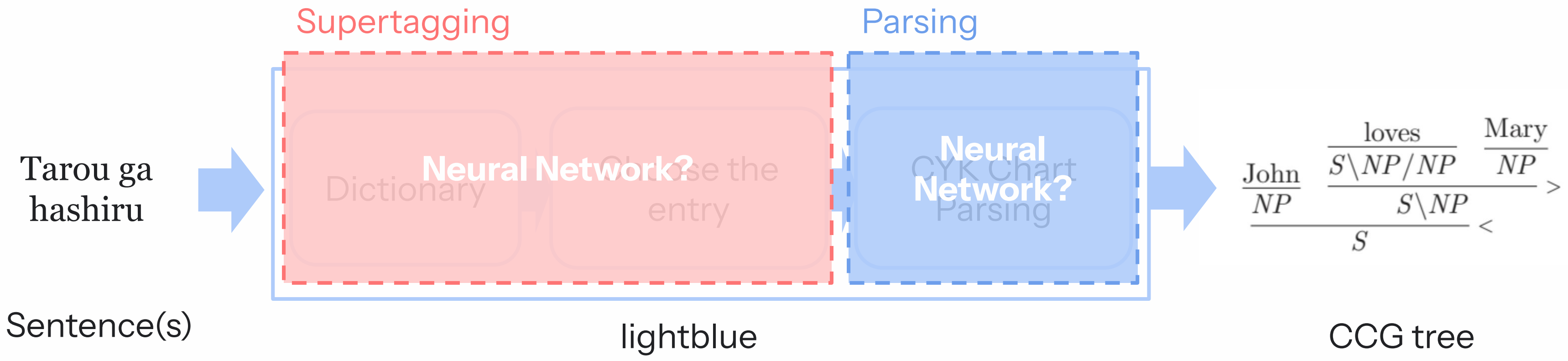


Neural-based CCG Syntactic Parsing in the lightblue

How to integrate with Neural network?

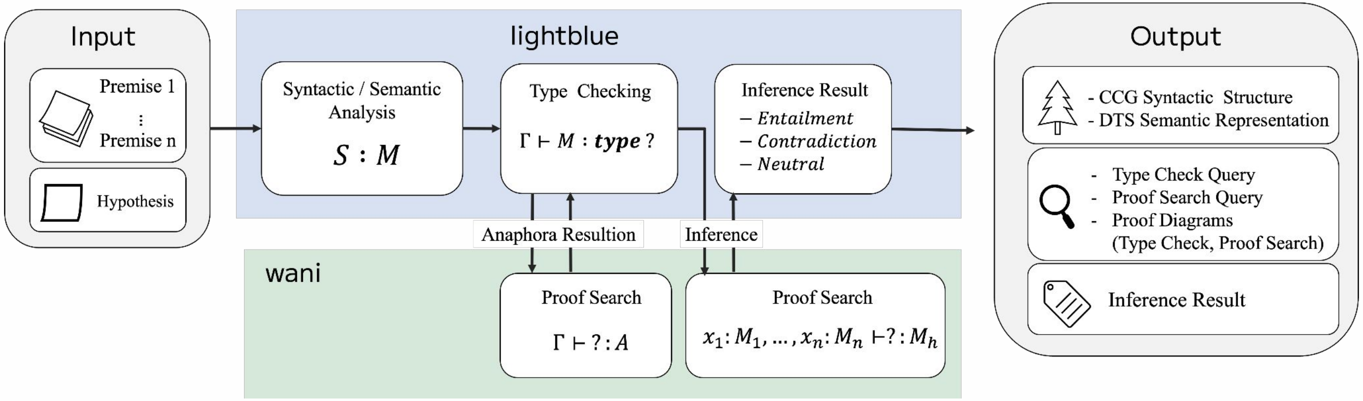
- Option 1: Replace the dictionary-based supertagging with a neural CCG supertagger.
- Option 2: Replace the CYK parser with a neural CCG parser.
- Option 3: Replace both components with an end-to-end neural model.

It is not that simple!



Challenge 1: The System Disconnect

- lightblue is a comprehensive system that performs syntax, semantics, and inference.
- However, standard neural parsers are trained only on syntactic information.
  - They cannot receive feedback from downstream components like semantic analysis or NLI.
  - This creates a disconnect, hindering the development of a fully integrated and robust system.



Challenge 2: Data limitation

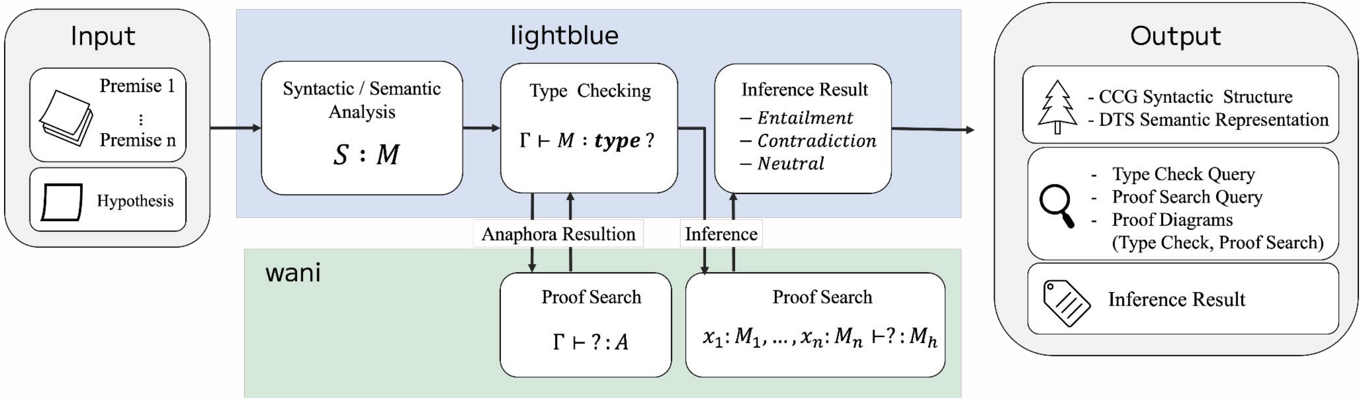
- Current state-of-the-art neural networks achieve high performance scores—96.64 in supertagging and 92.9 in parsing.
  - However, these results are based on the CCGbank test set, which has known issues, such as an abundance of unary rules
  - High scores on CCGbank do not guarantee strong performance on real-world data.
- **Flexibility In lightblue**
  - In contrast, lightblue offers greater flexibility, as its dictionary is not dependent on a specific corpus and can be easily modified.

Hybrid Approach for Semantics

Semantic Components in the lightblue

lightblue provides extensive information about semantics. One of its most important outputs is the result of **Natural Language Inference (NLI)**

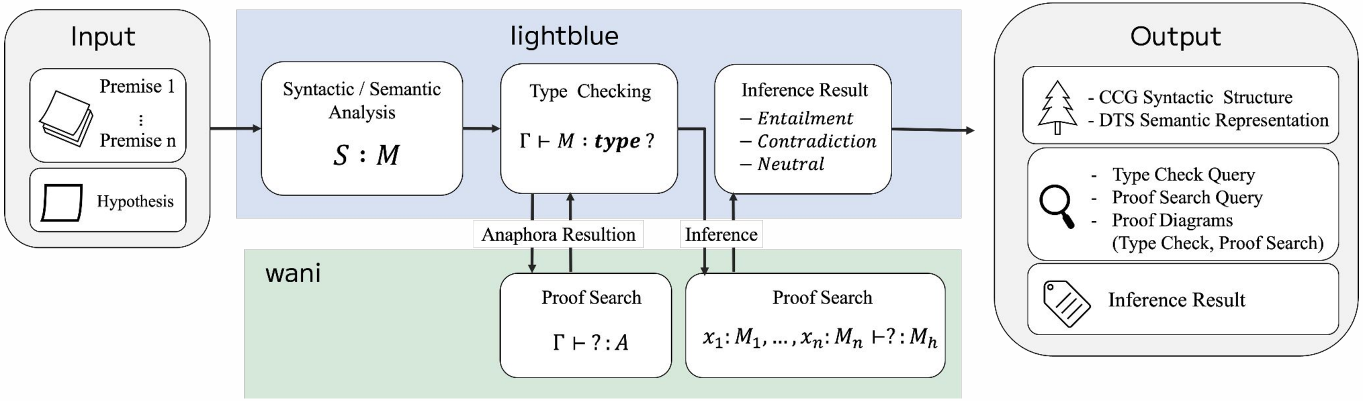
- NLI is the task of determining whether a given premise semantically entails a given hypothesis (Dagan et al., 2013).
- This has broad applications and can benefit many other language tasks, such as question answering, text summarization, and machine reading comprehension.



Semantic Components in the lightblue

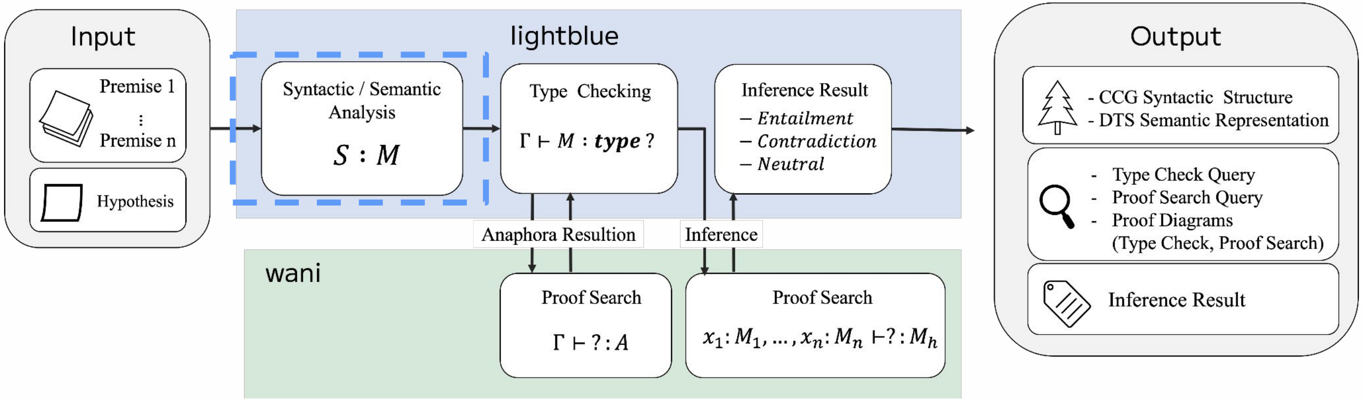
lightblue follows these semantic steps:

1. **Semantic parsing**: Provides a semantic representation based on DTS for both premises and hypotheses.
2. **Type check**: Verifies if the given semantic representations are well-formed (i.e., meet semantic felicity conditions) and resolves anaphora.
3. **NLI**: Determines if the premises entail the hypothesis from the semantic representation.



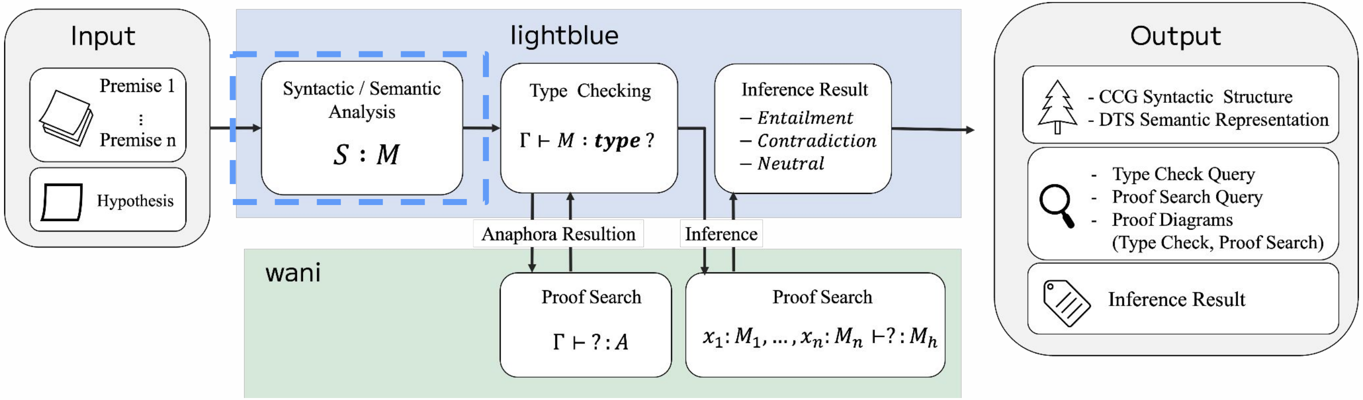
Hybrid Approach for Semantics: Neural-based Semantic Parsing

- Input: Sentence(s)
- Output: Semantic representation
- Several neural-based parsers for other semantic theories (e.g., Discourse Representation Theory (DRT)) have been published (Liu et al., 2018; Fancellu et al., 2019; van Noord et al., 2020; Yang et al., 2024).



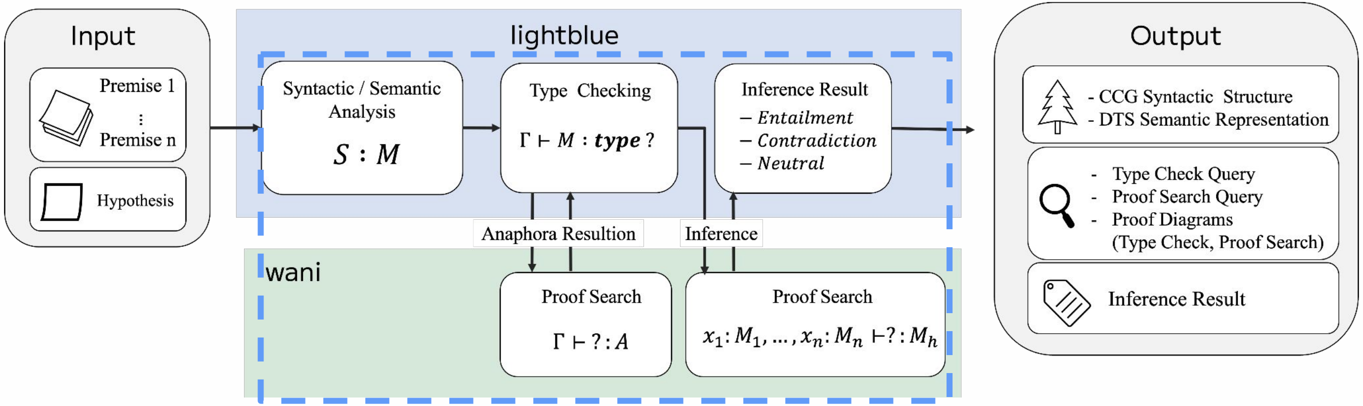
Hybrid Approach for Semantics: Neural-based Semantic Parsing

- Input: Sentence(s)
- Output: Semantic representation
- Several neural-based parsers for other semantic theories (e.g., Discourse Representation Theory (DRT)) have been published (Liu et al., 2018; Fancellu et al., 2019; van Noord et al., 2020; Yang et al., 2024).
- However, applying the same method is difficult due to the lack of a large, DTS-based corpus.



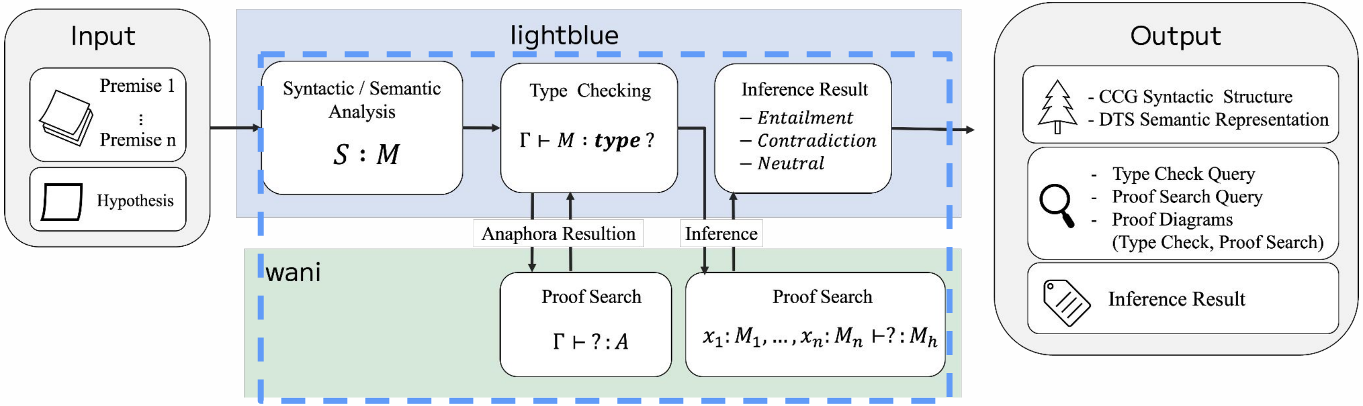
Hybrid Approach for Semantics: Hybrid models for NLI

- Input: Sentences/Semantic Representation
- Output: YES/NO/UNKNOWN
- Hybrid approaches for NLI systems have been proposed (Cooper 2019, Chen et al. 2021, Larsson, 2022), embedding a neural network into a symbolic approach in some manner.



Hybrid Approach for Semantics: Hybrid models for NLI

- Input: Sentences/Semantic Representation
- Output: YES/NO/UNKNOWN
- Hybrid approaches for NLI systems have been proposed (Cooper 2019, Chen et al. 2021, Larsson, 2022), embedding a neural network into a symbolic approach in some manner.
- We are also exploring this method to embed the neural network locally to avoid affecting other components.
- This approach mitigates concerns about data limitations and system disconnect.



Hybrid Approach for DTS based Reasoning

There are two ongoing hybrid project for DTS:

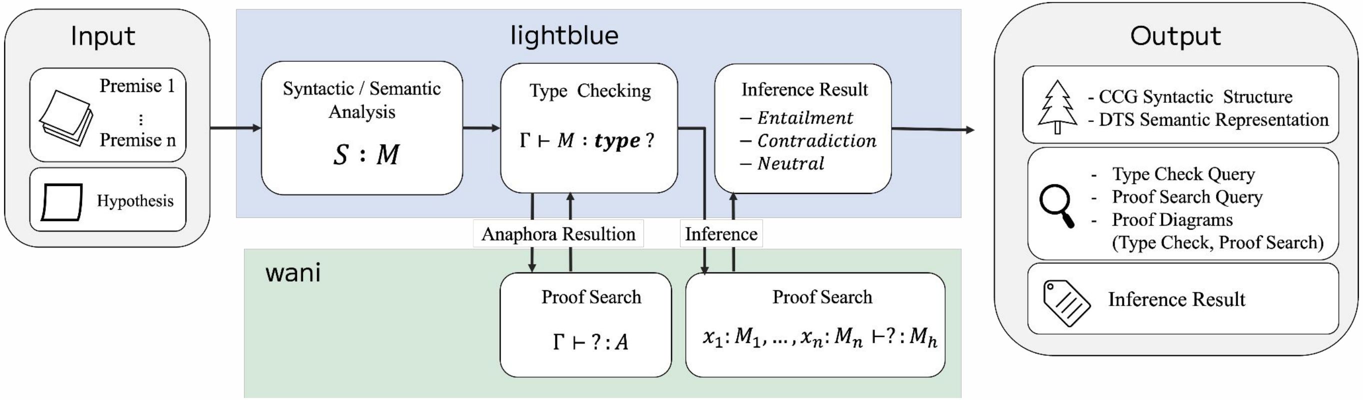
NeuralWani

Optimize wani (automated theorem prover) with neural classifier to choose a next rule to apply in the proof search.

NeuralDTS

Incorporate Neural network into DTS theory: replace predicates with simple neural networks.

Implemented in the wani.



Hybrid Approach for DTS based Reasoning: NeuralWani

wani: An automated prover for DTS (Daido and Bekki 2017)

- During backward inference, wani searches for the next rule that can be applied to the current judgment
- The order in which wani searches for rules is predefined.

$$(\textbf{?})\ \frac{\textbf{?}}{\Gamma \vdash ? : A}$$

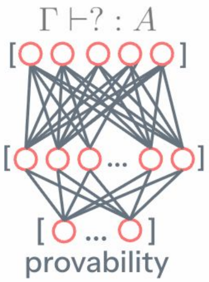
Hybrid Approach for DTS based Reasoning: NeuralWani

wani: An automated prover for DTS (Daido and Bekki 2017)

- During backward inference, wani searches for the next rule that can be applied to the current judgment
- The order in which wani searches for rules is predefined.

NeuralWani

- Uses a neural classifier to determine the search order for the next rule. It tries the rules output by the classifier in order of provability.



$$(?)\ \frac{?}{\Gamma \vdash ? : A}$$

Hybrid Approach for DTS based Reasoning: NeuralWani

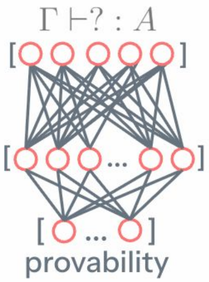
wani: An automated prover for DTS (Daido and Bekki 2017)

- During backward inference, wani searches for the next rule that can be applied to the current judgment
- The order in which wani searches for rules is predefined.

NeuralWani

- Uses a neural classifier to determine the search order for the next rule. It tries the rules output by the classifier in order of provability.
- **Miyagawa et al. (2023)** conducted a preliminary experiment with the full judgment, creating a dataset from the TPTP library and JSeM.
  - The results showed a sufficient F1 score for incorporation into wani.

	Prec	Rec	F1	Supp
Var	0.978	1.000	0.989	45
Con	1.000	1.000	1.000	45
TypeF	1.000	1.000	1.000	24
PiF	1.000	0.978	0.989	45
SigmaF	0.938	1.000	0.968	45
PiI	0.969	1.000	0.984	31
PiE	1.000	0.911	0.953	45
SigmaE	0.957	1.000	0.978	45
SigmaI	1.000	0.500	0.667	2
EnumF	1.000	1.000	1.000	45
IqF	1.000	0.500	0.667	2
IqE	1.000	1.000	1.000	5



(?)

$$\frac{?}{\Gamma \vdash ? : A}$$

$$\Gamma \vdash M : A$$

Hybrid Approach for DTS based Reasoning: NeuralDTS

Bekki (2022) proposes a theory of NeuralDTS for incorporating a Neural Network into DTS.

- The main idea is to represent **names** with vectors and replace **predicates** with a neural classifier.

Hybrid Approach for DTS based Reasoning: NeuralDTS

Bekki (2022) proposes a theory of NeuralDTS for incorporating a Neural Network into DTS.

- The main idea is to represent **names** with vectors and replace **predicates** with a neural classifier.
- For example...
  - When the lightblue inference system determines if the hypothesis (Cup noodle is cheap) is entailed by the given premise (Every noodle is cheap), the following proof search is conducted.

isCheap(cupNoodle) is a proposition, which is true
if and only if there is a proof for cup noodle being cheap.

Every noodle is cheap
c : (x:e)→isNoodle(x)→isCheap(x)

⊢

Cup noodle is cheap
? : isCheap(cupNoodle)

Hybrid Approach for DTS based Reasoning: NeuralDTS

Bekki (2022) proposes a theory of NeuralDTS for incorporating a Neural Network into DTS.

- The main idea is to represent **names** with vectors and replace **predicates** with a neural classifier.
- For example…
  - When the lightblue inference system determines if the hypothesis (Cup noodle is cheap) is entailed by the given premise (Every noodle is cheap), the following proof search is conducted.

Every noodle is cheap ⊢ **Cup noodle is cheap**

$c : (x{:}e) \to \mathrm{isNoodle}(x) \to \mathrm{isCheap}(x) \vdash ? : \mathrm{isCheap}(\mathrm{cupNoodle})$

$$\dfrac{\dfrac{c{:}(x{:}e)\to \mathrm{isNoodle}(x)\to \mathrm{isCheap}(x) \quad \dfrac{}{\mathrm{cupNoodle}{:}e}\,(\mathrm{CON})}{c(\mathrm{cupNoodle}){:}\mathrm{isNoodle}(\mathrm{cupNoodle})\to \mathrm{isCheap}(\mathrm{cupNoodle})}\,(\Pi E) \quad s{:}\mathrm{isNoodle}(\mathrm{cupNoodle})}{? : \mathrm{isCheap}(\mathrm{cupNoodle})}\,(\Pi E)$$

Hybrid Approach for DTS based Reasoning: NeuralDTS

Bekki (2022) proposes a theory of NeuralDTS for incorporating a Neural Network into DTS.

- The main idea is to represent **names** with vectors and replace **predicates** with a neural classifier.
- For example...
  - When the lightblue inference system determines if the hypothesis (Cup noodle is cheap) is entailed by the given premise (Every noodle is cheap), the following proof search is conducted.
  - Common-sense knowledge like "Every cup noodle is a noodle" must be manually added as a theorem in the current lightblue system.

Every noodle is cheap ⊢ **Cup noodle is cheap**

c : (x:e)→isNoodle(x)→isCheap(x) ⊢ ? : isCheap(cupNoodle)

$$\dfrac{\dfrac{c:(x{:}e)\rightarrow \mathrm{isNoodle}(x)\rightarrow \mathrm{isCheap}(x) \qquad \dfrac{}{\mathrm{cupNoodle}:e}\,(\mathbf{CON})}{c(\mathrm{cupNoodle}):\mathrm{isNoodle}(\mathrm{cupNoodle})\rightarrow \mathrm{isCheap}(\mathrm{cupNoodle})}\,(\Pi E) \qquad \overset{?}{\nabla}\; s:\mathrm{isNoodle}(\mathrm{cupNoodle})}{?:\mathrm{isCheap}(\mathrm{cupNoodle})}\,(\Pi E)$$

Can't infer unless it is contained in the Premises

Hybrid Approach for DTS based Reasoning: NeuralDTS

Bekki (2022) proposes a theory of NeuralDTS for incorporating a Neural Network into DTS.

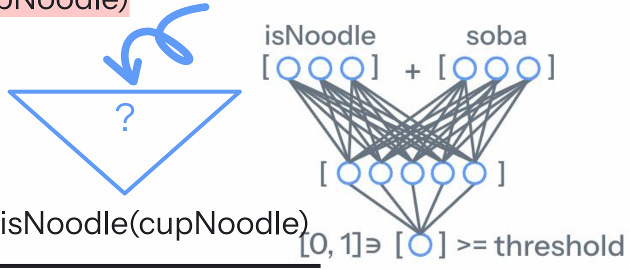
- The main idea is to represent **names** with vectors and replace **predicates** with a neural classifier.
- For example...
  - When the lightblue inference system determines if the hypothesis (Cup noodle is cheap) is entailed by the given premise (Every noodle is cheap), the following proof search is conducted.
  - Common-sense knowledge like “Every cup noodle is a noodle” must be manually added as a theorem in the current lightblue system.

Every noodle is cheap ⊢ **Cup noodle is cheap**

$c : (x:e) \to \mathrm{isNoodle}(x) \to \mathrm{isCheap}(x) \vdash\ ? : \mathrm{isCheap}(\mathrm{cupNoodle})$

Call neural classifier to judge the predicate-entity relationship

$$\dfrac{\dfrac{c:(x:e)\to \mathrm{isNoodle}(x)\to \mathrm{isCheap}(x) \qquad \dfrac{}{\mathrm{cupNoodle}:e}\,(\mathrm{CON})}{c(\mathrm{cupNoodle}):\mathrm{isNoodle}(\mathrm{cupNoodle})\to \mathrm{isCheap}(\mathrm{cupNoodle})}\,(\Pi E) \qquad s:\mathrm{isNoodle}(\mathrm{cupNoodle})}{?:\mathrm{isCheap}(\mathrm{cupNoodle})}\,(\Pi E)$$



Hybrid Approach for DTS based Reasoning: NeuralDTS

Bekki (2022) proposes a theory of NeuralDTS for incorporating a Neural Network into DTS.

- The main idea is to represent **names** with vectors and replace **predicates** with a neural classifier.
- However,
  - If the hypothesis is "Soba is cheap," soba would be a predicate.
  - The current neural classifier cannot determine if "soba is a noodle" (i.e., it cannot compare predicates).

Every noodle is cheap
c : (x:e)→noodle(x)→isCheap(x)

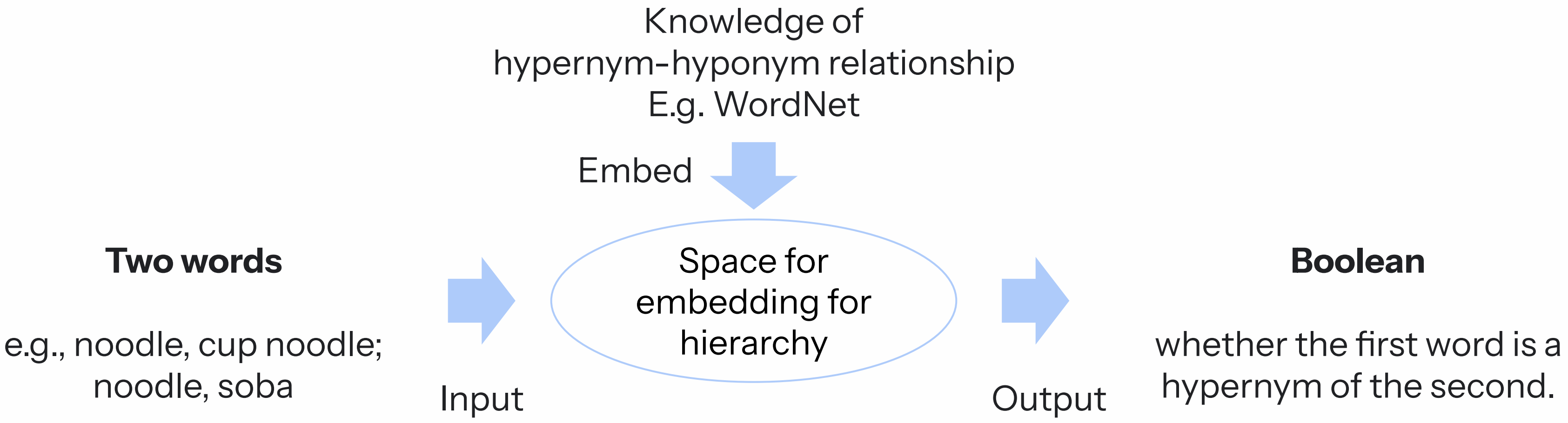
⊢

Soba is cheap
? :soba(x)→isCheap(x)

$$
\dfrac{\dfrac{\dfrac{c:(x:e)\to noodle(x)\to isCheap(x) \qquad \dfrac{}{x:e}\,(\mathrm{Var})}{c(x):noodle(x)\to isCheap(x)}\,(\Pi E) \qquad \dfrac{\dfrac{}{s:soba(x)}\,2 \quad \vdots}{?:noodle(x)}}{?:isCheap(x)}\,(\Pi E)}{?:soba(x)\to isCheap(x)}\,(\Pi I),2
$$

Hybrid Approach for DTS based Reasoning: NeuralDTS

- We plan to embed a hierarchy of general concepts (hypernym-hyponym relationships) into the space.
- Both predicates and names will be embedded in this space.



Hybrid Approach for DTS based Reasoning: NeuralDTS

- We plan to embed a hierarchy of general concepts (hypernym-hyponym relationships) into the space.
- Both predicates and names will be embedded in this space.

Space for embedding for hierarchy

Add the relationship as an axiom

Every noodle is cheap $\vdash$ **Soba is cheap**

$c : (x:e)\to noodle(x)\to isCheap(x) \vdash\ ? : soba(x)\to isCheap(x)$

$$
\dfrac{\dfrac{\dfrac{\dfrac{c:(x:e)\to noodle(x)\to isCheap(x) \quad \dfrac{}{x:e}\,(Var)\,2}{c(x):noodle(x)\to isCheap(x)}\,(\Pi E) \qquad \dfrac{\dfrac{}{s:soba(x)}\,1 \quad \dfrac{\dfrac{}{x:e}\,(Var)\,2 \quad n:(x:e)\to soba(x)\to noodle(x)}{n(x):soba(x)\to noodle(x)}}{n(x)(s): noodle(x)}\,(\Pi E)}{c(x)(n(x)(s)) : isCheap(x)}\,(\Pi E)}{\lambda s.\ c(x)(n(x)(s)) : soba(x)\to isCheap(x)}\,(\Pi I)\,1}{\lambda x.\lambda s.\ c(x)(n(x)(s)) : (x:e)\to soba(x)\to isCheap(x)}\,(\Pi I)\,2
$$

hasktorch

What is the hasktorch?

An open-source project providing a Haskell interface for PyTorch.

- Github: http://github.com/hasktorch/hasktorch
- Web page: http://hasktorch.org/

It has three implementations:

- hasktorch/src/: We are using this version.
  - The most common implementation, where a tensor is treated as a “Tensor” type.
- hasktorch/Typed/
  - This version includes type information for tensors, such as size and numerical type (e.g., int, float). It cannot handle tensors with dynamically changing sizes, like word embeddings.
- experimental/gradually-typed
  - This is still an experimental implementation.

Why do we use hasktorch?

- lightblue is implemented in Haskell.
- While we could train a model with PyTorch and call it from lightblue, using hasktorch directly offers several advantages:
- **Pros**: The codebase for lightblue will be synced with any updates. We can share enum definitions (e.g., Preterm, Rules) and functions between the lightblue and neural models.
- **Cons**: The main drawback is the lack of helpful library, detailed documentation and example implementations.

Hasktorch projects by Bekki lab.

Hasktorch-tools: https://github.com/DaisukeBekki/hasktorch-tools

- Owned by Prof. Bekki
- Provides:
  - Helpful libraries: src/Torch
  - Layers: src/Torch/Layer
  - Example implementation: app/

Hasktorch seminar

- We have held an introductory seminar for undergraduate students and interns from Bordeaux University for the past two years.
- It allows students to learn functional programming and machine learning simultaneously.
- Since there is limited documentation for hasktorch compared to PyTorch, it offers a unique opportunity for students to learn from scratch.

References

Ambati, Bharat Ram, Tejaswini Deoskar, and Mark Steedman. 2016. Shift-reduce CCG parsing using neural network models. In Proceedings of the 2016Conference of the North American Chapter of the Association for Computa-tional Linguistics: Human Language Technologies, pages 447–453. San Diego,California: Association for Computational Linguistics.

Bangalore, Srinivas. and Aravind K. Joshi. 1999. Supertagging: An Approach to Almost Parsing. Computational Linguistics, 25(2):237–265.

Bekki, Daisuke, Ribeka Tanaka, and Yuta Takahashi. Learning knowledge withneural DTS. In Proceedings of the 3rd Natural Logic Meets Machine LearningWorkshop (NALOMA III), pp. 17–25, Galway, Ireland, August 2022. Associa-tion for Computational Linguistics.

Bekki, Daisuke. 2010. Nihongo-Bunpoo-no Keisiki-Riron - Katuyootaikei, Toogohantyuu, Imigoosei -(trans. ‘Formal Japanese Grammar: the conjugationsystem, categorial syntax, and compositional seman-tics’). Kuroshio Publisher, Tokyo.

Bhargava, Aditya and Gerald Penn. 2020. Supertagging with CCG primitives.In Proceedings of the 5th Workshop on Representation Learning for NLP,pages 194–204. Online: Association for Computational Linguistics.

Chen, Zeming, Qiyue Gao, and Lawrence S. Moss. 2021. NeuralLog: Natural Language Inference with Joint Neural and Logical Reasoning. In Proceedings of *SEM 2021: The Tenth Joint Conference on Lexical and Computational Semantics, pages 78–88, Online. Association for Computational Linguistics.

Clark, Kevin, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018.Semi-supervised sequence modeling with cross-view training. In Proceedings ofthe 2018 Conference on Empirical Methods in Natural Language Processing,pages 1914–1925. Brussels, Belgium: Association for Computational Linguis-tics.

Clark, Stephen, Darren Foong, Luana Bulat, and Wenduan Xu. 2015. The Java version of the C&C parser: Version 0.95. Tech. rep., The University of Cambridge Computer Laboratory, Cambridge, UK.

References

Clark, Stephen, and James R. Curran. 2004. Parsing the WSJ Using CCG and Log-Linear Models. In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04), pages 103–110, Barcelona, Spain.

Clark, Stephen. 2002. Supertagging for Combinatory Categorial Grammar. In Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6), pages 19–24, Universitá di Venezia. Association for Computational Linguistics.

Clark, Stephen. 2021. Something old, something new:Grammar-based CCG parsing with transformer mod-els. CoRR, abs/2109.10044v2.

Collins, Michael and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In Proceedings of the 42nd Meeting of the ACL, pages111–118. Barcelona, Spain.

Collins, Michael. 1997. Three generative, lexicalised models for statistical pars-ing. In Proceedings of the 35th Meeting of the ACL, pages 16–23. Madrid,Spain.

Cooper,R.:Representingtypesasneuralevents. JournalofLogic,LanguageandInformation28, 131–155 (2019). DOI 10.1007/s10849-019-09285-4. URL https://link.springer.com/article/10.1007/s10849-019-09285-4

Dagan, I., Roth, D., Sammons, M. & Zanzotto, F.M. (2013). Recognizing Textual Entailment: Models andApplications. Synthesis Lectures on Human Language Technologies, pp. 1–220. Morgan and ClaypoolPublishers

Daido, Hinari and Daisuke Bekki. 2017. Developmentof an automated theorem prover for the fragment ofdts. In the 17th International Workshop on Logicand Engineering of Natural Language Semantics(LENLS17).

Fancellu, Federico, Sorcha Gilroy, Adam Lopez, andMirella Lapata. 2019. Semantic graph parsing withrecurrent neural network DAG grammars. In Pro-ceedings of the 2019 Conference on Empirical Meth-ods in Natural Language Processing and the 9th In-ternational Joint Conference on Natural LanguageProcessing (EMNLP-IJCNLP), pages 2769–2778,Hong Kong, China. Association for ComputationalLinguistics.

References

Hockenmaier, Julia and Mark Steedman. 2007. CCG-bank: A corpus of CCG derivations and dependencystructures extracted from the Penn treebank. Computational Linguistics, 33(3):355–396.

Hockenmaier, Julia, and Mark Steedman. 2002. Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pages 335–342, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Kogkalidis, Konstantinos, Michael Moortgat, and Tejaswini Deoskar. 2019. Constructive Type-Logical Supertagging With Self-Attention Networks. In Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019), pages 113–123, Florence, Italy. Association for Computational Linguistics.

Kogkalidis, Konstantinos, and Michael Moortgat. 2023. Geometry-Aware Supertagging with Heterogeneous Dynamic Convolutions. In Proceedings of the 2023 CLASP Conference on Learning with Small Data (LSD), pages 107–119, Gothenburg, Sweden. Association for Computational Linguistics.

Larsson,S.:Discreteandprobabilisticclassifier-basedsemantics.In:theProbabilityandMean-ing Conference (PaM 2020), pp. 62–68. Association for Computational Linguistics (2020).URL https://aclanthology.org/2020.pam-1.8

Lee, Kenton, Mike Lewis, and Luke Zettlemoyer. 2016. Global neural CCGparsing with optimality guarantees. In Proceedings of the 2016 Conference onEmpirical Methods in Natural Language Processing, pages 2366–2376. Austin,Texas: Association for Computational Linguistics.

Lee, Kenton, Mike Lewis, and Luke Zettlemoyer. 2016. Global neural CCGparsing with optimality guarantees. In Proceedings of the 2016 Conference onEmpirical Methods in Natural Language Processing, pages 2366–2376. Austin,Texas: Association for Computational Linguistics.

Lewis, Mike and Mark Steedman. 2014a. Improved CCG Parsing with Semi-supervised Supertagging. Transactions of the Association for Computational Linguistics, 2:327–338.

References

Lewis, Mike, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 221–231. San Diego, California: Association for Computational Linguistics.

Lewis, Mike, and Mark Steedman. 2014b. A* CCG Parsing with a Supertag-factored Model. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 990–1000, Doha, Qatar. Association for Computational Linguistics.

Liu, Jiangming, Shay B. Cohen, and Mirella Lapata.2018. Discourse representation structure parsing.In Proceedings of the 56th Annual Meeting of theAssociation for Computational Linguistics (Volume 1:Long Papers), pages 429–439, Melbourne, Australia.Association for Computational Linguistics.

Liu, Yufang, Tao Ji, Yuanbin Wu, and Man Lan.2021. Generating CCG categories. Proceedingsof the AAAI Conference on Artificial Intelligence,35(15):13443–13451.

Prange, Jakob, Nathan Schneider, and Vivek Srikumar. 2021. Supertagging the Long Tail with Tree-Structured Decoding of Complex Categories. Transactions of the Association for Computational Linguistics, 9:243–260.

Riezler, Stefan, Tracy H. King, Ronald M. Kaplan, Richard Crouch, JohnT. Maxwell III, and Mark Johnson. 2002. Parsing the Wall Street Journal us-ing a Lexical-Functional Grammar and discriminative estimation techniques.In Proceedings of the 40th Meeting of the ACL, pages 271–278. Philadelphia,PA.

Stanojevi´c, Miloˇs and Mark Steedman. 2020. Max-margin incremental CCGparsing. In Proceedings of the 58th Annual Meeting of the Association forComputational Linguistics, pages 4111–4122. Online: Association for Com-putational Linguistics.

References

Tian, Yuanhe, Yan Song, and Fei Xia. 2020. Supertagging Combinatory Cate-gorial Grammar with attentive graph convolutional networks. In Proceedingsof the 2020 Conference on Empirical Methods in Natural Language Processing(EMNLP), pages 6037–6044. Online: Association for Computational Linguis-tics.

Van Noord, Rik, Antonio Toral, and Johan Bos. 2020.Character-level representations improve DRS-basedsemantic parsing even in the age of BERT. InProceedings of the 2020 Conference on EmpiricalMethods in Natural Language Processing (EMNLP),pages 4587–4603, Online. Association for Computa-tional Linguistics.

Vaswani, Ashish, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertag-ging with LSTMs. In Proceedings of the 2016 Conference of the North Amer-ican Chapter of the Association for Computational Linguistics: Human Lan-guage Technologies, pages 232–237. San Diego, California: Association forComputational Linguistics.

Xu and Wenduan, Michael Auli, and Stephen Clark. 2015. CCG Supertagging with a Recurrent Neural Network. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 250–255, Beijing, China. Association for Computational Linguistics.

Xu, Wenduan. 2016. LSTM shift-reduce CCG parsing. In Proceedings ofthe 2016 Conference on Empirical Methods in Natural Language Processing,pages 1754–1764. Austin, Texas: Association for Computational Linguistics.

Yamaki, Ryosuke, Tadahiro Taniguchi, and Daichi Mochihashi. 2023. Holographic CCG Parsing. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 262–276, Toronto, Canada. Association for Computational Linguistics.

Yang, Xiulin, Jonas Groschwitz, Alexander Koller, and Johan Bos. 2024. Scope-enhanced Compositional Semantic Parsing for DRT. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 19602–19616, Miami, Florida, USA. Association for Computational Linguistics.

References

Yoshikawa, Masashi, Hiroshi Noji, and Yuji Matsumoto. 2017. A* CCG pars-ing with a supertag and dependency factored model. In Proceedings of the55th Annual Meeting of the Association for Computational Linguistics (Vol-ume 1: Long Papers), pages 277–287. Vancouver, Canada: Association forComputational Linguistics.

Zhang, Yue and Stephen Clark. 2011. Shift-reduce CCG parsing. In Proceedingsof the 49th Annual Meeting of the Association for Computational Linguistics:Human Language Technologies, pages 683–692. Portland, Oregon, USA: As-sociation for Computational Linguistics.

Zhao, Jinman, and Gerald Penn. 2024. LLM-supertagger: Categorial Grammar Supertagging via Large Language Models. In Findings of the Association for Computational Linguistics: EMNLP 2024, pages 697–705, Miami, Florida, USA. Association for Computational Linguistics.

宮川 夏菜子, 田上 青空, 戸次 大介, 依存型理論のための自動定理証明器Neural Waniの開発に向けて, 人工知能学会全国大会論文集 2025, JSAI2025 巻, 第39回 (2025), セッションID 3G5-GS-6-04, p. 3G5GS604, 公開日 2025/07/01, Online ISSN 2758-7347, https://doi.org/10.11517/pjsai.JSAI2025.0_3G5GS604, https://www.jstage.jst.go.jp/article/pjsai/JSAI2025/0/JSAI2025_3G5GS604/_article/-char/ja

Thank you for listening!