NeuralLog:
Natural Language Inference with Joint Neural and Logical Reasoning

Zeming Chen[†*] **Qiyue Gao**[†] **Lawrence S. Moss**[‡]
[†]Rose-Hulman Institute of Technology, Terre Haute, IN, USA
[‡]Indiana University, Bloomington, IN, USA
{chenz16,gaoq}@rose-hulman.edu
{lmoss}@indiana.edu

Abstract

Deep learning (DL) based language models achieve high performance on various benchmarks for Natural Language Inference (NLI). And at this time, symbolic approaches to NLI are receiving less attention. Both approaches (symbolic and DL) have their advantages and weaknesses. However, currently, no method combines them in a system to solve the task of NLI. To merge symbolic and deep learning methods, we propose an inference framework called NeuralLog, which utilizes both a monotonicity-based logical inference engine and a neural network language model for phrase alignment. Our framework models the NLI task as a classic search problem and uses the beam search algorithm to search for optimal inference paths. Experiments show that our joint logic and neural inference system improves accuracy on the NLI task and can achieve state-of-art accuracy on the SICK and MED datasets.

1 Introduction

Currently, many NLI benchmarks' state-of-the-art systems are exclusively deep learning (DL) based language models (Devlin et al., 2019; Lan et al., 2020; Liu et al., 2020; Yin and Schütze, 2017). These models often contain a large number of parameters, use high-quality pre-trained embeddings, and are trained on large-scale datasets, which enable them to handle diverse and large test data robustly. However, several experiments show that DL models lack generalization ability, adopt fallible syntactic heuristics, and show exploitation of annotation artifacts (Glockner et al., 2018; McCoy et al., 2019; Gururangan et al., 2018). On the other hand, there are logic-based systems that use symbolic reasoning and semantic formalism to solve NLI (Abzianidze, 2017; Martínez-Gómez et al., 2017; Yanaka et al., 2018; Hu et al., 2020). These systems show high precision on complex inferences involving difficult linguistic phenomena and present logical and explainable reasoning processes. However, these systems lack background knowledge and do not handle sentences with syntactic variations well, which makes them poor competitors with state-of-the-art DL models. Both DL and logic-based systems show a major issue with NLI models: they are too one-dimensional (either purely DL or purely logic), and no method has combined these two ap-

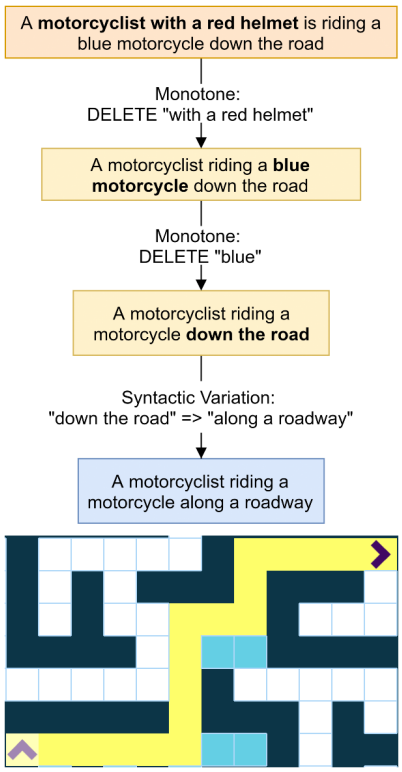


Figure 1: Analogy between path planning and an entailment inference path from the premise *A motorcyclist with a red helmet is riding a blue motorcycle down the road* to the hypothesis *A motorcyclist is riding a motorbike along a roadway*.

[*]The first two authors have equal contribution

Proceedings of the 10th Conference on Lexical and Computational Semantics, pages 78–88
August 5–6, 2021, Bangkok, Thailand (online)

proaches together for solving NLI.

This paper makes several contributions, as follows: first, we propose a new framework in section 3 for combining logic-based inference with deep-learning-based network inference for better performance on conducting natural language inference. We model an NLI task as a path-searching problem between the premises and the hypothesis. We use beam-search to find an optimal path that can transform a premise to a hypothesis through a series of inference steps. This way, different inference modules can be inserted into the system. For example, DL inference modules will handle inferences with diverse syntactic changes and logic inference modules will handle inferences that require complex reasoning. Second, we introduce a new method in section 4.3 to handle syntactic variations in natural language through sequence chunking and DL based paraphrase detection. We evaluate our system in section 6 by conducting experiments on the SICK and MED datasets. Experiments show that joint logical and neural reasoning show state-of-art accuracy and recall on these datasets.

2 Related Work

Perhaps the closest systems to NeuralLog are Yanaka et al. (2018), MonaLog (Hu et al., 2020), and Hy-NLI (Kalouli et al., 2020). Using Martínez-Gómez et al. (2016) to work with logic representations derived from CCG trees, Yanaka et al. (2018) proposed a framework that can detect phrase correspondences for a sentence pair, using natural deduction on semantic relations and can thus extract various paraphrases automatically. Their experiments show that assessing phrase correspondences helps improve NLI accuracy. Our system uses a similar methodology to solve syntactic variation inferences, where we determine if two phrases are paraphrases. Our method is rather different on this point, since we call on neural language models to detect paraphrases between two sentences. We feel that it would be interesting to compare the systems on a more theoretical level, but we have not done the comparison in this paper.

NeuralLog inherits the use of polarity marking found in MonaLog (Hu et al., 2020). (However, we use the dependency-based system of Chen and Gao (2021) instead of the CCG-based system of Hu and Moss (2018).) MonaLog did propose some integration with neural models, using BERT when logic failed to find entailment or contradiction. We are doing something very different, using neural models to detect paraphrases at several levels of "chunking". In addition, the exact algorithms found in Sections 3 and 4 are new here. In a sense, our work on alignment in NLI goes back to MacCartney and Manning (2009) where alignment was used to find a chain of edits that changes a premise to a hypothesis, but our work uses much that simply was not available in 2009.

Hy-NLI is a hybrid system that makes inferences using either symbolic or deep learning models based on how linguistically challenging a pair of sentences is. The principle Hy-NLI followed is that deep learning models are better at handling sentences that are linguistically less complex, and symbolic models are better for sentences containing hard linguistic phenomena. Although the system integrates both symbolic and neural methods, its decision process is still separate, in which the symbolic and deep learning sides make decisions without relying on the other side. Differently, our system incorporates logical inferences and neural inferences as part of the decision process, in which the two inference methods rely on each other to make a final decision.

3 Method

3.1 NLI As Path Planning

The key motivation behind our architecture and inference modules is that the Natural Language Inference task can be modeled as a path planning problem. Path planning is a task for finding an optimal path traveling from a start point to a goal containing a series of actions. To formulate NLI as path planning, we define the **premise** as the **start state** and the **hypothesis** as the **goal** that needs to be reached. The classical path planning strategy applies expansions from the start state through some search algorithms, such as depth-first-search or Dijkstra search, until an expansion meets the goal. In a grid map, two types of action produce an expansion. The vertical action moves up and down, and the horizontal action moves left and right. Similarly, language inference also contains these two actions. Monotonicity reasoning is a vertical action, where the monotone inference moves up and simplifies a sentence, and the antitone inference moves down and makes a sentence more specific. Syntactic variation and synonym replacement are horizontal actions. They change the form of a sentence while maintaining the original mean-

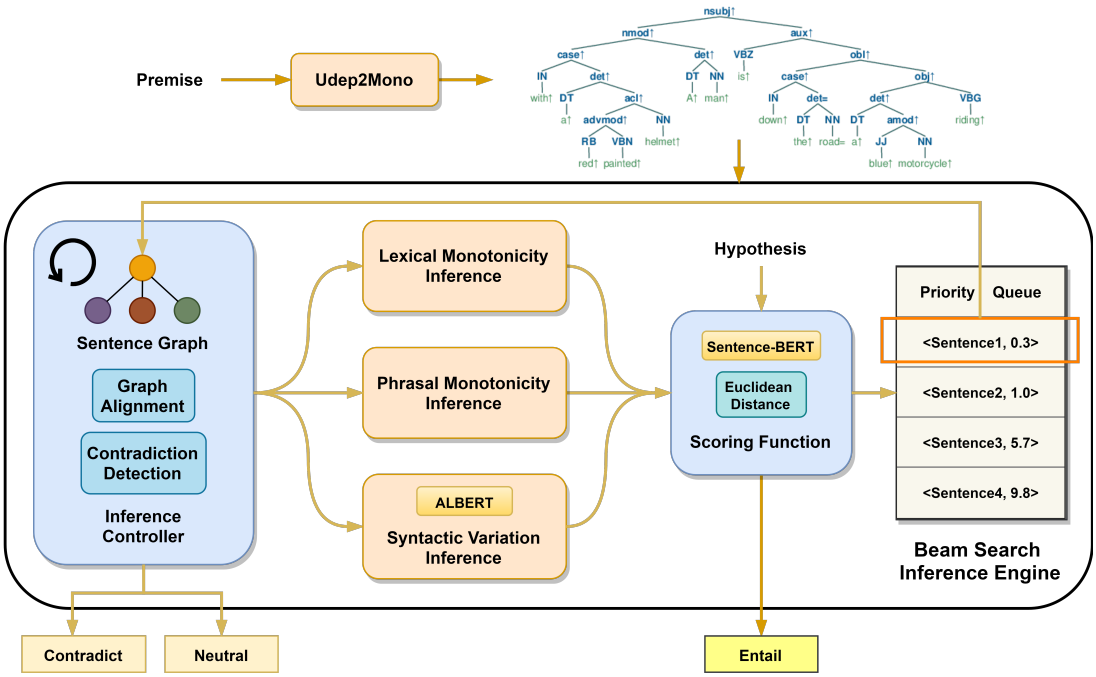


Figure 2: Overview system diagram of NeuralLog.

ing. Then, similar to path planning, we can continuously make inferences from the premise using a search algorithm to determine if the premise entails the hypothesis by observing whether one of the inferences can reach the hypothesis. If the hypothesis is reached, we can connect the list of inferences that transform a premise to a hypothesis to be the optimal path in NLI, a valid reasoning chain for entailment.

Figure 1 shows an analogy between an optimal path for the classical grid path planning problem and an example of an optimal inference path for NLI. On the top, we have a reasoning process for natural language inference. From the premise, we can first delete the modifier *with a red helmet*, then delete *blue* to get a simplified sentence. Finally, we can paraphrase *down the road* to *along a roadway* in the premise to reach the hypothesis and conclude the entailment relationship between these two sentences.

3.2 Overview

Our system contains four components: (1) a polarity annotator, (2) three sentence inference modules, (3) a search engine, and (4) a sentence inference controller. Figure 2 shows a diagram of the full system. The system first annotates a sentence with monotonicity information (polarity marks) using Udep2Mono (Chen and Gao, 2021). The polarity marks include monotone (↑), antitone (↓), and no monotonicity information (=) polarities. Next, the polarized parse tree is passed to the search engine. A beam search algorithm searches for the optimal inference path from a premise to a hypothesis. The search space is generated from three inference modules: lexical, phrasal, and syntactic variation. Through graph alignment, the sentence inference controller selects a inference module to apply to the premise and produce a set of new premises that potentially form entailment relations with the hypothesis. The system returns **Entail** if an inference path is found. Otherwise, the controller will determine if the premise and hypothesis form a contradiction by searching for counter example signatures and returns **Contradict** accordingly. If neither **Entail** nor **Contradict** is returned, the system returns **Neutral**.

3.3 Polarity Annotator

The system first annotates a given premise with monotonicity information using Udep2Mono, a polarity annotator that determines polarization of all constituents from universal dependency trees. The annotator first parses the premise into a binarized universal dependency tree and then conducts polarization by recursively marks polarity on each node . An example can be *Every*$^{\uparrow}$ *healthy*$^{\downarrow}$ *person*$^{\downarrow}$ *plays*$^{\uparrow}$ *sports*$^{\uparrow}$.

3.4 Search Engine

To efficiently search for the optimal inference path from a premise $\mathcal{P}$ to a hypothesis $\mathcal{H}$, we use a beam search algorithm which has the advantage of reducing search space by focusing on sentences with higher scores. To increase the search efficiency and accuracy, we add an inference controller that can guide the search direction.

Scoring In beam search, a priority queue $\mathcal{Q}$ maintains the set of generated sentences. A core operation is the determination of the highest-scoring generated sentence for a given input under a learned scoring model. In our case, the maximum score is equivalent to the minimum distance:

$$\mathbf{y}^{\star} = \arg\max_{\mathrm{s}\in\mathcal{S}} \mathrm{score}(\mathrm{s}, \mathcal{H})$$

$$\mathbf{y}^{\star} = \arg\min_{\mathrm{s}\in\mathcal{S}} \mathrm{dist}(\mathrm{s}, \mathcal{H})$$

where $\mathcal{H}$ is the hypothesis and $\mathcal{S}$ is a set of generated sentences produced by the three (lexical, phrasal, syntactic variation) inference modules. We will present more details about these inference modules in section 4. We formulate the distance function as the Euclidean distance between the sentence embeddings of the premise and hypothesis. To obtain semantically meaningful sentence embeddings efficiently, we use Reimers and Gurevych (2019)'s language model, Sentence-BERT (SBERT), a modification of the BERT model. It uses siamese and triplet neural network structures to derive sentence embeddings which can be easily compared using distance functions.

3.5 Sentence Inference Controller

In each iteration, the search algorithm expands the search space by generating a set of potential sentences using three inference modules: (1) lexical inference, (2) phrasal inference, and (3) syntactic variation inference. To guide the search engine to select the most applicable module, we designed a inference controller that can recommend which of the labels the overall algorithm should proceed with. For example, for a premise *All animals eat food* and a hypothesis *All dogs eat food*, only a lexical inference of *animals* to *dogs* would be needed. Then, the controller will apply the lexical inference to the premise, as we discuss below.

3.5.1 Sentence Representation Graph

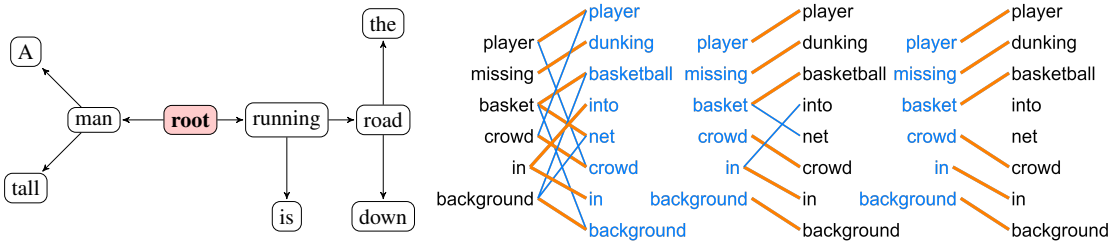
The controller makes its decision based on graph-based representations for the premise and the hypothesis. We first build a sentence representation graph from parsed input using Universal Dependencies. Let $\mathcal{V} = \mathcal{V}_m \cup \mathcal{V}_c$ be the set of vertices of a sentence representation graph, where $\mathcal{V}_m$ represents the set of modifiers such as *tall* in Figure 5, and V_c represents the set of content words (words that are being modified) such as *man* in Figure 5. While content words in $\mathcal{V}_c$ could modify other content words, modifiers in $\mathcal{V}_m$ are not modified by other vertices. Let $\mathcal{E}$ be the set of directed edges in the form $\langle v_c, v_m \rangle$ such that $v_m \in \mathcal{V}_m$ and $v_c \in \mathcal{V}_c$. A sentence representation graph is then defined as a tuple $\mathrm{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. Figure 3a shows an example graph.

3.5.2 Graph Alignment

To observe the differences between two sentences, we rely on graph alignment between two sentence representation graphs. We first align nodes from subjects, verbs and objects, which constitutes what we call a component level. Define G_p as the graph for a premise and G_h as the graph for a hypothesis. Then, $\mathcal{C}_p$ and $\mathcal{C}_h$ are component level nodes from the two graphs. We take the Cartesian product $\mathcal{C}_p \times \mathcal{C}_h = \{(\mathrm{c}_p, \mathrm{c}_h) : \mathrm{c}_p \in \mathcal{C}_p, \mathrm{c}_h \in \mathcal{C}_h\}$. In the first round, we recursively pair the child nodes of each c_p to child nodes of each c_h. We compute word similarity between two child nodes c_p^i and c_h^i and eliminate pairs with non-maximum similarity. We denote the new aligned pairs as a set $\mathcal{A}^*$. At the second round, we iterate through the aligned pairs in $\mathcal{A}^*$. If multiple child nodes from the first graph are paired to a child node in the second graph, we only keep the pair with maximum word similarity. In the final round, we perform the same check for each child node in the first graph to ensure that there are no multiple child nodes from the second graph paired to it. Figure 3b shows a brief visualization of the alignment process.

3.5.3 inference Module Recommendation

After aligning the premise graph $\mathcal{G}_p$ with hypothesis graph $\mathcal{G}_h$, the controller checks through each node in the two graphs. If a node does not get aligned, the controller considers to delete the node or insert it depending on which graph the node belongs to and recommends phrasal inference. If a node is different from its aligned node, the controller recommends lexical inference. If additional lexical or phrasal inferences are detected under this node, the controller decides that there is a more complex transition under this node and rec-



(a) Sentence representation graph

(b) Graph alignment visualization

Figure 3: (a) A sentence representation graph for *A tall man is running down the road*. (b) Visualization for the graph alignment. The lines between two words represent their similarity. The orange lines are the pairs with maximum similarities for a blue word. Through bi-directional alignment, we eliminate word pairs with non-maximum similarity and gets the final alignment pairs.

ommends a syntactic variation.

3.5.4 Contradiction Detection

We determine whether the premise and the hypothesis contradict each other inside the controller by searching for potential contradiction transitions from the premise to the hypothesis. For instance, a transition in the scope of the quantifier (*a* $\longrightarrow$ *no*) from the same subject could be what we call a contradiction signature (possible evidence for a contradiction). With all the signatures, the controller decides if they can form a contradiction as a whole. To avoid situations when multiple signatures together fail to form a complete contradiction, such as double negation, the controller checks through the contradiction signatures to ensure a contradiction. For instance, in the verb pair (*not remove*, *add*), the contradiction signature *not* would cancel the verb negation contradiction signature from *remove* to *add* so the pair as a whole would not be seen as a contradiction. Nevertheless, other changes from the premise to the hypothesis may change the meaning of the sentence. Hence, our controller would go through other transitions to make sure the meaning of the sentence does not change when the contradiction sign is valid. For example, in the neutral pair P: *A person is eating* and H: *No tall person is eating*, the addition of *tall* would be detected by our controller. But the aligned word of the component it is applied to, *person* in P, has been marked downward monotone. So this transition is invalid. This pair would then be classified as neutral.

For P2 and H2 in Figure 4, the controller notices the contradictory quantifier change around the subject *man*. The subject *man* in P2 is upward monotone so the deletion of *tall* is valid. Our controller also detects the meaning transition from *down the road* to *inside the building*, which affects the sentence's meaning and cancels the previous contradiction signature. The controller thus will not classify P2 and H2 as a pair of contradiction.

signature type	example
quantifier negation	**no** dogs $\Longrightarrow$ **some** dogs
verb negation	is **eating** $\Longrightarrow$ is **not eating**
noun negation	**some people** $\Longrightarrow$ **nobody**
action contradiction	is **sleeping** $\Longrightarrow$ is **running**
direction contradiction	The **turtle** is following the **fish** $\Longrightarrow$ The **fish** is following the **turtle**

Table 1: Examples of contradiction signatures.

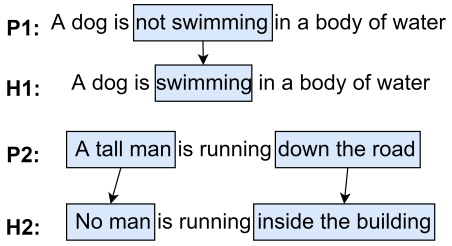


Figure 4: Example of contradiction signatures. P1 and H1 form a contradiction. P2 and H2 does not form a contradiction because the meaning after the verb *running* has changed.

4 Inference Generation

4.1 Lexical Monotonicity Inference

Lexical inference is word replacement based on monotonicity information for key-tokens including nouns, verbs, numbers, and quantifiers. The system uses lexical knowledge bases including WordNet (Miller, 1995) and ConceptNet (Liu and Singh, 2004). From the knowledge bases, we extract four word sets: hypernyms, hyponyms, synonyms, and antonyms. Logically, if a word has a monotone polarity (↑), it can be replaced by its hypernyms. For example, *swim* $\leq$ *move*; then *swim* can be replaced with *move*. If a word has an antitone polarity (↓),

it can be replaced by its hyponyms. For example, *flower* $\geq$ *rose*. Then, *flower* can be replaced with *rose*. We filter out irrelevant words from the knowledge bases that do not appear in the hypothesis. Additionally, we handcraft knowledge relations for words like quantifiers and prepositions that do not have sufficient taxonomies from knowledge bases. Some handcrafted relations include: *all* = *every* = *each* $\leq$ *most* $\leq$ *many* $\leq$ *several* $\leq$ *some* = *a*, *up* $\perp$ *down*.

4.2 Phrasal Monotonicity Inference

Phrasal replacements are for phrase-level monotonicity inference. For example, with a polarized sentence *A* $^{\uparrow}$ *woman*$^{\uparrow}$ *who*$^{\uparrow}$ *is*$^{\uparrow}$ *beautiful*$^{\uparrow}$ *is*$^{\uparrow}$ *walking*$^{\uparrow}$ *in*$^{\uparrow}$ *the*$^{\uparrow}$ *rain*$^{=}$, the monotone mark $^{\uparrow}$ on *woman* allows an upward inference: *woman* $\sqsupseteq$ *woman who is beautiful*, in which the relative clause *who is beautiful* is deleted. The system follows a set of phrasal monotonicity inference rules. For upward monotonicity inference, modifiers of a word are deleted. For downward monotonicity inference, modifiers are inserted to a word. The algorithm traverses down a polarized UD parse tree, deletes the modifier sub-tree if a node is monotone ($\uparrow$), and inserts a new sub-tree if a node is antitone ($\downarrow$). To insert new modifiers, the algorithm extracts a list of potential modifiers associated to a node from a modifier dictionary. The modifier dictionary is derived from the hypothesis and contains word-modifier pairs for each dependency relation. Below is an example of a modifier dictionary from *There are no beautiful flowers that open at night*:

- **obl**: [head: *open*, mod: *at night*]
- **amod**: [head: *flowers*, mod: *beautiful*]
- **acl:relcl**: [head: *flowers*, mod: *that open at night*]

4.3 Syntactic Variation Inference

We categorize linguistic changes between a premise and a hypothesis that cannot be inferred from monotonicity information as *syntactic variations*. For example, a change from *red rose* to *a rose which is red* is a syntactic variation. Many logical systems rely on handcrafted rules and manual transformation to enable the system to use syntactic variations. However, without accurate alignments between the two sentences, these methods are not robust enough, and thus are difficult to scale up for wide-coverage input.

Recent development of pretrained transformer-based language models are showing state-of-art performance on multiple benchmarks for Natural Language Understanding (NLU) including the task for paraphrase detection (Devlin et al., 2019; Lan et al., 2020; Liu et al., 2020) exemplify phrasal knowledge of syntactic variation. We propose a method that incorporates transformer-based language models to robustly handle syntactic variations. Our method first uses a sentence chunker to decompose both the premise and the hypothesis into chunks of phrases and then forms a Cartesian product of chunk pairs. For each pair, we use a transformer model to calculate the likelihood of a pair of chunks being a pair of paraphrases.

4.3.1 Sequence Chunking

To obtain phrase-level chunks from a sentence, we build a sequence chunker to extract chunks from a sentence using its universal dependency information. Instead of splitting a sentence into chunks, our chunker composes word tokens recursively to form meaningful chunks. First, we construct a sentence representation graph of a premise from the controller. Recall that a sentence representation graph is defined as $\mathrm{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V} = \mathcal{V}_m \cup \mathcal{V}_c$ is the set of modifiers ($\mathcal{V}_m$) and content words ($\mathcal{V}_c$), and $\mathcal{E}$ is the set of directed edges. To generate the chunk for a content word in $\mathcal{V}_c$, we arrange its modifiers, which are nodes it points to, together with the content word by their word orders in the original sentence to form a word chain. Modifiers that make the chain disconnected are discarded because they are not close enough to be part of the chunk. For instance, the chunk from the verb *eats* in the sentence *A person eats the food carefully* would not contain its modifier *carefully* because they are separated by the object *the food*. If the sentence is stated as *A person carefully eats the food*, *carefully* now is next to *eat* and it would be included in the chunk of the verb *eat*. To obtain chunks for a sentence, we iterate through each main component node, which is a node for subject, verb, or object, in the sentence's graph representation and construct verb phrases by combining verbs' chunks with their paired objects' chunks. There are cases when a word modifies other words and gets modified in the same time. They often occur when a chunk serves as a modifier. For example, in *The woman in a pink dress is dancing*, the phrase *in a pink dress* modifies *woman* whereas *dress* is modified by *in*, *a* and *pink*. Then edges from *dress* to *in*, *a*, *pink* with the edge from *woman* to *dress* can be drawn. Chunks *in a pink dress* and *the woman in a*

Type	Premise	Hypothesis
Verb Phrase Variation	Two men are standing near the water and are **holding fishing poles**	Two men are standing near the water and are **holding tools used for fishing**
Noun Phrase Variation	A man with climbing equipment is hanging from **rock which is vertical and white**	A man with equipment used for climbing is hanging from a **white, vertical rock**.

Table 2: Examples of phrasal alignments detected by the syntactic variation module

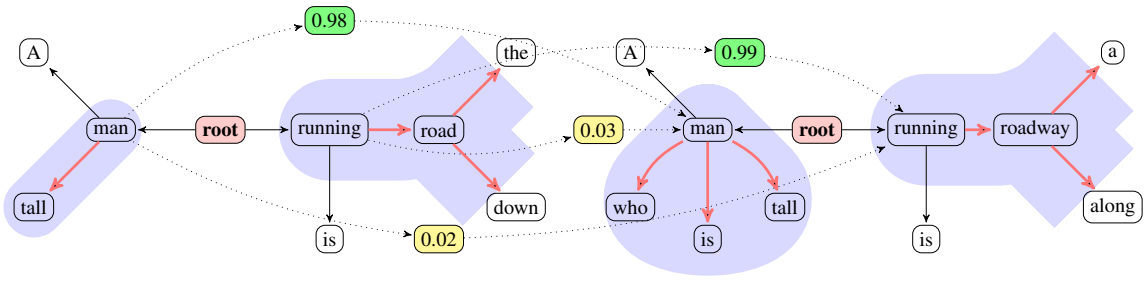


Figure 5: A graph representation of the monolingual phrase alignment process. Here the left graph represents the premise: *A tall man is running down the road*. The right graph represents the hypothesis *A man who is tall is running along a roadway*. The blue region represents phrase chunks extracted by the chunker from the graph. An alignment score is calculated for each pair of chunks. The pair ⟨ *tall man*, *man who is tall* ⟩ is a pair of paraphrases, and thus has a high alignment score (0.98). The pair ⟨ *tall man*, *running along a road way* ⟩ has two unrelated phrases, and thus has a low alignment score(0.03).

pink dress will be generated for *dress* and *woman* respectively.

4.3.2 Monolingual Phrase Alignment

After the chunker outputs a set of chunks from a generated sentence and from the hypothesis, the system selects chunk pairs that are aligned by computing an alignment score for each pair of chunks. Formally, we define $\mathcal{C}_s$ as the set of chunks from a generated sentence and $\mathcal{C}_h$ as the set of chunks from the hypothesis. We build the Cartesian product from $\mathcal{C}_s$ and $\mathcal{C}_h$, denoted $\mathcal{C}_s \times \mathcal{C}_h$. For each chunk pair $(c_{si}, c_{hj}) \in \mathcal{C}_s \times \mathcal{C}_h$, we compute an alignment score $\boldsymbol{\alpha}$:

$$\mathbf{y}_{\langle \mathbf{c_{si}}, \mathbf{c_{hi}} \rangle} = \text{ALBERT.forward}(\langle \mathbf{c_{si}}, \mathbf{c_{hi}} \rangle)$$

$$\boldsymbol{\alpha}_{\langle \mathbf{c_{si}}, \mathbf{c_{hi}} \rangle} = \text{p}(\mathbf{c_{si}} \mid \mathbf{c_{hj}})$$

$$\boldsymbol{\alpha}_{\langle \mathbf{c_{si}}, \mathbf{c_{hi}} \rangle} = \frac{\exp^{\mathbf{y}_{\langle \mathbf{c_{si}}, \mathbf{c_{hi}} \rangle_0}}}{\sum_{j=1}^{2} \exp^{\mathbf{y}_{\langle \mathbf{c_{si}}, \mathbf{c_{hi}} \rangle_j}}}$$

If $\boldsymbol{\alpha} > 0.85$, the system records this pair of phrases as a pair of syntactic variation. To calculate the alignment score, we use an ALBERT (Lan et al., 2020) model for the paraphrase detection task, fine tuned on the Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005). We first pass the chunk pair to ALBERT to obtain the logits. Then we apply a softmax function to the logits to get the final probability. A full demonstration of the alignment between chunks is shown in Figure 5.

5 Data

5.1 The SICK Dataset

The SICK (Marelli et al., 2014) dataset is an English benchmark that provides in-depth evaluation for compositional distribution models. There are 10,000 English sentence pairs exhibiting a variety of lexical, syntactic, and semantic phenomena. Each sentence pair is annotated as Entailment, Contradiction, or Neutral. we use the 4,927 test problems for evaluation.

5.2 The MED Dataset

The Monotonicity Entailment Dataset (MED), is a challenge dataset designed to examine a model's ability to conduct monotonicity inference (Yanaka et al., 2019a). There are 5382 sentence pairs in MED, where 1820 pairs are upward inference problems, 3270 pairs are downward inference problems, and 292 pairs are problems with no monotonicity information. MED's problems cover a variety of linguistic phenomena, such as lexical knowledge, reverse, conjunction and disjunction, conditional, and negative polarity items.

6 Evaluation

6.1 Experiment Setup

For Universal Dependency parsing, we follow Chen and Gao (2021)'s framework and use a parser

Model	P	R	acc.
ML/DL-based systems			
BERT (base, uncased)	86.8	85.4	86.7
Yin and Schütze (2017)	–	–	87.1
Beltagy et al. (2016)	–	–	85.1
Logic-based systems			
Abzianidze (2017)	98.0	58.1	81.4
Martínez-Gómez et al. (2017)	97.0	63.6	83.1
Yanaka et al. (2018)	84.2	77.3	84.3
Hu et al. (2020)	83.8	70.7	77.2
Abzianidze (2020)	94.3	67.9	84.4
Hybrid System			
Hu et al. (2020)+BERT	83.2	85.5	85.4
Kalouli et al. (2020)	–	–	86.5
Our System			
NeuralLog (full system)	88.0	**87.6**	**90.3**
– ALBERT-SV	68.9	79.3	71.4
– Monotonicity	74.5	75.1	74.7

Table 3: Performance on the SICK test set

from Stanford's natural language analysis package, Stanza (Qi et al., 2020). In the parser, we use a neural parsing model pretrained on the UD English GUM corpus (Zeldes, 2017) with 90.0 LAS (Zeman et al., 2018) evaluation score. For Sentence-BERT, we selected the BERT-large model pre-trained on STS-B (Cer et al., 2017). For ALBERT, we used textattack's ALBERT-base model pretrained on MRPC from transformers. For word alignment in the controller, we select Řehůřek and Sojka (2010)'s Gensim framework to calculate word similarity from pre-trained word embedding. We evaluated our model on the SICK and MED datasets using the standard NLI evaluation metrics of accuracy, precision, and recall. Additionally, we conducted two ablation tests focusing on analyzing the contributions of the monotonicity inference modules and the syntactic variation module.

6.2 Results

SICK Table 3 shows the experiment results tested on SICK. We compared our performance to several logic-based systems as well as two deep learning based models. As the evaluation results show, our model achieves the state-of-art performance on the SICK dataset. The best logic-based model is Abzianidze (2020) with 84.4 percent accuracy. The best DL-based model is Yin and Schütze (2017) with 87.1 percent accuracy. Our system outperforms both of them with 90.3 percent accuracy. Compare to Hu et al. (2020) + BERT, which also explores a way of combining logic-based methods and deep learning based methods, our system shows higher accuracy with a 4.92 percentage point increase. In addition, our system's accuracy has a 3.8 percentage point increase than another hybrid system, Hy-NLI (Kalouli et al., 2020). The good performance proves that our framework for joint logic and neural reasoning can achieve state-of-art performance on inference and outperforms existing systems.

Model	Up	Down	All
DeComp (Parikh et al., 2016)	71.1	45.2	51.4
ESIM (Chen et al., 2017)	66.1	42.1	53.8
BERT (Devlin et al., 2019)	82.7	22.8	44.7
BERT+ (Yanaka et al., 2019a)	76.0	70.3	71.6
NeuralLog (ours)	**91.4**	**93.9**	**93.4**

Table 4: Results comparing model compared to state-of-art NLI models evaluated on MED. **Up**, **Down**, and **All** stand for the accuracy on upward inference, downward inference, and the overall dataset.

Ablation Test In addition to the standard evaluation on SICK, we conducted two ablation tests. The results are included in Table 3. First, we remove the syntactic variation module that uses neural network for alignment (−ALBERT-SV). As the table shows, the accuracy drops 18.9 percentage points. This large drop in accuracy indicates that the syntactic variation module plays a major part in our overall inference process. The result also proves our hypothesis that deep learning methods for inference can improve the performance of traditional logic-based systems significantly. Secondly, when we remove the monotonicity-based inference modules (−Monotonicity), the accuracy shows another large decrease in accuracy, with a 15.6 percentage point drop. This result demonstrates the important contribution of the logic-based inference modules toward the overall state-of-the-art performance. Compared to the previous ablation test which removes the neural network based syntactic variation module, the accuracy does not change much (only 3.3 differences). This similar performance indicates that neural network inference in our system alone cannot achieve state-of-art performance on the SICK dataset, and additional guidance and constrains from the logic-based methods are essential parts of our framework. Overall, we believe that the results reveal that both modules, logic and neural, contribute equally to the final performance and are both important parts that are unmovable.

MED Table 4 shows the experimental results tested on MED. We compared to multiple deep

learning based baselines. Here, DeComp and ESIM are trained on SNLI and BERT is fine-tuned with MultiNLI. The BERT+ model is a BERT model fine-tuned on a combined training data with the HELP dataset, (Yanaka et al., 2019b), a set of augmentations for monotonicity reasoning, and the MultiNLI training set. Both models were tested in Yanaka et al. (2019a). Overall, our system (NeuralLog) outperforms all DL-based baselines in terms of accuracy, by a significant amount. Compared to BERT+, our system performs better both on upward (+15.4) and downward (+23.6) inference, and shows significant higher accuracy overall (+21.8). The good performance on MED validates our system's ability on accurate and robust monotonicity-based inference.

6.3 Error Analysis

For entailment, a large amount of inference errors are due to an incorrect dependency parse trees from the parser. For example, P: *A black, red, white and pink dress is being worn by a woman*, H: *A dress, which is black, red, white and pink is being worn by a woman*, has long conjunctions that cause the parser to produce two separate trees from the same sentence. Secondly, a lack of sufficient background knowledge causes the system to fail to make inferences which would be needed to obtain a correct label. For example, P: *One man is doing a bicycle trick in midair*, H: *The cyclist is performing a trick in the air* requires the system to know that *a man doing a bicycle trick* is a *cyclist*. This kind of knowledge can only be injected to the system either by handcrafting rules or by extracting it from the training data. For contradiction, our analysis reveals inconsistencies in the SICK dataset. We account for multiple sentence pairs that have the same syntactic and semantic structures, but are labeled differently. For example, P: *A man is folding a tortilla*, H: *A man is unfolding a tortilla* has gold-label **Neutral** while P: *A man is playing a guitar*, H: *A man is not playing a guitar* has gold-label **Contradiction**. These two pair of sentences clearly have similar structures but have inconsistent gold-labels. Both gold-labels would be reasonable depending on whether the two subjects refer to the same entity.

7 Conclusion and Future Work

In this paper, we presented a framework to combine logic-based inference with deep-learning based inference for improved Natural Language Inference performance. The main method is using a search engine and an alignment based controller to dispatch the two inference methods (logic and deep-learning) to their area of expertise. This way, logic-based modules can solve inference that requires logical rules and deep-learning based modules can solve inferences that contain syntactic variations which are easier for neural networks. Our system uses a beam search algorithm and three inference modules (lexical, phrasal, and syntactic variation) to find an optimal path that can transform a premise to a hypothesis. Our system handles syntactic variations in natural sentences using the neural network on phrase chunks, and our system determines contradictions by searching for contradiction signatures (evidence for contradiction). Evaluations on SICK and MED show that our proposed framework for joint logical and neural reasoning can achieve state-of-art accuracy on these datasets. Our experiments on ablation tests show that neither logic nor neural reasoning alone fully solve Natural Language Inference, but a joint operation between them can bring improved performance.

For future work, one plan is to extend our system with more logic inference methods such as those using dynamic semantics (Haruta et al., 2020) and more neural inference methods such as those for commonsense reasoning (Levine et al., 2020). We also plan to implement a learning method that allows the system to learn from mistakes on a training dataset and automatically expand or correct its rules and knowledge bases, which is similar to Abzianidze (2020)'s work.

Acknowledgements

We thank the anonymous reviewers for their insightful comments. We also thank Dr. Michael Wollowski from Rose-hulman Institute of Technology for his helpful feedback on this paper.

References

Lasha Abzianidze. 2017. LangPro: Natural language theorem prover. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 115–120, Copenhagen, Denmark. Association for Computational Linguistics.

Lasha Abzianidze. 2020. Learning as abduction: Trainable natural logic theorem prover for natural language inference. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Seman-*

tics, pages 20–31, Barcelona, Spain (Online). Association for Computational Linguistics.

I. Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. 2016. Representing meaning with a combination of logical and distributional models. *Computational Linguistics*, 42(4):763–808.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Zeming Chen and Qiyue Gao. 2021. Monotonicity marking from universal dependency trees. *CoRR*, abs/2104.08659.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Izumi Haruta, Koji Mineshima, and Daisuke Bekki. 2020. Combining event semantics and degree semantics for natural language inference. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1758–1764, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Hai Hu, Qi Chen, Kyle Richardson, Atreyee Mukherjee, Lawrence S. Moss, and Sandra Kuebler. 2020. MonaLog: a lightweight system for natural language inference based on monotonicity. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 334–344, New York, New York. Association for Computational Linguistics.

Hai Hu and Larry Moss. 2018. Polarity computations in flexible categorial grammar. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 124–129, New Orleans, Louisiana. Association for Computational Linguistics.

Aikaterini-Lida Kalouli, Richard Crouch, and Valeria de Paiva. 2020. Hy-NLI: a hybrid system for natural language inference. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5235–5249, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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 on Learning Representations*.

Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2020. SenseBERT: Driving some sense into BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4656–4667, Online. Association for Computational Linguistics.

H. Liu and P. Singh. 2004. Conceptnet — a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{bert}a: A robustly optimized {bert} pretraining approach.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the Eighth International Conference on Computational Semantics (IWCS-8)*, Tilburg, Netherlands.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2016. ccg2lambda: A compositional semantics system. In *Proceedings*

of ACL 2016 System Demonstrations, pages 85–90, Berlin, Germany. Association for Computational Linguistics.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2017. On-demand injection of lexical knowledge for recognising textual entailment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 710–720, Valencia, Spain. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019a. Can neural networks understand monotonicity reasoning? In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 31–40, Florence, Italy. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019b. HELP: A dataset for identifying shortcomings of neural models in monotonicity reasoning. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 250–255, Minneapolis, Minnesota. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, Pascual Martínez-Gómez, and Daisuke Bekki. 2018. Acquisition of phrase correspondences using natural deduction proofs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 756–766, New Orleans, Louisiana. Association for Computational Linguistics.

Wenpeng Yin and Hinrich Schütze. 2017. Task-specific attentive pooling of phrase alignments contributes to sentence matching. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 699–709, Valencia, Spain. Association for Computational Linguistics.

Amir Zeldes. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.