MonaLog: a Lightweight System for Natural Language Inference Based on Monotonicity

Hai Hu[†] **Qi Chen**[†] **Kyle Richardson**[‡]
Atreyee Mukherjee[†] **Lawrence S. Moss**[†] **Sandra Kübler**[†]
[†]Indiana University, Bloomington, IN, USA
[‡]Allen Institute for Artificial Intelligence, Seattle, WA, USA
{huhai,qc5,atremukh,lmoss,skuebler}@indiana.edu
kyler@allenai.org

Abstract

We present a new logic-based inference engine for natural language inference (NLI) called MonaLog, which is based on natural logic and the monotonicity calculus. In contrast to existing logic-based approaches, our system is intentionally designed to be as lightweight as possible, and operates using a small set of well-known (surface-level) monotonicity facts about quantifiers, lexical items and token-level polarity information. Despite its simplicity, we find our approach to be competitive with other logic-based NLI models on the SICK benchmark. We also use MonaLog in combination with the current state-of-the-art model BERT in a variety of settings, including for compositional data augmentation. We show that MonaLog is capable of generating large amounts of high-quality training data for BERT, improving its accuracy on SICK.

1 Introduction

There has been rapid progress on natural language inference (NLI) in the last several years, due in large part to recent advances in neural modeling (Conneau et al., 2017) and the introduction of several new large-scale inference datasets (Marelli et al., 2014; Bowman et al., 2015; Williams et al., 2018; Khot et al., 2018). Given the high performance of current state-of-the-art models, there has also been interest in understanding the limitations of these models (given their uninterpretability) (Naik et al., 2018; McCoy et al., 2019), as well as finding systematic biases in benchmark datasets (Gururangan et al., 2018; Poliak et al., 2018).

In parallel to these efforts, there have also been recent logic-based approaches to NLI (Mineshima et al., 2015; Martínez-Gómez et al., 2016; Martínez-Gómez et al., 2017; Abzianidze, 2017; Yanaka et al., 2018), which take inspiration from linguistics. In contrast to early attempts at using logic (Bos and Markert, 2005), these approaches have proven to be more robust. However they tend to use many rules and their output can be hard to interpret. It is sometimes unclear whether the attendant complexity is justified, especially given that such models are currently far outpaced by data-driven models and are generally hard to hybridize with data-driven techniques.

In this work, we introduce a new logical inference engine called MonaLog, which is based on natural logic and work on monotonicity stemming from van Benthem (1986). In contrast to the logical approaches cited above, our starting point is different in that we begin with the following two questions: 1) what is the *simplest* logical system that one can come up with to solve empirical NLI problems (i.e., the system with minimal amounts of primitives and background knowledge)?; and 2) what is the lower-bound performance of such a model? Like other approaches to natural logic (MacCartney and Manning, 2008; Angeli and Manning, 2014), our model works by reasoning over surface forms (as opposed to translating to symbolic representations) using a small inventory of monotonicity facts about quantifiers, lexical items and token-level polarity (Hu and Moss, 2018); *proofs* in the calculus are hence fully interpretable and expressible in ordinary language. Unlike existing work on natural logic, however, our model avoids the need for having expensive alignment and search sub-procedures (MacCartney et al., 2008; Stern and Dagan, 2011), and relies on a much smaller set of background knowledge and primitive relations than MacCartney and Manning (2009).

To show the effectiveness of our approach, we show results on the SICK dataset (Marelli et al., 2014), a common benchmark for logic-based NLI, and find MonaLog to be competitive with more complicated logic-based approaches (many

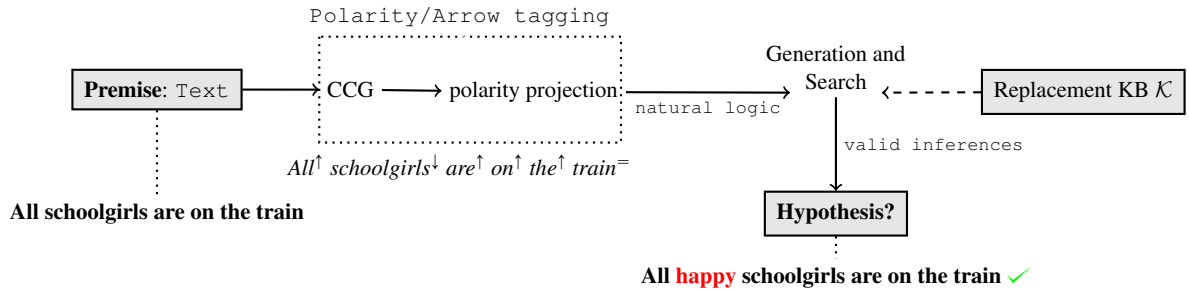

Figure 1: An illustration of our general monotonicity reasoning pipeline using an example premise and hypothesis pair: *All schoolgirls are on the train* and *All happy schoolgirls are on the train*.

of which require full semantic parsing and more complex logical machinery). We also introduce a supplementary version of SICK that corrects several common annotation mistakes (e.g., asymmetrical inference annotations) based on previous work by Kalouli et al. (2017, 2018)[1]. Positive results on both these datasets show the ability of lightweight monotonicity models to handle many of the inferences found in current NLI datasets, hence putting a more reliable lower-bound on what results the simplest logical approach is capable of achieving on this benchmark.

Since our logic operates over surface forms, it is straightforward to hybridize our models. We investigate using MonaLog in combination with the language model BERT (Devlin et al., 2019), including for *compositional data augmentation*, i.e, re-generating entailed versions of examples in our training sets. To our knowledge, our approach is the first attempt to use monotonicity for data augmentation, and we show that such augmentation can generate high-quality training data with which models like BERT can improve performance.

2 Our System: MonaLog

The goal of NLI is to determine, given a premise set P and a hypothesis sentence H, whether H follows from the meaning of P (Dagan et al., 2005). In this paper, we look at single-premise problems that involve making a standard 3-way classification decision (i.e., Entailment (H), Contradict (C) and Neutral (N)). Our general monotonicity reasoning system works according to the pipeline in Figure 1. Given a premise text, we first do `Arrow Tagging` by assigning polarity annotations (i.e., the arrows $\uparrow, \downarrow$, which are the basic primitives of our logic) to tokens in text. These *surface-level* annotations, in turn, are associated with a set of `natural logic` inference rules that provide instructions for how to generate entailments and contradictions by span replacements over these arrows (which relies on a library of span replacement rules). For example, in the sentence *All schoolgirls are on the train*, the token *schoolgirls* is associated with a polarity annotation $\downarrow$, which indicates that in this sentential context, the span *schoolgirls* can be replaced with a semantically more specific concept (e.g., *happy schoolgirls*) in order to generate an entailment. A `generation and search` procedure is then applied to see if the hypothesis text can be generated from the premise using these inference rules. A *proof* in this model is finally a particular sequence of edits (e.g., see Figure 2) that derive the hypothesis text from the premise text rules and yield an entailment or contradiction.

In the following sections, we provide the details of our particular implementation of these different components in MonaLog.

2.1 Polarization (Arrow Tagging)

Given an input premise P, MonaLog first polarizes each of its tokens and constituents, calling the system described by Hu and Moss (2018)[2], which performs polarization on a CCG parse tree. For example, a polarized P could be $every^{\uparrow}$ $linguist^{\downarrow}$ $swim^{\uparrow}$. Note that since we ignore morphology in the system, tokens are represented by lemmas.

2.2 Knowledge Base $\mathcal{K}$ and Sentence Base $\mathcal{S}$

MonaLog utilizes two auxiliary sets. First, a knowledge base $\mathcal{K}$ that stores the world knowledge needed for inference, e.g., $semanticist \leqslant linguist$ and $swim \leqslant move$, which captures the facts that $[\![semanticist]\!]$ denotes a subset of $[\![linguist]\!]$,

[1] Our correction can be found at: https://github.com/huhailinguist/SICK_correction

[2] https://github.com/huhailinguist/ccg2mono

and that $[\![swim]\!]$ denotes a subset of $[\![move]\!]$, respectively. Such world knowledge can be created manually for the problem at hand, or derived easily from existing resources such as WordNet (Miller, 1995). Note that we do not blindly add *all* relations from WordNet to our knowledge base, since this would hinge heavily on word sense disambiguation (we need to know whether the "bank" is a financial institution or a river bank to extract its relations correctly). In the current implementation, we avoid this by adding $x \leqslant y$ or $x \perp$[3] y relations only if both x and y are words in the premise-hypothesis pair.[4] Additionally, some relations that involve quantifiers and prepositions need to be hard-coded, since WordNet does not include them: *every* = *all* = *each* $\leqslant$ *most* $\leqslant$ *many* $\leqslant$ *a few* = *several* $\leqslant$ *some* = *a*; *the* $\leqslant$ *some* = *a*; *on* $\perp$ *off*; *up* $\perp$ *down*; etc.

We also need to keep track of relations that can potentially be derived from the P-H sentence pair. For instance, for all adjectives and nouns that appear in the sentence pair, it is easy to obtain: *adj* + *n* $\leqslant$ *n* (*black cat* $\leqslant$ *cat*). Similarly, we have *n* + *PP/relative clause* $\leqslant$ n (*friend in need* $\leqslant$ *friend*, *dog that bites* $\leqslant$ *dog*), *VP* + *advP/PP* $\leqslant$ *VP* (*dance happily/in the morning* $\leqslant$ *dance*), and so on. We also have rules that extract pieces of knowledge from P directly, e.g.: $n_1 \leqslant n_2$ from sentences of the pattern *every* n_1 *is a* n_2. One can also connect MonaLog to bigger knowledge graphs or ontologies such as DBpedia.

A sentence base $\mathcal{S}$, on the other hand, stores the generated entailments and contradictions.

2.3 Generation

Once we have a polarized CCG tree, and some $\leqslant$ relations in $\mathcal{K}$, generating entailments and contradictions is fairly straightforward. A concrete example is given in Figure 2. Note that the generated $\leqslant$ instances are capable of producing mostly monotonicity inferences, but MonaLog can be extended to include other more complex inferences in *natural logic*, hence the name Mo*na*Log. This extension is addressed in more detail in Hu et al. (2019).

Entailments/inferences The key operation for generating entailments is `replacement`, or substitution. It can be summarized as follows: 1) For upward-entailing (UE) words/constituents, replace them with words/constituents that denote bigger sets. 2) For downward-entailing (DE) words/constituents, either replace them with those denoting smaller sets, or add modifiers (adjectives, adverbs and/or relative clauses) to create a smaller set. Thus for *every*$^{\uparrow}$ *linguist*$^{\downarrow}$ *swim*$^{\uparrow}$, MonaLog can produce the following three entailments by replacing each word with the appropriate word from $\mathcal{K}$: *most*$^{\uparrow}$ *linguist*$^{\downarrow}$ *swim*$^{\uparrow}$, *every*$^{\uparrow}$ *semanticist*$^{\downarrow}$ *swim*$^{\uparrow}$ and *every*$^{\uparrow}$ *linguist*$^{\downarrow}$ *move*$^{\uparrow}$. These are results of one `replacement`. Performing `replacement` for multiple rounds/depths can easily produce many more entailments.

Contradictory sentences To generate sentences contradictory to the input sentence, we do the following: 1) if the sentence starts with "no (some)", replace the first word with "some (no)". 2) If the object is quantified by "a/some/the/every", change the quantifier to "no", and vice versa. 3) Negate the main verb or remove the negation. See examples in Figure 2.

Neutral sentences MonaLog returns Neutral if it cannot find the hypothesis H in $\mathcal{S}.entailments$ or $\mathcal{S}.contradictions$. Thus, there is no need to generate neutral sentences.

2.4 Search

Now that we have a set of inferences and contradictions stored in $\mathcal{S}$, we can simply see if the hypothesis is in either one of the sets by comparing the strings. If yes, then return Entailment or Contradiction; if not, return Neutral, as schematically shown in Figure 2. However, the exact-string-match method is too brittle. Therefore, we apply a heuristic. If the only difference between sentences S_1 and S_2 is in the set {"a", "be", "ing"}, then S_1 and S_2 are considered semantically equivalent.

The search is implemented using depth first search, with a default depth of 2, i.e. at most 2 replacements for each input sentence. At each node, MonaLog "expands" the sentence (i.e., an entailment of its parent) by obtaining its entailments and contradictions, and checks whether H is in either set. If so, the search is terminated; otherwise the systems keeps searching until all the possible entailments and contradictions up to depth 2 have been visited.

[3] $\perp$ means "is contradictory to".

[4] There may be better and robust ways of incorporating WordNet relations to $\mathcal{K}$; we leave this for future work.

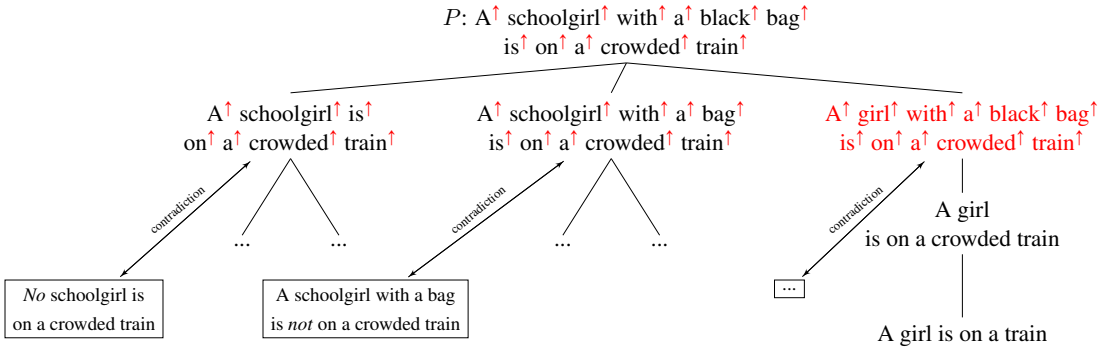

Figure 2: Example search tree for SICK 340, where P is *A schoolgirl with a black bag is on a crowded train*, with the H: *A girl with a black bag is on a crowded train*. Only one `replacement` is allowed at each step. Sentences at the nodes are generated entailments. Sentences in rectangles are the generated contradictions. In this case our system will return `entail`. The search will terminate after reaching the H in this case, but for illustrative purposes, we show entailments of depth up to 3. To exclude the influence of morphology, all sentences are represented at the lemma level in MonaLog, which is not shown here.

3 MonaLog and SICK

We perform two experiments to test MonaLog. We first use MonaLog to solve the problems in a commonly used natural language inference dataset, SICK (Marelli et al., 2014), comparing our results with previous systems. Second, we test the quality of the data generated by MonaLog. To do this, we generate more training data (sentence pairs) from the SICK training data using our system, and performe fine-tuning on BERT (Devlin et al., 2019), a language model based on the transformer architecture (Vaswani et al., 2017), with the expanded dataset. In all experiments, we use the Base, Uncased model of BERT[5].

3.1 The SICK Dataset

The SICK (Marelli et al., 2014) dataset includes around 10,000 English sentence pairs that are annotated to have either "Entailment", "Neutral" or "Contradictory" relations. We choose SICK as our testing ground for several reasons. First, we want to test on a large-scale dataset, since we have shown that a similar model (Hu et al., 2019) reaches good results on parts of the smaller FraCaS dataset (Cooper et al., 1996). Second, we want to make our results comparable to those of previous logic-based models such as the ones described in (Bjerva et al., 2014; Abzianidze, 2015; Martínez-Gómez et al., 2017; Yanaka et al., 2018), which were also tested on SICK. We use the data split provided in the dataset: 4,439 training problems, 4,906 test problems and 495 trial problems, see Table 1 for examples.

3.2 Hand-corrected SICK

There are numerous issues with the original SICK dataset, as illustrated by Kalouli et al. (2017, 2018).

They first manually checked 1,513 pairs tagged as "A entails B but B is neutral to A" (*AeBBnA*) in the original SICK, correcting 178 pairs that they considered to be wrong (Kalouli et al., 2017). Later, Kalouli et al. (2018) extracted pairs from SICK whose premise and hypothesis differ in only one word, and created a simple rule-based system that used WordNet information to solve the problem. Their WordNet-based method was able to solve 1,651 problems, whose original labels in SICK were then manually checked and corrected against their system's output. They concluded that 336 problems are wrongly labeled in the original SICK. Combining the above two corrected subsets of SICK, minus the overlap, results in their corrected SICK dataset[6], which has 3,016 problems (3/10 of the full SICK), with 409 labels different from the original SICK (see breakdown in Table 2). 16 of the corrections are in the trial set, 197 of them in the training set and 196 in the test set. This suggests that more than one out of ten problems in SICK are potentially problematic. For this reason, two authors of the current paper checked the 409 changes. We found that only 246 problems are labeled the same by our team and by Kalouli et al. (2018). For cases where there is disagreement, we adjudicated the differences after a

[5] https://github.com/google-research/bert

[6] https://github.com/kkalouli/SICK-processing

id	premise	hypothesis	orig. label	corr. label
219	There is no girl in white dancing	A girl in white is dancing	C	C
294	Two girls are lying on the ground	Two girls are sitting on the ground	N	C
743	A couple who have just got married are walking down the isle	The bride and the groom are leaving after the wedding	E	N
1645	A girl is on a jumping car	One girl is jumping on the car	E	N
1981	A truck is quickly going down a hill	A truck is quickly going up a hill	N	C
8399	A man is playing guitar next to a drummer	A guitar is being played by a man next to a drummer	E	n.a.

Table 1: Examples from SICK (Marelli et al., 2014) and corrected SICK (Kalouli et al., 2017, 2018) w/ syntactic variations. n.a.:example not checked by Kalouli and her colleagues. C: contradiction; E: entailment; N: neutral.

total	N → E	E → C	N → C	E → N
409	14	7	190	198

Table 2: Changes from SICK to corrected SICK (Kalouli et al., 2017, 2018).

discussion.

We are aware that the partially checked SICK (by two teams) is far from ideal. We therefore present results for two versions of SICK for experiment 1 (section 4): the original SICK and the version corrected by our team. For the data augmentation experiment in section 5, we only performed fine-tuning on the corrected SICK. As shown in a recent SICK annotation experiment by Kalouli et al. (2019), annotation is a complicated issue influenced by linguistic and non-linguistic factors. We leave checking the full SICK dataset to future work.

4 Experiment 1: Using MonaLog Directly

4.1 Setup and Preprocessing

The goal of experiment 1 is to test how accurately MonaLog solves problems in a large-scale dataset. We first used the system to solve the 495 problems in the trial set and then manually identified the cases in which the system failed. Then we determined which syntactic transformations are needed for MonaLog. After improving the results on the trial data by introducing a preprocessing step to handle limited syntactic variation (see below), we applied MonaLog on the test set. This means that the rule base of the system was optimized on the trial data, and we can test its generalization capability on the test data.

The main obstacle for MonaLog is the syntactic variations in the dataset, illustrated in some examples in Table 1. There exist multiple ways of dealing with these variations: One approach is to 'normalize' unknown syntactic structures to a known structure. For example, we can transform passive sentences into active ones and convert existential sentences into the base form (see ex. 8399 and 219 in Table 1). Another approach is to use some more abstract syntactic/semantic representation so that the linear word order can largely be ignored, e.g., represent a sentence by its dependency parse, or use Abstract Meaning Representation. Here, we explore the first option and leave the second approach to future work. We believe that dealing with a wide range of syntactic variations requires tools designed specifically for that purpose. The goal of MonaLog is to generate entailments and contradictions based on a polarized sentence instead.

Below, we list the most important syntactic transformations we perform in preprocessing[7].

1. Convert all passive sentences to active using *pass2act*[8]. If the passive does not contain a *by* phrase, we add *by a person*.
2. Convert existential clauses into their base form (see ex. 219 in Table 1).
3. Other transformations: *someone/anyone/no one* → *some/any/no person*; *there is no man doing sth.* → *no man is doing sth.*; etc.

4.2 Results

The results of our system on uncorrected and corrected SICK are presented in Table 3, along with comparisons with other systems.

[7]For the complete list of transformations see: https://github.com/huhailinguist/SICK_correction

[8]https://github.com/DanManN/pass2act

system	P	R	acc.
On **uncorrected** SICK			
majority baseline	–	–	56.36
hypothesis-only baseline (Poliak et al., 2018)	–	–	56.87
MonaLog (this work)			
MonaLog + all transformations	83.75	70.66	77.19
Hybrid: MonaLog + BERT	83.09	85.46	85.38
ML/DL-based systems			
BERT (base, uncased)	86.81	85.37	86.74
(Yin and Schütze, 2017)	–	–	**87.1**
(Beltagy et al., 2016)	–	–	85.1
Logic-based systems			
(Bjerva et al., 2014)	93.6	60.6	81.6
(Abzianidze, 2015)	97.95	58.11	81.35
(Martínez-Gómez et al., 2017)	97.04	63.64	83.13
(Yanaka et al., 2018)	84.2	77.3	84.3
On **corrected** SICK			
MonaLog + existential trans.	89.43	71.53	79.11
MonaLog + pass2act	89.42	72.18	80.25
MonaLog + all transformations	89.91	74.23	81.66
Hybrid: MonaLog + BERT	85.65	87.33	**85.95**
BERT (base, uncased)	84.62	84.27	85.00

Table 3: Performance on the SICK test set, original SICK above and corrected SICK below. P / R for MonaLog averaged across three labels. Results involving BERT are averaged across six runs; same for later experiments.

Our accuracy on the uncorrected SICK (77.19%) is much higher than the majority baseline (56.36%) or the hypothesis-only baseline (56.87%) reported by Poliak et al. (2018), and only several points lower than current logic-based systems. Since our system is based on *natural logic*, there is no need for translation into logical forms, which makes the reasoning steps transparent and much easier to interpret. I.e., with entailments and contradictions, we can generate a natural language trace of the system, see Fig. 2.

Our results on the corrected SICK are even higher (see lower part of Table 3), demonstrating the effect of data quality on the final results. Note that with some simple syntactic transformations we can gain 1-2 points in accuracy.

Table 4 shows MonaLog's performance on the individual relations. The system is clearly very good at identifying entailments and contradictions, as demonstrated by the high precision values, especially on the corrected SICK set (98.50 precision for E and 95.02 precision for C). The lower recall values are due to MonaLog's current inability to handle syntactic variation.

Based on these results, we tested a hybrid model of MonaLog and BERT (see Table 3) where we exploit MonaLog's strength: Since MonaLog has a very high precision on Entailment and Contradiction, we can always trust MonaLog if it predicts E or C; when it returns N, we then fall back to BERT. This hybrid model improves the accuracy of BERT by 1% absolute to 85.95% on the corrected SICK. On the uncorrected SICK dataset, the hybrid system performs worse than BERT. Since MonaLog is optimized for the corrected SICK, it may mislabel many E and C judgments in the *uncorrected* dataset. The stand-alone BERT system performs better on the uncorrected data (86.74%) than the corrected set (85.00%). The corrected set may be too inconsistent since only a part has been checked.

Overall, these hybird results show that it is possible to combine our high-precision system with deep learning architectures. However, more work is necessary to optimize this combined system.

4.3 Error Analysis

Upon closer inspection, some of MonaLog's errors consist of difficult cases, as shown in Table 5. For example, in ex. 359, if our knowledge base $\mathcal{K}$ contains the background fact *chasing* $\leqslant$ *running*, then MonaLog's judgment of C would be correct. In ex. 1402, if *crying* means *screaming*, then the label should be E; however, if *crying* here means *shedding tears*, then the label should probably be N. Here we also see potentially problematic labels (ex. 1760, 3403) in the original SICK dataset.

Another point of interest is that 19 of MonaLog's mistakes are related to the antonym pair *man* vs. *woman* (e.g., ex. 5793 in Table 5). This points to inconsistency of the SICK dataset: Whereas there are at least 19 cases tagged as Neutral (e.g., ex. 5793), there are at least 17 such pairs that are annotated as Contradictions in the test set (e.g., ex. 3521), P: *A man is dancing*, H: *A woman is dancing* (ex. 9214), P: *A shirtless man is jumping over a log*, H: *A shirtless woman is jumping over a log*. If *man* and *woman* refer to the same entity, then clearly that entity cannot be *man* and *woman* at the same time, which makes the sentence pair a contradiction. If, however, they do not refer to the same entity, then they should be Neutral.

	E		C		N	
	P	R	P	R	P	R
uncorr. SICK	97.75	46.74	80.06	70.24	73.43	94.99
corr. SICK	98.50	50.46	95.02	73.60	76.22	98.63

Table 4: Results of MonaLog per relation. C: contradiction; E: entailment; N: neutral.

id	premise	hypothesis	SICK	corr. SICK	Mona
359	There is no dog chasing another or holding a stick in its mouth	Two dogs are running and carrying an object in their mouths	N	n.a.	C
1402	A man is crying	A man is screaming	N	n.a.	E
1760	A flute is being played by a girl	There is no woman playing a flute	N	n.a.	C
2897	The man is lifting weights	The man is lowering barbells	N	n.a.	E
2922	A herd of caribous is not crossing a road	A herd of deer is crossing a street	N	n.a.	C
3403	A man is folding a tortilla	A man is unfolding a tortilla	N	n.a.	C
4333	A woman is picking a can	A woman is taking a can	E	N	E
5138	A man is doing a card trick	A man is doing a magic trick	N	n.a.	E
5793	A man is cutting a fish	A woman is slicing a fish	N	n.a.	C

Table 5: Examples of incorrect answers by MonaLog; n.a. = the problem has not been checked in corr. SICK.

5 Experiment 2: Data Generation Using MonaLog

Our second experiment focuses on using MonaLog to generate additional training data for machine learning models such as BERT. To our knowledge, this is the first time that a rule-based NLI system has been successfully used to generate training data for a deep learning application.

5.1 Setup

As described above, MonaLog generates entailments and contradictions when solving problems. These can be used as additional training data for a machine learning model. I.e., we pair the newly generated sentences with their input sentence, creating new pairs for training. For example, we take all the sentences in the *nodes* in Figure 2 as inferences and all the sentences in *rectangles* as contradictions, and then form sentence pairs with the input sentence. The additional data can be used directly, almost without human intervention.

Thus for experiment 2, the goal is to examine the quality of these generated sentence pairs. For this, we re-train a BERT model on these pairs. If BERT trained on the manually annotated SICK training data is improved by adding data generated by MonaLog, then we can conclude that the generated data is of high quality, even comparable to human annotated data, which is what we found.

More specifically, we compare the performance of BERT models trained on a) SICK training data alone, and b) SICK training data plus the entailing and contradictory pairs generated by MonaLog. All experiments are carried out using our corrected version of the SICK data set.

However, note that MonaLog is designed to only generate entailments and contradictions. Thus, we only have access to newly generated examples for those two cases, we do not acquire any additional neutral cases. Consequently, adding these examples to the training data will introduce a skewing that does not reflect the class distribution in the test set. Since this will bias the machine learner against neutral cases, we use the following strategy to counteract that tendency: We relabel all cases where BERT is not confident enough for either E or C into N. We set this threshold to 0.95 but leave further optimization of the threshold to future work.

5.2 Data Filtering and Quality Control

MonaLog is prone to over-generation. For example, it may wrongly add the same adjective before a noun (phrase) twice to create a more specific noun, e.g., *young young man* $\leqslant$ *young man* $\leqslant$

label	premise	hypothesis	comm.
E	A woman be not cooking something	A person be not cooking something	correct
E	A man be talk to a woman who be seat beside he and be drive a car	A man be talk	correct
E	A south African plane be not fly in a blue sky	A south African plane be not fly in a very blue sky in a blue sky	unnat.
C	No panda be climb	Some panda be climb	correct
C	A man on stage be sing into a microphone	A man be not sing into a microphone	correct
C	No man rapidly be chop some mushroom with a knife	Some man rapidly be chop some mushroom with a knife with a knife	unnat.
E	Few$^{\uparrow}$ people$^{\downarrow}$ be$^{\downarrow}$ eat$^{\downarrow}$ at$^{\downarrow}$ red$^{\downarrow}$ table$^{\downarrow}$ in$^{\downarrow}$ a$^{\downarrow}$ restaurant$^{\downarrow}$ without$^{\downarrow}$ light$^{\uparrow}$	Few$^{\uparrow}$ large$^{\downarrow}$ people$^{\downarrow}$ be$^{\downarrow}$ eat$^{\downarrow}$ at$^{\downarrow}$ red$^{\downarrow}$ table$^{\downarrow}$ in$^{\downarrow}$ a$^{\downarrow}$ Asian$^{\downarrow}$ restaurant$^{\downarrow}$ without$^{\downarrow}$ light$^{\uparrow}$	correct

Table 6: Sentence pairs generated by MonaLog, lemmatized.

label	total	correct	wrong	unnatural
E	56	49	0	7
C	44	41	0	3

Table 7: Quality of 100 manually inspected sentences.

man. Since it is possible that such examples influence the machine learning model negatively, we look into filtering such examples to improve the quality of the additional training data.

We manually inspected 100 sentence pairs generated by MonaLog to check the quality and naturalness of the new sentences (see Table 6 for examples). All of the generated sentences are correct in the sense that the relation between the premise and the hypothesis is correctly labeled as entailment or contradiction (see Table 7). While we did not find any sentence pairs with wrong labels, some generated sentences are unnatural, as shown in Table 6. Both unnatural examples contain two successive copies of the same PP.

Note that our data generation hinges on correct polarities on the words and constituents. For instance, in the last example of Table 6, the polarization system needs to know that *few* is downward entailing on both of its arguments, and *without* flips the arrow of its argument, in order to produce the correct polarities, on which the `replacement` of MonaLog depends.

In order to filter unnatural sentences, such as the examples in Table 6, we use a rule-based filter and remove sentences that contain bigrams of repeated words[9]. We experiment with using one quarter or one half randomly selected sentences in addition to a setting where we use the complete set of generated sentences.

5.3 Results

Table 8 shows the amount of additional sentence pairs per category along with the results of using the automatically generated sentences as additional training data.

It is obvious that adding the additional training data results in gains in accuracy even though the training data becomes increasingly skewed towards E and C. When we add all additional sentence pairs, accuracy increases by more than 1.5 percent points. This demonstrates both the robustness of BERT in the current experiment and the usefulness of the generated data. The more data we add, the better the system performs.

We also see that raising the threshold to relabel uncertain cases as neutral gives a small boost, from 86.51% to 86.71%. This translates into 10 cases where the relabeling corrected the answer.

Finally, we also investigated whether the hybrid system, i.e., MonaLog followed by the retrained BERT, can also profit from the additional training data. Intuitively, we would expect smaller gains since MonaLog already handles a fair amount of the entailments and contradictions, i.e., those cases where BERT profits from more examples. However the experiments show that the hybrid system reaches an even higher accuracy of 87.16%, more than 2 percent points above the

[9]We also investigated using a bigram based language model to filter out non-natural sentences. However, this affected the results negatively.

training data	# E	# N	# C	acc.
SICK.train: baseline	1.2k	2.5k	0.7k	85.00
1/4 gen. + SICK.train	8k	2.5k	4k	85.30
1/2 gen. + SICK.train	15k	2.5k	7k	85.81
all gen. + SICK.train	30k	2.5k	14k	86.51
E, C prob. threshold = 0.95	30k	2.5k	14k	86.71
Hybrid baseline	1.2k	2.5k	0.7k	85.95
Hybrid + all gen.	30k	2.5k	14k	87.16
Hybrid + all gen. + threshold	30k	2.5k	14k	**87.49**

Table 8: Results of BERT trained on MonaLog-generated entailments and contradictions plus SICK.train (using the corrected SICK set).

baseline, equivalent to roughly 100 more problems correctly solved. Setting the high threshold for BERT to return E or C further improves accuracy to 87.49%. This brings us into the range of the state-of-the-art results, even though a direct comparison is not possible because of the differences between the corrected and uncorrected dataset.

6 Conclusions and Future Work

We have presented a working natural-logic-based system, MonaLog, which attains high accuracy on the SICK dataset and can be used to generated natural logic proofs. Considering how simple and straightforward our method is, we believe it can serve as a strong baseline or basis for other (much) more complicated systems, either logic-based or ML/DL-based. In addiction, we have shown that MonaLog can generate high-quality training data, which improves the accuracy of a deep learning model when trained on the expanded dataset. As a minor point, we manually checked the corrected SICK dataset by Kalouli et al. (2017, 2018).

There are several directions for future work. The first direction concerns the question how to handle syntactic variation from natural language input. That is, the computational process(es) for inference will usually be specified in terms of strict syntactic conditions, and naturally occurring sentences will typically not conform to those conditions. Among the strategies which allow their systems to better cope with premises and hypotheses with various syntactic structures are sophisticated versions of alignment used by e.g. MacCartney (2009); Yanaka et al. (2018). We will need to extend MonaLog to be able to handle such variation. In the future, we plan to use dependency relations as representations of natural language input and train a classifier that can determine which relations are crucial for inference.

Second, as mentioned earlier, we are in need of a fully (rather than partially) checked SICK dataset to examine the impact of data quality on the results since the partially checked dataset may be inherently inconsistent between the checked and non-checked parts.

Finally, with regard to the machine learning experiments, we plan to investigate other methods of addressing the imbalance in the training set created by additional entailments and contradictions. We will look into options for artificially creating neutral examples, e.g. by finding reverse entailments[10], as illustrated by Richardson et al. (2019).

Acknowledgements

We thank the anonymous reviewers for their helpful comments. Hai Hu is supported by China Scholarship Council.

References

Lasha Abzianidze. 2015. A tableau prover for natural logic and language. In *Proceedings of EMNLP*, pages 2492–2502.

Lasha Abzianidze. 2017. LangPro: Natural language theorem prover. In *Proceedings of EMNLP: System Demonstrations*, pages 115–120, Copenhagen, Denmark.

Gabor Angeli and Christopher D. Manning. 2014. NaturalLI: Natural logic inference for common sense reasoning. In *Proceedings of EMNLP*, pages 534–545, Doha, Qatar.

Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J Mooney. 2016. Representing meaning with a combination of logical and

[10] In the set relations by MacCartney (2009), if $A \sqsubset B$, then A entails B, but B is neutral to A.

distributional models. *Computational Linguistics*, 42(4):763–808.

Johan van Benthem. 1986. *Essays in Logical Semantics*, volume 29 of *Studies in Linguistics and Philosophy*. D. Reidel Publishing Co., Dordrecht.

Johannes Bjerva, Johan Bos, Rob Van der Goot, and Malvina Nissim. 2014. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 642–646.

Johan Bos and Katja Markert. 2005. Recognising Textual Entailment with Logical Inference. In *Proceedings of EMNLP*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*, pages 632–642.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. Using the framework. Technical report, Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment*, pages 177–190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186, Minneapolis, Minnesota. 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 NAACL: HLT*, volume 2, pages 107–112.

Hai Hu, Qi Chen, and Lawrence S Moss. 2019. Natural language inference with monotonicity. In *Proceedings of the 13th International Conference on Computational Semantics (IWCS)*.

Hai Hu and Lawrence S. Moss. 2018. Polarity computations in flexible categorial grammar. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 124–129.

Aikaterini-Lida Kalouli, Annebeth Buis, Livy Real, Martha Palmer, and Valeria dePaiva. 2019. Explaining simple natural language inference. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 132–143.

Aikaterini-Lida Kalouli, Livy Real, and Valeria de Paiva. 2017. Textual inference: Getting logic from humans. In *IWCS 2017: 12th International Conference on Computational Semantics*.

Aikaterini-Lida Kalouli, Livy Real, and Valeria de Paiva. 2018. Wordnet for easy textual inferences. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*, Miyazaki, Japan.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Bill MacCartney. 2009. *Natural Language Inference*. Ph.D. thesis, Stanford University.

Bill MacCartney, Michel Galley, and Christopher D Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP*, pages 802–811. Association for Computational Linguistics.

Bill MacCartney and Christopher D Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of COLING*, pages 521–528.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *IWCS-8, Proceedings of the Eighth International Conference on Computational Semantics*, pages 140–156.

M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC 2014*.

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 EACL*, pages 710–720.

R Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. *arXiv preprint arXiv:1902.01007*.

George A Miller. 1995. Wordnet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of EMNLP*, pages 2055–2061.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. *arXiv preprint arXiv:1806.00692*.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191.

Kyle Richardson, Hai Hu, Lawrence S Moss, and Ashish Sabharwal. 2019. Probing natural language inference models through semantic fragments. *arXiv preprint arXiv:1909.07521*.

Asher Stern and Ido Dagan. 2011. A confidence model for syntactically-motivated entailment proofs. In *Proceedings of RANLP*, pages 455–462.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL-HLT*, volume 1, pages 1112–1122.

Hitomi Yanaka, Koji Mineshima, Pascual Martínez-Gómez, and Daisuke Bekki. 2018. Acquisition of phrase correspondences using natural deduction proofs. In *Proceedings of NAACL-HLT*, pages 756–766, New Orleans, LA.

Wenpeng Yin and Hinrich Schütze. 2017. Task-specific attentive pooling of phrase alignments contributes to sentence matching. In *Proceedings of EACL*, pages 699–709.