IWCS 2021

Natural Logic Meets Machine Learning (NALOMA)

Proceedings of the 1st and 2nd Workshops

June 16, 2021

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-954085-23-7

Preface

This volume consists of papers presented at the first and second workshops entitled NAtural LOgic Meets MAchine Learning (NALOMA). Both workshops were held online; the first with the Web Summer School in Logic, Language, and Information (WeSSLLI) in 2020, and the second with the 14th International Conference on Computational Semantics (IWCS) in 2021.

NALOMA aims to bridge the gap between machine learning/deep learning and symbolic/logic-based approaches to Natural Language Inference (NLI), and it is one of the only workshops organized to do so. The workshop also lays focus on theoretical notions of NLI which influence the way approaches to NLI can and should operate.

We thank everyone who submitted papers to the meeting, including the authors who submitted non-archival extended abstracts. These contributions are not part of the proceedings but can be found within the schedule and on our website (`https://typo.uni-konstanz.de/naloma21/`) The meetings were enriched by the inspiring talks of our invited speakers: Lauri Karttunen and Ignacio Cases, Ellie Pavlick, and Mark Steedman, in 2020; Vered Shwartz, and Benjamin Van Durme in 2021.

We also thank all and everyone who served on the program committee (most served twice): Lasha Abziniadize, Stergios Chatzikyriakidis, Katrin Erk, Hai Hu, Thomas Icard, Valeria de Paiva, and Hitomi Yanaka. We are also thankful to the Research Unit FOR 2111 “Questions at the Interfaces" of the University of Konstanz, funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), for its support and for hosting our web pages, and also the Indiana University Program in Pure and Applied Logic. The meeting would not have been possible without the encouragement and organizational support that we received from Sophia Malamud and James Pustejovsky in 2020, and from Rik van Noord and Lasha Abziniadize in 2021.

When people combine research communities, the intent is not merely to talk together but also to find joint intellectual projects. NALOMA's parents are logic and symbolic AI on one side, and machine learning on the other side. As we welcome NALOMA to its third year, we watch expectantly for those joint projects.

Aikaterini-Lida Kalouli and Lawrence S. Moss

Organizers:
Aikaterini-Lida Kalouli, University of Konstanz
Lawrence S. Moss, Indiana University

Program Committee:
Lasha Abzianidze, Utrecht University
Stergios Chatzikyriakidis, University of Gothenburg
Katrin Erk, University of Texas at Austin
Hai Hu, Indiana University
Thomas Icard, Stanford University
Valeria de Paiva, Topos Institute
Hitomi Yanaka, Riken

Invited Speakers:
2020
Lauri Karttunen and Ignacio Cases, Stanford University
Ellie Pavlick, Brown University
Mark Steedman, University of Edinburgh

2021
Benjamin Van Durme, Johns Hopkins University
Vered Shwartz, Allen Institute for AI (AI2) / University of Washington

Table of Contents

Workshop Program NALOMA'20

Sunday, July 12, 2020

4:20–4:30 *Opening Remarks*
Larry Moss

4:30–5:30 *Invited Talk: Reasoning with Implicatives: Modular and Compositional Learning for Language Understanding*
Lauri Karttunen and Ignacio Cases

5:40–6:20 ***Research Exchange***

Monday, July 13, 2020

8:45–9:20 *Learning General Event Schemas with Episodic Logic*
Lane Lawley, Benjamin Kuehnert and Lenhart Schubert

9:20–9:55 *Applied Medical Code Mapping with Character-based Deep Learning Models and Word-based Logic*
John Langton and Krishna Srihasam

9:55–10:10 ***Coffee Break***

10:10–10:45 *Learning lexical knowledge with Natural Tableau*
Lasha Abzianidze

Tuesday, July 14, 2020

8:45–9:45 *Invited Talk: Linguistic Semantics and Contemporary NLI*
Mark Steedman

9:45–10:00 ***Coffee Break***

10:00–10:45 *Formalizations and Implementations of Monotonicity Reasoning*
Larry Moss

Wednesday, July 15, 2020

8:45–9:20 *Investigating the Generalization Ability of Neural Models through Monotonicity Reasoning*
Hitomi Yanaka

9:20–9:55 *Attentive Tree-structured Network for Monotonicity Reasoning*
Zeming Chen

9:55–10:10 ***Coffee Break***

10:10–10:45 *Using Monotonicity for Natural Language Inference*
Hai Hu

Thursday, July 16, 2020

8:45–9:45 *Invited Talk: "I don't know what you mean" semantics is hard: Challenges in evaluation of semantic phenomena*
Ellie Pavlick

9:45–10:00 ***Coffee Break***

10:00–10:45 *NLI Data and Annotations: a Look Behind the Scenes*
Aikaterini-Lida Kalouli

Friday, July 17, 2020

8:45–9:20 *Transferring Representations of Logical Connectives*
Aaron Traylor, Ellie Pavlick and Roman Feiman

9:20–9:55 *Monotonic Inference for Underspecified Episodic Logic*
Gene Kim, Mandar Juvekar and Lenhart Schubert

9:55–10:45 ***Panel discussion and Closing remarks***

Workshop Program NALOMA'21

Wednesday, June 16, 2021

14:30–14:45 *Welcome and Opening Remarks*
Aikaterini-Lida Kalouli and Larry Moss

14:45–15:10 *Do Neural Models Learn Transitivity of Veridical Inference?*
Hitomi Yanaka, Koji Mineshima and Kentaro Inui

15:10–15:35 *Does Logic-based Reasoning Work for Dutch?*
Lasha Abzianidze and Konstantinos Kogkalidis

15:35–16:00 *Supporting Context Monotonicity Abstractions in Neural NLI Models*
Julia Rozanova, Deborah Ferreira, Mokanarangan Thayaparan, Marco Valentino and André Freitas

16:00–16:45 *Invited Talk: Expanding Natural Language Inference*
Benjamin Van Durme

16:45–17:15 ***(Coffee/Lunch/Dinner) Break***

17:15–17:40 *Bayesian Classification and Inference in a Probabilistic Type Theory with Records*
Staffan Larsson and Robin Cooper

17:40–18:05 *From compositional semantics to Bayesian pragmatics via logical inference*
Julian Grove, Jean-Philippe Bernardy and Stergios Chatzikyriakidis

18:05–18:30 *A (Mostly) Symbolic System for Monotonic Inference with Unscoped Episodic Logical Forms*
Gene Kim, Mandar Juvekar, Junis Ekmekciu, Viet Duong and Lenhart Schubert

18:30–19:15 *Invited Talk: Natural Language Inference: Challenges and Opportunities*
Vered Shwartz

19:15–20:00 ***Panel discussion and Closing remarks***

Learning General Event Schemas with Episodic Logic

Lane Lawley and Benjamin Kuehnert and Lenhart Schubert
University of Rochester
Department of Computer Science
{llawley@cs, bkuehner@u, schubert@cs}.rochester.edu

Abstract

We present a system for learning generalized, stereotypical patterns of events—or "schemas"—from natural language stories, and applying them to make predictions about other stories. Our schemas are represented with Episodic Logic, a logical form that closely mirrors natural language. By beginning with a "head start" set of *protoschemas*—schemas that a 1- or 2-year-old child would likely know—we can obtain useful, general world knowledge with very few story examples—often only one or two. Learned schemas can be combined into more complex, composite schemas, and used to make predictions in other stories where only partial information is available.

1 Introduction

We present a novel approach to learning rich, symbolic event schemas from natural language texts. While most modern approaches to automated script learning (e.g. (Chambers and Jurafsky, 2011; Pichotta and Mooney, 2016a; Yuan et al., 2018)) learn linear sequences of simple tuple representations of events, our schema representation allows for typed and interrelated participating entities; multiple temporally related subevents; specification of goals, preconditions, and postconditions; and nesting of subschemas as steps in another schema.

We mitigate the "brittleness" of past symbolic approaches (e.g., GENESIS (Mooney, 1990) and IPP (Lebowitz, 1980)) by parsing stories into Episodic Logical Form (ELF) (Schubert and Hwang, 2000), a logical representation that closely resembles natural English, but allows for complex event representation and powerful inference procedures. As Stratos et al. (2011) argue, Episodic Logic facilitates "Natural Logic-like inference while also providing greater generality". EL, and its underspecified variant ULF, facilitate NLog-like inferences using a combination of lexical and semantic knowledge (Schubert, 2014; Kim et al., 2019). Because most nouns and verbs are preserved as predicates in ELFs, we also utilize existing lexical resources, like WordNet's hypernym hierarchy for generalizing schema predicates (e.g. `DOG.N` and `ELEPHANT.N` to `PLACENTAL_MAMMAL.N`), and semantic word embeddings for retrieving relevant schema candidates for a story from a large number of known schemas.

We also bypass the need for large amounts of data by giving the system a "head start" in the form of a relatively small number of initial schemas targeting the commonsense knowledge of a very young child, from which more complex schemas can be learned and composed. These "protoschemas" describe basic action types—e.g., eating, searching, moving from place to place, transferring possession of objects—at a very general level, along with their underlying motivations and pre- and postconditions. More complex schemas—e.g., "a monkey climbs a tree, gets a coconut, and eats the coconut"—can be composed by "chaining" these simpler ones together after matching them to a story.

From a corpus of several hundred short children's stories, we have acquired dozens of schema matches, generalized them into new schemas, automatically composed some of them into more complex schemas, and used those generalized schemas to make predictions on unseen stories with only partial information.

2 Episodic Logic

Our schema representation is based on Episodic Logic (EL) (Schubert and Hwang, 2000), a formal knowledge representation with semantic types and operators common to many natural languages. EL

Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA), pages 1–6

uses first-order quantification, but has type-shifting and reification operators to map predicate and sentence intensions to domain individuals, allowing it to represent higher-order propositions.

EL is a good fit for schemas in part because of its *characterizing* operator `**`, which relates an EL formula to a situational argument, an "episode" that it characterizes. For example, the EL formula `((I.PRO EAT.V (K STEAK.N)) ** E1)` says that `E1` is a (possibly repetitive, habitual) episode of me eating steak[1]. Episodes can have multiple formulas "true in" them, where these formulas characterize subepisodes with limited temporal bounds. This makes them ideal for representing entire schemas, which are "packages" of formulas all true together within some span of time.

2.1 Overview

Although an adequate explanation of the features and syntax of EL would not fit within these margins—please refer to (Schubert and Hwang, 2000) for more detail—we offer a brief guide to understanding some of the formulas in, e.g., Figure 2.

2.1.1 Propositions

An atomic EL proposition has a **prefix argument** (sentential subject), an infixed **predicate**, and zero or more **postfix arguments**. In exact EL syntax, if there are postfixed arguments then the monadic predicate formed by the infix predicate together with its postfixed arguments is bracketed (e.g., see Figure 1. Monadic predicates as well as complete formulas may have **modifiers** applied to them. In the formula `(I.PRO (QUICKLY.ADV-A (EAT.V (K STEAK.N))))`, the prefix argument is the individual `I.PRO`, the infix predicate is the verbal predicate `EAT.V`, the postfix argument is the kind-level individual `(K STEAK.N)`, and the modifier is the adverb `QUICKLY.ADV-A`. When there are no predicate modifiers, atomic formulas with postfix arguments can be "flattened", as in the formula `(I.PRO EAT.V (K STEAK.N))` above.

Not all EL formulas use verbal predicates: type constraint formulas, like `(?X STEAK.N)` or `?D RED.A`, are examples of formulas with nominal and adjectival predicates.

[1]Here, the `STEAK.N` predicate is reified into an abstract individual—the kind of food, steak—by the `K` operator so it can be used as an argument of the `EAT.V` predicate.

2.1.2 Quantifiers

Although explicit quantifiers are not present in the schemas we present here—a schema's variables are implicitly Skolem functions of the schema's head episode—we will note that EL supports the standard first-order quantifiers $\exists$ and $\forall$. It also has nonstandard quantifiers like `Most` and `Few`, to represent sentences like "Most students who have studied here have gone on to be successful". Nonstandard quantifiers use "restrictors" to filter the quantified variables with an arbitrary predicate.

3 Schema Representation

In this section, we will describe our schema representation. Although sequential and causally connected events play a large role in our schemas, our schema language is differentiated from causal representations such as (Luo et al., 2016) and sequential script representations such as (Pichotta and Mooney, 2016b) by the expressiveness and interconnectedness of its constituent logical formulas. The language is designed to model the schema's **Steps**, the **Roles** (types) of participating entities, and the motivating **Goals**, **Preconditions**, and **Postconditions** of the schema as a whole.

An example schema our system has learned can be seen in Figure 1. The EL formulas specifying the semantic contents of a schema organized into sections; we describe the sections below.

```
(EPI-SCHEMA ((?X_D EAT.379.V ?X_C)
                ** ?X_E)
  :ROLES
    !R1 (?X_D AGENT.N)
    !R2 (?X_C FOOD.N)
    !R3 (?X_C GRASS.N)
    !R4 (?X_D COW.N)
  :GOALS
    ?G1 (?X_D (WANT.V (THAT (NOT
        (?X_D HUNGRY.A)))))
  :PRECONDS
    ?I1 (?X_D HAVE.V ?X_C)
    ?I2 (?X_D HUNGRY.A)
  :POSTCONDS
    ?P1 (NOT (?X_D (HAVE.V ?X_C)))
    ?P2 (NOT (?X_D HUNGRY.A))
  :EPISODE-RELATIONS
    !W1 (?P1 AFTER ?X_E)
    !W2 (?I1 BEFORE ?X_E)
  :NECESSITIES
    !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
)
)
```

Figure 1: A schema learned by applying the eating protoschema to the sentence "The cow ate the grass".

3.1 Overall Structure

A schema is represented by its *header*, seen in line 1 of Figure 2. A schema's header is an EL proposition and an episode characterized by the proposition, here `?E`. The header episode summarizes the entire schema, and can be used to embed a schema as a step inside another schema.

The rest of the schema is laid out in two types of sections: *fluent* and *nonfluent* sections. Nonfluent sections such as `Roles` and `Episode-relations` contain formulas that hold true regardless of time, such as the types or physical properties of objects. Fluent sections such as `Steps` and `Preconds` contain formulas whose truth values are time-dependent, such as an action taken by someone. We will now examine these sections, and what they're used for, in more detail.

3.2 Roles

The **Roles** section of a schema is a *nonfluent* section meant for putting "eternal" type constraints on the participating entities in the schema. In addition to type constraints, e.g. `(?X DOG.N)`, nonfluent relational constraints between entities can also be specified in this section, e.g. `(?X PERTAINS_TO.N ?Y)`.

When individuals from story formulas are bound to slot variables in the schema, these "type" constraints are evaluated to judge how well the individuals fit those slots. Some constraints may be broken—this is a key part of the generalization process—but the soft scoring metric in Section 4.3.1 helps identify good matches.

3.3 Preconditions, Postconditions, and Goals

Schemas specify preconditions, postconditions, and goals characterize the motivations of the agents involved. Fluent constraints in the precondition section are tacitly assumed to start before the schema's header episode (adjoining or overlapping it), and those in the postcondition section extend beyond the header episode (post-adjoining or overlapping it). Schema matches can be "chained together" into composite, multi-step schemas by unifying their pre- and postconditions, or their goals and postconditions. The schema in Figure 2 examplifies a learned "chained" schema.

3.4 Temporal Relations

The episodes characterized by fluent formulas within the body of a schema can all be complexly interrelated using constraints from the Allen Interval Algebra (Allen, 1983) as well as causal and quantitative temporal constraints. Pre- and postconditions are implicitly constrained to be true at the start and end of the schema's header episode, respectively, and steps, by default, are ordered sequentially as listed in the schema, but additional constraints can be specified in the **Episode-relations** section of each schema. To evaluate these interval constraint propositions, we implemented a time graph specialist module (Gerevini and Schubert, 1993). The time graph models the temporal projection of each episode as a pair of time points, corresponding to the beginning and end of the episode. The time graph has time points as vertices, and an edge between t_1 and t_2 if $t_1 \leq t_2$. Then, querying the graph for propositions can be done with a graph transversal. The time graph also keeps track of "chains", which are long consecutive sequences of time points in the graph. This allows the module to exploit the often linear structure of stories, and it achieves high efficiecy on the subalgebra of Allen's Interval Algebra that can be expressed in terms of $\leq$ point-relations.

4 Schema Learning

In this section, we describe how our system learns new schemas from natural language stories. We describe our story parsing process, the process of matching parsed stories to schemas, how schema matches can be generalized to create new schemas, and how partial schema matches can be used to predict events in similar stories with missing details.

4.1 The Protoschema Approach

As noted, we generate new schemas from stories by starting with an initial set of *protoschemas* that we would expect a 1- or 2-year-old child to have; these encode very general knowledge about physical and communicative actions, with their preconditions and effects. Examples of protoschemas we've already written include movement of an agent from one location to another, consumption of food, and possession and transfer of possession. These protoschemas are then invoked by actions in stories—for example, the "travel" protoschema matched a "climb" action in a story to yield a "monkey climbs a tree" schema, which was eventually incorporated

as the first step of the chained schema in Figure 2. [2]

4.2 Story Parsing

We first process raw stories with the AllenNLP coreference analyzer (Gardner et al., 2017), and then use the first stage of the BLLIP parser (Charniak, 2000) for an initial syntactic parse. The syntactic parse is then converted to *Unscoped Logical Form* (ULF) (Kim and Schubert, 2019), an underspecified variant of EL, by tree transductions, and then a second transduction phase processes the ULF into full EL.

Our current parsing pipeline converts about 50 percent of (very brief, typically 2-5 sentence) stories to valid Episodic Logic formulas; our rules cannot transduce some grammatical features into ULF, including quotations and rhetorical questions. Kim (2019) is investigating direct English-to-ULF conversion using a cache transition parser, and we hope that this approach will boost our parsing accuracy.

4.3 Matching

Matching formulas in semantically parsed stories to formulas in schemas underlies both learning and prediction. The formulas comprising a schema are intended to be relatively simple—with complex conjunctions split into separate formulas—and unifiable with formulas parsed from real stories. Unification of a story formula with a schema formula binds individual constants from the former to variables in the latter. These bindings are then substituted in the rest of the schema instance, thereby "filling in" some of the missing information. This information is likely to be correct if the story events and participant types matched to the schema can be assumed to provide good evidence for an occurrence of the stereotyped pattern of events the schema captures. We refer to any schema instance with one or more bound variables as a *match*.

Using EL formula unification as a primitive, we implement schema matching by iterating through the formulas in an EL parse of a story, matching each formula to any schema formula retrieved as a candidate, and applying the bindings to the schema. When the story has been fully iterated through, or all schema variables have been bound, the match is complete.

We randomly permute story formulas and unify them, in the randomized order, with schema formulas. We try multiple permutations to explore the space of possible matches, and cache low-level unification results to speed up the process.

4.3.1 Partial Matches and Scoring

When a schema is matched to a story, some constraints may be broken; this is a natural part of the learning process. A schema for a cow eating grass matched to a story about a dog eating grass violates the cow constraint on a participating entity, but is a valuable source of knowledge if properly generalized. On the other hand, too many broken constraints are indicative of a poor match between a schema candidate and a story.

Schema matches are heuristically scored by counting satisfied constraints, weighted by constraint type. Confirmed Role constraints are worth half as many points as confirmed events in the Steps section. Confirming the schema's header formula is worth twice the points of any other event.

For inexact matches—e.g., `(?X COW.N)` and `(ROVER.NAME DOG.N)`—the score of the binding is further weighted by the approximate semantic similarity of the two words. If one subsumes the other in a hypernym hierarchy, the strength is scaled by the distance of the two in that hierarchy. If neither subsumes the other, but they share a common ancestor hypernym, the strength is half their average distance to that ancestor.

The hypernym score accounts for half of the overall weight of an inexact match; the other half is provided by their semantic similarity according to a pre-trained word embedding model. [3]

4.4 Generalizing Matches

To generalize a match into a new, "learned" schema, we need to incorporate incidental information about the matched value. For example, the variables of the `travel.v` protoschema can be bound by the constants in the formula `((MONKEY27.SK (CLIMB.V TREE28.SK)) ** E34.SK)` in a story about a monkey climbing a tree, but re-generalizing the constants `MONKEY27.SK` and `TREE28.SK` into unconstrained variables would remove all the information we learned. However, if we incorporate formulas about the types of those objects into our new schema—such as the formulas `(MONKEY27.SK MONKEY.N)` and

[2] "travel" was invoked by "climb" by way of the WordNet hypernym hierarchy.

[3] *GoogleNews-vectors-negative300.bin*, Mikolov et al. (2013)

(TREE28.SK TREE.N)—we can then generalize the constants but maintain knowledge of their types.

4.4.1 Re-Matching Learned Schemas

Once a protoschema has been matched to a story and generalized into a learned schema, it may contain extraneous details or overly specific constraints. To filter out such details or constraints, we search for at least one more match of the learned schema to another story, downgrading details and constraints that were not matched again. To learn (potentially) more abstract versions of learned schemas, we retain both basic types and generalized types in the abstract versions, with certainties reflecting their match frequencies.

4.5 Prediction

Prediction is relatively straightforward: Given a story, we try to identify a similar schema, such as the learned schema in Figure 2, and match as many formulas as we can to it. We find similar schemas by average pairwise distance between story words and schema word predicates in the pre-trained word vector space. After we've substituted story entities for variables, we may fill in other formulas in the schema. Schema formulas whose variables have all been filled in, but are not present in the story, are predictions: in effect, we guess that the schema underlies the observed events, and infer further aspects of the situation from its explicitly provided aspects.

5 Results

Using 511 simple stories taken from a children's first reader (McGuffey, 1901) and the ROCstories corpus (Mostafazadeh et al., 2017), and 13 protoschemas[4], we obtained 665 schemas, with a mean score of -0.899, a median score of 0.292, a minimum score of -19.304, and a maximum score of 4.5 according to the scoring metric in Section 4.3.1. After filtering out the 314 negative-scoring schemas, we obtained 314 "specified" schemas, including six multi-step schemas, examples of which can be found in Figure 1 and Figure 2.

[4]These 13 protoschemas, including traveling from place to place, eating food, taking possession of an object, and searching for something, were selected to cover a large number of sentences in a "development set" of 50 held-out stories from our corpus of 561 stories; 511 were used in the test set. We intend to eventually construct dozens to hundreds of initial protoschemas.

```
(EPI-SCHEMA ((?X_B CLIMB_GET_EAT.PR
                ?X_A ?X_C) ** ?E)
  :ROLES
    !R1 (?X_A TREE.N)
    !R2 (?X_C INANIMATE_OBJECT.N)
    !R3 (?X_B MONKEY.N)
    !R4 (?X_C FOOD.N)
    !R5 (?X_C COCOANUT.N)
  :STEPS
    ?E1 (?X_B CLIMB.481.V
           (FROM.P-ARG ?L1) ?X_A)
    ?E2 (?X_B GET.511.V ?X_C
           (AT.P-ARG ?X_A))
    ?E3 (?X_B EAT.541.V ?X_C)
  :EPISODE-RELATIONS
    !W1 (?E1 BEFORE ?E2)
    !W2 (?E2 BEFORE ?E3)
    !W3 (?E1 DURING ?E)
    !W4 (?E2 DURING ?E)
    !W5 (?E3 DURING ?E)
)
```

Figure 2: An example of a multi-step schema learned by our system from protoschema matches to a story about a monkey climbing a tree to get and eat a cocoanut.

The schema in Figure 2 inferred, given the sentences "Simeon can climb the tree" and "He gets the cocoanuts for his mother", that Simeon was a monkey, that he got the cocoanuts in the tree, and that he later ate the cocoanuts. The schema in Figure 1 inferred, given the sentences "The bees like it", "They find sweet nectar in the clover flowers", and "It grows in the fields", that the bees went to the fields to find the nectar. These predictions about unseen stories are reasonable and fill in details absent in the stories themselves.

6 Future Work

The schemas learned and predictions generated by the system with only 13 protoschemas are encouraging; we've obtained many simple schemas, like "person sits in a chair" or "dogs run around outside", as well as complex, multi-step schemas used for predictions like the ones in Section 5. Because complex schemas are made by stringing together protoschema matches, we plan to develop more protoschemas—possibly dozens to hundreds—to more fully cover the general knowledge of a two-year-old child. With those protoschemas as a base, we expect to generate many more useful, multi-step schemas, use them to generate predictions about stories, and have human judges evaluate those predictions.

Acknowledgments

This work was supported by NSF EAGER award IIS-1940981, DARPA CwC subcontract W911NF-15-1-0542, and NSF NRT award 1449828.

References

James F Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 976–986. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Alfonso Gerevini and Lenhart Schubert. 1993. Efficient temporal reasoning through timegraphs. In *IJCAI*, pages 648–654.

Gene Kim, Benjamin Kane, Viet Duong, Muskaan Mendiratta, Graeme McGuire, Sophie Sackstein, Georgiy Platonov, and Lenhart Schubert. 2019. Generating discourse inferences from unscoped episodic logical formulas. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 56–65.

Gene Kim and Lenhart Schubert. 2019. A type-coherent, expressive representation as an initial step to language understanding. In *Proceedings of the 13th International Conference on Computational Semantics*, Gothenburg, Sweden. Association for Computational Linguistics.

Gene Louis Kim. 2019. Towards parsing unscoped episodic logical forms with a cache transition parser. In *the Poster Abstracts of the Proceedings of the 32nd International Conference of the Florida Artificial Intelligence Research Society*.

Michael Lebowitz. 1980. *Generalization and Memory in an Integrated Understanding System*. Ph.D. thesis, New Haven, CT, USA. AAI8109800.

Zhiyi Luo, Yuchen Sha, Kenny Q Zhu, Seung-won Hwang, and Zhongyuan Wang. 2016. Commonsense causal reasoning between short texts. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

William Holmes McGuffey. 1901. *The New McGuffey First Reader*. American Book Company.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Raymond J Mooney. 1990. *A general explanation-based learning mechanism and its application to narrative understanding*. Morgan Kaufmann.

Nasrin Mostafazadeh, Chris Brockett, Bill Dolan, Michel Galley, Jianfeng Gao, Georgios Spithourakis, and Lucy Vanderwende. 2017. Image-grounded conversations: Multimodal context for natural question and response generation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 462–472, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Karl Pichotta and Raymond Mooney. 2016a. Statistical script learning with recurrent neural networks. In *Proceedings of the Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*, pages 11–16. Association for Computational Linguistics.

Karl Pichotta and Raymond J. Mooney. 2016b. Using sentence-level LSTM language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 279–289, Berlin, Germany. Association for Computational Linguistics.

Lenhart Schubert. 2014. Nlog-like inference and commonsense reasoning. *LiLT (Linguistic Issues in Language Technology)*, 9.

Lenhart K. Schubert and Chung Hee Hwang. 2000. Episodic Logic meets Little Red Riding Hood: A comprehensive natural representation for language understanding. In Lucja M. Iwańska and Stuart C. Shapiro, editors, *Natural Language Processing and Knowledge Representation*, pages 111–174. MIT Press, Cambridge, MA, USA.

Karl Stratos, Lenhart Schubert, and Jonathan Gordon. 2011. Episodic logic: Natural logic + reasoning. *KEOD 2011 - Proceedings of the International Conference on Knowledge Engineering and Ontology Development*.

Quan Yuan, Xiang Ren, Wenqi He, Chao Zhang, Xinhe Geng, Lifu Huang, Heng Ji, Chin-Yew Lin, and Jiawei Han. 2018. Open-schema event profiling for massive news corpora. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 587–596, New York, NY, USA. ACM.

Applied Medical Code Mapping with Character-based Deep Learning Models and Word-based Logic

John T. Langton
Wolters Kluwer Health
230 3rd Avenue
Waltham, MA 02451
jlangton@wolterskluwer.com

Krishna Srihasam
Wolters Kluwer Health
230 3rd Avenue
Waltham, MA 02451
ksrihasam@wolterskluwer.com

Abstract

Logical Observation Identifiers Names and Codes (LOINC) is a standard set of codes that enable clinicians to communicate about medical tests. Laboratories depend on LOINC to identify what tests a doctor orders for a patient. However, clinicians often use site-specific, custom codes in their medical records systems that can include shorthand, spelling mistakes, and invented acronyms. Software solutions must map from these custom codes to the LOINC standard to support data interoperability. A key challenge is that LOINC is comprised of six elements. Mapping requires not only extracting those elements, but also combining them according to LOINC logic. We found that character-based deep learning excels at extracting LOINC elements while logic-based methods are more effective for combining those elements into complete LOINC values. In this paper, we present an ensemble of machine learning and logic that is currently used in several medical facilities to map from custom codes to standard LOINC values.

1 Introduction

LOINC supports several use cases in the medical domain. For instance, a doctor can use LOINC codes to precisely indicate which blood tests they want a laboratory to perform. A major challenge is that clinicians often use custom codes when entering data into medical records systems. Custom codes may be more intuitive for humans to understand but also suffer from personal nuance and error. They can contain misspellings, shorthand, and invented acronyms. Further, custom codes are site-specific such that the codes in one facility may differ from those in another. These differences make it difficult for facilities to communicate. For instance, if a doctor uses one set of codes to order tests, and a laboratory uses a different set of codes to perform tests, then the laboratory may not be able to correctly identify which tests to perform what tests the doctor is ordering. It is necessary for software solutions to map from custom codes to standards like LOINC to eliminate differences between the codes that facilities use and support data interoperability.

2 The Task

There are around 40,000 LOINC codes. Each code contains six elements as shown in Table 1. Our task is to first extract the six elements from a noisy input string (e.g. custom code), then combine those elements to form a standard LOINC output. Equation (1) shows a real-world example with a custom hospital code on the left mapped to a standard LOINC code on the right. The following are five additional input strings from different hospitals that refer to the exact same LOINC code: {*"Ur Leukocyte Esterase", "LEUK ESTER", "UR Leuko Est–Clinitek", "LEUKOCYTE E URINE", "Leuk Est Test Strip U"*}. One can readily see the differences that complicate communication and data interoperability.

$$\text{U Leuk Est} \rightarrow \begin{cases} \text{5799-2 Leukocyte esterase,} \\ \text{Urine, Ordinal, PT,} \\ \text{Test Strip, Presence} \end{cases} \tag{1}$$

3 Data

The data for our project came from prior mappings that were performed manually by clinical informaticists. The distribution of LOINC values was skewed, with ten codes making up 87.7% of the data. The remaining codes made up a long tail distribution but not all possible LOINC codes were present. Table 3 shows the possible unique LOINC element values along with the coverage of those values in the available data. If we treat each LOINC

Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA), pages 7–11

Element	Example	Description
Component	Leukocyte Esterase	What is being measured, observed, or evaluated
Specimen	Urine	specimen type collected for measurement
Scale	Ordinal	the scale of measure such as ordinal or nominal
Timing	PT	interval of time for measurement or observation
Method	Test Strip	Method of measurement
Property	Presence	Property of what is being measured such as mass or volume

Table 1: The six LOINC elements with examples.

Challenge	Example
Invented acronyms and missing letters	Lkct for Leukocyte
Misspellings	Luykocite
Missing delimiters	UrLeukEst
Missing LOINC elements	Leuk Est (component only)
Parsing and input / output errors	00001

Table 2: Challenges of mapping custom codes to LOINC standards.

code as a class, then the data distribution corresponds with severe class imbalance.

Medical facilities rarely have local codes for every possible LOINC. Instead, they maintain a subset of codes that are most commonly used in their practice. As a result, custom codes at a facility often exclude LOINC elements that are irrelevant to their practice. For example, a blood laboratory may exclude specimen because they implicitly know the value is always "blood". Our data, therefore, contained many codes with only a subset of the elements necessary to specify a full LOINC. Efforts to map custom codes to LOINC standards must contend with several challenges as enumerated in Table 2.

4 Related Research

Both machine learning and logic-based methods for NLP struggle with noisy text inputs. Vectorization methods including *Term Frequency – Inverse Document Frequency* (TF-IDF) vectorization (Xu et al., 2009) and word embeddings (Kim, 2014; Pennington et al., 2014) are particularly sensitive, though the use of sub-words (i.e. n-grams of characters) can somewhat ameliorate the issue (Edizel et al., 2019). Pre-trained transformer models have recently topped some NLP benchmarks but are also sensitive to noisy text (Devlin et al., 2018; Wang et al., 2019; Rajpurkar et al., 2018). Pruthi, Dhingra, and Lipton have shown that misspellings reduce BERT performance by significant margins and propose an independent model for spelling correction (Pruthi et al., 2019). Luong and Manning have used hybrids of character and word based recurrent neural networks (RNN) to address unknown words in translation (Luong and Manning, 2016). Zhang and Yang have used a lattice of Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) models (variants of RNNs) at the character and word level to improve performance of named entity extraction (Zhang and Yang, 2018). We evaluated multiple methods and determined to use a similar hybrid approach for extracting canonical LOINC terms from noisy text inputs.

5 Hybrid Solution for LOINC Mapping

We evaluated several methods for addressing noisy text inputs. Logic-based methods were paired with fuzzy matching, word frequency analysis, and synonym dictionaries. Only around 3% of incoming strings could be mapped in this manner. We found that character-based GRUs excelled at extracting LOINC elements from noisy text inputs but plateaued around 60% accuracy when combining those elements into a final code. A combination of machine learning and logic-based approaches achieved much higher accuracy and coverage. The resulting hybrid model is shown in Figure 1. A given input string is first processed by six, character-based GRUs for each of the LOINC elements (though the figure shows only three). The outputs of these models are then input to logic that combines them in a final LOINC code. The following sections describe these processing steps and provide a final evaluation.

Name	Unique Possible Values	Unique Values in Data	Coverage
Component	19,507	4,783	25%
Specimen	344	143	42%
Scale	6	6	100%
Timing	668	380	57%
Method	504	212	42%
Property	116	91	78%
LOINC Codes	46,156	11,190	24%

Table 3: Data coverage of possible LOINC values.

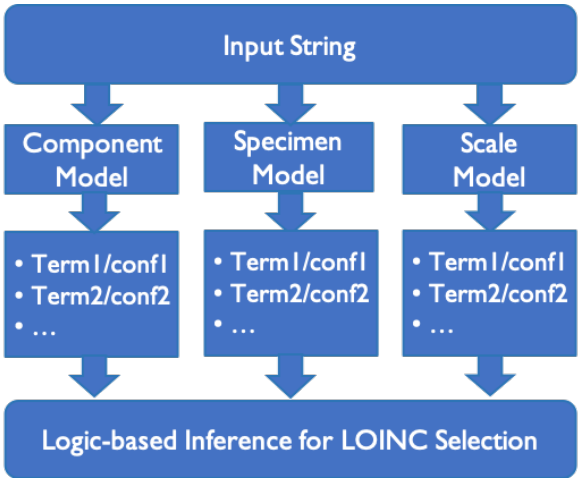

Figure 1: Hybrid solution combining machine learning models with logic for LOINC mapping.

Prediction	Conf
LEUKOCYTE ESTERASE	0.875
LEUKOCYTES ESTERASE NITRITE	0.81
ALBUMIN	0.002

Table 4: Example component predictions for the input string ”Ur Leukocyte Esterase”.

5.1 Extracting Terms with Deep Learning

A character-based GRU was trained for each of the six LOINC elements using the scikit-learn and Keras packages (Pedregosa et al., 2011). The output classes for each element model were the possible values for that element. For instance, the model for the component element was trained to output 19,507 possible classes. A softmax activation function was used with categorical cross-entropy for the loss function. This approach enabled the model to output a probability between 0 and 1 for each of the possible class values. Table 4 shows the top three predictions for the component element given the input string ”Ur Leukocyte Esterase”. Table 5 shows the top predictions for each LOINC element for the same input string.

Element	Prediction	Conf
Component	LEUKOCYTE ESTERASE	.875
Specimen	URINE	.95
Scale	ORD	.86
Method	NONE	.84
Timing	NONE	.856
Property	PRTHR	.763

Table 5: Example predictions for each of the 6 LOINC elements for the input string ”Ur Leukocyte Esterase”.

5.2 Combining Terms with Logic

We hypothesize that several factors contribute to the poor performance of machine learning when predicting a final code from the LOINC elements:

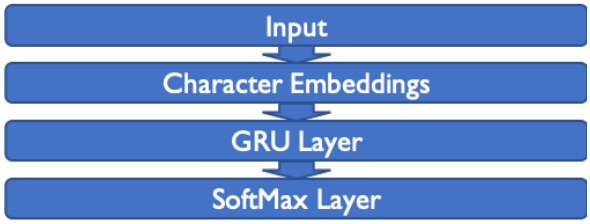

Figure 2: Character-based RNN for extracting a LOINC element.

Class imbalance was severe. While there were sufficient samples of LOINC elements, there were insufficient and imbalanced samples for complete LOINC codes. Machine learning methods are sensitive to class imbalance, whereas logic-based methods are not.

LOINC logic dictates that some elements can be combined while others cannot. For instance, liquid units of measure cannot be combined with specimen that are not liquids. It is easier to explicitly represent these constraints than train models to adhere to them.

Implicit knowledge about the relative importance of LOINC elements is only revealed through conversations with clinical informaticists. For example, clinicians often prioritize the component LOINC element over other elements. A code that has the correct component but incorrect specimen may be acceptable for certain use cases. Because of this implicit prioritization, it is useful to weight elements based on use case rather than taking a completely data-driven approach that treats each element equally in a classifier.

An inference engine was built to combine element predictions into a final LOINC code. The general processing steps are as follows:

1. Start with the highest priority element and generate all candidate LOINC codes that contain the predicted value for that element.

2. Go to the next highest priority element and filter out any candidates that do not have the predicted value for that element.

3. Repeat for all elements, then sort the remaining candidates by a priority weighted average of element confidence values.

For example, the component GRU predicts "Leukocyte Esterase" for the example input string "Ur Leukocyte Esterase". If we start with component as the most essential element, we generate a candidate list of LOINC codes with the predicted value for component: $\{2563-5, 27297-1, 5799-2, 59262-6, 60026-2, 77563-5\}$. We then filter out candidates that do not contain the predicted specimen (or top n predicted specimen). We continue filtering for the rest of the elements in order of priority. Note that it is entirely possible for an input string to be lacking any value for one of the six elements.

Bin	Accuracy %		Coverage %	
	H	R	H	R
$c >.99$	90	71	10	4
$.99 \leq c < 0.75$	85	62	81	63
$.75 \leq c < .5$	80	37	8	23
$c \leq .5$	56	18	2	11

Table 6: Accuracy and coverage percentages binned according to confidence intervals for hybrid and rules-only models. H columns represent hybrid models and R columns represent rules-only models.

5.3 Evaluation

Clinical informaticists average 80% accuracy in a completely manual mapping process. Initial approaches to automate mapping were purely rules-based. Approaches using purely machine learning scored high for element prediction but were less than 70% accurate at predicting final LOINC codes. A hybrid approach combining logic and machine learning provided a dramatic increase in accuracy and coverage. Table 6 shows a comparison. Accuracy metrics are broken into bins based on confidence intervals where c = confidence. Binning was performed to simplify decisions for clinicians. By accepting predictions with a confidence higher than .5, we can achieve human performance of 80% accuracy on a combined coverage of the top three bins or 98% of all incoming custom codes.

5.4 Conclusion

Practical applications of artificial intelligence often require an ensemble of approaches. Combining the multiple approaches can overcome their respective weaknesses in particular use cases. We found that machine learning approaches were best equipped to extract LOINC elements from noisy text inputs, whereas logic-based methods were better at combining those elements into final LOINC codes.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Bora Edizel, Aleksandra Piktus, Piotr Bojanowski, Rui Ferreira, Edouard Grave, and Fabrizio Silvestri. 2019. Misspelling oblivious word embeddings. *CoRR*, abs/1905.09755.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *CoRR*, abs/1604.00788.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. *CoRR*, abs/1905.11268.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Jinzhong Xu, Jie Liu, and Ming Liu. 2009. Research on topic relevancy of sentences based on how net semantic computation. volume 2, pages 195–198.

Yue Zhang and Jie Yang. 2018. Chinese NER using lattice LSTM. *CoRR*, abs/1805.02023.

Attentive Tree-structured Network for Monotonicity Reasoning

Zeming Chen
Computer Science and Software Engineering Department,
Rose-Hulman Institute of Technology
5500 Wabash Ave, Terre Haute, IN, USA
chenz16@rose-hulman.edu

Abstract

Many state-of-art neural models designed for monotonicity reasoning perform poorly on downward inference. To address this shortcoming, we developed an attentive tree-structured neural network. It consists of a tree-based long-short-term-memory network (Tree-LSTM) with soft attention. It is designed to model the syntactic parse tree information from the sentence pair of a reasoning task. A self-attentive aggregator is used for aligning the representations of the premise and the hypothesis. We present our model and evaluate it using the Monotonicity Entailment Dataset (MED). We show and attempt to explain that our model outperforms existing models on MED.

1 Introduction

In this paper, we present and evaluate a tree-structured long-short-term-memory (LSTM) network in which the syntactic information of a sentence is encoded and the alignment between the premise-hypothesis pair is calculated through a self-attention mechanism. Our work builds on the Child-Sum Tree-LSTM from Tai et al. (2015). We evaluate our model on several datasets to show that it performs well on both upward and downward inference. Particularly, our model demonstrated good performance on downward inference, which is a difficult task for most NLI models.

Natural language inference (NLI), also known as recognizing textual entailment (RTE) is one of the important benchmark tasks for natural language understanding. Many other language tasks can benefit from NLI, such as question answering, text summarization, and machine reading comprehension. The goal of NLI is to determine whether a given premise **P** semantically entails a given hypothesis **H** (Dagan et al., 2013). Consider the example below:

- **P**: An Irishman won the Nobel prize for literature.
- **H**: An Irishman won the Nobel prize.

The hypothesis can be inferred from the premise and therefore the premise entails the hypothesis. To arrive at a correct determination, an NLI model often needs to perform different inferences including various types of lexical and logical inferences. In this paper, we are concerned with monotonicity reasoning, a type of logical inference that is based on word replacement. Below is an example of monotonicity reasoning:

1. (a) **All** students↓ carry a MacBook↑.
   (b) All students carry a laptop.
   (c) All new students carry a MacBook.
2. (a) **Not All** new students↑ carry a laptop.
   (b) Not All students carry a laptop.

An upward entailing phrase (↑) can allow inference from (1a) to (1b), where a more general concept laptop replaces the more specific *MacBook*. A downward entailing phrase (↓) allows an inference from (1a) to (1c), where a more specific context *new students* replaces the word *students*. The direction of the monotonicity can be reversed by adding a downward entailing phrase like "Not"; thus (2a) entails (2b).

Recently, Yanaka et al. (2019a) constructed a new dataset called the Monotonicity Entailment Dataset (MED). The purpose of that dataset is to evaluate the ability of a neural inference model to perform monotonicity reasoning. It is the first dataset ever created for such purpose. While many neural language models have shown state-of-art performance on large annotated NLI dataset such as the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015a; Chen et al., 2017; Parikh et al., 2016), many of these models did not

Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA), pages 12–21

perform well on monotonicity reasoning. In particular, they had low accuracy when performing downward monotonicity inference. Additionally, most of the state-of-art inference models that do well on upward monotonicity inference perform poorly on downward inference (Yanaka et al., 2019a).

2 Related Work

Existing work in this area has adopted a recursive tree-structured neural network for natural language inference. Bowman et al. (2015b) proposed a tree-structured neural tensor network (TreeRNTNs) that can learn representations to correctly identify logical relationships such as entailment.

Zhou et al. (2016) extended the recursive neural tensor networks to a recursive long-short term memory network, a tree-LSTM, which combines the advantages of both the recursive neural network structure and the sequential recurrent neural network structure. The tree-LSTM can learn memory cells that reflect the historical memories of the descendant cells and thus improved the model's ability to process long-distance interaction over hierarchies, such as the language parse information.

Parikh et al. (2016) proposed a simple decompose attention model for natural language inference. Their model relies on the attention to decompose the problem into sub-problems so that the smaller problems can be solved separately and in parallel.

Chen et al. (2017), proposed the Enhanced Sequential Inference Model (ESIM) for natural language inference task. It incorporated the sequential LSTM encoder with the syntactic parsing information from the tree-LSTM structure to form a hybrid neural inference mode. They found that incorporating the parsing information can improve the performance of the model.

A new type of inference model that relies on external knowledge called the knowledge-based inference model (KIM) was introduced by Chen et al. (2018). They incorporated neural NLI models with external knowledge in co-attention, local inference collection, and inference composition components. The KIM model achieved state-of-art performance on the SNLI and MNLI datasets.

3 Our Model

In this section we present an attentive tree structured network (AttentiveTreeNet) with self-attention based aggregation. This model is composed of the following main components: input

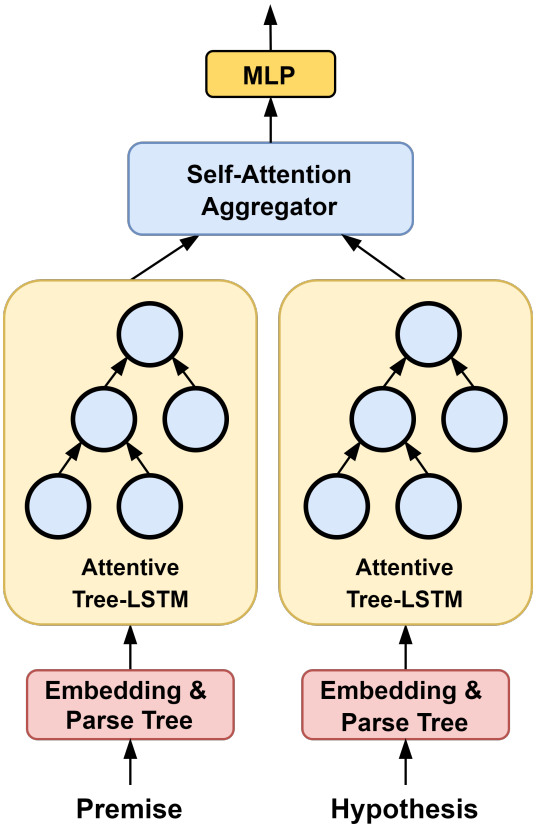

Figure 1: Architecture of our model.

sentence embedding, attentive tree-LSTM encoder, self-attention aggregator and a multi-layer perceptron (MLP) classifier. Figure 1 shows the architecture of our model. Given an input sentence pair, consisting of a premise **P** and a hypothesis **H**, the objective of the model is to determine whether **P** entails **H**. Our model takes in four inputs: the word embeddings of the premise and hypothesis and the dependency parse trees of the premise and hypothesis. The model initializes the embedding of **P** and **H** with some pre-trained word embedding; the parse trees are produced by a dependency parser. Our model forms a Siamese neural network structure (Mueller and Thyagarajan, 2016), in which the premise and the hypothesis are passed into a pair of identical tree-LSTMs that share the same parameters and weights. The main idea is to find a function that can map the input sentences into a target space such that we can approximate the semantic distance in the input space.

3.1 Attentive Tree-LSTM Encoder

Child-Sum Tree-LSTM We employ Child-Sum Tree-LSTMs (Tai et al., 2015) as the basic building blocks for our model. A standard sequential LSTM

network only permits sequential information propagation. However, the *lingistic principle of compositionality* states that an expression's meaning is derived from the meanings of its parts and of the way they are syntactically combined (Partee, 2007). A tree-structured LSTM network allows each LSTM unit to be able to incorporate information from multiple children units. This takes advantage of the fact that sentences are syntactically formed bottom-up tree-structures.

A Child-Sum Tree-LSTM is a type of tree-LSTM which contains units that conditioned their components on the sum of their children's hidden states. While a standard sequential LSTM network computes the current hidden state from the current input and the previous hidden state, a child-sum tree-LSTM computes the hidden state from the input and the hidden states of an arbitrary number of children nodes. This property allows relation representations of non-leaf nodes to be recursively computed by composing the relations of the children, which can be viewed as natural logic for neural model (MacCartney and Manning, 2009; Zhao et al., 2016). Using the child-sum tree structure is beneficial in interpreting the entailment relations between parts of the two sentences.

When encoding the sentence in a forward manner, hidden states are passed recursively in a bottom-up fashion. The information flow in each LSTM cell is controlled by a gating mechanism similar to the one in a sequential LSTM cell. The computations in an LSTM cell are as follows:

$$\tilde{h} = \Sigma_{1 \leq k \leq n} h_k,$$
$$i = \sigma(W^{(i)}x + U^{(i)}\tilde{h} + b^{(i)}),$$
$$o = \sigma(W^{(o)}x + U^{(o)}\tilde{h} + b^{(o)}),$$
$$u = tanh(W^{(u)}x + U^{(u)}\tilde{h} + b^{(u)}),$$
$$f_k = \sigma(W^{(f)}x + U^{(f)}h_k + b^{(f)}),$$
$$c = i \odot u + \Sigma_{1<n} f_k \odot c_k,$$
$$h = o \odot tanh(c),$$

Here, k is the number of children of the current node, and $\tilde{h}$ is the sum of the hidden states from the children of the current node. The forget gate f_k controls the amount of memory being passed from the kth child. The input gate i controls the amount of internal input u being updated and the output gate o controls the degree of exposure of the memory. The σ is the sigmoid activation function, $\odot$ is the element-wise product and W and U are both trainable weights to be learned.

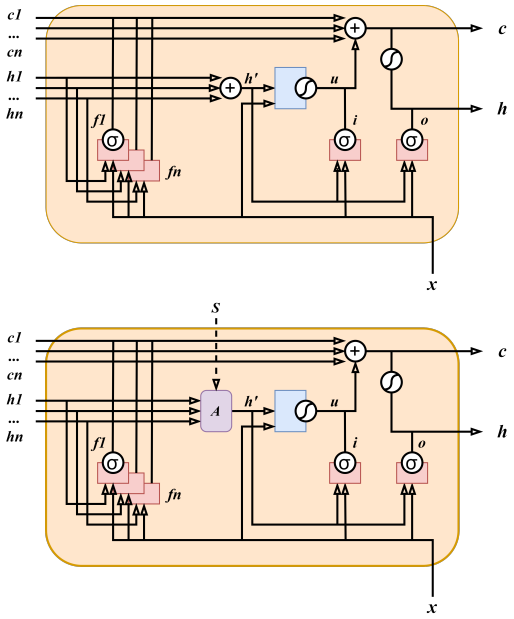

Figure 2: A comparison between a standard LSTM cell and an attentive LSTM cell.

Attentive Tree-LSTM In our model, the standard tree-LSTM is extended to an attentive tree-LSTM (Zhou et al., 2016) by incorporating the attention mechanism into the LSTM cell. In a sentence, some words are more related to the overall context of the sentence than others. The benefit of applying attention is that it considers this semantic relevance by weighting each child according to how relative that child is to the given context. The attention mechanism can assign a higher weight to a child node that is more relevant to the context of the sentence and a lower weight to a child node that is not relevant to the context.

To apply the attention mechanism, a common soft-attention layer is used in the model. That layer receives a set of hidden states $\{h_1, h_2, ..., h_n\}$ and an external vector s, which is a vector representation of a sentence from a layer of sequential LSTM. The layer then computes a weight α for each hidden state, and sums up the product of each hidden state and its weight to output the context vector g. Below are the equations for the soft-attention layer:

$$m_k = tanh(W^{(m)}h_k + U^{(m)}s),$$
$$\alpha_k = \frac{exp(w^\top m_k)}{\Sigma_{j=1}^{n} exp(w^\top m_j)},$$
$$g = \Sigma_{1 \leq k \leq n} \alpha_k h_k$$

A new previous hidden state is then computed through a transformation $\tilde{h} = tanh(W^{(a)}g + b^{(a)})$.

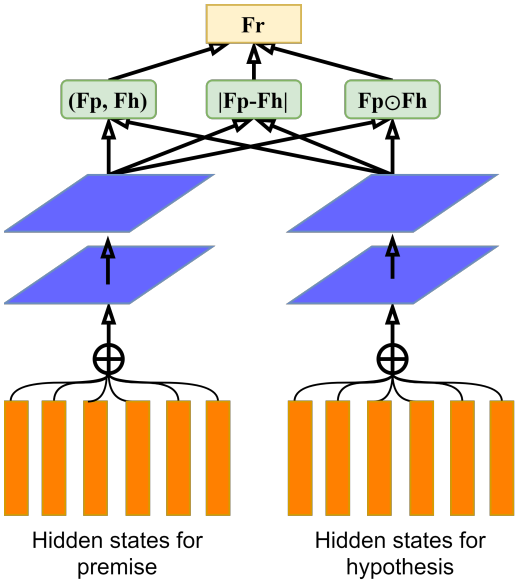

Figure 3: Detailed view of the self-attention aggregator

Figure 2 illustrates the standard tree-LSTM cell and the attentive tree-LSTM cell.

3.2 Self-Attention Aggregator

After both the premise and the hypothesis are encoded through the tree-LSTM, each tree's hidden states from the nodes are concatenated into a pair of matrices H_p and H_h and passed to a self-attentive aggregator. The aggregator contains a multi-hop self-attention mechanism (Lin et al., 2017). A sentence has multiple components such as groups of related words and phrases to form an overall context, especially for long sentences. By performing multiple hops of attention, the model can get multiple attentions that each focus on different parts of the sentence. Given a matrix H, the self-attention mechanism performs multiple hops of attention and outputs an annotation matrix A which consists of the weight vector from each hop. A is calculated from a 2-layer multi-layer perceptron (MLP) and a softmax function. Below is the equation to calculate A:

$$A = softmax(W_{s2}tanh(W_{s1}H^{\top}))$$

The annotation matrix A is then multiplied by the hidden state matrix H to obtain a context matrix: $M = AH$. In the model, there will be a pair of context matrices M_p and M_h. A batch dot product and a *tanh* function is then applied to the context matrices with a trainable weight to obtain a pair of output F_p and F_h matrices:

$$F_p = tanh(bmm(M_p, W_f)),$$
$$F_h = tanh(bmm(M_h, W_f))$$

To aggregate F_p and F_h, we follow Conneau et al. (2017)'s generic NLI training scheme, which includes three matching methods: (i) a concatenation of F_p and F_h, (ii) an absolute distance between F_p and F_h, and (iii) an element wise product of F_p and F_h. Results from the three methods are then concatenated to F_r as the factor of semantic relation between the two sentences which can measure how close the two vector representations of the sentence pair are in the target space. This relatedness information will help the classifier to determine whether the hypothesis is entailed by the premise.

$$F_r = [F_p; F_h; |F_p - F_h|; F_p \odot F_h],$$

3.3 MLP

The factor of relation F_r is fed to a classic three layer MLP classifier. The final prediction is a probability p_θ representing the degree to which the hypothesis is entailed by the premise. It is calculated by a softmax function, which is a standard activation function used to calculate the probability of the input being in a category for multi-way classification tasks:

$$Y_1 = ReLU(W_{f_1}F_r + b_{f_1}),$$
$$Y_2 = \sigma(W_{f_2}Y_1 + b_{f_2}),$$
$$y_\theta = softmax(W_{f_3}Y_2 + b_{f_3}),$$

For the classification, the binary cross-entropy loss is used as the objective function:

$$-\sum_c \mathbb{1}(X, c)log(p(c|X)),$$

where $\mathbb{1}$ is the binary indicator (0 or 1) whether the label c is the correct class for X.

4 Evaluation

4.1 Data

Six different types of training data are used to train our model. Initially, we used the HELP dataset (Yanaka et al., 2019b) to train our model. HELP is a dataset for learning entailment with lexical and logical phenomena. It embodies a combination of lexical and logical inferences focusing on

Model	Train Data	Upward	Downward	None	All
BiMPM (Wang et al., 2017)	SNLI	53.5	57.6	27.4	54.6
ESIM (Chen et al., 2017)	SNLI	71.1	45.2	41.8	53.8
DeComp (Parikh et al., 2016)	SNLI	66.1	42.1	**64.4**	51.4
KIM (Chen et al., 2018)	SNLI	78.8	30.3	53.1	48.0
BERT (Devlin et al., 2019)	MNLI	**82.7**	22.8	52.7	44.7
BERT (Devlin et al., 2019)	HELP+MNLI	76.0	70.3	59.9	71.6
AttentiveTreeNet (ours)	MNLI	54.7	60.4	37.8	58.6
AttentiveTreeNet (ours)	HELP	55.7	72.6	57.9	66.0
AttentiveTreeNet (ours)	HELP+SubMNLI	81.4	**74.5**	53.8	**75.7**

Table 1: Accuracy of our model and other state-of-art NLI models evaluated on MED.

monotonicity. HELP consists of 36K sentence pairs including those for upward monotone, downward monotone, non-monotone, conjunction, and disjunction. Next we trained our model with the Multi-Genre NLI Corpus (MNLI) dataset (Williams et al., 2018). MNLI contains 433k pairs of sentences annotated with textual entailment information. That dataset covers a wide range of genres of spoken and written language. The majority of the training examples in that dataset is upward monotone. In order to provide more balanced training data, we combined a subset of the MNLI dataset with the HELP dataset to reduce the effect of the large number of downward monotone examples in the HELP dataset, we call this combined training data HELP+SubMNLI. The fourth training data contains both the HELP+SubMNLI training data and the training set for simple monotonicity from Richardson et al. (2019)'s Semantic Fragments. The fifth training data contains both the HELP+SubMNLI training data and the training set for hard monotonicity from Semantic Fragments. Finally, the last training data contains the HELP+SubMNLI training data and the training set for simple and hard monotonicity from Semantic Fragments.

To validate our model's ability for monotonicity reasoning and to evaluate its performance on upward and downward inference, the Monotonicity Entailment Dataset (MED) was used (Yanaka et al., 2019a), which is designed to examine a model's ability of performing monotonicity reasoning. MED contains 5382 premise-hypothesis pairs including 1820 upward inference examples, 3270 downward inference examples, and 292 non-monotone examples. The sentences in MED cover a variety of linguistic phenomena, including lexical knowledge, reverse, conjunction, disjunction, conditional and negative polarity items. We removed sentence pair with the label "contradict" from MNLI dataset since the test dataset MED and the training dataset HELP do not contain the label "contradict". We furthermore tested our model on the simple and hard monotonicity fragments test sets from Semantic Fragments.

4.2 Training

Word embeddings are a common way to represent words when training neural networks (Mikolov et al., 2013). To train our model we used Stanford's pre-trained 300-D Glove 840B vectors (Pennington et al., 2014) to initialize the word embeddings. The Stanford Dependency Parser (Chen and Manning, 2014) was used to parse each sentence in the dataset. The model is trained with the Adam optimizer (Kingma and Ba, 2014) which is computationally efficient and helps a model to quickly converge to an optimal result. A standard learning rate for Adam, 0.001, is also used. Dropout with a standard rate of 0.5 is applied to the feed-forward layer in the self-attention aggregator and the classifier to reduce the over-fitting of the model. For the number of hops of the self-attention, we used the default 15 hops. The metric for evaluation is accuracy based. The system is implemented using a common deep learning framework, PyTorch and is trained on a GPU for 20 epochs.

5 Results

5.1 Overall Performance

In this section, we evaluated our model's ability of performing monotonicity reasoning. Table 1 shows a comparison of the performance of different models on the Monotonicity Entailment Dataset (MED), including our model. The data for all models except for ours was developed byYanaka

Test	Model	Training Data	Upward	Downward	None	All
-	Full Model w/ vector-concat	HELP	55.7	72.6	57.9	66.0
1	–Self-Attentive Aggregator	HELP	65.1	67.1	53.7	65.7
2	–Tree-LSTM	HELP	36.6	65.5	94.8	49.5
3	Full Model w/ mean-dist	HELP	59.3	71.2	46.2	65.9
-	Full Model w/ vector-concat	HELP+SubMNLI	**81.4**	**74.5**	53.8	**75.7**
1	–Self-Attentive Aggregator	HELP+SubMNLI	70.5	66.9	85.6	69.1
2	–Tree-LSTM	HELP+SubMNLI	54.7	60.4	37.8	58.6
3	Full Model w/ mean-dist	HELP+SubMNLI	68.9	73.7	**91.0**	73.0

Table 2: This table shows the accuracy of ablation tests trained on HELP and HELP+SubMNLI and tested on MED. Three ablation test were performed: (i) Remove self-attentive aggregator (–Self-Attentive Aggregator), (ii) Replace tree-LSTM with regular LSTM (–Tree-LSTM) (iii) Use mean distance as a matching method (Full Model w/ mean-dist). The final model (Full Model w/ vector-concat) uses a concatenation of the sentence vectors as one of the matching methods instead of mean distance.

et al. (2019a) who developed the MED dataset. Our model achieves an overall accuracy of 75.7% which outperforms all other models, even a state-of-art language model like BERT. Table 1 shows the ability of different models on performing upward and downward inference. Our attentive tree model performed better on downward inference than other models with an accuracy of 74.5% . Our model's performance on upward inference outperforms other models except BERT. However, the upward inference accuracy of our model (81.4) is very close to the accuracy of BERT (82.7). We believe the good performance on upward and downward inference is due to considering parse tree information. Furthermore, the accuracy on upward inference increased significantly when trained with a combination of HELP and MNLI (HELP+SubMNLI) then trained only with HELP; the accuracy increased from 55.7 to 81.4 while the downward accuracy did not change much. Such phenomena suggests that adding MNLI to HELP does reduce the effect of the large number of downward monotone examples in the HELP dataset and thus improve the model's ability on upward inference.

5.2 Robustness of Model

To demonstrate the robustness of our model, we experimented with training the model on various datasets. First, the model was trained on the HELP dataset alone. The overall accuracy was 66.0%, which outperformed other models from Table 1 except BERT trained with HELP+SubMNLI and our model trained with HELP+SubMNLI. Even on downward inference alone our model outperforms all other models with an accuracy of 72.6% except our model trained with HELP+SubMNLI. This result indicates that with a rich set of downward monotone examples, the model can learn to better predict a downward inference problem.

We then trained a model with the MNLI dataset alone. It contains a large amount of upward inference examples and only a rare number of downward inference examples. The result shows that the model generalized to the training data, and had an accuracy of 58.6% which is still higher than most models from Table 1. Interestingly, the model's performance on downward inference is still better than its performance on upward inference, even though the training dataset contains a large number of upward monotone examples. This suggests that the model is immune to significant change of training data possibly due to the multiple dropout layer added to the aggregator and the classifier which forces a the model to learn more robust features. As Table 1 show, comparing to BERT trained with MNLI along, our model trained with MNLI along has better performance on downward inference than BERT's performance from Yanaka et al. (2019a).

Finally, we trained our model on a combination of the MNLI dataset and the HELP dataset (HELP+SubMNLI). Because of the large number of upward training examples in MNLI, we suspected that the combination would alleviate the effects of this distortion and as such increase the accuracy for upward inference. We selected 20% of the complete MNLI dataset due to the long training period. As the results in Table 1 show, our model still performs well on downward inference with 74.5% accuracy, it also showed significant improvements on upward inference with an accuracy of 81.4% . The overall performance also increased

substantially to 75.7% . Compared to the results of BERT trained with HELP+MNLI from Yanaka et al. (2019a), our model performs better on both upward inference and downward inference, and achieves a higher overall accuracy. The result validates our hypothesis that training on a combination of upward and downward monotone sentences can help the model achieve good performance on both upward and downward monotone, and that the use of AttentiveTreeNet is a good choice.

5.3 Ablation Test

To further evaluate which part of the model contributed the most for monotonicity reasoning, we performed several ablation tests on the model. The ablation tests were trained with HELP and HELP+SubMNLI separately and the models were evaluated on the MED dataset. The results are shown in Table 2. We will focus our evaluation on the HELP+SubMNLI data.

For ablation test 1, we removed the self-attentive aggregator and built the feature vector for classification right after the tree-LSTM encoder. As Table 2 (–Self-Attentive Aggregator) shows, performance of the model trained on HELP+SubMNLI shows a significant, 6.6 percentage point drop in overall accuracy, a 10.9 percentage point drop in upward inference accuracy and a 7.6 percentage point drop in downward inference accuracy. The results of this test suggest that the self-attentive aggregator is an important component of the model that cannot be removed.

For ablation test 2, we replaced the tree-LSTM encoder with a standard LSTM encoder. Here, we see an even larger drop in performance. As Table 2 (–Tree-LSTM) shows, performance of the model trained on HELP+SubMNLI shows a large, 17.1 percentage point drop in overall accuracy, a 26.7 percentage point drop in upward inference accuracy and a 14.1 percentage point drop in downward inference accuracy. Based on the results, replacing tree-LSTM with standard LSTM has significant negative impact on the model's monotonicity reasoning performance. Thus, tree-LSTM is a major component of the model that cannot be replaced.

For ablation test 3, we compared two matching methods for aggregating the two sentence vectors. In our final model (Full Model w/ vector-concat), we updated the matching method by following the generic NLI training scheme (Conneau et al., 2017). In it, we concatenate the two sentence vectors with an absolute distance and an element-wise product as the input vector for the classifier. We compared the performance to our original model (Full Model w/ mean-dist) which contains the tree-LSTM encoder, the self-attentive aggregator, and the concatenation of an absolute distance, an element-wise product, and a mean distance as the input vector for the classifier. For this ablation test, the results from Table 2 (Full Model w/ mean-dist) are mixed, yet important. While the overall accuracy decreases just slightly, by 2.7 percentage points and the downward inference accuracy only decreases by 0.8 percentage points, the accuracy for upward inference decreases by a significant 12.5 percentage points. We believe that these results justify the use of concatenation of the sentence vector pair.

Overall, the removal of the Tree-LSTM encoder affected the model's performance most. Thus, we conclude that the Tree-LSTM encoder contributes the most to the model's performance on monotonicity reasoning.

Training Data	**SF**	**HF**	**MED**
Pre-Trained Models			
HELP	57.0	56.8	66.0
HELP+SubMNLI	46.0	63.0	75.7
Re-trained Models w/ SF-training fragments			
HELP+frag	98.1	80.6	64.5
HELP+SubMNLI+frag	97.8	74.8	81.5
Re-trained Models w/ HF-training fragments			
HELP+frag	74.3	95.6	68.9
HELP+SubMNLI+frag	73.9	93.2	73.3
Re-trained Models w/ SF and HF-training fragments			
HELP+frag	96.9	94.6	64.5
HELP+SubMNLI+frag	96.4	98.3	75.4

Table 3: This table shows the result of the model tested on MED and the simple monotonicity fragments test set (SF) and hard monotonicty fragments test set (HF) from the Semantic Fragments dataset. The table includes three subsections: (i) test accuracy on the three test sets using models pre-trained on HELP and HELP+SubMNLI; (ii) test accuracy on the three test sets using the model re-trained after adding simple monotonicity training set to HELP and HELP+SubMNLI; (iii) test accuracy on the three test sets using the model re-trained after adding hard monotonicity training set to HELP and HELP+SubMNLI; (iv) test accuracy on the three test sets using the model re-trained after adding both simple and hard monotonicity training sets to HELP and HELP+SubMNLI.

5.4 Additional Testings

To check if our pre-trained model can be generalized to other monotonicity dataset, and to see if the model can be easily trained to master the new dataset while retaining its performance on the original benchmark, we conducted some additional testings on the model. We tested our pre-trained models on the Semantic Fragments test dataset which provides a more in-depth test for an NLI model's performance with semantic phenomena, see (Richardson et al., 2019). Since our model focuses on monotonicity reasoning, we only selected the simple and hard monotonicity fragments for testing. Additionally, since our models are pre-trained on datasets that only contain two labels: "Entailment" and "Neutral", we removed sentence pairs with the third label "contradict" from the test dataset.

Table 3 shows the results of our testing. While we show the results for both, the HELP and HELP+SubMNLI data sets, we will focus our discussion again on the data obtained with the HELP+SubMNLI data set.

The top portion of Table 3 shows that the model trained on just HELP+SubMNLI performs poorly on the simple and hard monotonicity fragments. This performance is on par with other state-of-art model's, see (Richardson et al., 2019).

The first middle portion on Table 3 shows the results of our model's performance when only the *simple* training fragments were added to the HELP+SubMNLI training set. As the data shows, the model masters the simple monotonicity reasoning tests, does well on the hard monotonicity reasoning tests and retains its accuracy on the original benchmark MED.

The second middle portion of Table 3 shows the results of our model's performance when only the *hard* training fragments were added to the HELP+SubMNLI training set. In this case, the model masters the hard monotonicity reasoning tests, does well on the simple monotonicity reasoning tests and again retains its accuracy on the original benchmark MED.

The bottom portion of Table 3 shows the results of our model's performance when both the *simple* and *hard* training fragments were added to the HELP+SubMNLI training set. As the results show, the model masters both the simple and hard monotonicity reasoning tests while retaining its accuracy on the original benchmark MED.

Overall, the results show that the model trained on the fragments can be generalized to both simple and hard monotonicity reasoning.

6 Conclusions

In this paper, we explained our attentive tree-structured network to perform monotonicity reasoning. Our model combines a tree-structured LSTM network and a self-attention mechanism, which is a potential mechanism for future natural language inference models, to incorporate syntactic structures of the sentence to improve sentence-level monotonicity reasoning. We evaluated our model and showed that it achieves better accuracy on monotonicity reasoning than other inference models. In particular, our model is performing significantly better on downward inference than others. We interpret the results of the experiments as supporting the thesis that using parse trees of a sentence are helpful in inferring the entailment relation.

Future research on the attentive tree network might extend a tree-LSTM architecture by replacing the LSTM cell with newer language models that have much better performance on various number of natural language processing tasks. One such model is the transformer model. Furthermore, future work might want to investigate how different attention mechanism affect a model's performance.

Acknowledgments

We thank Michael Wollowski for reading and giving feedback on drafts and revisions of this paper. We also thank the anonymous reviewers for providing helpful suggestions and feedback.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. 2015b. Recursive neural networks can learn logical semantics. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 12–21, Beijing, China. Association for Computational Linguistics.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on*

Empirical Methods in Natural Language Processing (EMNLP), pages 740–750, Doha, Qatar. 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.

Yufei Chen, Sheng Huang, Fang Wang, Junjie Cao, Weiwei Sun, and Xiaojun Wan. 2018. Neural maximum subgraph parsing for cross-domain semantic dependency analysis. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 562–572, Brussels, Belgium. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool Publishers.

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.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *ArXiv*, abs/1703.03130.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the Eight International Conference on Computational Semantics*, pages 140–156, Tilburg, The Netherlands. Association for Computational Linguistics.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2786–2792. AAAI Press.

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.

Barbara Partee. 2007. Compositionality and coercion in semantics: The dynamics of adjective meaning 1.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

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

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4144–4150.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122. 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.

Kai Zhao, Liang Huang, and Mingbo Ma. 2016. Textual entailment with structured attentions and composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2248–2258, Osaka, Japan. The COLING 2016 Organizing Committee.

Yao Zhou, Cong Liu, and Yan Pan. 2016. Modelling sentence pairs with tree-structured attentive encoder. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2912–2922, Osaka, Japan. The COLING 2016 Organizing Committee.

Transferring Representations of Logical Connectives

Aaron Traylor
Dept. of Computer Science
Brown University

Ellie Pavlick
Dept. of Computer Science
Brown University

Roman Feiman
Dept. of Cognitive, Linguistic, and Psychological Sciences
Brown University

{aaron_traylor, ellie_pavlick, roman_feiman}@brown.edu

Abstract

In modern natural language processing pipelines, it is common practice to "pretrain" a generative language model on a large corpus of text, and then to "finetune" the created representations by continuing to train them on a discriminative textual inference task. However, it is not immediately clear whether the logical meaning necessary to model logical entailment is captured by language models in this paradigm. We examine this pretrain-finetune recipe with language models trained on a synthetic propositional language entailment task, and present results on test sets probing models' knowledge of axioms of first order logic.

1 Introduction

In modern natural language processing pipelines, it is common practice to "pretrain" a generative language model on a large corpus of text, and then to "finetune" the created representations by continuing to train them on a discriminative textual inference task. This pretrain-finetune recipe has led to state-of-the-art accuracy on natural language inference tasks (Wang et al., 2018), including tasks that explicitly target logical, compositional reasoning (Williams et al., 2017). However, *a priori*, it is not obvious that language model pretraining should lead models to encode anything mirroring the functions of logical concepts such as entailment, negation, and disjunction, despite their importance in language, and in fact there are many reasons why we might directly expect language modeling *not* to encode logical meaning: for example, language modeling does not provide models with access to variable bindings or truth values (a key component for defining logical functions), and it only provides models with access to positive training examples (sentences that are true in some possible world, assuming they were true at the time they were uttered) but not negative examples (sentences that have never been attested). This difference between our expectations for these models and their empirical performance gives rise to a broader question: under what conditions can core logical concepts emerge, and what information is sufficient to learn them? We assume entailment as a task is crucial to beginning to form an answer; if a system can accurately determine whether a given premise logically entails a given hypothesis, it may have a representation of logical conjunction, disjunction, conditionals, and negation. Thus, by observing how language model pretraining affects the performance of a model of logical entailment, we can better understand the extent to which these models capture logical reasoning capabilities, if at all. The primary question we aim to answer is: does the traditional language modelling objective result in representations which readily support classical logical reasoning?

Due to the pervasiveness of statistical artifacts in natural language inference datasets (Gururangan et al., 2018; Tsuchiya, 2018), directly assessing the logical reasoning capabilities of neural models using these datasets is challenging. Thus, we design a set of experiments using a toy dataset and task which uses propositional logic sentences in place of natural language, but otherwise uses the same common pretraining+finetuning recipe.

Note that we do not aim to answer the question "can neural networks do logical reasoning" in general. This question has been explored extensively elsewhere (see e.g. (Bowman et al., 2014; Geiger et al., 2018; Evans et al., 2018)). Rather, we are interested in understanding what aspects of logical reasoning can be encoded during unsupervised pretraining.

Specifically, we test whether sentence encodings learned on a language modeling task transfer to a "downstream" entailment task, i.e. improving the efficiency and/or accuracy of the model trained on the downstream task.

Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA), pages 22–25

2 Experimental Design

2.1 Dataset creation

We create two propositional logic language datasets– one for each of the pretraining and entailment steps. For the pretraining corpus, we create 500,000 unique well-formed propositional logic sentences containing atomic symbols and logical connectives, which is roughly the size of a large natural language corpus. These sentences are generated by nesting between 2 and 14 clauses, each containing a unary or binary logical connective from the set $(\neg, \models, \&, \|)$ and atomic symbols. In order to parallel a natural language pretraining corpus, we construct our data such that only logically consistent sentences are present in the propositional logic language. E.g. $A\&B$ might appear as a sentence in our corpus, whereas $A\&\neg A$ is logically inconsistent and would not be included. There are 30,000 unique symbols across our dataset, and symbols are distributed between atomic sentences uniformly, as opposed to the Zipfian distribution of natural language: this is designed in order to make it extremely challenging for the model to make judgments based on co-occurence statistics of symbols.

For the entailment training dataset, 100,000 unique propositional logic premise/hypothesis pairs are generated using the same sentence-generating algorithm as used for the pretraining dataset. As done by (Evans et al., 2018), the dataset is balanced such that, for each premise/hypothesis pair (A_1, A_2), there is a corresponding premise/hypothesis pair (B_1, B_2) such that $A_1 \models A_2$ and $B_1 \models B_2$ but $A_1 \not\models B_2$ and $B_1 \not\models B_2$. This reduces the effect of artifacts resulting from sentence generation, as each sentence is used evenly with each of the *entailed* and *not entailed* labels. We hold out an additional 5,000 pairs containing sentences not seen in training as a validation dataset.

2.2 Language model pretraining and finetuning

The objective of our model during pretraining is the typical language modelling objective. We use unidirectional LSTMs (Hochreiter and Schmidhuber, 1997), as well as Transformers (Vaswani et al., 2017) as sequence models. We train models until the perplexity converges to a local minimum, using early stopping on a validation set.

We use a simple linear classifier based on the last step of the sequence model to determine model predictions on the entailment task. We compare the performance at the entailment task of pretrained sequence models to those initialized from scratch.

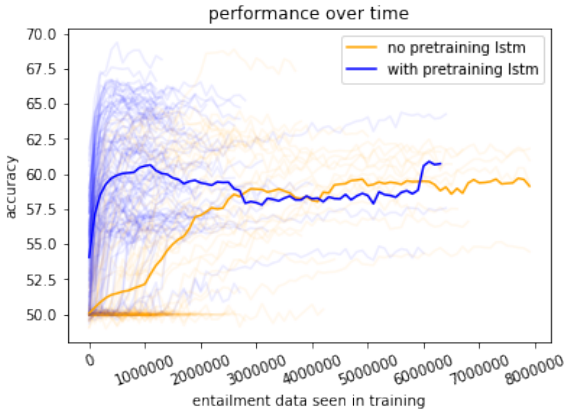

Figure 1: Average accuracy of 100 different hyperparameter settings of LSTMs on the entailment validation dataset over time.

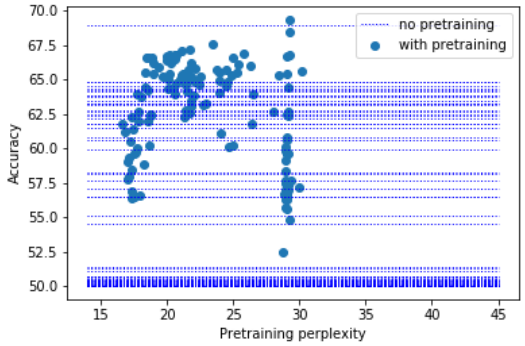

Figure 2: Perplexity versus accuracy for 100 different LSTM hyperparameter settings.

2.3 Axiomatic test sets

We create several diagnostic test datasets representing axioms of first order logic (e.g. the double negation diagnostic dataset contains entailment pairs of the structure A entails $\neg\neg A$.) These diagnostic datasets allow a fine-grained look at which functional elements of logical connectives the network is or is not able to capture after training. Sentences within a diagnostic dataset range across lengths to observe whether length significantly impacts the models' judgments. Performance on these datasets is judged by F1 score– there are several distractor negative examples based on each pattern.

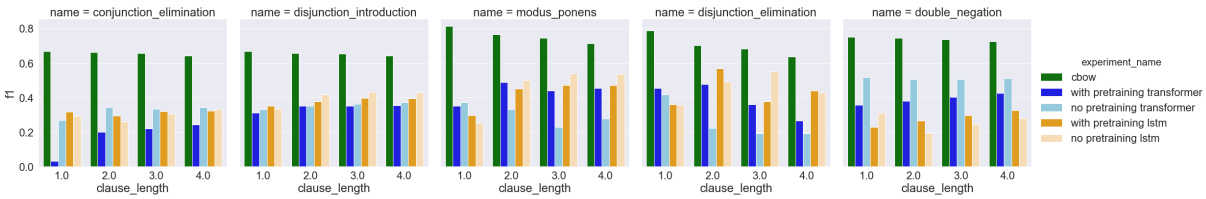

Figure 3: F1-score on axiomatic test sets of the best performing classifier model of each of the four conditions.

Model	w/ Pretraining	w/o Pretraining
CBOW	-	51.47
LSTM	70.41	68.74
Transformer	70.54	63.95

Table 1: Best performance on entailment validation set across hyperparameter sweep of each model.

3 Experiments

We compare the performance on the entailment task of a classifier on top of a model pretrained on the language modelling objective described in 2.1 to that of a classifier on top of a model with randomly initialized parameters. Models were implemented in pytorch (Paszke et al., 2019) and were trained using GPUs. We conduct a large hyperparameter sweep for each of our four settings: LSTMs and Transformers as classifier models with and without pretraining before the entailment task. We search across learning rate, hidden dimension, symbol embedding dimension, dropout, weight decay, and, respective to each model, number of stacked LSTMs and number of transformer heads/layers.

4 Results

The entailment dataset is evenly balanced, so maximum class accuracy is 50%. As shown in Table 1, the performance of the different types of classification models is comparable. The baseline continuous bag of words model performs slightly above chance.

Figure 1 displays the validation accuracy trajectory over time of 100 different hyperparameter settings for LSTM initialized with and without pretraining. Models that do not reach a new maximum accuracy within 10 epochs have their training stopped early. Most of the pretrained models' performance spikes early, while many models initialized from scratch take much longer and must see the training data many more times before their accuracy begins to increase.

Figure 2 plots the relationship between the perplexity and accuracy that LSTM models converge to. There appears to be a local minimum of perplexity that the models achieve in pretraining, and this data suggests no relationship between lower perplexity of the language model and higher accuracy.

The models do not perform well on the diagnostic axiomatic test set dataset across the board, as observed in Figure 3. Despite the extensive training in each scenario, the models are unable to regularly capture even extremely simple patterns, such as double negation of one clause. The effect of clause length varies– the models may be picking up on dataset artifacts for shorter premises and hypotheses. Furthermore, despite its low performance at the entailment task, the F1 score of the continuous bag of words model is significantly higher than that of the other models. This is an interesting trend that could be due to the pretraining leading the model into a poor initialization, which requires further investigation.

5 Discussion

In our experiments, we find mixed evidence to suggest that language model pretraining on sentences gives a reliable performance increase over models trained entirely from scratch with regards to purely logical entailment. On average, pretrained LSTMs and Transformers perform slightly better than those initialized entirely from scratch, but the performance of these models at held-out test sets which specifically probe fundamental axioms of logical entailment is underwhelming and worse than the baseline model. Furthermore, no model performs especially well at the entailment task– logical operators are applied inflexibly to their input, and no model performs near 100% accuracy.

It is potentially the case that, within modern natural language inference pipelines, the logical information encoded at the pretraining step is partially responsible for the increase in accuracy at downstream tasks. However, it is hard to say that any such transfer of representations that facilitate

explicit logical capabilities occurs. It is possible that language models, if they do not preserve logical information, lead models to perform well at entailment tasks in general by capturing complex lexical heuristics and associations between topics at the pretraining step.

In future work, we will test whether pretraining on both positive and negative examples affects entailment performance. Furthermore, we will test whether adjusting from a uniform distribution of the symbols to a Zipfian distribution causes the performance of the pretrained model to improve, which would suggest that these models rely much more heavily on exploiting frequency heuristics than captured logical capabilities to more accurately model the training data.

References

Samuel R Bowman, Christopher Potts, and Christopher D Manning. 2014. Recursive neural networks can learn logical semantics. *arXiv preprint arXiv:1406.1827*.

Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. 2018. Can neural networks understand logical entailment? *arXiv preprint arXiv:1802.08535*.

Atticus Geiger, Ignacio Cases, Lauri Karttunen, and Christopher Potts. 2018. Stress-testing neural models of natural language inference with multiply-quantified sentences. *arXiv preprint arXiv:1810.13033*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. *arXiv preprint arXiv:1804.08117*.

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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Monotonic Inference for Underspecified Episodic Logic

Gene Louis Kim[♡], **Mandar Juvekar**[♢], **and Lenhart Schubert**[♣]
University of Rochester
Department of Computer Science
{gkim21[♡],schubert[♣]}@cs.rochester.edu
mjuvekar[♢]@u.rochester.edu

Abstract

We present a method of making natural logic inferences from Unscoped Logical Form of Episodic Logic. We establish a correspondence between inference rules of scope-resolved Episodic Logic and the natural logic treatment by Sánchez Valencia (1991a), and hence demonstrate the ability to handle foundational natural logic inferences from prior literature as well as more general nested monotonicity inferences.

1 Introduction

Natural Logic is an approach to generating inferences from language directly over the grammatical structure through knowledge of entailment monotonicity in the lexicon. Monotonicity is a characteristic of functions within an ordering, i.e. f is upward monotone if $x \leq y$ implies $f(x) \leq f(y)$ (and downward monotone for the opposite). For example, *not* is downward monotone in entailment since it flips the entailment ordering of "Fido is a dog" entails "Fido is an animal" to "Fido is *not* an animal" entails "Fido is *not* a dog". Natural Logic can be seen as an extension of Aristotelian syllogistic reasoning (Van Benthem et al., 1986) and was first formally related to higher-order logic entailments by Sánchez Valencia (1991a). Icard and Moss (2014) and Icard et al. (2017) later construed Natural Logic as a formal system of its own, independent of a separate logical formalism.

Unscoped Logical Form (ULF) of Episodic Logic was developed with the aim to integrate machine learning into automatic natural language inference by simplifying the semantic parsing task that presupposes symbolic inference (Kim and Schubert, 2019). ULFs retain certain ambiguities in the sentence while strictly defining the core semantic type structure that is necessary to specify the compositional structure. This results in a parsing task that is similar in form and complexity to constituency parsing, for which the community has built effective parsers (Mrini et al., 2019; Zhou and Zhao, 2019). Promising preliminary results show parsability of ULF from a small dataset and minimal representation-specific knowledge (Kim, 2019). Automatic inference generation from ULF has been demonstrated for dialogue-focused structural inferences which correspond to simple presuppositions and implicatures over questions, requests, counterfactual constructions, and clause-taking verbs (Kim et al., 2019).

Here we present a proof-based Natural Logic inference formalism for ULF. We show that this method covers inferences presented by Sánchez Valencia (1991a) and can support Rule Instantiation from Episodic Logic (EL) inference for nested polarity inference. The contributions of this paper are two-fold: (1) this marks the first formalized inference procedure for ULF and (2) we present an alternative to parsing full logical formulas in symbolic Natural Logic inference through the use of an underspecified representation. An implementation of the inference procedure that we describe is beyond the scope of this paper, but is an important next step for empirically evaluating the efficacy of this approach against existing Natural Logic inference systems. Due to space limitations, we leave fully formalized definitions and proofs to the appendix and use the main document for condensed explanations and demonstrative examples.

2 Natural Logic

We will limit our discussion of Natural Logic to that presented by Sánchez Valencia (1991a) and follow his notation and terminology.[1] The semantics of Sánchez-Valencia's Natural Logic is rooted in an undirected, typed lambda calculus constructed

[1] Much of the recent work in Natural Logic uses the terminology and notation introduced by Icard and Moss (2014).

Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA), pages 26–40

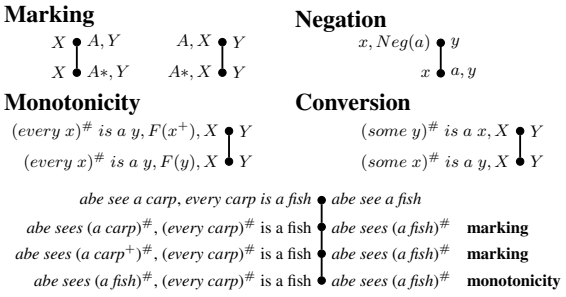

Figure 1: The basic inference rules for Sánchez-Valencia's natural logic proof system and an example.

from derivations of Lambek cum Permutation Calculus (Lambek, 1988). The primitives semantic types are e and t (for entities and truth values) which denote sets, with complex types of the form $\langle a, b\rangle$ where a and b are semantic types in the language. There is one inference rule over these, $\langle \alpha, \beta\rangle$, $\alpha \rightarrow \beta$, where the order of the functor and the argument do not matter.

Sánchez-Valencia formalizes the monotonicity of polarity lexical terms from the linguistic literature of Natural Logic in relation to this semantic framework in order to make soundness claims of predicate substitutions within positive and negative polarity contexts. Polarity contexts are determined by counting the number of downward entailing arguments that lie in between a constituent and the root of the Lambek derivation.

Inference Reasoning is done with a tableau proof system (Beth, 1955) starting with a node with the premises, a_i, on the left and conclusion, b, on the right like so, $a_1, ..., a_n \bullet b$, where all statements are in plain English. This is accompanied by the Lambek analyses for each of the statements, which supply grammatical information (scoping and polarity) to the proof. The tableau is closed when all paths in the proof tree are closed and a path is closed when the leaf of the path has the same statement (including scope marking) on both sides of the node. A^+ and A^- mark positive and negative polarity, respectively, and $(A)^\#$ marks the outermost operator scope. The inference rules in Sánchez's proof system needed for demonstrating basic monotonic inference and an example are displayed in Figure 1.

3 Episodic Logic: Unscoped Logical Form

Episodic Logic (EL) is an extended FOL that closely matches the form and expressivity of natural language, using type-shifters and a liberal ontology of individuals (e.g. basic individuals, situations, propositions, kinds, etc.) to keep the logic first-order while allowing for intensionality, general quantifiers, etc. (Schubert, 2000). EL supports deductive and uncertain inference, including forward and goal-chaining inference that uses polarity-based substitution in a Natural Logic-like manner (Hwang and Schubert, 1993; Morbini and Schubert, 2009; Schubert, 2014). The forward inference rules are basically the same as nested inference rules proposed by (Traugott, 1986).

(|Ali| (do.aux-s **not** (know.v (that
(i.pro (work.v (adv-a (with.p **(a.d dog.n)**))))))))
↓ **scoping**
(not (|Ali| (do.aux-s (know.v (that
(a.d x: (x dog.n)
(i.pro (work.v (adv-a (with.p **x**)))) **)**)))) **)**

Figure 2: Example of a ULF scoping into an SLF for the sentence *Ali does not know that I work with a dog*.

ULF fully specifies the semantic type structure of EL by specifying the types of the atoms and all of the predicate-argument relationships while leaving operator scope, anaphora, and word sense unresolved (Kim and Schubert, 2019). The name, Unscoped Logical Form, is a label for its stage in the interpretation process of EL and does not mean that scoping is the only unresolved aspect of the logical form. Kim and Schubert (2019) describe the role of ULF in the interpretation process.

The types of ULF atoms that correspond to surface words and are not logical or macro symbols are marked with suffixed tags resembling the part-of-speech (e.g. .v, .n, .pro, .d for verbs, nouns, pronouns, and determiners). Case-sensitive symbols such as names and titles are marked with pipes (e.g. |John|). Pipe-marked symbols may be left without a type tag in which case they default to having an entity type. A closed set of logical and macro symbols have unique types so the type marking is omitted. Each suffix indicates a set of possible semantic denotations, e.g. .pro always denotes an *entity* and .v denotes an *n-ary predicate* where *n* can vary.

Type shifters in ULF maintain coherence of the semantic type structure. For example, the type shifter adv-a maps a predicate into a predicate modifier as in the prepositional phrase *"with a dog"* in Figure 2, as opposed to its predicative use *"I am*

with my dog".

The syntactic structure is closely reflected in ULF even under syntactic movement through the use of simple rewriting *macros* which explicitly mark these occurrences and upon expansion make available the exact semantic argument structure.

The ordering of operator-argument relations in ULF can have the operator in the first or second position, disambiguated by the types of the participating expressions. The EL type system only allows function application for combining types, $\langle A, B\rangle, A \rightarrow B$, much like Montagovian semantics (Montague, 1970) without type-raising.

Scoped Logical Form (SLF) SLF is ULF with explicit scoping. Since polarity propagates through scoped operator relations, scopes must be fully specified before adding polarities. While inferences will interface with ULFs, auxiliary SLFs are necessary to model the polarities and bookkeep scope-related assumptions in the inferences. Scoped operator orderings are represented using parentheses, and are lifted around the sentence that it scopes around. Scoped determiners are represented as $(\delta\ \nu\colon\ \phi\ \psi)$ where δ is a determiner, ν is a variable, ϕ is the restrictor wff, and ψ is the scope wff. Figure 2 shows examples of the scoping process.

4 Inference with ULF

Scope marking Rather than using a Lambek analysis for identifying operator scopes and hence polarities, we use SLFs.[2] The scoping of determiners leads to decoupled representations of the scoped constituent, so we must define a correspondence that allows us to mark the scoping of the ULF based on a fixed realization of the scoping.

For a ULF, ψ, that contains a quantified expression $\wr$ of form the $(\delta\ \pi)$, where δ is a determiner and π is a predicate, the corresponding formula with $(\delta\ \pi)$ at the top-level scope is $(\delta\ x\colon\ (x\ \pi)\ \psi[\wr/x])$.

Top-level scope marking process Given the SLF that defines the scope ordering, the constituent of the form $(\delta\ \pi)$ in ψ at the position of x in $\psi[\wr/x]$ is marked with $\#$ as the top scope of ψ.[3] Below is an example to help illustrate the mapping.

"Abelard sees a carp"
SLF (a.d x: (x carp.n) (|Abelard| (see.v x)))
Marked ULF (|Abelard| (see.v (a.d carp.n)$^{\#}$))
$\boldsymbol{\delta}$: a.d, $\boldsymbol{\pi}$: carp.n

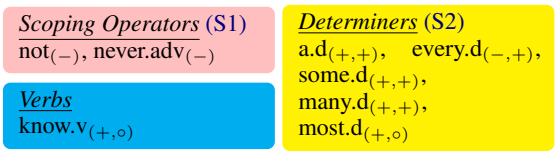

Figure 3: Examples of lexical monotonicity markings.

Polarity marking We perform polarity marking in a two stage process that mirrors the process used by Sánchez Valencia (1991a). First we classify lexical entries according to their monotonicity properties—in what entailment contexts they place their arguments—and mark them in the SLF with parenthesized subscripts. The possible entailment options are $+$ for upward, $-$ for downward, and $\circ$ for none. Figure 3 provides a few examples.[4]

Using the lexical annotations, we mark the local entailment direction of the constituents in the SLF using subscripts *without parentheses*. Finally, the global polarity is derived from these local entailment directions and marked with superscripts. The global polarity is computed by traversing the SLF from the root and counting the number of occurrences of negative and flat entailments, with the following rules.

1. Node a has no polarity if any node in the path from the root to a is marked with $\circ$.
2. Else, node a has negative polarity if there are an odd number of nodes between the root and a (inclusive) marked with $-$.
3. Otherwise, node a has positive polarity.

Figure 4 shows all of these markings in a tree format. We then mark the global polarity in the ULF according to the corresponding SLF to get.
((no.d^{+} scientist.n^{-})
(know.v^{-} (every.d^{-} (scientific.a^{+} fact.n^{+})$^{+}$))$^{-}$)

Inference Rules Figure 5 lists the ULF versions of the monotonicity and conversion rules from Figure 1, but in a standard rule of inference format. Sánchez-Valencia's Marking and Negation rules are specific to the tableau system and not relevant

[2] In accordance with Sánchez-Valencia's treatment, we do not address the possible scoping complexities of including sentential modifiers, tenses, and aspect as scoped operators.

[3] $\wr$ is not an alias for the pattern $(\delta\ \pi)$. Rather it refers to a unique constituent of ψ that has the form $(\delta\ \pi)$. This is an important distinction in order to properly handle sentences with multiple constituents of the same form, e.g., *"A dog greets a dog"*.

[4] Unmarked lexical entries are assumed to have upward entailment on all of their arguments.

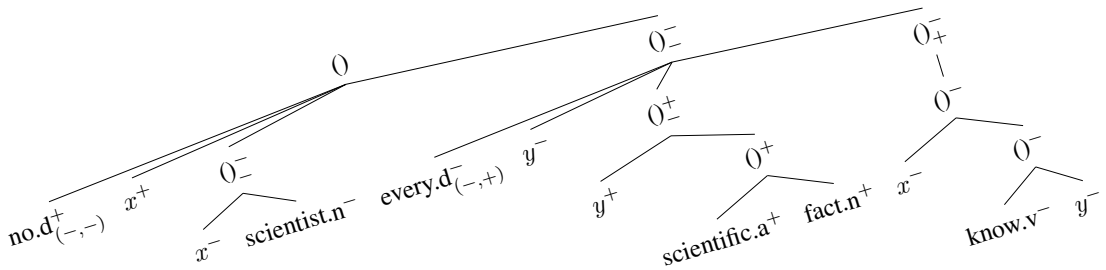

Figure 4: A tree representation of the SLF for "No scientist knows every scientific fact." with all lexical monotonicity, local entailment context, and global polarity markings.

Monotonicity (UMI)

$$\frac{\phi[(\delta\ P1)^{+}],\ ((\text{every.d}\ P1)\ (\text{be.v}\ (=\ (\text{a.d}\ P2))))}{\phi[(\delta\ P2)]}$$

where δ is a determiner.

Conversion (UCI)

$$\frac{((d1\ P)\ (\text{be.v}\ (=\ (d2\ Q))))}{((d1\ Q)\ (\text{be.v}\ (=\ (d2\ P))))}$$ where $d1 \in$ {some.d, a.d, no.d} and $d2 \in$ {some.d, a.d}.

Figure 5: Inference rules in ULF corresponding to basic inference rules for Sánchez-Valencia's natural logic proof system.

as a general logical inference rule. Derivations and proofs are available in Appendix B.

We can also define the corresponding monotonicity rule for the negative polarity context. The monotonicity rule in ULF handles the explicit copula, through the transparent semantic interpretation of 'be.v'.

Example Now we use these ULF rules to perform the inferences from Figure 1.[5]

Basic Monotonicity Example with ULF

1. (|Abelard| (see.v (a.d carp.n))) — Assumption
2. ((every.d carp.n) (be.v (= (a.d fish.n)))) — Assumption
3. (a.d x: (x carp.n)$^{+}$ (|Abelard| (see.v x)$^{+}$)$^{+}$) — SLF of 1. w/ polarity
4. (|Abelard| (see.v (a.d carp.n)$^{+}$)) — Pol marking 1.,3.
5. (|Abelard| (see.v (a.d fish.n))) — UMI 2.,4.

It turns out that the monotonicity rules so far are special cases of EL Rule Instantiation, which operates on substitution under arbitrarily nested polarity contexts (Schubert and Hwang, 2000).

$$\text{RI-1}\ \frac{MAJ(\phi^{-}), MIN(\phi'^{+})}{MAJ_{\sigma}(\neg MIN_{\sigma}(\bot^{+})^{-})} \qquad \text{RI-2}\ \frac{MAJ(\phi^{-}), MIN(\phi'^{+})}{MIN_{\sigma}(MAJ_{\sigma}(\top^{-})^{+})}$$

[5] Appendix D demonstrates how to handle all traditional Aristotelian syllogisms.

where RI-1 is sound if the only variables in the matching expression (ϕ') of the minor premise (MIN) are "matchably bound,"—bound within ϕ' or by a universal quantifier in positive polarity context or existential quantifier in negative polarity context—and RI-2 is sound if the only variables in the matching expression (ϕ) of the major premise (MAJ) are "matchably bound."

UMI is a special case of RI-2 and the negative polarity version is a special case of RI-1. These can handle inferences where the major premise is a more complex construction than *every p is a q*. RI-2 can be used to conclude *Something is a cap or pretty if Little Red Riding Hood wears it* from *Every dress or hood that Little Red Riding Hood wears is pretty* and *Something is a cap or a hood*. See Appendix C for a thorough discussion on Rule Instantiation.

5 Integration with Machine Learning

While a working inference system is beyond the scope of this paper, in this section we discuss some ways in which machine learning can be leveraged in conjunction with the inference formalism that we describe in this paper. An obvious and important role for machine learning in building a ULF-based inference system is to train a semantic parser to provide ULFs for English sentences. Our preliminary work in this direction (Kim, 2019) using an annotated dataset has shown promising results. Our current method is to train an LSTM to parse action sequences for a cache transition parser (Gildea et al., 2018). Including contextual embeddings such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) as inputs to such as model will allow the parser to use the representational power of these embeddings to select the most appropriate parse.

Similarly, we can expect polarity labeling algorithms to improve with the introduction of contextual embeddings, though we are unaware of any work that has tried to do this. This labeling could

also have a collaborative effect with a symbolic polarity labeling. With a partially complete lexicon of negative polarity inducing operators, a ULF could verify parts of the sequentially labeled polarities or correct them if inconsistencies are found in the graph where lexical knowledge is available.

For tasks like SICK (Marelli et al., 2014) which rely largely on lexical specializations, we envision using a lexical resource like WordNet (Miller, 1995). There still remains the issue of word sense, which is not resolved in ULF. Again distributional word representations could be used here to select the most appropriate word sense or set of word senses. Tasks that provide all the necessary relationships such as FraCaS (Cooper et al., 1996) do not require any additional axioms beyond inference rules for basic logical operators and for introducing and eliminating macro operators. For example, modeling relative clauses, which appear frequently in FraCaS, simply requires properly handling the relativizer and post-nominal modification macros to get a monotonicity ordering between predicates "A" and "A that B" that is fully modeled by logical conjunction "A" and "(λ x: ((x A) $\wedge$ (x B)))".

The near-syntactic nature of ULF allows accurate generation of English sentences corresponding to formulas (Kim et al., 2019). An interesting question is whether generative language models could be used to enhance inference generation. Such models are trained to learn the patterns of language use, and as such do not necessarily reflect valid entailments. But anchoring the use of language models to a symbolic representation like ULF would potentially enable constraining inferences to interpretable ones.

6 Conclusion

We have presented a proof-based formalism for making natural logic inferences from ULFs with in-proof scoping assumption declarations using less ambiguous, scoped LFs. This inferential capacity of ULF, in conjunction with its ease of parsing, positions ULF as a promising representational basis for automatically generating natural logic inferences. Machine learning tools can then be deployed for semantic parsing (ULFs) and sequence labeling (polarities), both well-researched paradigms, rather than building a model of Natural Logic directly on top of statistical tools.

Acknowledgments

This work was supported by NSF EAGER grant NSF IIS-1908595, DARPA CwC subcontract W911NF-15-1-0542, and a Sproull Graduate Fellowship from the University of Rochester. We are grateful to Hannah An, Sapphire Becker and the anonymous reviewers for their helpful feedback.

References

Evert Willem Beth. 1955. Semantic entailment and formal derivability. In *Proceedings of the Section of Sciences*, 18, page 309–342. Koninklijke Nederlandse Akademie van Wentenschappen.

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

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.

Daniel Gildea, Giorgio Satta, and Xiaochang Peng. 2018. Cache transition systems for graph parsing. *Computational Linguistics*, 44(1):85–118.

C.H. Hwang and L.K. Schubert. 1993. Episodic Logic: A situational logic for natural language processing. In P. Aczel, D. Israel, Y. Katagiri, and S. Peters, editors, *Situation Theory and its Applications 3 (STA-3)*, pages 307–452. CSLI.

Thomas Icard, Lawrence Moss, and William Tune. 2017. A monotonicity calculus and its completeness. In *Proceedings of the 15th Meeting on the Mathematics of Language*, pages 75–87, London, UK. Association for Computational Linguistics.

Thomas F Icard and Lawrence S Moss. 2014. Recent progress on monotonicity. In *Linguistic Issues in Language Technology*. Citeseer.

Gene Kim, Benjamin Kane, Viet Duong, Muskaan Mendiratta, Graeme McGuire, Sophie Sackstein, Georgiy Platonov, and Lenhart Schubert. 2019. Generating discourse inferences from unscoped episodic logical formulas. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 56–65, Florence, Italy. Association for Computational Linguistics.

Gene Kim and Lenhart Schubert. 2019. A type-coherent, expressive representation as an initial step to language understanding. In *Proceedings of the 13th International Conference on Computational Semantics*, Gothenburg, Sweden. Association for Computational Linguistics.

Gene Louis Kim. 2019. Towards parsing unscoped episodic logical forms with a cache transition parser. In *the Poster Abstracts of the Proceedings of the 32nd International Conference of the Florida Artificial Intelligence Research Society*.

Joachim Lambek. 1988. Categorial and categorical grammars. In *Categorial grammars and natural language structures*, pages 297–317. Springer.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

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).

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

Richard Montague. 1970. Universal grammar. *Theoria*, 36(3):373–398.

Fabrizio Morbini and Lenhart Schubert. 2009. Evaluation of Epilog: A reasoner for Episodic Logic. In *Proceedings of the Ninth International Symposium on Logical Formalizations of Commonsense Reasoning*, Toronto, Canada.

Khalil Mrini, Franck Dernoncourt, Trung Bui, Walter Chang, and Ndapa Nakashole. 2019. Rethinking self-attention: An interpretable self-attentive encoder-decoder parser.

Victor Sánchez Valencia. 1991a. Categorial grammar and natural logic. ILTI Prepublication: Logic, Philosophy and Linguistics (LP) Series.

Victor Sánchez Valencia. 1991b. *Studies on natural logic and categorial grammar*. Ph.D. thesis, University of Amsterdam.

Lenhart Schubert. 2014. From treebank parses to Episodic Logic and commonsense inference. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 55–60, Baltimore, MD. Association for Computational Linguistics.

Lenhart K. Schubert. 2000. The situations we talk about. In Jack Minker, editor, *Logic-based Artificial Intelligence*, pages 407–439. Kluwer Academic Publishers, Norwell, MA, USA.

Lenhart K. Schubert and Chung Hee Hwang. 2000. Episodic Logic meets Little Red Riding Hood: A comprehensive natural representation for language understanding. In Lucja M. Iwańska and Stuart C. Shapiro, editors, *Natural Language Processing and Knowledge Representation*, pages 111–174. MIT Press, Cambridge, MA, USA.

Jonathan Traugott. 1986. Nested resolution. In *8th International Conference on Automated Deduction*, pages 394–402, Berlin, Heidelberg. Springer Berlin Heidelberg.

Johan Van Benthem et al. 1986. *Essays in Logical Semantics*. Springer.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on Penn treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.

Appendix A Sánchez-Valencia's Treament of Natural Logic

This appendix lays out the formal treatment of Sánchez-Valencia's Natural Logic which is described in Section 2. The semantics of Sánchez-Valencia's Natural Logic is rooted in an undirected, typed lambda calculus constructed from derivations of Lambek cum Permutation Calculus (Lambek, 1988). The primitives semantic types are e and t (for entities and truth values) which denote sets, with complex types of the form $\langle a, b\rangle$ where a and b are semantic types in the language.There is one inference rule over these, $\langle \alpha, \beta\rangle$, $\alpha \rightarrow \beta$, where the order of the functor and the argument do not matter.

Definition A.1. Monotonicity is defined over the partial ordering relation $\leq_a$ which is defined as follows, where a is a semantic type and $\mathcal{D}_a$ is the corresponding set:

- If $\alpha, \beta \in \mathcal{D}_e$ then $\alpha \leq_e \beta$ iff $\alpha = \beta$.
- If $\alpha, \beta \in \mathcal{D}_t$ then $\alpha \leq_t \beta$ iff $\alpha = \bot$ or $\beta = \top$.
- If $\alpha, \beta \in \mathcal{D}_{(c,d)}$ then $\alpha \leq_{(c,d)} \beta$ iff for each $\kappa \in \mathcal{D}_c$, $\alpha(\kappa) \leq_d \beta(\kappa)$.

Definition A.2. Monotonicity for functions $f \in \mathcal{D}_{(a,b)}$ is defined over this ordering as follows:

- f is *upward monotone* iff for all $x, y \in \mathcal{D}_a$, $x \leq_a y$ entails $f(x) \leq_b f(y)$.
- f is *downward monotone* iff for all $x, y \in \mathcal{D}_a$, $x \leq_a y$ entails $f(y) \leq_b f(x)$.
- f is *non-monotone* if it is neither upward or downward monotone.

Definition A.3. Monotonicity of an occurrence M in N is defined relative to its semantic interpretation, where I is the interpretation function such that:

- M is *upward monotone* in N iff $I(M) \leq I(M')$ entails $I(N) \leq I(N\{M/M'\})$ for all models and assignments.
- M is *downward monotone* in N iff $I(M') \leq I(M)$ entails $I(N\{M/M'\}) \leq I(N)$ for all models and assignments.[6]

Using this, Sánchez-Valencia proves that positive and negative polarity items from the prior Natural Logic literature corresponds to upward and downward monotone occurrences. From this correspondence the soundness of substituting supersets for subsets in positive polarities and vice versa is realized.

Appendix B Detailed Inference System Correspondence

Sánchez-Valencia reasons using a tableau proof system (Beth, 1955) with nodes of the form

$$a_1, a_2, ..., a_n \bullet b_1, b_2, ..., b_m$$

where $a_i, 1 \leq i \leq n$ and $b_j 1 \leq j \leq m$ are English expressions with corresponding Lambek derivations a_i' and b_j'. The proof starts with the premises on the left side (a_i) and the desired conclusions on the right side (b_j). The proof concludes when all paths of the proof tree are closed. A path is closed when the leaf node of the path has the same statement (including scope markings) on both sides of the node. For those unaware of the notation of tableau systems, this node can be interpreted as the following well-formed formula.

$$(A_1 \wedge A_2 \wedge ... \wedge A_n) \rightarrow (B_1 \vee B_2 \vee ... \vee B_m)$$

A tableau step, e.g.

$$\begin{array}{l} a \bullet b \\ a' \mid \\ \bullet\ b' \end{array}$$

can be interpreted as the formula

$$(A \rightarrow B) \longleftrightarrow (A' \rightarrow B')$$

[6] $N\{M/M'\}$ is shorthand for M' substitutes for M in N.

B.1 Marking in NLog

Sánchez-Valencia's (1991a) monotonicity and scope marking rules are

$$\begin{array}{l} X \bullet A, Y \\ X \mid \\ \bullet\ A^*, Y \end{array} \qquad \begin{array}{l} A, X \bullet Y \\ A^*, X \mid \\ \bullet\ Y \end{array}$$

where A^* is a monotonicity or scope marking of A provided by the fixed Lambek analysis of A. Monotonicity can take values $+$ and $-$ and scoping is marked with $()^{\#}$ where the parentheses circumscribe the words associated with the top-level scope.

B.2 Scope Marking in ULF

Rather than using a Lambek analysis for identifying the operator scopes and as a result the polarities, scoped logical forms (SLFs) are used, which are ULFs with disambiguated scopes. The conversion from ULF to SLF can be denoted mid-proof so that a specific scoping does not need to be committed to at the start of the proof.

B.2.1 Mapping ULFs to SLFs

Scoping for ULFs comes in two flavors:

(S1) **Independent scoped operators.** An independent scoped operator is one that simply raises up to any wff level and introduces some information to only that scope of the overall formula. This includes tense, aspect, and sentence-level adverbials operators. These operators add temporal, locative, or general additional contextual information to the wff.

(|Abelard| ((past see.v) him.pro yesterday.adv-e))

↓ scoping

(past (yesterday.adv-e (|Abelard| (see.v him.pro))))

(S2) **Determiners with restrictors.** When determiners are scoped, they bring with them the restrictor predicate. A variable is introduced which is placed in the position of the lifted constituent and this variable is quantified with the lifted determiner and restricted by the restrictor predicate.

(|Abelard| (see.v (a.d carp.n)))

↓ scoping

(a.d x: (x carp.n) (|Abelard| (see.v x)))

B.2.2 Scope Marking with SLFs

In accordance with Sánchez-Valencia's treatment, we will only perform scope marking on the (S2) classes of scoping operators. Fortunately, this is the more interesting one from a structural perspective. The scoping of ULFs with (S2) classes leads to a more decoupled representation of the constituent, so we must define a correspondence between these components that allows us to still mark the scoping of the ULF based on the fixed scoped realization.

The key to making a correspondence between the (S2)-type constituent in ULF and SLF is the quantified variable of the SLF. For a ULF, ψ which contains a quantified expression $\wp$ of form the $(\delta\ \pi)$ where δ is a determiner and π is a predicate, the corresponding formula with $(\delta\ \pi)$ at the top-level scope is $(\delta\ x\text{: }(x\ \pi)\ \psi[\wp/x])$.

Top-level scope marking process Knowing this correspondence, we have a path to marking the ULF quantified expression constituent that is the top-level scope. Given the SLF which defines the scope ordering, the constituent of the form $(\delta\ \pi)$ in psi at the position of x in $\psi[\wp/x]$ is marked as the top scope of ψ. Note that $\wp$ does is not an alias for the pattern $(\delta\ \pi)$. Rather it refers to a unique constituent of ψ which has the form $(\delta\ \pi)$. This is an important distinction in order to properly handle sentences with multiple constituents of the same form, e.g. *"A dog greets a dog"*. Below is an example to help illustrate the mapping.

"Abelard sees a carp"
SLF (a.d x: (x carp.n) (|Abelard| (see.v x)))
Marked ULF (|Abelard| (see.v (a.d carp.n)$^{\#}$))
$\boldsymbol{\delta}$: a.d, $\boldsymbol{\pi}$: carp.n

In practice, we will not use the actual natural logic marking for inference since we don't use the tableau method for inference. Rather, we use this process to identify the ULF constituent with the top-level scope on the fly using the process which retains the same inferential capacity to the marking in tableau method.

B.3 Polarity marking in ULF

We perform polarity marking in a two stage process that mirrors the process used by (Sánchez Valencia, 1991a). First we classify lexical entries according to their monotonicity properties—in which direction they place entailment contexts on their arguments—and mark them in the SLF with subscripts. For example, the determiner no.d, which has negative downward entailment on both the restrictor and scope is marked as no.d$_{(-,-)}$. Here are a few lexical polarity annotated items. The possible entailment options are $+$ for upward, $-$ for downward, and $\circ$ for flat. Figure 6 provides more examples.

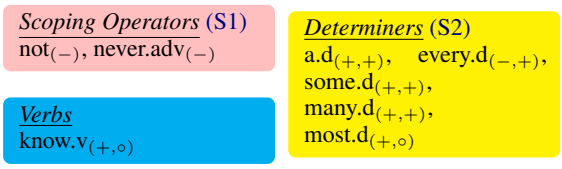

Figure 6: Examples of lexical monotonicity markings.

Unmarked lexical entries are assumed to have upward entailment on all of their arguments. Using the lexical annotations, we mark the local entailment direction of the argument constituents in the SLF using subscripts again. For example, the SLF for "no scientist knows every scientific fact"

(no.d$_{(-,-)}$ x: (x scientist.n)
(every.d$_{(-,+)}$ y: (y (scientific.a fact.n))
(x (know.v y))))

gets its arguments marked as follows.

(no.d$_{(-,-)}$ x: (x scientist.n)$_{-}$
(every.d$_{(-,+)}$ y: (y (scientific.a fact.n))$_{-}$
(x (know.v y))$_{+}$)$_{-}$)

Finally, the global polarity is derived from these local entailment directions and marked with superscripts. The global polarity is derived by traversing the SLF from the root and counting the number of occurrences of negative and flat entailments. The global polarity is computed form this using the following rules, applied in order.

1. If node a has no polarity if any node in the path from the root to a is marked with $\circ$ (flat local entailment).

2. Else, if node a has negative polarity if there are an odd number of nodes between the root and a (including a) marked with $-$ (downward local entailment).

3. Otherwise, node a has positive polarity.

Following these rules the argument marked SLF gets marked with global polarity as (limiting global polarity marking to just nodes with local entailment marking for readability)

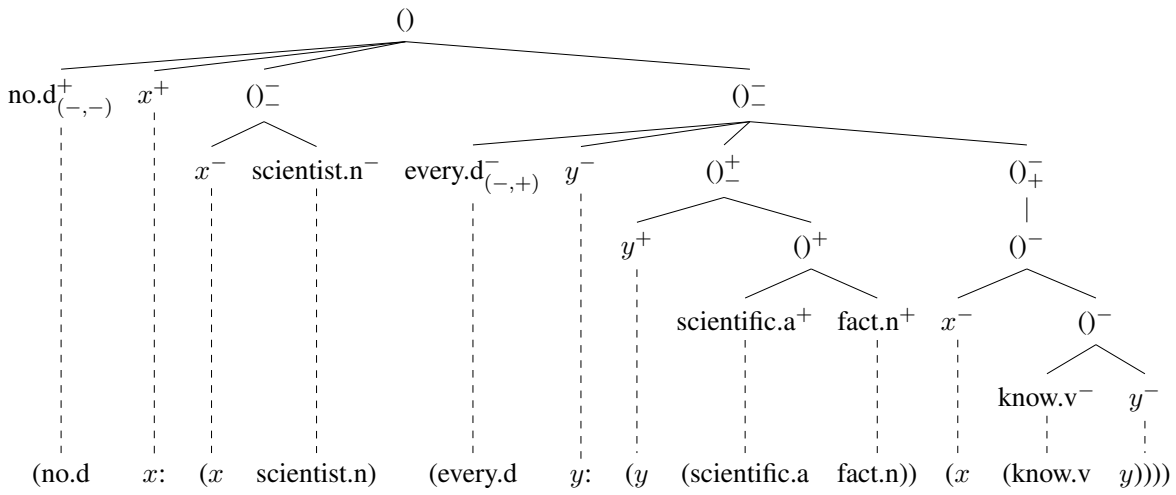

Figure 7: A tree representation of the SLF for "No scientist knows every scientific fact." with all lexical monotonicity, local entailment context, and global polarity markings.

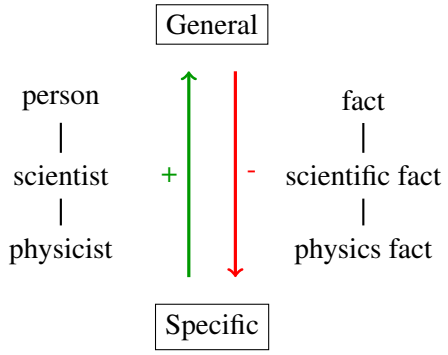

Figure 8: Monotonicity orderings of the predicates that are involved in the polarity inference example. Lower predicates are subsets of predicates above them. Predicates can be replaced with those above them in positive polarity and below them in negative polarity.

(no.d$_{(-,-)}$ x: (x scientist.n)$^{-}_{-}$
(every.d$_{(-,+)}$ y: (y (scientific.a fact.n))$^{+}_{-}$
(x (know.v y))$^{-}_{+}$)$^{-}_{-}$)

The propagation of the polarity is easier to see when the SLF is written in a tree diagram, Figure 7.

Then we can mark the global polarity in the ULF according to the corresponding constituent in the SLF. The following is the resulting ULF for the above SLF, only marking constituents that correspond to predicates in predicative position in the SLF.

((no.d scientist.n^{-})
(know.v^{-} (every.d (scientific.a fact.n)$^{+}$)))

We now show that these markings result in inferences that follow our intuition. From the sentence *"No scientists knows every scientific fact"* we want to infer

(1) a. No physicist knows every scientific fact

b. No scientist knows every fact

but not

(2) a. *No person knows every scientific fact

b. *No scientist knows every physics fact

Figure 8 shows the monotonicity orderings of the predicates that are involved in this example and the direction of warranted inference in each polarity.

In positive polarity, we are warranted inference upward on this diagram, and for negative polarity downward. That is, 'scientist' is in negative polarity which warrants replacement by 'physicist', which is indeed intuitively warranted in English. As is desired from the motivating sentences, "scientist" is in negative polarity and can be replaced with "physicist" (1a) and "scientific fact" is in positive polarity and can be replaced with "fact" (1b).

B.4 Inferences with ULFs

B.4.1 Negation

Sánchez-Valencia's (1991a) negation rule is

$$\begin{array}{l} x, Neg(a) \\ x \end{array} \begin{array}{l} \bullet\, y \\ \bullet\, a, y \end{array}$$

In simple logical terms, this rule is

$$((X \wedge \neg A) \rightarrow Y) \leftrightarrow (X \rightarrow (A \vee Y)) \qquad (1)$$

Negation of ULFs is performed by applying the logical 'not' operator so no special rule is necessary.

For instance, the negation of the ULF (|Abelard| walk.v) is (not (|Abelard| walk.v)).

Some identities that are useful for inferences are listed below, in the form of inference rules. Proofs for these identities can be given using the formal definitions of the respective generalized quantifiers.

$$\frac{\text{(not (not } \phi))}{\phi} \qquad \frac{\phi}{\text{(not (not } \phi))}$$

$$\frac{\text{(not (some.d } \nu\text{: } \psi\ \phi))}{\text{(no.d } \nu\text{: } \psi\ \phi)} \qquad \frac{\text{(no.d } \nu\text{: } \psi\ \phi)}{\text{(not (some.d } \nu\text{: } \psi\ \phi))}$$

$$\frac{\text{(not (a.d } \nu\text{: } \psi\ \phi))}{\text{(no.d } \nu\text{: } \psi\ \phi)} \qquad \frac{\text{(no.d } \nu\text{: } \psi\ \phi)}{\text{(not (a.d } \nu\text{: } \psi\ \phi))}$$

B.4.2 Handling 'be.v'

While 'be.v' are included in ULFs for simplifying the interface to natural language since the copula can act as an anchor for modifications from adverbial phrases and temporal information from its conjugation, we can consider it to be semantically void with respect to its arguments.

SLF Rule 1 (be.v Elimination).

$$\frac{\text{(a.d } y\text{: } (y\ P)\ (x\ \text{(be.v (= } y\text{))))}}{(x\ P)}$$

where P is an arbitrary unary predicate and x is an arbitrary term.

Proof

1. (a.d y: $(y\ P)$ $(x$ (be.v (= y))))	Assumption
2. I((a.d y: $(y\ P)$ $(x$ (be.v (= y)))))	Interp. fn.
3. There exists $d \in \mathcal{D}$ s.t. $d \in I(P)$ and $I((x\ \text{(be.v (= } y\text{))))}^{\mathcal{U}_{y:d}}$	Satisfaction conds of $\exists$
4. There exists $d \in \mathcal{D}$ s.t. $d \in I(P)$ and $I((x\ \text{(= } y\text{)))}^{\mathcal{U}_{y:d}}$	Def of be.v
5. There exists d s.t. $d \in I(P)$ and $(I(x) = d)$	I(=)
6. $I(x) \in I(P)$	Variable substitution
7. $(x\ P)$	Interp. fn. of predication

With this inference rule we can easily derive a predicate subset defining inference rule which is necessary for polarity inferences.

$$\frac{\text{(every.d } x\text{: } (x\ P1)\ \text{(a.d } y\text{: } (y\ P2)\ (x\ \text{(be.v (= } y\text{))))))}}{\text{(every.d } x\text{: } (x\ P1)\ (x\ P2))}$$

where $P1$ and $P2$ are arbitrary unary predicates.

For example, from the initial SLF for "every carp is a fish" we get a nice relationship between the predicates 'carp.n' and 'fish.n'.

(every.d x: (x carp.n) (a.d y: (y fish.n) (x (be.v (= y)))))
$$\Downarrow$$
(every.d x: (x carp.n) (x fish.n))

B.4.3 Monotonicity Inference

The basic monotonicity inference rule in Natural Logic takes a subset relationship between two predicates, $P1 \subseteq P2$, and a formula, f, where something of type $P1$ occurs in positive polarity. Then we can assert f' which is the same as f except that $P2$ is substituted for $P1$. We can state a direct analog of this rule using SLFs.

We also formulate a similar rule which takes the subset relationship $P1 \subseteq P2$ and a formula g, where something of type $P2$ appears in negative polarity. In this case we can assert g' which is the same as g except that $P1$ is substituted for $P2$. Using SLFs the inferences looks as follows.

SLF Rule 2 (Monotonicity Inference, SMI).

$$\frac{(\delta\ x\text{: } (x\ P1)^{+}\ \phi(x)),\ \text{(every.d } y\text{: } (y\ P1)\ (y\ P2))}{(\delta\ x\text{: } (x\ P2)\ \phi(x))}$$

$$\frac{(\delta\ x\text{: } (x\ P2)^{-}\ \phi(x)),\ \text{(every.d } y\text{: } (y\ P1)\ (y\ P2))}{(\delta\ x\text{: } (x\ P1)\ \phi(x))}$$

where δ is a determiner.

The SLFs are necessary for keeping track of the outer scope and determining the polarities, but the core inference can be written using ULFs, closer to surface form, with the SLFs acting as auxiliary information to ensure consistency of the formulas. Using ULFs and chaining SMI and be.v elimination we get the following inferences.

ULF Rule 1 (Monotonicity Inference, UMI).

$$\frac{\phi[(\delta\ P1)^{+}],\ \text{((every.d } P1)\ \text{(be.v (= (a.d } P2))))}{\phi[(\delta\ P2)]}$$

$$\frac{\phi[(\delta\ P2)^{-}],\ \text{((every.d } P1)\ \text{(be.v (= (a.d } P2))))}{\phi[(\delta\ P1)]}$$

where δ is a determiner.

It is worth noting that if the restrictor of a determiner is a conjunction of predicates restricting the variable, then due to the upward entailing nature of the $\wedge$ operator we can propagate the polarity induced by the determiner on its restrictor to each term in the conjunction. Also, we know that the $\wedge$ operator preserves subset relations, that is if x satisfies predicates $P1$ and $P2$ and if every element of $P1$ is also in Q, then x must satisfy the predicates Q and $P2$. Therefore, in a case where a variable is restricted by a conjunction of predicates,

it is possible to use the monotonicity inference rule on individual predicates in the restrictor. This is particularly useful when dealing with extensionally modified predicates (see B.4.4).

Example 1. Now we will use the presented ULF/SLF marking and inference rules to perform an inference over generalized quantifiers that Sánchez Valencia (1991a) demonstrated: from "Abelard sees a carp" and "Every carp is a fish" we will conclude "Abelard sees a fish". Before we start the inference, we walk through the scoping and polarity derivation of assumption (1), which will be used in the inference. This derivation takes the place of the Lambek derivations used by Sánchez-Valencia to get polarity and scoping information into the proof.

Scoping and Polarity Derivation

1.	(|Abelard| (see.v (a.d carp.n)))	ULF
2.	(a.d x: (x carp.n) (|Abelard| (see.v x)))	Only possible scoping
3.	(a.d$_{(+,+)}$ x: (x carp.n) (|Abelard| (see.v x)))	a.d lexical monotonicity
4.	(a.d x: (x carp.n)$_+$ (|Abelard| (see.v x))$_+$)	Local entail. context
5.	(a.d x: (x carp.n)$^+$ (|Abelard| (see.v x)$^+$)$^+$)	Global polarity

Now for the actual proof.

Proof

1.	(|Abelard| (see.v (a.d carp.n)))	Assumption
2.	((every.d carp.n) (be.v (= (a.d fish.n))))	Assumption
3.	(|Abelard| (see.v (a.d carp.n)$^+$))	Polarity marking, 1.
4.	(|Abelard| (see.v (a.d fish.n)))	UMI, 2.,3.

B.4.4 Inferences with Predicate Modifiers

Let P' be an extensional modification of a predicate P, and let P_m be the modifying predicate. Since the modification is extensional, we know that an entity x satisfies P' if and only if it satisfies both P and P_m. Hence we get the following rule.

SLF Rule 3.

$$\frac{(\delta\ x\text{: } (x\ P')\ \phi(x))}{(\delta\ x\text{: } ((x\ P) \wedge (x\ P_m))\ \phi(x))}$$

where P' is an extensional modification of the predicate P with modifying predicate P_m, and δ is a determiner.

This rule can then be combined with monotonicity inference to get

ULF Rule 2.

$$\frac{\phi[(\delta\ (M\ P1))^+],\ ((\text{every.d}\ P1)\ (\text{be.v}\ (=\ (\text{a.d}\ P2))))}{\phi[(\delta\ (M\ P2))]}$$

where M is an extensional modifier and δ is a determiner. A similar rule for negative polarities can be written as well.

This allows us to make another inference demonstrated by Sánchez Valencia (1991a):

Example 2. From "Abelard sees a male carp" and "Every carp is a fish", we will conclude "Abelard sees a male fish".

Scoping and Polarity Derivation

1.	(|Abelard| (see.v (a.d (male.a carp.n))))	ULF
2.	(a.d x: (x (male.a carp.n)) (|Abelard| (see.v x)))	Only possible scoping
3.	(a.d x: ((x male.a) $\wedge$ (x carp.n)) (|Abelard| (see.v x)))	Assume intersective modification
4.	(a.d$_{(+,+)}$ x: ((x male.a) $\wedge$ (x carp.n)) (|Abelard| (see.v x)))	a.d lexical monotonicity
5.	(a.d x: ((x male.a) $\wedge$ (x carp.n))$_+$ (|Abelard| (see.v x))$_+$)	Local entail. context
6.	(a.d x: ((x male.a)$_+$ $\wedge$ (x carp.n)$_+$) (|Abelard| (see.v x))$_+$)	Upward entail. of $\wedge$
7.	(a.d x: ((x male.a)$^+$ $\wedge$ (x carp.n)$^+$) (|Abelard| (see.v x)$^+$)$^+$)	Global polarity

Proof

1.	(|Abelard| (see.v (a.d (male.a carp.n))))	Assumption
2.	((every.d carp.n) (be.v (= (a.d fish.n))))	Assumption
3.	(|Abelard| (see.v (a.d (male.a carp.n$^+$))))	Polarity marking, 1.
4.	(|Abelard| (see.v (a.d (male.a fish.n))))	UMI, 2.,3.

The need for addressing the intersective nature of the modification in *male fish* brings up a benefit of using ULFs as a basis for the inferences. Since ULF is explicitly underspecified, the assumptions made during the inference process must be stated. The corresponding proof presented by Sánchez-Valencia hides the assumption of intersective modification in the lexical monotonicity marking of *male* (as $((e,t)^+,(e,t))$). In ULF, modifications are assumed to be intensional unless otherwise assumed, so the intersective nature of the modifier male.n must be explicitly stated.

B.4.5 Conversion Rules

Sánchez-Valencia's (1991a) conversion rule is

$$\begin{array}{l}(some\ y)^{\#}\ is\ a\ x, X \\ (some\ x)^{\#}\ is\ a\ y, X\end{array} \Big\} Y$$

Before stating the corresponding rule for ULFs, we note that the rule also works for the determiners a.d and no.d. Thus we state the ULF conversion rules as follows:

SLF Rule 4 (Conversion).

$$(d\ x\text{: }(x\ P)\ (x\ Q)) \leftrightarrow (d\ y\text{: }(y\ Q)\ (y\ P))$$

where $d \in \{$some.d, a.d, no.d$\}$.

Correctness of this rule can be argued using the definitions of the generalized quantifiers 'some', 'a', and 'no' and subset relations under interpretation.

Using ULFs and chaining this rule with the be.v inference, we get the following rule.

ULF Rule 3 (Conversion).

$$\begin{aligned}&((d1\ P)\ \text{(be.v (= }(d2\ Q))))\\ &\qquad\leftrightarrow ((d1\ Q)\ \text{(be.v (= }(d2\ P))))\end{aligned}$$

where $d1 \in \{$some.d, a.d, no.d$\}$ and $d2 \in \{$some.d, a.d$\}$.

B.5 Boolean Connectives

Sánchez-Valencia (1991a) handles generalized boolean connectives by allowing connectives (*and*, *or*) to have the type $(a,(a,a))$, where a is any complex category ending in t. Then the inference rules appropriately substitute one of the connective constituents for the entire phrase. The rules, which we will not list here, have a few versions depending on the position of the connective due to the left side of the tableau nodes being interpreted as connected with conjunctions and the right side with disjunctions.

ULF handles this similarly, without the tableau-specific details by allowing connectives to be interpreted as $\langle A, \langle A, A\rangle\rangle$ for an arbitrary type A. They are interpreted as generalized lambda expressions.

Definition B.1 (ULF Generalized Connective)**.**

$$\begin{aligned}&(A\ \chi\ B) \leftrightarrow\\ &(\lambda\ x_1, ..., x_n\text{: }((A\ x_1 ... x_n)\ \chi\ (B\ x_1 ... x_n)))\end{aligned}$$

where $\chi \in \{$and.cc, or.cc$\}$ and both A and B are prefix operators with arity n. Infix operators are defined in the equivalent way while respecting the predicate position relative to the arguments.

This along with the observation that the following formulas hold true through the intersective and unionistic nature of conjunction and disjunction, respectively, allow us to use simple monotonicity rules in the context of boolean connectives.

$$(\lambda\ x_1, ..., x_n\text{: }((A\ x_1 ... x_n)\ \text{and.cc}\ (B\ x_1 ... x_n))) \subseteq A, B$$

$$A, B \subseteq (\lambda\ x_1, ..., x_n\text{: }((A\ x_1 ... x_n)\ \text{or.cc}\ (B\ x_1 ... x_n)))$$

Appendix C Polarity-based EL Inference

EL supports two forward inference rules and two goal-based inference rules that operate on substitutions under appropriate polarity contexts (Schubert and Hwang, 2000). Here we present a couple of examples and connect it to the inference rules in ULF. First, the forward inference rules, called Rule Instantiation (RI):

$$\text{RI-1}\ \frac{MAJ(\phi^{-}), MIN(\phi'^{+})}{MAJ_{\sigma}(\neg MIN_{\sigma}(\bot^{+})^{-})} \qquad \text{RI-2}\ \frac{MAJ(\phi^{-}), MIN(\phi'^{+})}{MIN_{\sigma}(MAJ_{\sigma}(\top^{-})^{+})}$$

where RI-1 is sound if the only variables in the matching expression (ϕ') of the minor premise (MIN) are "matchably bound,"—bound within ϕ' or by a universal quantifier in positive polarity context or existential quantifier in negative polarity context—and RI-2 is sound if the only variables in the matching expression (ϕ) of the major premise (MAJ) are "matchably bound."

It turns out that the monotonicity rule presented by Sánchez Valencia (1991a) is a special case of RI-2. Here is an example to demonstrate a monotonicity inference over SLFs, which for this inference is sufficiently disambiguated.

Proof

1.	*Every carp is a fish*	Assumption
2.	*Abelard sees a carp*	Assumption
3.	(every.d x: (x carp.n) (a.d y: (y fish.n) (be.v (= y))))	SLF for 1.
4.	(a.d x: (y carp.n) (|Abelard| (see.v y)))	SLF for 2.
5.	(every.d x: (x carp.n) (x fish.n))	be.v Elim, 3.
6.	(every.d x: (x carp.n)$^{-}$ (x fish.n)$^{+}$)	Polarity marking, 5.
7.	(a.d y: (y carp.n)$^{+}$ (|Abelard| (see.v y))$^{+}$)	Polarity marking, 4.
8.	(a.d y: (y fish.n)$^{+}$ (|Abelard| (see.v y))$^{+}$)	RI-2, 6.,7. (see C)
9.	*Abelard sees a fish*	English for 8.

Step-by-step RI-2 application

1.	(every.d x: (x carp.n)$^{-}$ (x fish.n)$^{+}$)	MAJ

2. (a.d y: (y carp.n)$^+$ (|Abelardl (see.v y))$^+$) — MIN

3. $\top \rightarrow$ (y fish.n)$^+$ — Converted MAJ, $\{x/y\}$, 1.

4. $\bot \vee$ (y fish.n)$^+$ — $\rightarrow$ Def, 3.

5. (y fish.n)$^+$ — $\bot$ Elim, 4.

6. (a.d y: (y fish.n)$^+$ (|Abelardl (see.v y))$^+$) — Subst. of converted MAJ, 2.,5.

Notice that this proof holds for an arbitrary predicates in place of *carp* and *fish* and an arbitrary sentence where *carp* occurs in positive polarity context in place of *Abelard sees a carp*. Thus, RI-2 is a generalization of Sánchez's monotonicity rule.

$$\begin{array}{l} (every\ x)^{\#}\ is\ a\ y, F(x^+), X \\ (every\ x)^{\#}\ is\ a\ y, F(y), X \end{array} \Big\} \begin{array}{l} Y \\ Y \end{array}$$

Note that RI-2 can also handle inferences where the major premise is a more complex construction than "every p is a q". In episodic logic, RI-2 can be used to conclude *Something is a cap or pretty if Little Red Riding Hood wears it* from *Every dress or hood that Little Red Riding Hood wears is pretty* and *Something is a cap or a hood* (Schubert and Hwang, 2000).

Additionally, RI-1 is a generalization of the reverse inference: substituting in more specific predicates when in negative polarity.

Step-by-step RI-1 application

1. (every.d x: (x carp.n)$^-$ (x fish.n)$^+$) — MIN

2. (no.d y: (y fish.n)$^-$ (|Abelardl (see.v y))$^-$) — MAJ

3. (no.d x: (x fish.n)$^-$ (|Abelardl (see.v x))$^-$) — Converted MAJ, $\{y/x\}$, 2.

4. $\neg$((x carp.n)$^- \rightarrow \bot^+$) — Converted MIN, 1.

5. $\neg(\neg$(x carp.n)$^- \vee \bot^+$) — $\rightarrow$ Def, 4.

6. $\neg\neg$(x carp.n)$^- \wedge \top^+$ — de Morgan, 5.

7. (x carp.n)$^- \wedge \top^+$ — $\neg$ Elim, 6.

8. (x carp.n)$^-$ — $\top$ Annih, 7.

9. (no.d x: (x carp.n)$^-$ (|Abelardl (see.v x))$^-$) — Subst. of converted MIN, 3.,8.

We've already shown that RI subsumes the specialized monotonicity inference presented by Sánchez Valencia (1991a). Now, we will show that in first order contexts RI also subsumes a more general presentation of natural logic inference by Sánchez Valencia (1991b). The upward monotonicity inference in positive contexts is written by Sánchez Valencia as

$$\frac{[\![M]\!] \leq [\![M']\!] \quad [\![N(M)]\!]}{[\![N(M')]\!]}$$

Where $[\![]\!]$ is the denotation function and $\leq$ is the monotonicity ordering from Definition A.1. We will refer to this rule as SVMI. First, we show that the RI-2 inference in first order contexts can be interpreted in this form.

Since RI-2 substitutes $MAJ_\sigma(\top)$ for ϕ'^+ in $MIN(\phi'^+)$, if we can show that $[\![\phi']\!] \leq [\![MAJ_\sigma(\top)]\!]$, then RI-2 can be justified through SVMI. As $MAJ(\phi)$ is assumed in RI-2, if $\phi = \top$, then for all models satisfying the assumptions, $MAJ(\top) = \top$.[7] Basically, $MAJ(\phi), (\phi = \top) \rightarrow MAJ(\top)$. This by definition of $\leq$ (A.1) satisfies $[\![\phi]\!] \leq [\![MAJ(\top)]\!]$: in any case where ϕ is true, so is $MAJ(\top)$. Since ϕ and ϕ' are matchably bound, their differences are irrelevant in the above justification and can be substituted for each other. Thus, for any application of RI-2 $[\![\phi']\!] \leq [\![MAJ_\sigma(\top)]\!]$ holds, and therefore the inference can be justified through SVMI.

Now, we show that for any SVMI inference where M and M' are wffs, it can be written in the form of RI-2. $[\![M]\!] \leq [\![M']\!]$ can be restated as $(\forall_x M \rightarrow M')$, which we identify as $MAJ(\phi^-)$, where M is phi and we know that M is in a negative polarity context due to $\forall$. $N(M)$ is identified as $MIN(\phi'^+)$ where M is ϕ' and we know is in a positive polarity context by assumption. $MAJ(\top) = \top \rightarrow M' = M'$ so $MIN(MAJ(\top)) = N(M')$.

We conjecture that this generalizes to all M and M' that are not wffs. The monadic predicate case seems simple enough through a connection with modus ponens, but proofs for cases such as determiners and variables are more elusive.

Appendix D Traditional Aristotelian Syllogisms in ULF

In this appendix, we show that similar to Sánchez-Valencia's (1991a) Natural Logic, the inference system described for ULFs can explain traditional syllogistic inference. We will give proofs for the

[7] This step requires the first-order context. In intensional contexts, the substitution via equality is not justified.

syllogisms of the first figure using ULFs. Since all other syllogisms can be derived from these, this is sufficient to show that all traditional syllogisms can be derived.

D.1 Scoping and Polarity Derivations

Before proving the syllogisms we go through the scoping and polarity derivations of the propositions used in the syllogisms.

Proposition i. "Every X is a Y"

1. ((every.d X) (be.v (= (a.d Y)))) — ULF
2. (every.d x: (x X) (x (be.v (= (a.d Y))))) — Scope every.d
3. (every.d x: (x X)
(a.d y: (y Y) (x (be.v (= y))))) — Scope a.d
4. (every.d$_{(-,+)}$ x: (x X)
(a.d$_{(+,+)}$ y: (y Y)
(x (be.v (= y))))) — Lexical monotonicity
5. (every.d x: (x X)$_-$
(a.d y: (y Y)$_+$
(x (be.v (= y)))$_+$)$_+$) — Local entail. context
6. (every.d x: (x X)$^-$
(a.d y: (y Y)$^+$
(x (be.v (= y)))$^+$)$^+$) — Global polarity

Proposition ii. "Some X is a Y"

1. ((some.d X) (be.v (= (a.d Y)))) — ULF
2. (some.d x: (x X)
(a.d y: (y Y) (x (be.v (= y))))) — Scoping
3. (some.d$_{(+,+)}$ x: (x X)
(a.d$_{(+,+)}$ y: (y Y)
(x (be.v (= y))))) — Lexical monotonicity
4. (some.d x: (x X)$_+$
(a.d y: (y Y)$_+$
(x (be.v (= y)))$_+$)$_+$) — Local entail. context
5. (some.d x: (x X)$^+$
(a.d y: (y Y)$^+$
(x (be.v (= y)))$^+$)$^+$) — Global polarity

Proposition iii. "No X is a Y"

1. ((no.d X) (be.v (= (a.d Y)))) — ULF
2. (no.d x: (x X)
(a.d y: (y Y) (x (be.v (= y))))) — Scoping
3. (no.d$_{(-,-)}$ x: (x X)
(a.d$_{(+,+)}$ y: (y Y)
(x (be.v (= y))))) — Lexical monotonicity
4. (no.d x: (x X)$_-$
(a.d y: (y Y)$_+$
(x (be.v (= y)))$_+$)$_-$) — Local entail. context
5. (no.d x: (x X)$^-$
(a.d y: (y Y)$^-$
(x (be.v (= y)))$^-$)$^-$) — Global polarity

Proposition iv. "Not every X is a Y"

1. (not (every.d X) (be.v (= (a.d Y)))) — ULF
2. (not (every.d x: (x X)
(a.d y: (y Y)
(x (be.v (= y)))))) — Scoping
3. (not$_{(-)}$ (every.d$_{(-,+)}$ x: (x X)
(a.d$_{(+,+)}$ y: (y Y)
(x (be.v (= y)))))) — Lexical monotonicity
4. (not (every.d x: (x X)$_-$
(a.d y: (y Y)$_+$
(x (be.v (= y)))$_+$)$_+$)$_-$) — Local entail. context
5. (not (every.d x: (x X)$^+$
(a.d y: (y Y)$^-$
(x (be.v (= y)))$^-$)$^-$)$^-$) — Global polarity

D.2 Deriving the Syllogisms

Syllogism 1 (Barbara). "Every M is a P" and "Every S is a M" entail "Every S is a P".

Proof

1. ((every.d M) (be.v (= (a.d P)))) — Assumption
2. ((every.d S) (be.v (= (a.d M)))) — Assumption
3. ((every.d M)$^-$ (be.v (= (a.d P)))) — Polarity marking, 1.
4. ((every.d S) (be.v (= (a.d P)))) — UMI, 2.,3.

Syllogism 2 (Darii). "Every M is a P" and "Some S is a M" entail "Some S is a P".

Proof

1. ((every.d M) (be.v (= (a.d P)))) — Assumption
2. ((some.d S) (be.v (= (a.d M)))) — Assumption
3. ((some.d S) (be.v (= (a.d M)$^+$))) — Polarity marking, 2.
4. ((some.d S) (be.v (= (a.d P)))) — UMI, 1.,3.

Syllogism 3 (Celarent). "No M is a P" and "Every S is a M" entail "No S is a P".

Proof

1. ((no.d M) (be.v (= (a.d P)))) — Assumption
2. ((every.d S) (be.v (= (a.d M)))) — Assumption
3. ((no.d M)$^-$ (be.v (= (a.d P)))) — Polarity marking, 1.
4. ((no.d S) (be.v (= (a.d P)))) — UMI, 2.,3.

Syllogism 4 (Ferio). "No M is a P" and "Some S is a M" entail "Not every S is a P".

Using the logical interpretation of Sánchez-Valencia's Negation Rule twice (1), we see that the

syllogism is true iff "Every S is a P" and "Some S is a M" entail "Some M is a P". We prove this as follows.

Proof

1.	((every.d S) (be.v (= (a.d P))))	Assumption
2.	((some S) (be.v (= (a.d M))))	Assumption
3.	((some S)$^+$ (be.v (= (a.d M))))	Polarity marking, 2.
4.	((some P) (be.v (= (a.d M))))	UMI, 1.,3.
5.	((some M) (be.v (= (a.d P))))	Conversion, 4.

This could alternatively be proved by contradiction without the use of the equivalent of the Negation Rule, where "Not every S is a P" becomes negated to "every S is a P", after which, applying UMI and Conversion leads to a contradiction.

Supporting Context Monotonicity Abstractions in Neural NLI Models

Julia Rozanova [†], **Deborah Ferreira**[†], **Mokanarangan Thayaparan**[†], **Marco Valentino**[†], **André Freitas**[†‡]
Department of Computer Science, University of Manchester, United Kingdom[†]
Idiap Research Institute, Switzerland[‡]
{firstname.surname} @manchester.ac.uk

Abstract

Natural language **contexts** display logical regularities with respect to substitutions of related concepts: these are captured in a functional order-theoretic property called *monotonicity*. For a certain class of NLI problems where the resulting entailment label depends only on the context monotonicity and the relation between the substituted concepts, we build on previous techniques that aim to improve the performance of NLI models for these problems, as consistent performance across both upward and downward monotone contexts still seems difficult to attain even for state of the art models. To this end, we reframe the problem of **context monotonicity classification** to make it compatible with transformer-based pre-trained NLI models and add this task to the training pipeline. Furthermore, we introduce a sound and complete simplified monotonicity logic formalism which describes our treatment of contexts as abstract units. Using the notions in our formalism, we adapt targeted challenge sets to investigate whether an intermediate context monotonicity classification task can aid NLI models' performance on examples exhibiting monotonicity reasoning.

1 Introduction

NLI has seen much success in terms of performance on large benchmark datasets, but there are still expected systematic reasoning patterns that we fail to observe in the state of the art NLI models. We focus in particular on *monotonicity reasoning*: a large class of NLI problems that can be described as a form of *substitutional* reasoning which displays logical regularities with respect to substitution of related concepts. In this setting, a subphrase $\mathbf{a}$ of a premise $p(\mathbf{a})$ is replaced with a phrase $\mathbf{b}$, yielding a hypothesis $p(\mathbf{b})$.

Usually, the resulting entailment label relies on exactly two properties: the inclusion relation between concepts $\mathbf{a}$ and $\mathbf{b}$, and the *systematic behaviour* of the context p with respect to such relations.

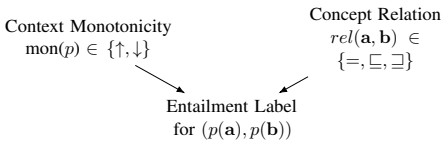

Figure 1: The class of entailment problems under consideration: premise-hypothesis pairs $(p(\mathbf{a}), p(\mathbf{b}))$ whose entailment label depends only on the monotonicity of the context p and the relation between $\mathbf{a}$ and $\mathbf{b}$.

In formal semantics, this is referred to as the *monotonicity* of p (where p is either upward or downward monotone), and this reasoning pattern is referred to as *monotonicity reasoning*. Monotonicity reasoning is incredibly systematic, and thus is a much-probed behaviour in enquiries into the *systematicity* and *generalization capability* of neural NLI models (Goodwin et al., 2020; Yanaka et al., 2020, 2019; Richardson et al., 2020; Geiger et al., 2020).

Determining both the concept relation and the context monotonicity requires significant linguistic understanding of syntactic structure and scope of operators, but in terms of reasoning, this is a very systematic pattern that nevertheless has a history of causing problems for neural models. It has been observed (Yanaka et al., 2019; Geiger et al., 2020) that current state of the art transformer-based NLI models tend to routinely fail in downward monotone contexts, such as those arising in the presence of negation or generalized quantifiers. Recent strategies (Richardson et al., 2020) to address the shortcomings of NLI models in downward-monotone contexts have followed the *inoculation* method (Liu et al., 2019a): additional training data which exhibits the target phenomenon (in this case, downward-monotone reasoning) is used to fine-

Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA), pages 41–50

tune existing models. This is done with some success in (Yanaka et al., 2019; Richardson et al., 2020) and (Geiger et al., 2020). In contrast, we wish to investigate a *transfer learning* strategy that directly targets the monotonicity question as an *additional training task* to see if this can *further* improve the monotonicity reasoning performance of popular transformer-based NLI models.

Our contributions are as follows:

- Emphasizing monotonicity as a property of a *context*, we introduce a sound and complete logical formalism which models the monotonicity reasoning phenomenon but abstracts away from specific linguistic operators, treating the context as a single abstract object.
- Extending our treatment of contexts as individual objects to an experimental setting, we introduce an improvement in neural NLI model performance on monotonicity reasoning challenge datasets by employing a context monotonicity classification task in the training pipieline of NLI models. To the best of our knowledge, this is the first use of neural models for this specific task.
- For this purpose, we adapt the **HELP** dataset (Yanaka et al., 2019) into a **HELP-Contexts** dataset, mimicking the structure of our logical formalism.
- For the class of NLI problems described as *monotonicity reasoning*, we demonstrate the impact of the proposed transfer strategy: we show that there can be a strong improvement on downward monotone contexts, previously known to be a bottleneck for neural NLI models. As such, this shows the benefit of directly targeting intermediate abstractions (in this case, monotonicity) present in logical formalisms that underly the true label.

2 Contexts and Monotonicity

2.1 Contexts

Informally, we treat a natural language *context* as a sentence with a "gap", represented by a variable symbol.

A context $p(x)$:
I ate some x for breakfast.
A sentence $S = p(\text{'fruit'})$:
I ate some fruit for breakfast.

Although every sentence can be viewed as a context with an insertion in as many ways as there are n-grams in the sentence, in this work we shall consider in particular those contexts where the variable corresponds to a slot in the expression that may be filled by an *entity* reference, such as a noun or noun phrase. In the view of Montague-style formal semantics, these contexts correspond to expressions of type $< e, t >$.

2.2 Monotonicity

Given a natural language context p and a pair of nouns/noun phrases $(\mathbf{a}, \mathbf{b})$, we may create a natural language sentence pair $(p(\mathbf{a}), p(\mathbf{b}))$ by substituting the respective subphrases into the natural language context. When treated as a premise-hypothesis pair (as in the experimental NLI task setting), the gold entailment label has a strong relationship with the kinds of *relations* that exist between the insertions $\mathbf{a}$ and $\mathbf{b}$.

In the seminal works on monotonicity (Valencia, 1991; van Benthem, 1988), the relations that are studied are *semantic containment* relations, which are defined analogously to set-theoretic containment relations ($\subseteq$).

	$\mathbf{a}$	$\mathbf{b}$
$\equiv$	couch	sofa
$\sqsubset$	apples	fruit
	South African soccer players	soccer players
	dogs with hats	dogs

Table 1: Examples of the semantic containment relation between concept pairs.

For insertions related by $\sqsubset$, the gold entailment label depends on one other property: the combined *monotonicity profile* of all the linguistic operators within whose scope the insertion is located. If the final monotonicity marking in the insertion's position is "upward", the gold label is entailment. However, if it is "downward", we can deduce entailment of the reversed sentence pair, $(p(\mathbf{b}), p(\mathbf{a}))$. Linguistic operators such as "not" are downward monotone, while generalized quantifiers such as "every" have a more complex monotonicity profile: downward-monotone in the first argument and upward-monotone in the second argument. The monotonicity properties of all the operators compose along the syntax tree, culminating in a final monotonicity marking for the "x" position in the

context (the monotonicity is independent of the inserted word). It is this final monotonicity label that determines the entailment patterns with respect to insertion relations. Although there are formalisms that model this logical behaviour (Icard et al., 2017), they aim to model the behaviour of each linguistic linguistic operator and the way they compose given the parse tree of a sentence.

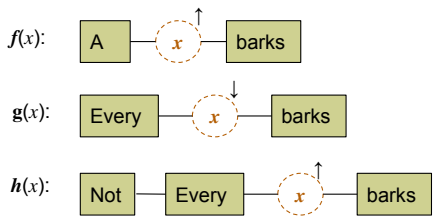

Figure 2: Natural language *contexts* have a property which dictates logical regularities with respect to concept hierarchies: like numerical functions, they can be upward monotone or downward monotone.

We consider a simplification of this behaviour by abstracting away the linguistic specifics of the context, treating it as a single abstract object. As such, we are not concerned with the exact monotonicity profiles of all the linguistic operators that culminate in the monotonicity of the final context. We describe this behaviour with a simple logic system extending the $\mathcal{L}(all)$ logic of (Moss, 2008a) with the abstracted behaviour of upward and donward monotone contexts. We include a proof that this adaptation is still sound and complete.

2.3 A Context-Abstracted Monotonicity Logic

We extend the syllogistic syntax of the language $\mathcal{L}(all)$ included in (Moss, 2008b) and (Moss, 2008a). In keeping with that style, we present the syntax as natural language sentences. However, we include the corresponding first order formulae as well. In the subsequent proofs, we mix the stylizations somewhat for readability, but the table below should serve as a reminder for the exact correspondence.

Definition 2.1. Let the language $\mathcal{L}$ consist of the following:

1. A countable set $\mathbf{A}$ of constant symbols $\mathbf{a}, \mathbf{b}, \mathbf{a}_1, \mathbf{b}_1, \ldots$
2. Exactly two variables, x and y
3. A binary relation symbol $\sqsubseteq$.
4. A set $\mathbf{P}$ of relation symbols $\sqsubseteq_p$ indexed by a countable set $p, p_1, \ldots$.

Only the following are considered sentences in the language $\mathcal{L}$:

Natural Language Stylization	FOL Stylization
all $\mathbf{a}$ are $\mathbf{b}$	$\mathbf{a} \sqsubseteq \mathbf{b}$
if $p(\mathbf{a})$ then $p(\mathbf{b})^*$	$\mathbf{a} \sqsubseteq_p \mathbf{b}^*$
p is upward monotone	$\forall_{x,y}(x \sqsubseteq y \leftrightarrow x \sqsubseteq_p y)$
p is downward monotone	$\forall_{x,y}(x \sqsubseteq y \leftrightarrow y \sqsubseteq_p x)$

Table 2: * For every natural language context p in a set P of contexts, and where $p(\mathbf{a})$ is the substitution of $\mathbf{a}$ into the natural language context p.

This can in many ways be seen as a simplification of previous formalisms (Icard et al., 2017; Hu and Moss, 2018) based on either extending the syllogistic logic $\mathcal{L}(all)$ (Moss, 2008a) or extending typed lambda calculus with monotonicity behaviour. The key difference of this approach is the abstraction away from specific linguistic operators and their monotonicity profiles. On one hand, we are thus only modeling one level of linguistic compositionality, but since the monotonicity profile of every linguistic operator composes into one monotonicity marker which affects the final entailment label (for this class of problem), it encompasses all of the linguistically-specific approaches. This is useful when the monotonicity of a context can be determined by an external system such as a neural classifier or the ccg-2-mono system (Hu and Moss, 2018) . In this case, the monotonicity marking of the entire context is explicit.

2.4 Semantics

Definition 2.2. A model $\mathcal{M}$ of the language $\mathcal{L}$ is a structure

$$\mathcal{M} = (M, [\![\cdot]\!])$$

consisting of a set M and an interpretation function $[\![\cdot]\!]$ where $[\![\mathbf{a}]\!] \subseteq M$, $[\![\sqsubseteq]\!]$ is the $\subseteq$ relation on the powerset $\mathcal{P}(M)$ and $[\![\sqsubseteq_p]\!] \subseteq \mathcal{P}(M) \times \mathcal{P}(M)$ is any binary relation on $\mathcal{P}(M)$. Truth of a formula with respect to a given model is defined as follows:

2.5 Proof Calculus

Our language will be equipped with the following deduction rules and axioms:

$$\begin{array}{lcc} \mathcal{M} \models \mathbf{a} \sqsubseteq \mathbf{b} & \iff & [\![\mathbf{a}]\!] \subseteq [\![\mathbf{b}]\!] \\ \mathcal{M} \models \mathbf{a} \sqsubseteq_p \mathbf{b} & \iff & [\![\mathbf{a}]\!], [\![\mathbf{b}]\!] \in [\![\sqsubseteq_p]\!] \\ \mathcal{M} \models \forall_{x,y}(x \sqsubseteq y \leftrightarrow x \sqsubseteq_p y) & \iff & \subseteq = [\![\sqsubseteq_p]\!] \\ \mathcal{M} \models \forall_{x,y}(x \sqsubseteq y \leftrightarrow y \sqsubseteq_p x) & \iff & \supseteq = [\![\sqsubseteq_p]\!] \end{array}$$

$$\frac{\text{ALL } \mathbf{a} \text{ ARE } \mathbf{b} \quad \text{ALL } \mathbf{b} \text{ ARE } \mathbf{c}}{\text{ALL } \mathbf{a} \text{ ARE } \mathbf{c}}\ BARBARA$$

$$\frac{\text{ALL } \mathbf{a} \text{ ARE } \mathbf{b} \quad p \text{ IS UPWARD MONOTONE}}{\text{IF } p(\mathbf{a}) \text{ THEN } p(\mathbf{b})}\ \uparrow$$

$$\frac{\text{ALL } \mathbf{a} \text{ ARE } \mathbf{b} \quad p \text{ IS DOWNWARD MONOTONE}}{\text{IF } p(\mathbf{b}) \text{ THEN } p(\mathbf{a})}\ \downarrow$$

$$\frac{}{\text{ALL } \mathbf{a} \text{ ARE } \mathbf{a}}\ Axiom1$$

$$\frac{}{\text{IF } p(\mathbf{a}) \text{ THEN } p(\mathbf{a})}\ Axiom2$$

2.6 Soundness and Completeness

Definition 2.3. For a set of $\mathcal{L}$-sentences Γ, we can define the *canonical model* $\mathcal{M}_\Gamma$ as follows:

First, let M be the set of atomic constant symbols $\mathbf{A}$ and define a relation $\leq$ on $\mathbf{A}$ where $\mathbf{a} \leq \mathbf{b} \iff \Gamma \vdash a \sqsubset b$. The interpretation function $[\![\cdot]\!]$ is defined as follows:

Define $[\![\mathbf{a}]\!] = {\downarrow}\,\mathbf{a} = \{\mathbf{b} \in P \mid \mathbf{b} \leq \mathbf{a}\}$.

Define $[\![\sqsubseteq]\!]$ as the $\subseteq$ relation on $\mathcal{P}(M)$.

For each $p \in \mathbf{P}$, we have a conditional definition:

If and only if "p is upward monotone" is the only sentence about p entailed by Γ, we define $[\![\sqsubseteq_p]\!] = \subseteq$.

If and only if "p is downward monotone" is the only sentence about p entailed by Γ, we define $[\![\sqsubseteq_p]\!] = \subseteq$.

In all other cases, $[\![\sqsubseteq_p]\!]$ is defined as set equality in $\mathcal{P}(M)$.

Lemma 1. *For a set Γ of $\mathcal{L}$-sentences, the canonical model $\mathcal{M}_\Gamma \models \Gamma$.*

Proof. The key parts of the proof are a consequence of the fact that ${\downarrow}\, a \subseteq {\downarrow}\, b \iff a \leq b$, and ${\downarrow}\, a \supseteq {\downarrow}\, b \iff b \leq a$ which is crucial to the case that Γ contains both "p is upward monotone" and "p is downward monotone". The rest is a routine consequence of the definitions. □

Theorem 2. *Soundness and Completeness*

Proof. We leave the perfunctory soundness proof as an exercise to the reader. Towards showing completeness, let Γ be a set of $\mathcal{L}$-sentences and ϕ another $\mathcal{L}$-sentence. Suppose that for every model $\mathcal{M}$ we have that $\Gamma \models \phi$. In particular, by lemma 1, $\mathcal{M}_\Gamma \models \phi$. All further discussion occurs in this specific model. The sentence ϕ may have one of four forms.

Suppose firstly that ϕ is "if $p(\mathbf{a})$ then $p(\mathbf{b})$". Thus, $([\![\mathbf{a}]\!], [\![\mathbf{b}]\!]) \in [\![\sqsubseteq_p]\!]$. Since the interpretation of $\sqsubseteq_p$ depends on the description of p entailed by Γ, there are three cases: Firstly, if $\Gamma \vdash$ "p is upward monotone" only, then it follows that $[\![\mathbf{a}]\!] \subseteq [\![\mathbf{b}]\!]$. Since this holds if and only if $a \leq b$ by a basic property of down-sets, then we will have $\Gamma \vdash a \sqsubseteq b$ and $\Gamma \vdash$ "p is upward monotone", so that $\Gamma \vdash$ "if $p(\mathbf{a})$ then $p(\mathbf{b})$" by the $\uparrow$ deduction rule.

On the other hand, if we had that $\Gamma \vdash$ "p is downward monotone" only, we can similarly deduce that $[\![\mathbf{a}]\!] \supseteq [\![\mathbf{b}]\!]$, and repeating the same reasoning arrive at $\Gamma \vdash$ "if $p(\mathbf{a})$ then $p(\mathbf{b})$". In the last option for p, we either have that Γ proves neither or both of the statements "p is upward monotone" and "p is downward monotone", and in either case $[\![\sqsubseteq_p]\!]$ is set equality in $\mathcal{M}_\Gamma$. As such, we will be able to conclude that $[\![\mathbf{a}]\!] = [\![\mathbf{b}]\!]$. Equal down-sets imply that $a = b$, so that trivially $\Gamma \vdash$ "if $p(\mathbf{a})$ then $p(\mathbf{b})$" Hence, in all of these cases, $\Gamma \vdash \phi$.

If ϕ is the sentence "p is upward monotone" (we omit the dual, which is similar), then truth in the canonical model gives us that $\subseteq = [\![\sqsubseteq_p]\!]$. In the $\mathcal{M}_\Gamma$, this happens exactly when $\Gamma \vdash$ "p is upward monotone" . The last option for ϕ is covered in the completeness theorem for the basic syllogistic logic with the "BARBARA" deduction rule and Axiom 1.

In conclusion, in all cases we may deduce that $\Gamma \vdash \phi$. □

3 Related Work

The study of monotonicity in natural language has a strongly developed linguistic and mathematical theoretical groundwork, dating back to the monotonicity calculus of (Valencia, 1991) and in semantic studies such as (van Benthem, 1988). Its formal treatments have led to the expansion of typed lambda calculus with an order relation so as to model this order-theoretic behaviour, resulting in a corresponding deduction system and completeness theorem in (Icard et al., 2017). There are varying presentations and some variation in terminology, but for the most part *monotonicity* refers to the

order-theoretic property of the context as a function, while the term *polarity* usually refers to the tag assigned to the node in a CCG parse tree or a word in a sentence. The inferential mechanism based on monotonicity properties of quantifiers, determiners and contexts in general is sometimes referred to as *natural logic*, and the underlying principles of natural logic applying to set-theoretic concept relations has led to work on *generalized monotonicity* (MacCartney and Manning, 2009). However, the additional relations such as negation, alternation and cover are no longer order-theoretic notions.

Symbolic Implementations There are two flavours of implementations that result in the deductions allowed by monotonicity reasoning. Firstly, works such as (Hu et al.; Abzianidze, 2015) rely on linguistically-informed polarity markings on the nodes of CCG parse trees. They require accurate parses and expertly hand-crafted linguistic rules to mark the nodes with polarity tags, as in (Hu and Moss, 2018). In (Hu et al.), a premise is tagged for monotonicity and a knowledge base of hypotheses created by a substitution known to be truth-preserving is generated. Candidate hypotheses are compared with this set, checking for exact matches. On the other hand, (Abzianidze, 2015) uses the CCG parses to further translate to a lambda logical form for use in a deduction method inspired by tableau calculus. These approaches differ from strategies such as (MacCartney and Manning, 2009), which require an *edit sequence* which transforms the premise into the hypothesis. Atomic edits are tagged with generalized entailment relations which are combined with a join operator based on relational composition to determine whether the transformation is overall truth-preserving, hence yielding a hypothesis entailed by the premise. Later, (Angeli and Manning, 2014) treated these atomic edits as edges in a graph and phrased entailment detection as a graph search problem. Concepts from symbolic approaches to NLI have also been applied in symbolic question answering systems (such as in (Bobrow et al., 2007)), and hybridized with neural systems (such as in (Kalouli et al., 2020)).

Neural NLI Models and Monotonicity State of the art NLI models have previously been shown (Yanaka et al., 2019; Geiger et al., 2020) to perform poorly on examples where the context f is *downward monotone*, as occurs in the presence of negation and various generalized quantifiers such as "every" and "neither". Benchmark datasets such as MNLI are somewhat starved of such examples, as observed by (Yanaka et al., 2019). As a consequence, the models trained on such benchmark datasets as MNLI not only fail in downward monotone contexts, but *systematically* fail: they tend to treat all examples as if the contexts are upward monotone, predicting the *opposite* entailment label with high accuracy (Yanaka et al., 2019; Geiger et al., 2020). Data augmentation techniques and additional fine-tuning with an inoculation (Liu et al., 2019a) strategy have been attempted in (Yanaka et al., 2019; Richardson et al., 2020) and (Geiger et al., 2020). In the latter case, performance on a challenge test set improved without much performance loss on the original benchmark evaluation set (SNLI), but in (Yanaka et al., 2019) there was a significant decrease in performance on the MNLI evaluation set. These studies form the basis on which we aim to build, and their choice of evaluation datasets and models inspires our own choices.

		Previous Work			
Evaluation Datasets		Geiger 2020 (Neural)	Yanaka 2020 (Neural)	Moss 2019 (Neural)	Hu 2020 (Symbolic)
Large, Broad Coverage	MNLI Test		x		
	MNLI Dev (Mismatched)			x	
	SNLI Test	x		x	
Small, Targeted Phenomena	MED		x		
	SICK		x*		x
	FraCaS		x*		x
	MoNLI Test	x			
	Monotonicity Fragments			x	x

Table 3: Evaluation datasets used in previous work investigating monotonicity reasoning. Positions marked $*$ indicate that the dataset is included in another used evaluation dataset.

Neural Transformer-based language models have been shown to implicitly model syntactic structure (Hewitt and Manning, 2019). There is also evidence to suggest that these NLI models are at least representing the concept relations quite well and using this information to predict the entailment label, as corroborated by a study based on *interchange interventions* in (Geiger et al., 2020).

We hypothesise that such models have the capacity for learning monotonicity features. The extent to which the representations capture monotonicity information in the contextual representations of tokens in the sequence is not yet well understood, and this is an investigation we wish to initiate and

encourage with this work.

4 Experiments

Building on the observations in the above-mentioned previous papers, we ask the following questions:

- Can a context monotonicity classification task in the model training pipeline further improve performance on targeted evaluation sets which test monotonicity reasoning?
- Does this mitigate the decrease in performance on benchmark NLI datasets?

Our investigation proceeds in three parts: Firstly, we attempt to fine-tune a SOTA NLI model for a context monotonicity classification task.

Secondly, we retrain the above model for NLI and evaluate the performance on several evaluation datasets which specifically target examples of both upward and downward monotonicity reasoning. We examine whether there is any improvement over a previously suggested approach on fine-tuning on a large, automatically generated dataset (HELP) from (Yanaka et al., 2019).

Models We start with existing NLI models pretrained on benchmark NLI datasets. In particular (and for best comparison with related studies) we use RoBERTa (Liu et al., 2019b) pretrained on MNLI (Williams et al., 2018) and BERT (Devlin et al., 2019) pretrained on SNLI (Bowman et al., 2015). These are two benchmark NLI datasets which contain examples derived from naturally occurring text and crowd-sourced labels, aiming for scale and broad coverage. We do not deviate from the architecture, as we are only investigating the effect of training on different tasks (monotonicity classification and NLI) and datasets.

4.1 Retraining NLI Models to Classify Context Monotonicity

Traditionally, symbolic approaches treat monotonicity classification as the task of labeling of each node in a CCG parse tree with either an upward or downward polarity marking. Our emphasis of monotonicity as a property of a *context* allows for a different framing of this problem: we consider monotonicity classification as a binary classification task by explicitly indicating (with a variable) the "slot" in the sentence for which we wish to know the polarity. Different positions of the variable in a partial sentence may yield a context with a different monotonicity label; a typical example of this is sentences featuring generalized quantifiers such as "every", which may be monotone up in one argument but monotone down in another.

4.1.1 Input Representation

The NLI models which we wish to start with are transformer-based models, in line with the current state of the art approaches to NLI. Transformer models represent a sentence as a sequence of tokens: we take a naive approach to representing a context by indicating the variable with an uninformative 'x' token. We refrain from using the mask token to indicate the variable, as the underlying pretrained transformer language models are trained to embed the mask token in such a way as to correspond with high-likelihood insertions in that position, which we would prefer to avoid.

4.1.2 Dataset

In order to ensure our monotonicity classification task does not add any unseen data (when compared to only fine-tuning on the HELP dataset) we adapt the HELP dataset for this task. The HELP dataset (Yanaka et al., 2019) consists of a set of $(p(\mathbf{a}), p(\mathbf{b}))$ pairs which included labels for the entailment relationship between the full sentences, the inclusion relation between $\mathbf{a}$ and $\mathbf{b}$, and the monotonicity classification of p. As such, we extract only the contexts p and the monotonicity label into dataset which we will call "HELP-contexts", which we split into a train, development and test set in a 50:20:30 ratio. Examples of this dataset are presented on Table 4. [1]

Context	Context Monotonicity
There were no x today.	downward monotone
There is no time for x.	downward monotone
Every x laughed.	downward monotone
There is little if any hope for his x .	downward monotone
Some x are allergic to wheat.	upward monotone
Tom is buying some flowers for x.	upward monotone
You can see some wild rabbits in the x.	upward monotone

Table 4: Examples from the HELP-contexts dataset, with respective labels.

[1]The original HELP dataset also contains a few non-monotone examples: in the current state of this work, these are omitted in favor of a focus on the specific confusion in existing models where downwards monotone contexts are often treated as upwards monotone ones.

4.1.3 Results

As presented in Table 5, the task of predicting the monotonicity of a context can be solved using fine-tuned transformer models. This suggests a potential path for inducing a bias for context classification in downstream tasks such as NLI, which could benefit from better modeling of context monotonicity.

Model	Evaluation Data					
	HELP-Contexts Dev			HELP-Contexts Test		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
bert-base	98.74	99.08	98.91	98.00	95.24	96.54
bert-large	98.23	98.88	98.55	97.51	95.70	96.57
roberta-large-mnli	99.62	98.92	99.26	98.73	96.64	97.64
roberta-large	99.81	99.46	99.27	98.99	96.41	97.62
roberta-base	99.81	99.46	99.63	98.10	95.56	96.76
bert-base-uncased-snli	98.88	98.19	8.53	98.92	97.29	98.07

Table 5: Performance of state of the art models for the context prediction task. Each model was trained on HELP context (training set).

4.2 Improving NLI Performance on Monotonicity Reasoning

A few datasets have been curated to either fine-tune or evaluate NLI models with monotonicity reasoning in mind: their uses in previous related works are detailed in table 3. We use the following datasets for training and evaluation respectively:

4.2.1 Training Data

We start by once again using the HELP dataset (Yanaka et al., 2019), which was designed specifically as a balanced additional training set for the improvement of NLI models with respect to monotonicity reasoning. We create a split of this dataset which is based on the HELP-contexts dataset by assigning each example either to the train, development or test set depending on which split its associated context f is in the HELP-contexts dataset. This is to ensure there is no overlap between the examples' contexts accross the three data partitions. Our approach combined this strategy with an additional step based on the context monotonicity task described in section 4.1.

4.2.2 Training Procedure

We rely on the architecture implementations and pretrained models available with the *transformers* library (Wolf et al., 2020). Starting with the pretrained models (which we shall henceforth tag as "bert-base-uncased-snli" and "roberta-large-mnli"), we first fine-tune these models for the context monotonicity classification task using the training partition of the HELP-contexts dataset. We re-use the classification head of the pretrained models for this purpose, but only use two output states for the classification.

4.2.3 Evaluation Data

Evaluation datasets are typically small, challenging and categorized by certain target semantic phenomena. Following previous work in this area, we evaluate our approach using the MED dataset introduced in (Yanaka et al., 2020), which is annotated with monotonicity information and draws from various expertly-curated diagnostic challenge sets in NLI such as SICK, FraCaS and the SuperGlue Diagnostic set. It features a balanced split between upward and downward monotone contexts, in contrast to the benchmark MNLI dataset. Additionally, we include evaluation on the MoNLI dataset (Geiger et al., 2020) which also features a labeled balance of upward and downward monotone examples. However, the latter dataset's downward monotone examples are only exemplary of contexts featuring the negation operator *"not"*, whereas MED (Yanaka et al., 2020) also includes more complex downward monotone operators such as generalized quantifiers and determiners. We refer to these respective papers (Yanaka et al., 2020; Geiger et al., 2020) for full breakdowns and analyses of these datasets.

4.2.4 Baselines

Although the main comparison to be made is the improvement introduced when including the context-monotonicity-classification training on top of the current state-of-the art roberta-large-mnli model trained on HELP, we include an additional baselines: roberta-large-mnli fine-tuned on the *monotonicity fragment* from the *semantic fragments* (Richardson et al., 2020) dataset. The strategy in this work is the same as with the HELP dataset, but we include this in the evaluation on the chosen challenge sets for a more complete comparison.

4.2.5 Results

We present the results on the challenge sets MED and MoNLI in Table 6, with a break-down by upward and downward monotone contexts. Furthermore, we have re-run each model on the original benchmark evaluation datasets SNLI and MNLI, with the results visible in Table 7.

Model	Additional Training Data	Challenge Datasets					
		MoNLI Test			MED		
		Upward Monotone	Downward Monotone	**All**	Upward Monotone	Downward Monotone	**All**
bert-base-uncased-snli	-	37.74	56.49	46.15	53.58	43.91	49.36
bert-base-uncased-snli	HELP	30.89	85.02	**55.19**	43.4	72.43	60.18
bert-base-uncased-snli	**HELP + HELP-Contexts**	21.6	97.67	**55.19**	32.56	87.13	**66.22**
roberta-large-mnli	-	95.19	5.32	58.84	82.12	25.76	46.09
roberta-large-mnli	Monotonicity Fragments (Easy)	92.68	79.62	86.81	74.54	65.68	70.05
roberta-large-mnli	Monotonicity Fragments (All)	50.00	50.00	50.00	35.42	61.80	49.78
roberta-large-mnli	HELP	94.72	98.67	96.48	64.47	86.25	**77.4**
roberta-large-mnli	**HELP + HELP-Contexts**	98.78	97.17	**98.06**	65.24	85.12	76.44

Table 6: Performance of NLI models on challenge datasets designed to test performance on monotonicity reasoning.

Model	Additional Training Data	Benchmark Datasets							
		MNLI (m*) Dev		MNLI (mm*) Dev		SNLI Dev		SNLI Test	
		Acc	Δ	Acc	Δ	Acc	Δ	Acc	Δ
bert-base-uncased-snli	-	44.96	-	45.52	-	41.54	-	40.78	-
bert-base-uncased-snli	HELP	35.13	-9.83	34.37	-11.5	25.93	-15.61	25.92	-14.86
bert-base-uncased-snli	**HELP + HELP-Contexts**	36.91	**-8.05**	37.36	**-8.16**	36.54	**-5.00**	37.20	**-3.58**
roberta-large-mnli	-	94.11	-	93.88	-	93.33	-	93.14	-
roberta-large-mnli	HELP	82.66	**-11.45**	83.38	**-10.50**	74.77	-18.56	74.39	-18.75
roberta-large-mnli	**HELP + HELP-Contexts**	81.00	-13.11	82.01	-11.87	82.99	**-10.34**	82.31	**-10.83**

Table 7: Fine-tuning state of the art NLI models with the aim of improving monotonicity has tended to result in lower performance on the original benchmark NLI datasets. We compare these performance losses in addition to tracking performance on the the challenge datasets. * MNLI (m) and (mm) refers to the matched and mismatched dataset respectively. For MNLI, only the *Dev* set is publically available.

5 Discussion

Average Performance Firstly, we confirm previous observations that the starting pretrained transformer model roberta-large-mnli (which is considered a high-performing NLI model, achieving over 93% accuracy on the large MNLI development set) has a dramatic performance imbalance with respect to context monotonicity. The fact that performance on downward monotone contexts is as low as 5% suggests that this model perhaps routinely assumes upward monotone contexts. It was noted in (Yanaka et al., 2019) that the MNLI benchmark dataset is strongly skewed in favor of upward monotone examples, which may account for this.

Our approach outperforms or matches the baseline models in three of the summary accuracy scores, and is competitive in the fourth. Furthermore, in most cases we observe less performance loss on the benchmark sets.

Performance by Monotonicity Category As evident from Table 6, we observe a substantial improvement for the bert-base-uncased NLI models for downward monotone contexts. For the much larger roberta-large-mnli models, any gains over the model trained on HELP only are quite small. A common observation is the notable trade-off between accuracy on upward and downward monotone contexts; training that improves one of these over a previous baseline generally seem to decrease performance of the other. This is especially evident in the MED dataset, which is larger and representative of a more diverse set of downward monotone examples (the MoNLI dataset is limited to the "No" operator). Sensibly, a decrease in performance in upward monotone contexts also leads to a decrease in performance on the original SNLI and MNLI datasets 7 (which are skewed in favor of upward monotone examples). However, in most cases (except for the roberta-large-mnli model on the MNLI benchmark) our method results in a *smaller* performance loss.

6 Conclusion and Future Work

Introducing context monotonicity classification into the training pipeline of NLI models provides some performance gains on challenge datasets designed to test monotonicity reasoning. We see contexts as crucial objects of study in future approaches to natural language inference. The ability to detect their logical properties (such as monotonicity) opens the door for hybrid neuro-symbolic NLI models and reasoning systems, especially in so far as dealing with out of domain insertions that may confuse out-of-the-box NLI models. The linguistic flexibility that transformer-based language

models bring is too good to lose; leveraging their power in situations where only part of our sentence is in a model's distribution would be helpful for domain-specific use cases with many out-of-distribution nouns. Overall, we are interested in furthering both the *analysis* and *improvement* of emergent modelling of abstract logical features in neural natural language processing models.

References

Lasha Abzianidze. 2015. A tableau prover for natural logic and language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2492–2502, Lisbon, Portugal. Association for Computational Linguistics.

Gabor Angeli and Christopher D Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 534–545.

Johan van Benthem. 1988. Essays in logical semantics. *Studia Logica*, 47(2):172–173.

D. Bobrow, Bob Cheslow, C. Condoravdi, Tracy Holloway King, R. Nairn, and A. Zaenen. 2007. Parc's bridge and question answering system.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

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.

A. Geiger, Kyle Richardson, and Christopher Potts. 2020. Neural natural language inference models partially embed theories of lexical entailment and negation. *arXiv: Computation and Language*.

Emily Goodwin, Koustuv Sinha, and Timothy J. O'Donnell. 2020. Probing linguistic systematicity. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1958–1969, Online. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. 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 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Hai Hu, Qi Chen, Kyle Richardson, Atreyee Mukherjee, Lawrence S Moss, and Sandra Kuebler. Monalog: a lightweight system for natural language inference based on monotonicity. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2020*, page 319.

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.

Thomas Icard, Lawrence Moss, and William Tune. 2017. A monotonicity calculus and its completeness. In *Proceedings of the 15th Meeting on the Mathematics of Language*, pages 75–87, London, UK. 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.

Nelson F. Liu, Roy Schwartz, and Noah A. Smith. 2019a. Inoculation by fine-tuning: A method for analyzing challenge datasets. *CoRR*, abs/1904.02668.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the Eight International Conference on Computational Semantics*, pages 140–156, Tilburg, The Netherlands. Association for Computational Linguistics.

Lawrence S. Moss. 2008a. *Completeness theorems for syllogistic fragments*, pages 143–174. De Gruyter Mouton.

Lawrence S. Moss. 2008b. Syllogistic Logics with Verbs. *Journal of Logic and Computation*, 20(4):947–967.

Kyle Richardson, H. Hu, L. Moss, and A. Sabharwal. 2020. Probing natural language inference models through semantic fragments. *ArXiv*, abs/1909.07521.

V.M.S. Valencia. 1991. *Studies on Natural Logic and Categorial Grammar*. Universiteit van Amsterdam.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, and Kentaro Inui. 2020. Do neural models learn systematicity of monotonicity inference in natural language? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6105–6117, Online. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019. 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.

Bayesian Classification, Inference in a Probabilistic Type Theory with Records

Staffan Larsson
Centre for Linguistic Theory and Studies in Probability (CLASP)
Dept. of Philosophy, Linguistics and Theory of Science
University of Gothenburg
sl@ling.gu.se

Robin Cooper
Centre for Linguistic Theory and Studies in Probability (CLASP)
Dept. of Philosophy, Linguistics and Theory of Science
University of Gothenburg
cooper@ling.gu.se

Abstract

We propose a probabilistic account of semantic inference and classification formulated in terms of probabilistic type theory with records, building on Cooper et al. (2014, 2015). We suggest probabilistic type theoretic formulations of Naive Bayes Classifiers and Bayesian Networks. A central element of these constructions is a type-theoretic version of a random variable. We illustrate this account with a simple language game combining probabilistic classification of perceptual input with probabilistic (semantic) inference.

1 Introduction

A probabilistic type theory was presented in Cooper et al. (2014) and Cooper et al. (2015), which extends Cooper's Type Theory with Records (TTR, Cooper, 2012; Cooper and Ginzburg, 2015; Cooper, in prep). Non-probabilistic TTR (in common with other type theories) works with judgements of the form $a : T$ ("a is of type T") and assumes that such judgements are categorical. In probabilistic TTR (probTTR) we associate probabilities with judgements: $p(a : T)$ ("the probability that a is of type T").

TTR has been used previously for natural language semantics (see, for example, Cooper, 2005 and Cooper, 2012), and to analyze semantic coordination and learning (for example, Larsson and Cooper, 2009; Cooper and Larsson, 2009). It has also been applied to the analysis of interaction in dialogue (for example, Ginzburg, 2012 and Breitholtz, 2020), and in modelling robotic states and spatial cognition (for example, Dobnik et al., 2013).

Two main considerations motivated recasting TTR in probabilistic terms. First, a probabilistic type theory offers a natural framework for capturing the gradience of semantic judgements. This allows it to serve as the basis for an account of vagueness in interpretation, as shown by Fernández and Larsson (2014). Second, such a theory lends itself to developing a model of semantic learning that can be straightforwardly integrated into more general probabilistic explanations of learning and inference.

Furthermore, we believe this will provide the foundation for a unified probabilistic account of natural language semantics that accounts for reasoning (logical as well as non-logical/enthymematic as in Breitholtz, 2020), learning (semantic and factual) and interaction, and that integrates low-level, subsymbolic real-valued perceptual information and high-level symbolic information (Larsson, 2015).

In this paper we suggest a way of incorporating a probabilistic inference and classification into ProbTTR. We do this because we believe that vagueness, learning, inference and classification are central rather than peripheral notions in semantics, and that probabilistic reasoning is central to all of them. Also, in contrast to an approach where e.g. classifiers are implemented outside the semantic theory, we want the reasoning underlying an agent's behaviour to be as transparent as possible to the agent itself (and thereby potentially also to its interlocutors).

To incorporate a probabilistic inference and classification into ProbTTR, we will need to introduce a ProbTTR version of a random variable, not discussed in Cooper et al. (2015). We will also show how probabilistic classification of perceptual evidence can be combined with probabilistic reasoning.

We first provide a brief overview of TTR and Probabilistic TTR. Section 4 provides some background on probabilistic inference and classification. Section 5 introduces conditional probabilities and defines a type theoretic version of a random variable. We use these variables to characterise a Naive Bayes classifier in Section 6. We illustrate Naive

Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA), pages 51–59

Bayes classification with an example of semantic classification. In Section 7 we show how probabilistic perception and reasoning can be combined in ProbTTR. We then introduce a ProbTTR characterisation of Bayesian Networks, and briefly discuss semantic learning. In Section 10 we present our conclusions and discuss directions for future work.

2 TTR: A brief introduction

We give a brief sketch of those aspects of TTR which we will use in this paper. For more detailed accounts see Cooper and Ginzburg (2015); Cooper (in prep).

$s : T$ represents a judgement that s is of type T. A second kind of judgement (often written Ttrue in Martin-Löf type theory) is the judgement that there is something of type T (T is non-empty).Types may be either *basic* or *complex* (in the sense that they are structured objects which have types or other objects introduced in the theory as components). One basic type that we will use is *Ind*, the type of individuals; another is *Real*, the type of real numbers. Among the complex types are *ptypes* which are constructed from a predicate and arguments of appropriate types as specified for the predicate. Examples are 'man(a)', 'see(a,b)' where a, b : *Ind*. The objects or *witnesses* of ptypes can be thought of as situations, states or events in the world which instantiate the type. Thus $s : \text{man}(a)$ can be glossed as "s is a situation which shows (or proves) that a is a man".

Another kind of complex type is *record types*. In TTR *records* are modelled as finite sets of fields. Each field is an ordered pair, $\langle \ell, o \rangle$, where ℓ is a *label* (drawn from a countably infinite stock of labels) and o is an object which is a witness of some type. No two fields of a record can contain the same label. Importantly, o can itself be a record. A *record type* is like a record except that the fields are of the form $\langle \ell, T \rangle$ where ℓ is a label as before and T is a type. The basic intuition is that a record, r is a witness for a record type, T, just in case for each field, $\langle \ell_i, T_i \rangle$, in T there is a field, $\langle \ell_i, o_i \rangle$, in r where $o_i : T_i$. (Note that this allows for the record to have additional fields with labels not included in the fields of the record type.) The types within fields in record types may *depend* on objects which can be found in the record which is being tested as a witness for the record type. We use a graphical display to represent both records and record types where each line represents a field. Example (1) represents the type of records which can be used to model situations where a man runs.

$$(1)\quad \begin{bmatrix} \text{ref} & : & \textit{Ind} \\ \text{c}_{\text{man}} & : & \text{man(ref)} \\ \text{c}_{\text{run}} & : & \text{run(ref)} \end{bmatrix}$$

A record of this type would be of the form

$$(2)\quad \begin{bmatrix} \text{ref} & = & a \\ \text{c}_{\text{man}} & = & s \\ \text{c}_{\text{run}} & = & e \\ \dots & & \end{bmatrix}$$

where a : *Ind*, $s : \text{man}(a)$ and $e : \text{run}(a)$.

We will introduce further details of TTR as we need them in subsequent sections.

3 Probabilistic TTR fundamentals

The core of ProbTTR is the notion of probabilistic judgement. There are two kinds of judgement corresponding to the two kinds of judgement in non-probabilistic TTR. The first is a judgement that a situation, s, is of type, T, with some probability. $p(s : T)$ is the probability that s is a witness for T. The second is a judgement that there is some witness of type T. $p(T)$ is the probability that there is some witness for T. This introduces a distinction that is not normally made explicit in the notation used in probability theory.

It is useful to have type theoretic objects corresponding to judgements that a situation is of a type. Following terminology first introduced in Barwise (1989, Chap. 11), we call these *Austinian propositions*. A *probabilistic Austinian proposition* is an object (a record) that corresponds to, or encodes, a probabilistic judgement. Probabilistic Austinian propositions are records of the type in (3).

$$(3)\quad \begin{bmatrix} \text{sit} & : & \textit{Sit} \\ \text{sit-type} & : & \textit{Type} \\ \text{prob} & : & [0,1] \end{bmatrix}$$

(where $[0, 1]$ represents the type of real numbers between 0 and 1). A probabilistic Austinian proposition φ of this type corresponds to the judgement that φ.sit is of type φ.sit-type with probability φ.prob. That is,

(4) p(φ.sit:φ.sit-type)= φ.prob

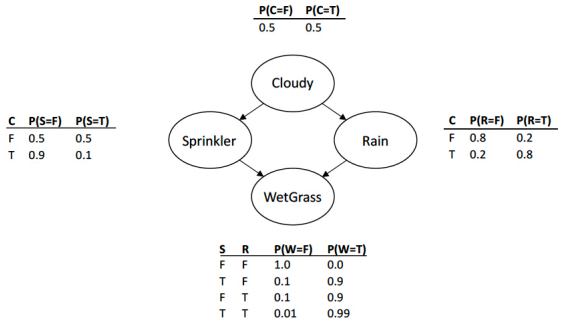

Figure 1: Example Bayesian Network

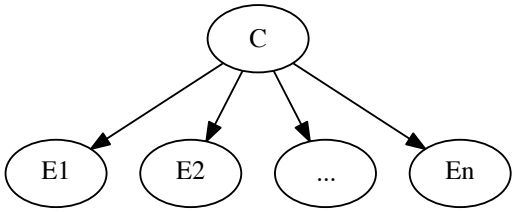

Figure 2: Naive Bayes classifier

4 Probabilistic Inference and Classification

A Bayesian Network is a Directed Acyclic Graph (DAG)[1]. The nodes of the DAG are random variables, each of whose values is the probability of one of the set of possible states that the variable denotes. Its directed edges express dependency relations among the variables. When the values of all the variables are specified, the graph describes a complete joint probability distribution (JPD) for its random variables (Pearl, 1990; Halpern, 2003).

Russell and Norvig (1995) give the example Bayesian Network in Figure 1. The only directly observable evidence is whether it is cloudy or not, and the queried variable is whether the grass is wet or not. We do not know if it is raining, or whether the sprinkler is on. Both of these factors depend on whether it is cloudy, and both affect the grass being wet.

From this Bayesian Network we can compute the marginal probability of the grass being wet (W = T).

(5) $p(W\text{=T})=\sum_{s,r,l} p(W\text{=T}, S\text{=}s, R\text{=}r, C\text{=}c)$

Here, s, r and l can be either T(rue) or F(alse).

The Bayesian network in Figure 1 allows us to simplify the computation of this JPD by encoding independence relations between variables, so that:

(6) $p(W, S, R, C)=p(W|S, R)p(S|C)p(R|C)p(C)$

and hence

(7) $p(W\text{=T})=$
$\sum_{s,r,l} p(W\text{=T}|S\text{=}s, R\text{=}r)p(S\text{=}s|C\text{=}c)$
$p(R\text{=}r|C\text{=}c)p(C\text{=}c)$

[1]This section briefly explains Bayesian nets and Naive Bayes classifiers, and introduces examples that will be used later. Readers familiar with this material can safely skip ahead to Section 5.

A standard Naive Bayes model is a Bayesian network with a single class variable C that influences a set of evidence variables $E_1, \ldots, E_n$ (the evidence), which do not depend on each other. Figure 2 illustrates the relation between evidence variables and a class variable in a Naive Bayes classifier.

A Naive Bayes classifier computes the marginal probability of a class, given the evidence:

(8) $p(c) =$
$\sum_{e_1,\ldots,e_n} p(c \mid e_1, \ldots, e_n)p(e_1) \ldots p(e_n)$

where c is the value of C, e_i is the value of E_i $(1 \leq i \leq n)$ and the conditional probability of the class given the evidence is estimated thus:

(9) $\hat{p}(c \mid e_1, \ldots, e_n) =$

$$\frac{p(c)p(e_1 \mid c) \ldots p(e_n \mid c)}{\sum_{C=c'} p(c')p(e_1 \mid c') \ldots p(e_n \mid c')}$$

Of course, if the assumption regarding the independence of the evidence variables does not hold, this estimation may be incorrect; this is the price to pay for the relative simplicity of the Naive Bayes classifier.

5 Type theoretic probabilistic inference and classification

We now turn to an account of probabilistic classification in ProbTTR. We first show how probabilistic inference can be modelled in ProbTTR. We then provide a Naive Bayes classifier with a detailed example. Finally, we generalise this account to Bayesian Networks.

5.1 Conditional probabilities in ProbTTR

We use $p(T_1||T_2)$ to represent the estimated[2] conditional probability that any situation, s, is of type

[2]Estimating $p(T_1||T_2)$ is part of the learning theory.

T_1 given that it is of type T_2. This contrasts with two other probability judgements in probTTR: $p(s_1 : T_1|s_2 : T_2)$, the probability that a particular situation, s_1, is of type T_1 given that s_2 is of type T_2 , and $p(T_1|T_2)$, the probability that there is a situation of type T_1 given that there is a situation of type T_2. In addition there are "mixed" probabilities such as $p(T_1|s : T_2)$, the probability that there is a situation of type T_1 given that $s : T_2$.

5.2 Random variables in TTR

To do probabilistic inference in ProbTTR, we need a type theoretic counterpart of a random variable in probabilistic inference. Assume a single (discrete) random variable with a range of possible (mutually exclusive) values. We introduce a *variable type* V whose range is a set of *value types* $\mathfrak{R}(V) = \{A_1, \ldots, A_n\}$ such that the following conditions hold.

(10) a. $A_j \sqsubseteq V$ for $1 \leq j \leq n$

b. $A_j \perp A_i$ for all i, j such that $1 \leq i \neq j \leq n$

c. for any s, $p(s : V) \in \{0, 1.0\}$ and $p(s : V) = \sum_{T \in \mathfrak{R}(V)} p(s : T)$

(10 a) says that all value types for a variable type V are subtypes of V. (A type T_1 is a subtype of type T_2, $T_1 \sqsubseteq T_2$, just in case $a : T_1$ implies $a : T_2$ no matter what we assign to the basic types.) A simple way of achieving this is to let $V = A_i \vee \ldots \vee A_n$. ($T_1 \vee T_2$ is the *join type* of T_1 and T_2. $a : T_1 \vee T_2$ just in case either $a : T_1$ or $a : T_2$). (10 b) says that all value types for a given variable type V are mutually exclusive, i.e. there are no objects that are of two value types for V. (10 c) says that the probability of a situation s being of a variable type V is either 0 or 1.0. If it is 0 (i.e., the variable has no value for the situation), then the probabilities that s is of each of the value types for V sum to 0; otherwise these probabilities sum to 1.0.

(10) encodes a conceptual difference between the probability that something has a property (such as colour, $p(s$:Colour)), and the probability that it has a certain value of a variable (e.g. $p(s$:Green)). If the probability distribution over different values (colours) sums to 1.0, then the probability that the object in question has a colour is 1.0. The probability that an object has colour is either 0 or 1.0. We assume that certain ontological/conceptual type judgements of the form "physical objects have colour" are categorical (which in a probabilistic framework means they have probability 0 or 1.0).

We can now formulate the example in Figure 1 in ProbTTR. We assume four binary variable types *Grass*, *Sprinkler*, *Raining* and *Cloudy* with corresponding variable value types as given in (11).

(11) $\mathfrak{R}$(*Grass*)={*GrassWet*, *GrassDry*}
$\mathfrak{R}$(*Sprinkler*)={*SprinklerOn*, *SprinklerOff*}
$\mathfrak{R}$(*Raining*)={*IsRaining*, *IsNotRaining*}
$\mathfrak{R}$(*Cloudy*)={*ItIsCloudy*, *ItIsNotCloudy*}

We specify that *Grass*=*GrassWet*$\vee$*GrassDry*, and similarly for the other variable types. This will ensure that *GrassWet*$\sqsubseteq$*Grass*, and similar subtyping constraints hold. Assuming that the variable types and variable value types are related as in (11) also entails that *GrassIsWet*$\perp$*GrassIsDry*, and similarly for the other variable value type pairs.

5.3 A ProbTTR Naive Bayes classifier

Corresponding to the evidence, class variables, and their values, we associate with a ProbTTR Naive Bayes classifier κ

(12) a. a collection of m evidence variable types $\mathbb{E}^\kappa_1, \ldots, \mathbb{E}^\kappa_n$,

b. associated sets of evidence value types $\mathfrak{R}(\mathbb{E}^\kappa_1), \ldots, \mathfrak{R}(\mathbb{E}^\kappa_n)$,

c. a class variable type $\mathbb{C}^\kappa$, and

d. an associated set of class value types $\mathfrak{R}(\mathbb{C}^\kappa)$.

To classify a situation s using a classifier κ, the evidence is acquired by observing and classifying s with respect to the evidence types. This can be done through another layer of probabilistic classification based on yet another set of evidence types. Type judgements can also be obtained directly from probabilistic or non-probabilistic classification of low-level sensory readings supplied by observation.

We define a ProbTTR Bayes classifier κ as a function from a situation s (of the meet type of the evidence variable types $\mathbb{E}^\kappa_1, \ldots, \mathbb{E}^\kappa_n$) to a set of probabilistic Austinian propositions that define a probability distribution over the values of the class variable type $\mathbb{C}^\kappa$, given probability distributions over the values of each evidence variable type $\mathbb{E}^\kappa_1, \ldots, \mathbb{E}^\kappa_n$. Formally, a ProbTTR Naïve Bayes classifier is a function κ of the type

(13) $(\mathbb{E}_1^\kappa \wedge \ldots \wedge \mathbb{E}_n^\kappa \rightarrow$
$\text{Set}(\begin{bmatrix} \text{sit} & : & \textit{Sit} \\ \text{sit-type} & : & \textit{Type} \\ \text{prob} & : & [0,1] \end{bmatrix})$

such that if[3] $s : \mathbb{E}_1^\kappa \wedge \ldots \wedge \mathbb{E}_n^\kappa$, then

(14) $\kappa(s) = \{ \begin{bmatrix} \text{sit} = s \\ \text{sit-type} = C \\ \text{prob} = p^\kappa(s : C) \end{bmatrix} \mid C \in \mathfrak{R}(\mathbb{C}^\kappa) \}$

where

(15) $p^\kappa(s : C) =$

$$\sum_{\substack{E_1 \in \mathfrak{R}(E_1^\kappa) \\ \ldots \\ E_n \in \mathfrak{R}(E_n^\kappa)}} p^\kappa(C||E_1 \wedge \ldots \wedge E_n) p(s : E_1) \ldots p(s : E_n)$$

($T_1 \wedge T_2$ is the *meet type* of T_1 and T_2. $a : T_1 \wedge T_2$ just in case $a : T_1$ and $a : T_2$.)

We are interested in the marginal probability $p^\kappa(s : C)$ of the situation s being of a class value type C in light of the evidence concerning s. As in the case of standard Bayesian Networks, we obtain the marginal probabilities of a class value type C by summing over all combinations of evidence value types. The classifier gives a probability distribution over the class value types.

Note that the probabilities associated with the evidence are probabilities that the situation s (the situation being classified) is of the various evidence value types. We do not assume that the evidence variables are known, only that we have a probability distribution over judgements of s being of the associated evidence value types. We also do not use the priors of the evidence value types here, as that would give us the marginal probability of *any* situation being of the class value type C, rather than the situation s being classified. Our ProbTTR notation allows us to make this distinction clear.

As above in (9), for the Naive Bayes classifier we estimate the conditional probability of the class given the evidence using the assumption that the evidence variable types are independent:

(16) $\hat{p}^\kappa(C||E_1 \wedge \ldots \wedge E_n) =$

$$\frac{p(C)p(E_1||C)\ldots p(E_n||C)}{\sum_{C' \in \mathfrak{R}(\mathbb{C}^\kappa)} p(C')p(E_1||C')\ldots p(E_n||C')}$$

6 Semantic Classification: Example

We will now illustrate classification in ProbTTR using a Naive Bayes classifier for fruits. We can imagine this classification taking place in the setting of an *Apple Recognition Game*. In this game a teacher shows a learning agent fruits (for simplicity, we assume there are only apples and pears in this instance of the game). The agent makes a guess, the teacher provides the correct answer, and the agent learns from these observations. (This paper describes only the classification step, leaving the learning step for future work.)

We will use shorthand for the types corresponding to an object being an apple vs. a pear

(17) a. *Apple* = $\begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{c}_{\text{apple}} & : & \text{apple(x)} \end{bmatrix}$

b. *Pear* = $\begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{c}_{\text{pear}} & : & \text{pear(x)} \end{bmatrix}$

We take it that the probability of judgements that something is of type *Ind* is always 1.0, and that

(18) $p(s : \begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{c} & : & \textit{T}(\text{x}) \end{bmatrix}) = p(s.c : T(s.x))$

so that e.g. if

(19) $s = \begin{bmatrix} \text{x} & = & \text{a} \\ \text{c} & = & \text{prf} \end{bmatrix}$,

then

(20) $p(s : \textit{Apple}) = p(\text{prf} : \text{apple(a)})$

.

Furthermore, we will assume that the objects in the Apple Recognition Game have one of two shapes (a-shape or p-shape) and one of two colours (green or red). We define shorthands for the record types involved.

(21) a. *Ashape* = $\begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{c} & : & \text{ashape(x)} \end{bmatrix}$

b. *Pshape* = $\begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{c} & : & \text{pshape(x)} \end{bmatrix}$

[3] Recall that that $\mathbb{E}_1^\kappa \ldots \mathbb{E}_n^\kappa$ are variable types and that for any variable type V and situation s, $p(s : V) \in \{0, 1.0\}$. Therefore, any type judgement regarding a variable type, such as that involved in the classifier function, can be regarded as categorical.

c. *Green* = $\begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{c} & : & \text{green(x)} \end{bmatrix}$

d. *Red* = $\begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{c} & : & \text{red(x)} \end{bmatrix}$

The class variable type is *Fruit*, with value types $\mathfrak{R}(\textit{Fruit}) = \{\textit{Apple}, \textit{Pear}\}$. The evidence variable types are (i) *Col*(our), with value types $\mathfrak{R}(\textit{Col}) = \{\textit{Green}, \textit{Red}\}$, and (ii) Shape, with value types $\mathfrak{R}(\textit{Shape}) = \{\textit{Ashape}, \textit{Pshape}\}$. Figure 3 shows the evidence and class types of the Apple Recognition Game in a simple Bayesian Network.

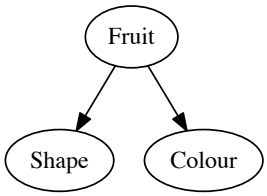

Figure 3: Bayesian Network for the Apple Recognition Game

For a situation s, the classifier FruitC(s) returns a set of probabilistic Austinian propositions asserting that s instantiates a certain type of fruit. This set is a probability distribution over the variable types of *Fruit*.

(22) FruitC(s) =
$$\{ \begin{bmatrix} \text{sit} = s \\ \text{sit-type} = F \\ \text{prob} = p^{\text{FruitC}}(s : F) \end{bmatrix} \mid F \in \mathfrak{R}(\textit{Fruit}) \}$$

We compute the probability of a classification in the Apple Recognition Game with the equation in (23), which is a special case of (15).

(23) for each $F \in \mathfrak{R}(\textit{Fruit})$, $p^{\textit{FruitC}}(s : F) =$

$$\sum_{\substack{L \in \mathfrak{R}(\textit{Col}) \\ S \in \mathfrak{R}(\textit{Shape})}} p(F||L \wedge S) p(s : L) p(s : S)$$

Therefore, to determine the probability that a situation is of the apple type, we sum over the various evidence type values for apple.

(24) $p^{\text{FruitC}}(s{:}\textit{Apple}) =$
$p(\textit{Apple}||\textit{Green} \wedge \textit{Ashape}) p(s{:}\textit{Green}) p(s{:}\textit{Ashape}) +$
$p(\textit{Apple}||\textit{Green} \wedge \textit{Pshape}) p(s{:}\textit{Green}) p(s{:}\textit{Pshape}) +$
$p(\textit{Apple}||\textit{Red} \wedge \textit{Ashape}) p(s{:}\textit{Red}) p(s{:}\textit{Ashape}) +$
$p(\textit{Apple}||\textit{Red} \wedge \textit{Pshape}) p(s{:}\textit{Red}) p(s{:}\textit{Pshape})$

Conditional probabilities for the fruit classifier are derived from previous judgements of the form $p(F||C \wedge S)$. The example values in the matrix in (25) illustrate a JPD for the Bayesian Network in Figure 3.

(25)

Apple/Pear	*Ashape*	*Pshape*
Green	0.93/0.07	0.63/0.37
Red	0.56/0.44	0.13/0.87

For each square with *Apple*/*Pear* type values, the conditional probabilities of the fruit being an apple and of its being a pear sum to 1. These probabilities are estimated using (16). For example:

(26) $\hat{p}(\textit{Apple}||\textit{Green} \wedge \textit{Ashape}) =$

$$\frac{p(\textit{Apple}) p(\textit{Green}||\textit{Apple}) p(\textit{Ashape}||\textit{Apple})}{\sum_{F' \in \{\textit{Apple}, \textit{Pear}\}} p(F') p(\textit{Green}||F') p(\textit{Ashape}||F')}$$

The non-conditional probabilities in (24) are derived from the agents' take on the particular situation being classified; let us call it s_5.

(27)

$T =$	*Ashape*	*Pshape*	*Green*	*Red*
p(s_5:T)	0.90	0.10	0.80	0.20

With these numbers in place, we can compute the probability that the fruit shown in $s_5$is an apple:

(28) $p^{\text{FruitC}}(s_5{:}\,\textit{Apple}) =$
$0.93 * 0.80 * 0.90 + 0.63 * 0.80 * 0.10 +$
$0.56 * 0.20 * 0.90 + 0.13 * 0.20 * 0.10 =$
$0.67 + 0.05 + 0.10 + 0.00 =$
0.82

In this section, we have shown how a Naive Bayes classifier, taking as input [1] judgements about how a situation s is classified with respect to a set of evidence value types, [2] conditional probabilities of some situation being of an evidence value type given that it is of a class value type, can be cast in ProbTTR.

7 Perceiving evidence

We might at this point ask, where do the non-conditional probabilities concerning the situation s being classified (exemplified in 27) come from? We suggest regarding these probabilities as resulting from probabilistic classification of real-valued (non-symbolic) visual input, where a classifier assigns to each image a probability that the image shows a situation of the respective type. Such a classifier can be implemented in a number of different ways, e.g. as a neural network, as long as it outputs a probability distribution.

Larsson (2015) shows how perceptual classification can be modelled in TTR, and Larsson (2020) reformulates and extends this formalisation to probabilistic classification. Adapting the notion of a probabilistic TTR classifier to the current setting, a probabilistic perceptual (here, visual) classifier corresponding to an evidence value type $E_i (1 \leq i \leq n)$ provides a mapping from perceptual input (of a type $\mathfrak{V}$, e.g. a digital image) onto a probability distribution over evidence value types in $\mathfrak{R}(E_i^\kappa)$, encoded as a set of probabilistic Austinian propositions:

(29) $\pi_{E_i^\kappa}$:$Sit_\mathfrak{V} \rightarrow$
$\{ \begin{bmatrix} \text{sit} : Sit_\mathfrak{V} \\ \text{sit-type} : RecType_R \\ \text{prob} : [0,1] \end{bmatrix} \mid R \in \mathfrak{R}(E_i^\kappa) \}$

where $Sit_\mathfrak{V}$ is the type of situations where perception of some object (labelled x) yields visual information (labelled c) concerning x:

(30) $Sit_\mathfrak{V} = \begin{bmatrix} \text{x} & : & Ind \\ \text{c} & : & \mathfrak{V} \end{bmatrix}$

and where $RecType_R$ is the (singleton) type of record types identical to R, so that e.g.

(31) T:$RecType_{Green}$ iff T:$RecType$ and $T = Green$

In the Apple game, an agent would be equipped with visual classifiers corresponding to *Shape* and *Col*, where e.g.

(32) $\pi_{Col} : \begin{bmatrix} \text{x} & : & Ind \\ \text{c} & : & \mathfrak{V} \end{bmatrix} \rightarrow$
$\{ \begin{bmatrix} \text{sit} : Sit_\mathfrak{V} \\ \text{sit-type} : RecType_{Green} \\ \text{prob} : [0,1] \end{bmatrix}, \begin{bmatrix} \text{sit} : Sit_\mathfrak{V} \\ \text{sit-type} : RecType_{Red} \\ \text{prob} : [0,1] \end{bmatrix} \}$

If we take s_5 to be e.g.

(33) $\begin{bmatrix} \text{x} & = & \text{a}_{453} \\ \text{c} & = & \text{Img}_{9876} \end{bmatrix}$

where

(34) a. a_{453}:*Ind*

b. Img_{9876}:$\mathfrak{V}$

and we assume that

(35) $\pi_{Col}(s_5)$=
$\{ \begin{bmatrix} \text{sit} = s_5 \\ \text{sit-type} = Green \\ \text{prob} = 0.8 \end{bmatrix}, \begin{bmatrix} \text{sit} = s_5 \\ \text{sit-type} = Red \\ \text{prob} = 0.2 \end{bmatrix} \}$

then (4) yields that

(36) a. p(s_5:*Green*)=0.8

b. p(s_5:*Red*)=0.2

which, incidentally, are the probabilities also shown in (27). This illustrates how ProbTTR allows combining probabilistic perceptual classification and probabilistic reasoning.

8 Bayesian networks in TTR

To extend the above to full Bayesian networks, we need to distinguish evidence variables from *unobserved variables*, and incorporate the latter into our classifier. A TTR Bayes net classifier is associated with

- $\mathbb{E}_1^\kappa, \ldots, \mathbb{E}_n^\kappa$ is a collection of evidence variable types,
- $\mathfrak{R}(\mathbb{E}_1^\kappa), \ldots, \mathfrak{R}(\mathbb{E}_n^\kappa)$ are sets of evidence value types,
- $\mathbb{I}_1^\kappa, \ldots, \mathbb{I}_m^\kappa$ is a collection of unobserved variable types,
- $\mathfrak{R}(\mathbb{I}_1^\kappa), \ldots, \mathfrak{R}(\mathbb{I}_m^\kappa)$ are sets of unobserved value types.

Given this, a TTR Bayes net classifier is a function κ of type

(37) $\mathbb{E}_1^\kappa \wedge \ldots \wedge \mathbb{E}_n^\kappa \rightarrow \text{Set}(\begin{bmatrix} \text{sit} & : & Sit \\ \text{sit-type} & : & Type \\ \text{prob} & : & [0,1] \end{bmatrix})$

such that if $s : \mathbb{E}_1^\kappa \wedge \ldots \wedge \mathbb{E}_n^\kappa$ and $1 \leq j \leq m$, then

(38) $\kappa(s) = \{ \begin{bmatrix} \text{sit} = s \\ \text{sit-type} = I_j \\ \text{prob} = p^\kappa(s : I_j) \end{bmatrix} \mid I_j \in \mathfrak{R}(\mathbb{I}_j^\kappa) \}$

$$p^{\kappa}(s : I_j) = \sum_{\substack{I_1 \in \mathfrak{R}(\mathbb{I}_1^{\kappa}) \\ I_{j-1} \in \ddot{\mathfrak{R}}(\mathbb{I}_{j-1}^{\kappa}) \\ I_{j+1} \in \mathfrak{R}(\mathbb{I}_{j+1}^{\kappa}) \\ I_m \in \ddot{\mathfrak{R}}(\mathbb{I}_m^{\kappa}) \\ E_1 \in \mathfrak{R}(\mathbb{E}_1^{\kappa}) \\ E_n \in \ddot{\mathfrak{R}}(\mathbb{E}_n^{\kappa})}} p(I_j || I_1 \wedge \ldots \wedge I_{j-1} \wedge I_{j+1} \wedge \ldots \wedge I_m \wedge E_1 \wedge \ldots \wedge E_n) p(s : E_1) \ldots p(s : E_n)$$

Figure 4: A TTR Bayes net classifier

where $p^{\kappa}(s : I_j)$ is as in Figure 4.

The dependencies encoded in a Bayes net will affect how the conditional probability $p(C||I_1 \wedge \ldots I_{j-1} \wedge I_{j+1} \wedge I_m \wedge E_1 \wedge \ldots \wedge E_n)$ is computed. In the sprinkler example, we have three unobserved variable types *Grass*, *Sprinkler* and *Rain*, and one evidence variable type *Cloudy*. For $S \in \mathfrak{R}(\textit{Sprinkler}), R \in \mathfrak{R}(\textit{Rain}), L \in \mathfrak{R}(\textit{Cloudy})$ and $G \in \mathfrak{R}(\textit{Grass})$, the dependencies encoded in the Bayesian network in Figure 1 entail that

(39) $p(G||S \wedge R \wedge L) =$

$$p(G||S \wedge R) p(S||L) p(R||L)$$

and hence for $G \in \mathfrak{R}(\textit{Grass})$,

(40) $p^{\kappa}(s : G) =$

$$\sum_{\substack{S \in \mathfrak{R}(\textit{Sprinkler}) \\ R \in \mathfrak{R}(\textit{Raining}) \\ L \in \mathfrak{R}(\textit{Cloudy})}} p(G||S \wedge R) p(S||L) p(R||L) p(s : L)$$

9 Semantic learning

A central question is, of course, how we get the conditional and prior probabilities used for classification. This is the role of the semantic learning component. For a ProbTTR classifier, the learning component needs to estimate the probabilities required for computing $p(C||E_1 \wedge \ldots \wedge E_n)$.

In Cooper et al. (2015) a solution is sketched, based on the idea that an agent makes judgements based on a finite string of probabilistic Austinian propositions, the *judgement history* $\mathfrak{J}$. When an agent A encounters a new situation s and wants to know if it is of type T or not, A uses probabilistic reasoning to determine $p(s : T)$ on the basis of A's previous judgements $\mathfrak{J}$. We are currently working on casting a couple of learning theories in ProbTTR, and this will be reported in future work.

10 Conclusions

Cooper et al. (2014) and Cooper et al. (2015) presented a probabilistic formulation of a rich type theory with records, and used it as the foundation for a compositional semantics in which a probabilistic judgement that a situation is of a certain type plays a central role. The basic types and type judgements at the foundation of the type system correspond to perceptual judgements concerning objects and events in the world, rather than to entities in a model, and set theoretic constructions defined on them. This approach grounds meaning in observational judgements concerning the likelihood of situations holding in the world.

Here, we have proposed a Bayesian account of semantic classification and inference formulated in terms of probabilistic type theory. We have suggested probabilistic type theoretic formulations of Naive Bayes Classifiers and Bayesian Networks. A central element of these constructions is a ProbTTR version of a random variable.

Future work includes applying Bayesian inference and classification in ProbTTR to a variety of problems in natural language semantics, including vagueness (where some initial steps are taken in Fernández and Larsson (2014)), probabilistic reasoning in dialogue, and learning grounded meanings from interaction (along the lines of Larsson (2013)). We will also implement this integrated system in order to demonstrate its viability as a computational model of natural language learning, reasoning and interaction.

Acknowledgments

This work was supported by Swedish Research Council grants 2014-39, Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg, and 2016-01162, Incremental Reasoning in Dialogue.

References

Jon Barwise. 1989. *The Situation in Logic*. CSLI Publications, Stanford.

Ellen Breitholtz. 2020. *Enthymemes and Topoi in Dialogue: The Use of Common Sense Reasoning in Conversation*. Brill, Leiden, The Netherlands.

Robin Cooper. 2005. Records and record types in semantic theory. *Journal of Logic and Computation*, 15(2):99–112.

Robin Cooper. 2012. Type theory and semantics in flux. In Ruth Kempson, Nicholas Asher, and Tim Fernando, editors, *Handbook of the Philosophy of Science*, volume 14: Philosophy of Linguistics. Elsevier BV. General editors: Dov M. Gabbay, Paul Thagard and John Woods.

Robin Cooper. in prep. From perception to communication: An analysis of meaning and action using a theory of types with records (TTR). Draft available from https://sites.google.com/site/typetheorywithrecords/drafts.

Robin Cooper, Simon Dobnik, Shalom Lappin, and Staffan Larsson. 2014. A probabilistic rich type theory for semantic interpretation. In *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, pages 72–79. Gothenburg, Association of Computational Linguistics.

Robin Cooper, Simon Dobnik, Shalom Lappin, and Staffan Larsson. 2015. Probabilistic type theory and natural language semantics. *Linguistic Issues in Language Technology 10*, pages 1–43.

Robin Cooper and Jonathan Ginzburg. 2015. Type theory with records for natural language semantics. In Shalom Lappin and Chris Fox, editors, *The Handbook of Contemporary Semantic Theory, Second Edition*, pages 375–407. Wiley-Blackwell, Oxford and Malden.

Robin Cooper and Staffan Larsson. 2009. Compositional and ontological semantics in learning from corrective feedback and explicit definition. In *Proceedings of DiaHolmia: 2009 Workshop on the Semantics and Pragmatics of Dialogue*, pages 59–66. Department of Speech, Music and Hearing, KTH.

Simon Dobnik, Robin Cooper, and Staffan Larsson. 2013. Modelling language, action, and perception in Type Theory with Records. In Denys Duchier and Yannick Parmentier, editors, *Constraint Solving and Language Processing - 7th International Workshop on Constraint Solving and Language Processing, CSLP 2012, Orleans, France, September 13-14, 2012. Revised Selected Papers*, number 8114 in Publications on Logic, Language and Information (FoLLI). Springer, Berlin, Heidelberg.

Raquel Fernández and Staffan Larsson. 2014. Vagueness and learning: A type-theoretic approach. In *Proceedings of the 3rd Joint Conference on Lexical and Computational Semantics (∗SEM 2014)*.

Jonathan Ginzburg. 2012. *The Interactive Stance: Meaning for Conversation*. Oxford University Press, Oxford.

J. Halpern. 2003. *Reasoning About Uncertainty*. MIT Press, Cambridge MA.

Staffan Larsson. 2013. Formal semantics for perceptual classification. *Journal of Logic and Computation*.

Staffan Larsson. 2015. Formal semantics for perceptual classification. *Journal of Logic and Computation*, 25(2):335–369. Published online 2013-12-18.

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

Staffan Larsson and Robin Cooper. 2009. Towards a formal view of corrective feedback. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, pages 1–9. EACL.

J. Pearl. 1990. Bayesian decision methods. In G. Shafer and J. Pearl, editors, *Readings in Uncertain Reasoning*, pages 345–352. Morgan Kaufmann.

Stuart Russell and Peter Norvig. 1995. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Englewood Cliffs, New Jersey.

From compositional semantics to Bayesian pragmatics via logical inference

Julian Grove **Jean-Philippe Bernardy** **Stergios Chatzikyriakidis**
Centre for Linguistic Theory and Studies in Probability
Department of Philosophy, Linguistics and Theory of Science
University of Gothenburg
firstname.lastname@gu.se

Abstract

Formal semantics in the Montagovian tradition provides precise meaning characterisations, but usually without a formal theory of the pragmatics of contextual parameters and their sensitivity to background knowledge. Meanwhile, formal pragmatic theories make explicit predictions about meaning in context, but generally without a well-defined compositional semantics. We propose a combined framework for the semantic and pragmatic interpretation of sentences in the face of probabilistic knowledge. We do so by (1) extending a Montagovian interpretation scheme to generate a distribution over possible meanings, and (2) generating a posterior for this distribution using a variant of the Rational Speech Act (RSA) models, but generalised to arbitrary propositions. These aspects of our framework are tied together by evaluating entailment under probabilistic uncertainty.[1]

We apply our model to anaphora resolution and show that it provides expected biases under suitable assumptions about the distributions of lexical and world-knowledge. Further, we observe that the model's output is robust to variations in its parameters within reasonable ranges.

1 Introduction

A goal of much work in computational semantics is to determine how responsibility should be apportioned between discrete, logical techniques and stochastic, probabilistic ones in explanations of inference. A current tradition that has roots in symbolic AI leverages the power of theorem provers to model inference in corpora, oftentimes grappling with both deductive and abductive modes of reasoning (Blackburn and Bos, 2005; Bos and Markert, 2005; Raina et al., 2005; van Eijck and Unger, 2010; Abzianidze, 2015; Emerson and Copestake, 2017a,b; Abzianidze, 2020, i.a.). Such approaches, while explicitly compositional, often attempt to combine both semantic and pragmatic meaning into a single inferential module, with the goal of capturing naturally occurring patterns.

Simultaneously (in the last decade), Rational Speech Act (RSA) models have provided a promising avenue for integrating logical and probabilistic approaches to meaning by modelling utterance interpretation as a process of updating probability distributions over logically characterised meanings (Goodman and Stuhlmüller, 2013; Lassiter and Goodman, 2013; Goodman and Frank, 2016; Lassiter and Goodman, 2017, i.a.). According to the RSA perspective, interpreting an utterance involves reasoning pragmatically about a speaker's intended message according to Bayesian principles of belief update. The reasoning of rational conversation participants, moreover, reflects principles of cooperative communication according to which speakers make true and informative utterances. Thus such models aim to capture a central feature of rational discourse known since the work of Grice (1975): that it is constrained by principles of appropriate social behaviour, which, through the reasoning of interlocutors, serve to enrich the very meanings which are communicated.

The goal of the current work is to integrate these two approaches to meaning and inference by using, on the one hand, a theorem prover to reason about compositionally derived semantic meanings and, on the other hand, Bayesian inference, as applied within the RSA framework, to give a computational account of pragmatic reasoning in discourse. Our contribution is thus to tie work in the logical tradition into a successful probabilistic framework for pragmatic reasoning. While logical entailment is at the core of evaluating truth values, we use probabilistic reasoning to deal with epistemic un-

[1] The code for this paper is available on GitHub at: https://github.com/juliangrove/grove-bernardy-chatzikyriakidis-naloma2021

Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA), pages 60–70

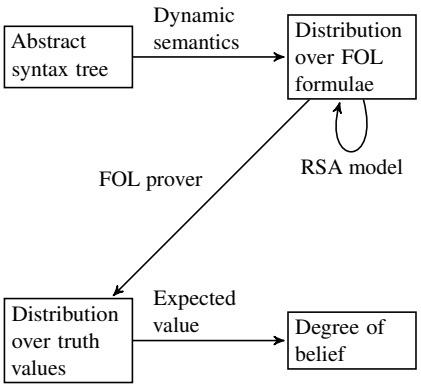

Figure 1: Phases of our system. Syntax is first interpreted into a distribution over FOL formulae. The truth values of such formulae can then be extracted using a theorem prover, allowing an expected value for the resulting distribution to be computed. The RSA model acts on the distribution over FOL formulae. This refinement step may itself invoke the theorem prover and distributions over truth values (we omit these dependencies to avoid clutter).

certainty. As such, this paper contributes a hybrid logical/probabilistic semantics consisting of both a standard Montagovian compositional scheme and an RSA model.

We pay special attention to the resolution of linguistic ambiguity in discourse—in particular, anaphora—as a test-case for our approach. By considering anaphora resolution as a Bayesian inference problem, we show how both prior world and lexical knowledge may influence the choice of antecedent for a given pronoun. Moreover, because our computational implementation combines a probabilistic approach to *inference* with a compositional, logical approach to *meaning* in the tradition of Montague (1973), i.e., by integrating numeric computation and theorem proving, we are able to explicitly and robustly characterise the contribution of conventional meaning to the task of pragmatic inference, as well as how the latter serves to modulate uncertainty about the former. We illustrate our approach on two test cases which differ in the priors they involve, thus demonstrating the importance of background knowledge to the behaviour of our model.

2 The framework

2.1 Compositional semantics under ambiguity

The goal of Montagovian compositional semantics is to map syntactic representations of an utterance u onto a meaning in some predefined domain. Typically, such meanings are propositions (for some logical system, like first order logic or type theory). We can write '$\phi = [\![u]\!]$' to represent such a mapping. However, there are, in general, several ways to map utterances to propositions, due to semantic ambiguity; thus our compositional semantics should instead produce a *distribution* of propositions.

The structure of our framework is illustrated in Figure 1. The first step in mapping an utterance to a pragmatically enriched meaning involves taking that utterance onto a probability distribution over expressions in some metalanguage: those which represent the dynamic semantic meaning of the utterance. As will become clear, any metalanguage for which one may define some computable notion of entailment suffices. For our implementation, we choose standard first order logic, so that utterances are mapped to distributions over FOL formulae.

Generalising the work of Lassiter and Goodman (2013), we may formalise this distribution in terms of the equation $\phi = [\![u]\!]^\theta$, where θ is a set of random variables, each having some *a priori* initial distribution. One can understand the above equation as invoking an interpretation function, $[\![\cdot]\!]$, defined inductively on expressions, and depending on the set θ of parameters whose role is to select among possible interpretations, as illustrated by the following scheme for Functional Application.

$$\frac{f = [\![np]\!]^\theta \quad x = [\![vp]\!]^\theta}{f(x) = [\![np\ vp]\!]^\theta}$$

In this way, a compositional semantics simultaneously produces distributions for ϕ and the parameters in θ.

2.2 Reasoning under probabilistic knowledge

One can evaluate the truth value of a proposition, given some background context, by evaluating its provability under a system of deduction (the system in question representing the reasoning capabilities of agents). We represent background knowledge as a distribution over sets of FOL formulae Γ, each of which may be regarded as representing a world-state; that is, a way things might be. We can then evaluate the truth value of ϕ, given some fixed world-state (i.e., set of hypotheses) Γ, as '$[\Gamma \vdash \phi]$', that is, 1 if $\Gamma \vdash \phi$ holds logically, and 0, otherwise. Even though entailment in many logical systems is undecidable, we may circumvent this issue, for

example, by limiting them to a certain depth of deduction, perhaps modelling finite reasoning capabilities. In our implementation, we use a regular FOL tableau prover limited to depth 10; this way, we can work with any set of propositions expressible in FOL, and in particular, those delivered by a Montagovian interpretation procedure. Calculating entailment constitutes the second step in our framework; it takes us from a probability distribution over formulae ϕ to a probability distribution over truth values reflecting whether ϕ holds at a world-state Γ, given an initial probability distribution over world-states.

Hereafter, we let Γ be a random variable ranging over sets of FOL formulae, whose distribution represents epistemic uncertainty of an agent about background knowledge, i.e., the actual world-state. Such a formulation of uncertainty is very flexible. For example, uncertainty about John's height can be represented as Γ = {John's height is H}, where H is a random variable with a normal distribution of mean 1.8 meters and standard deviation 0.05 meters. Discrete uncertainty may be represented using Bernoulli distributions. If b is a Boolean variable with a Bernoulli distribution, uncertainty about weather conditions can be represented as follows: Γ = if b then {it will rain tomorrow} else {it will not rain tomorrow}. Given a set of propositions $\Psi = \{\psi_1, \ldots, \psi_n\}$, each true or false according to one of a sequence of Bernoulli random variables $\gamma = b_1, \ldots, b_n$, we may take Γ to be equal to $\gamma\Psi$, i.e., the set containing either ψ_i or its negation $\neg\psi_i$, as according to whether b_i is True or False.

Given this setup, we can define a notion of 'expected truth value', which we encode as a real number between 0 and 1. We notate the truth value of ϕ, given some set of background hypotheses Γ, as '$[\Gamma \vdash \phi]$' and thus denote the expected truth value of ϕ, which takes into account the distribution associated with Γ, '$\mathbb{E}_\Gamma[\Gamma \vdash \phi]$'. As discussed in the next section, we will more often invoke the probability of non-entailment, given as $\mathbb{E}_\Gamma[\Gamma \nvdash \phi]$, and which is equal to $1 - \mathbb{E}_\Gamma[\Gamma \vdash \phi]$.

In general, we compute probability distributions over formulae, world-states, and truth values compositionally, in terms of probabilistic programs. A probabilistic program that returns a value of type α is a function of type $(\alpha \to \mathbb{R}) \to \mathbb{R}$; that is, one which consumes probability density functions (PDFs), i.e., from values of type α to real numbers, in order to derive a real number. For example, a probabilistic program that returns values of type α from some finite list l with a uniform distribution is the function $\lambda f.\texttt{sum}(\texttt{map}\, f\, l)/(\texttt{length}\, l)$. Given a PDF f, this program computes its sum across the members of l and divides the result by the length of l, thus returning the mean. If α is itself $\mathbb{R}$, the program may be used to compute an expected value by simply feeding it the identity function.

Crucially, probabilistic programs may be *composed*: given a probabilistic program m returning values of type $\alpha \to \beta$ and a probabilistic program n returning values of type α, a new program returning values of type β can be derived as $\lambda k.m(\lambda f.n(\lambda x.k(fx)))$. Such a composition scheme may appear familiar to many as applicative composition in the continuation monad. Indeed, probabilistic programs are composed by passing their input PDFs as continuations. More generally, complex probabilistic programs are easy to write and compose in monadic style, thus allowing us to keep our implementation pure and squarely within the simply typed λ-calculus. This approach, importantly, sets our framework apart from previous attempts at integrating natural language semantics with probabilistic computation, e.g., Goodman and Lassiter (2015).

2.3 RSA

We first review the general assumptions of the RSA model, and then we present the particular variant of RSA that we use in this paper. We discuss the differences between our presentation and that of the original RSA model of Lassiter and Goodman (2013) in §6.2.

RSA assumes two agents, a listener L and a speaker S. S utters a declarative sentence u heard by L, without transmission error. The point of RSA is to model how, assuming Gricean cooperativeness between S and L, L should disambiguate among possible interpretations of u.

Our model is defined by the following relations:

$$P_{L_1}(\phi \mid u) \propto P_{S_1}(u \mid \phi) \times P(\phi)$$
$$P_{S_1}(u \mid \phi) \propto (P_{L_0}(\phi \mid u)/C(u))^\alpha$$
$$P_{L_0}(\phi \mid u) = \mathbb{E}_{\theta,\Gamma}[\Gamma, \phi, [\![u]\!]^\theta \nvdash \bot]$$

In the above, relations for P_{L_1} and P_{L_0} represent listener models. Their primary function is to yield a distribution over interpretations of a given utterance u as propositions ϕ. $P_{L_1}(\phi|u)$ corresponds to a Bayesian update to the probability of the proposition ϕ, given an observation of the utterance u.

Following Bayes' theorem, this conditional probability is determined by multiplying a *likelihood*, $P_{S_1}(u \mid \phi)$, by a *prior*, $P(\phi)$.

The likelihood is the probability that the pragmatic listener S will utter u, given an intention to communicate the proposition ϕ. In other words, the output of the model P_{S_1} is an estimate of the probability of S uttering u if S *means* ϕ. This estimate is, in turn, obtained by considering utterances u in proportion to how likely they are to skew the (literal) listener towards interpreting u as ϕ, while taking into account an intrinsic utterance cost $C(u)$. Furthermore, an exponent α is applied to model the tendency of S to behave rationally, i.e., by choosing utterances in view of L_0's tendencies in conjunction with utterance cost.

The prior probability of the proposition ϕ is determined, in part, by the distribution over θ and, in part, by the distribution over prior knowledge Γ. Thus we have that $P(\phi) \propto \mathbb{E}_\Gamma[\Gamma, \phi \not\vdash \bot] * P(\theta)$, where $\phi = [\![u]\!]^\theta$ (for some θ), and $P(\phi) = 0$, otherwise. Priors are thus assessed using a *non-contradiction* model of interpretation: intuitively, Γ describes a world state—a way things could be—and ϕ is accepted if it is compatible with the world-state Γ.

The literal listener L_0 similarly uses a non-contradiction model of interpretation. It rejects interpretations ϕ incompatible with the utterance, in proportion to the *a priori* distribution of meanings for u, namely $[\![u]\!]^\theta$.

A final point deserving mention is that, in general, the priors over Γ and θ need not be the same in P_{L_1} and P_{L_0}. In P_{L_1}, they are L's actual priors, while in P_{L_0} they are those that L believes that S believes L has. In what follows, we consider only those priors which constitute common ground knowledge, i.e., in which case they are equal.

3 Anaphora resolution as a case study

To apply the above theory to anaphora resolution, we let θ be a set of parameters that determine the mapping of anaphoric expressions to antecedents. (In our experiments, we will consider only pronominal anaphora.)

For example, if $u =$ 'he runs', then (singleton) θ could be taken in the set $\Theta = \{John, Bill, Bob\}$, if those three antecedents are available in the discourse context. The logical representation of the utterance is then $[\![u]\!]^\theta = run(\theta)$. The factor $P(\theta)$ might be used to give lower probabilities to antecedents further back in the discourse; a value for this prior might be estimated from psycholinguistic experimentation. For the sake of simplicity, we let $P(\theta)$ be uniform across Θ in what follows.

Alternative utterances Within the RSA framework, P_{S_1} gives the distribution over alternative utterances considered by the speaker to express ϕ. The set underlying this distribution, moreover, must be supplied by the modeller *a priori*. We determine this set as follows. First, the utterance observed by L is itself in this set. Moreover, for any utterance u in the set, and for any anaphor x present in u, we include in the set the alternative utterance u' just like u, but in which the anaphor is substituted by a noun phrase denoting the antecedent actually meant by S (given S's intention to communicate ϕ). That is, u' is less ambiguous than u. For example, when evaluating $P(\text{'he runs'} | run(bill))$, S considers both 'he runs' and 'Bill runs'. This set is important because it is used to normalise S's distribution over utterances:

$$P_{S_1}(u|\phi) = \frac{(P_{L_0}(\phi|u)/C(u))^\alpha}{\sum_{u'}(P_{L_0}(\phi|u')/C(u'))^\alpha}$$

Background knowledge In our experiments, we let Γ be governed by a finite sequence of Boolean variables $b_1, \ldots, b_n$ drawn from Bernoulli distributions. Concretely, we work with a set of potential propositions $\{\psi_1, \ldots, \psi_n\}$ and write '$\ldots b_i \ldots \{\ldots \psi_i \ldots\}$' to denote the set containing ψ_i if b_i is True and $\neg\psi_i$ if b_i is False. The set of propositions in Γ and the parameters of their associated Bernoulli distributions vary from example to example.

The model then predicts a posterior distribution over mappings θ from anaphora to antecedents, along with a corresponding posterior distribution over meanings ϕ. As a result, one may also obtain a posterior distribution over Boolean variables b_i representing the common ground, now updated with ϕ.

4 Examples

We provide two examples to illustrate our model and, in particular, the effect of prior knowledge on its behaviour. Our first example is (1).

(1) Emacs is waiting for the command. It is prepared.

Here, the noun phrases *Emacs* and *the command* are in competition as potential antecedents for the

pronoun *it*.[2] Intuitively, the most likely antecedent for the pronoun is *Emacs*, which we take to be due (at least in part) to the fact that the verbs *waiting* and *prepared* lexically entail that their subjects are animate.[3] Thus a rational listener who infers that the antecedent for *it* in (1) is *Emacs* is doing so (at least in part) on the basis of the following reasoning: because animacy is entailed of the pronoun in virtue of its role as subject of the verb *prepared*, it is more likely, all else being equal, to co-refer with *Emacs*, which is also entailed to be animate (in virtue of being the subject of the verb *waiting*), than *the command*, which is subject to no such entailment. Such an inference is thus obtained on the basis of abductive reasoning about the source of the animacy of the pronoun.

The availability of this reasoning in (1) contrasts with its relative unavailibility in the second example in (2).

(2) Ashley is waiting for Amy. She sees her.

In contrast to *Emacs* and *the command*, proper names referring to humans, like *Ashley* and *Amy*, are very likely to denote animate individuals. As such, their prior probability of being animate will be higher than that of the noun phrases in (1), and the animacy entailment contributed by the verb *waiting* will therefore provide less of a basis for using animacy as a cue to distinguish potential antecedents for the pronouns. With the impact of animacy attenuated in (2), the candidate antecedents for the subject pronoun should be in closer competition, and anaphora resolution should be less certain. Intuitively, this seems to be the case: it appears more difficult to determine the referent of the subject pronoun in (2) than in (1) (though experimental investigation would be required to confirm this intuition).

We can model the difference between these examples by assuming different priors for the animacy of the referents of noun phrases like *Emacs* and *the command*, on the one hand, and *Ashley* and *Amy*, on the other. In our model, we encode such priors by associating probabilities with sentences translated into first order formulae; each such formula ψ is then associated with an independent Bernoulli random variable b in the definition of a probabilistic program that returns a world-state consisting of a set of hypotheses encoded as logical formulae. That is, such a set contains ψ if b is True, and it contains $\neg\psi$ if b is False.

$animate(emacs)$	0.2
$animate(the\_command)$	0.2
$animate(ashley)$	0.9
$animate(amy)$	0.9

As the table shows, we model world knowledge as dictating that the referents of *Emacs* and *the command* are only 20% likely to be animate, while individuals such as Ashley and Amy are 90% likely to be animate. Though these priors are somewhat arbitrary, they are meant to reflect qualitative differences in the knowledge we have about noun phrases referring to humans and those referring to other objects.

In addition to the priors listed above, we include priors for the truth of the following formulae, which, in each case, we take to be 0.05.

$$\exists x : wait\_for(emacs, x)$$
$$\exists x : wait\_for(the\_command, x)$$
$$\exists x : wait\_for(ashley, x)$$
$$\exists x : wait\_for(amy, x)$$
$$prepared(emacs)$$
$$prepared(the\_command)$$
$$\exists x : see(ashley, x)$$
$$\exists x : see(amy, x)$$

Finally, we encode the lexical entailments of the verbs *waiting*, *prepared*, and *sees* in terms of the following formulae:

$$\forall x : (\exists y : wait\_for(x, y)) \rightarrow animate(x)$$
$$\forall x : prepared(x) \rightarrow animate(x)$$
$$\forall x : (\exists y : see(x, y)) \rightarrow animate(x)$$

In our model, these formulae act as filters of background knowledge: any world-state that contradicts them is given probability 0, and the probability distribution over world-states is re-normalised. As a result, the Bernoulli random variables associated with individual hypotheses in the final model of background knowledge will not be entirely independent.

5 Results and analysis

To illustrate the model's performance, we give results for the examples discussed in the previous section, fixing values for parameters in the speaker model; in particular, the exponent α, as well as the log-cost associated with an utterance that uses either a pronoun or a full noun phrase to refer to a

[2] This example comes from Lappin and Leass (1994), who resolve anaphora on the basis of a number syntactic and semantic heuristics, with no specific pragmatic analysis.

[3] These inferences may, in fact, be presuppositions, a point we gloss over here.

given antecedent. Table 1 provides the model's calculations of the pragmatic listener's bias to choose *Emacs* (as opposed to *the command*) as the antecedent of the subject pronoun of (1), across two values of α and two sets of values for log-cost. Log-costs for pronouns (PN) and full noun phrases (NP) are summed, for any given utterance, to provide its total log-cost. For example, if the log-cost of a pronoun is 1, and that of a full noun phrase is 2 (as in the models reported in rows 2 and 4), then an utterance with one pronoun and one noun phrase will have a total log-cost of 3, and the probability P_{L_0} is scaled by a factor of $e^{-3\alpha}$ in the calculation of P_{S_1}.

α	PN	NP	*Emacs* bias
0.5	0	0	87.9%
0.5	1	2	86.9%
4.0	0	0	99.9%
4.0	1	2	98.6%

Table 1: Example (1)

The results of Table 1 highlight three notable features of our model. First, anaphora resolution displays the expected bias, based on the prior world and lexical knowledge governing inference. In particular, lexical knowledge associated with the verbs *waiting* and *prepared* determines that their subjects be animate; thus the pragmatic listener performs a kind of abductive inference, based on these entailments: a pronoun which is entailed to be animate displays a high probability of seeking animacy in its antecedent. Comparison with the results for (2) (which we discuss next) illustrates the importance of the low animacy priors (0.2) for the antecedents in achieving pragmatic reasoning of this kind.

Second, even though high values of α increase the bias in favour of *Emacs* (as expected), the model is not very sensitive to its precise choice. As α approaches 0, the speaker model approaches a uniform distribution over utterances, but even as low a value as 0.5 yields sensible results.

Third, incorporating a measure of cost into the reasoning of the pragmatic speaker has a dampening effect on the model's bias, as can be seen by comparing rows 1 and 2, as well as rows 3 and 4. This effect consists in about 1% of difference, and it is due to the fact that making reference to cost has the pragmatic listener reason about a "lazier" pragmatic speaker; such a speaker, who finds full noun phrases costlier to utter than pronouns, will more likely choose a pronoun to *minimise their effort*, rather than as a result of their reasoning about a literal listener who will choose the expected antecedent for the pronoun.

Table 2 provides the model's calculations of the pragmatic listener's bias to choose *Ashley* (as opposed to *Amy*) as the antecedent for both the subject pronoun *she* and the object pronoun *her* in (2). We show results for the same values of α and log-cost.

α	PN	NP	*Ashley* bias	
			for *she*	for *her*
0.5	0	0	53.0%	50%
0.5	1	2	52.9%	50%
4.0	0	0	60.7%	50%
4.0	1	2	54.2%	50%

Table 2: Example (2)

We note, first, that the same general patterns across values of α and log-cost obtain for this example as for the previous one: higher values of α exaggerate the pragmatic listener's bias, while increasing noun phrase cost relative to pronoun cost dampens it.

Second, comparing the results of this model with those for (1) demonstrates clearly the effect of prior knowledge on the model's behaviour. Because the antecedents have high animacy priors (0.9), the animacy entailment of the verb *waiting* provides less of a basis for distinguishing them; as a result, they are in closer competition as antecedents for the subject pronoun, which is entailed to be animate, and bias toward the subject antecedent is greatly reduced (though still present).

Last, we note that the object pronoun is exactly split in its probability of taking *Ashley* versus *Amy* as its antecedent in (2). Because there is no animacy entailment from the verb for the object pronoun, the pragmatic listener has no basis for distinguishing the antecedents, e.g., through abductive inference. This result supports our explanation for the biases displayed in the other cases.

6 Related work

The work presented in this paper is related to a number of attempts in both the formal and computational semantics communities to bridge logical and probabilistic approaches to natural language semantics. These approaches, in addition to their formal differences, can be categorised into those which have been computationally implemented and

those which have not. In the first category, one finds approaches such as Beltagy et al. (2013); Goodman and Stuhlmüller (2013); Goodman and Frank (2016); Lassiter and Goodman (2013, 2017); Bernardy et al. (2018); Emerson and Copestake (2017b) , while in the latter category, those such as van Eijck and Lappin (2012); Cooper et al. (2015); Sutton (2018).

A common theme among probabilistic approaches to interpretation is that they describe a set of possible world-states as a distribution. Predicates are then evaluated at each world-state, and probabilistic truth is the expected value over all possible world-states. In implemented accounts, one often uses Monte Carlo sampling methods to estimate truth values. We refrain from a further comparison with approaches lacking a computational implementation: even though they contain fruitful ideas, it is unclear how they should be realised computationally.

Another way to classify approaches is by the representation of world-states that they employ. Goodman and Stuhlmüller (2013); Goodman and Frank (2016); Lassiter and Goodman (2013) use an *ad hoc* set of variables, chosen according to the problem at hand. Bernardy et al. (2018) use vector representations inspired by machine-learning approaches. Bernardy et al. (2019b) present a system that tries to minimise (and in cases, eliminate) the need for sampling by modelling predicates as (the unions of) boxes and individuals as points.

A unique characteristic of the present account is our use of a small number of Bernoulli random variables to represent world-states, where each variable captures the applicability of a proposition. This choice is afforded by the use of logical entailment as the basis of evaluating truth values. Together, this means that we can provide exact calculations for truth values, i.e., by taking the average over finite probability distributions. An additional benefit of using the knowledge-as-propositions approach is that we have all the expressivity of the underlying logic at our disposal. Hence, we have no difficulty dealing with predicates with multiple arguments, contrary to Bernardy et al. (2019b, 2018). Even though weighted formulae can be interpreted as possible world-states via a Markov Logic Networks (Domingos and Lowd, 2009), as Beltagy et al. (2013) showed for natural language semantics, our simpler approach is sufficient for our purposes.

6.1 Logical approaches to semantic inference

Our framework aspires to connect two traditions in the study and computational implementation of semantics: logical, compositional semantics on the one hand, and Bayesian pragmatics, on the other. This connection is achieved by reasoning about propositional entailment via theorem proving, while modelling pragmatic inference as Bayesian reasoning, using a variant of RSA. Thus there are important connections to other approaches to semantics and natural language inference that rely on a compositional semantics to translate abstract syntax trees into logical formulae and then evaluate inference patterns via theorem proving (Bos and Markert, 2005; Mineshima et al., 2015; Abzianidze, 2015; Bernardy and Chatzikyriakidis, 2017, 2019, 2021). These accounts vary in their details; for example, in the type of parser used: Bos and Markert (2005); Mineshima et al. (2015); Abzianidze (2015) use variants of CCG parsers, while Bernardy and Chatzikyriakidis (2017, 2019) use the GF parser (Ranta, 2011). They also vary in the types of meaning representations they employ, as well as in the underlying logical systems they use (e.g., first order versus higher order). Finally, they differ in their choice of theorem provers, and whether they are automated or interactive. But the connections between such approaches and ours are clear: all employ a compositional semantics to generate logical formulae, which are further reasoned about with theorem provers. A crucial difference between our approach and the aforementioned ones, however, is that ours supports a designated pragmatic module that accomplishes pragmatic inference with Bayesian reasoning. Thus our framework may be seen as involving a pragmatic enrichment of a logical component, afforded by Bayesian reasoning in the guise of RSA. Finally, despite the fact that our account follows previous work in the RSA tradition (Goodman and Stuhlmüller, 2013; Goodman and Frank, 2016; Lassiter and Goodman, 2013, 2017), it employs a couple of different assumptions than usual—a point to which we now turn.

6.2 The relation between our model and standard RSA

Lassiter and Goodman (2013) give an RSA model governed by the following relations (modulo re-

naming of some parameters):

$$P_{L_1}(\Gamma, \theta \mid u) \propto P_{S_1}(u \mid \Gamma, \theta) \times P_{L_1}(\Gamma)$$
$$P_{S_1}(u \mid \Gamma, \theta) \propto (P_{L_0}(\Gamma \mid u, \theta)/C(u))^{\alpha}$$
$$P_{L_0}(\Gamma \mid u, \theta) = P_{L_0}(\Gamma \mid [\![u]\!]^{\theta})$$

This model differs from ours in two notable ways. First, the model of Lassiter and Goodman directly marginalises the distribution of world-states (Γ in the above formalisation), while we only consider the possible meanings of an utterance ($[\![u]\!]^{\theta}$). In other words, we regard pragmatic inference as a problem of inferring utterance meanings, rather than one of directly updating the common ground.

This choice has practical consequences from a modelling perspective. When applying the framework of Lassiter and Goodman, one needs to choose prior distributions carefully, in order to cover all possible aspects of a given world-state which may be relevant to the truth value of any of the possible meanings of u; i.e., those, which, in our example of anaphora resolution, we obtained as mappings from utterances to propositions that varied along the set of parameters θ.

Second, we have allowed the pragmatic speaker model P_{S_1} to marginalise over θ. In contrast, the model of Lassiter and Goodman uses a value of θ which is fixed throughout the model; i.e., it is passed up from the literal listener to the pragmatic listener. Since our distribution over θ depends on the utterance whose interpretation it parameterises, we allow our pragmatic speaker to re-sample θ in its model of the literal listener.

Finally, in comparison to previous RSA work which attempts to combine a natural language semantics with probabilistic reasoning (see Goodman and Lassiter, 2015), the approach we advocate is, we believe, conservative, flexible, and modular:

- It allows for the usual approach to compositional semantics, i.e., in a pure logical language.
- Any such logic can be chosen, so long as it is equipped with a computable notion of entailment.
- Probabilistic computation is added in terms of continuation passing, i.e., as a monadic side effect.
- Even such a side effect does not extend the basic semantics of the metalanguage, which is just the simply typed λ-calculus. We therefore end up with a compositional mathematical theory of the phenomena under investigation.

This situation contrasts, for example, with the implementation of Goodman and Lassiter (2015), using Church. While Goodman and Lassiter are innovative in their integration of probabilistic computation into a functional language, they extend the simply typed λ-calculus with a probabilistic semantics, which, as far as we can tell, is not entirely compositional and thus difficult to reason about.

7 Future directions

The model presented in this paper relies on techniques that are widely used in computational semantics; by combining them in a novel way, we believe that our approach has important potential to generate applications in semantic analysis, inference, and Bayesian cognitive modelling. One immediate avenue for extending our model concerns its applicability to a range of semantic problems that could benefit from a system that leverages both logical semantics and Bayesian reasoning. An obvious candidate is predication vagueness, a classic problem for logical semantics and the target of discussion of a number of Bayesian approaches to semantics (Sutton and Filip, 2016; Lassiter and Goodman, 2017; Bernardy et al., 2019a; Emerson, 2020). Thus extending the coverage of the present model and checking the predictions it makes with respect to these phenomena is one of our goals.

We are also interested in designing a general natural language inference system based on the approach proposed in this paper; such a system could then be evaluated against various test suites. To start, one can check whether the proposed system accounts for pragmatic aspects of the FraCaS test suite (Cooper et al., 1996), the RTE test suite (Dagan et al., 2006), or the small probabilistic test suite of Bernardy et al. (2019a).

As a realistic anaphora resolution algorithm, the model presented here falls short in some respects. First, we take no account of the well-studied grammatical restrictions on the relation between pronominal anaphora and their antecedents (Reinhart, 1976; Chomsky, 1981). Second, our model currently shows no sensitivity to the discourse factors which are well known to affect the acceptability of anaphora in various contexts. And third, it incorporates no sensitivity to psycholinguistic constraints on anaphora, which, like discourse factors,

affect acceptability. There are different ways one might make the model sensitive to such constraints, which may be decided on a case-by-case basis. In principle, any non-pragmatic factor may be accounted for by imposing the right prior on θ. However, other solutions suggest themselves. Grammatical constraints on the anaphora-antecedent relation, for example, might be implemented in an improved compositional semantics which makes antecedents available for certain anaphora depending on their relative syntactic positions. Sensitivity to discourse factors might be incorporated into our model as declarative knowledge that contributes to the prior (i.e., on a par with world and lexical knowledge). And, psycholinguistic (and, perhaps, discourse) constraints might, for example, be incorporated into our pragmatic speaker model as a more realistic measure of cost (see, e.g., Orita et al., 2015), or our literal listener model, by sampling antecedents for anaphora according to their retrieval costs. In principle, antecedent retrieval cost could be incorporated into the distribution over antecedents accessed by the pragmatic listener, as well, perhaps depending on whether our model is viewed as giving a computational-level versus algorithmic-level characterisation of anaphora resolution (Marr, 1982).

The ultimate success of our approach relies on obtaining an accurate account of prior knowledge. Prior world knowledge can be obtained through experiment, following approaches to RSA that have assessed prior beliefs using surveys (Xiang et al., 2021a,b). Given that our model characterises prior knowledge declaratively, we can use similar methods.

Finally, although we have paid specific attention to anaphora resolution, our model makes way for a general approach to semantic ambiguity resolution. We might, for example, extend our model to other anaphora-like phenomena, e.g., ellipsis, as well as the resolution of structural and quantifier-scope ambiguities. The success of such extensions depends on generating an appropriate set of alternatives, given an utterance (and vice versa). For ellipsis, semantic alternatives can be generated by a free parameter, as above; in the case of, e.g., quantifier scope, one might incorporate a parser to provide alternative semantic representations for a given utterance. In all cases, one requires an appropriate set of alternative utterances from a proposition in the speaker model.

Acknowledgements

The research reported in this paper was supported by grant 2014-39 from the Swedish Research Council, which funds the Centre for Linguistic Theory and Studies in Probability (CLASP) in the Department of Philosophy, Linguistics, and Theory of Science at the University of Gothenburg. We thank the anonymous reviewers for their useful comments on an earlier draft of the paper.

References

Abzianidze, L. (2015). A Tableau Prover for Natural Logic and Language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2492–2502, Lisbon, Portugal. Association for Computational Linguistics.

Abzianidze, L. (2020). Learning as Abduction: Trainable Natural Logic Theorem Prover for Natural Language Inference. *arXiv:2010.15909 [cs]*. arXiv: 2010.15909.

Beltagy, I., Chau, C., Boleda, G., Garrette, D., Erk, K., and Mooney, R. (2013). Montague Meets Markov: Deep Semantics with Probabilistic Logical Form. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 11–21, Atlanta, Georgia, USA. Association for Computational Linguistics.

Bernardy, J.-P., Blanck, R., Chatzikyriakidis, S., and Lappin, S. (2018). A Compositional Bayesian Semantics for Natural Language. In *Proceedings of the First International Workshop on Language Cognition and Computational Models*, pages 1–10, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Bernardy, J.-P., Blanck, R., Chatzikyriakidis, S., Lappin, S., and Maskharashvili, A. (2019a). Bayesian Inference Semantics: A Modelling System and A Test Suite. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 263–272, Minneapolis, Minnesota. Association for Computational Linguistics.

Bernardy, J.-P., Blanck, R., Chatzikyriakidis, S., Lappin, S., and Maskharashvili, A. (2019b). Predicates as Boxes in Bayesian Semantics for Natural Language. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 333–337, Turku, Finland. Linköping University Electronic Press.

Bernardy, J.-P. and Chatzikyriakidis, S. (2017). A Type-Theoretical system for the FraCaS test suite: Grammatical Framework meets Coq. In *IWCS 2017 - 12th International Conference on Computational Semantics - Long papers*.

Bernardy, J.-P. and Chatzikyriakidis, S. (2019). A Wide-Coverage Symbolic Natural Language Inference System. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 298–303, Turku, Finland. Linköping University Electronic Press.

Bernardy, J.-P. and Chatzikyriakidis, S. (2021). Applied temporal analysis: A complete run of the fracas test suite. In *IWCS 2021 - 14th International Conference on Computational Semantics - Long papers*.

Blackburn, P. and Bos, J. (2005). *Representation and Inference for Natural Language: A First Course in Computational Semantics*. Studies in Computational Linguistics. University of Chicago Press, Chicago.

Bos, J. and Markert, K. (2005). Recognising Textual Entailment with Logical Inference. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 628–635, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Chomsky, N. (1981). *Lectures on Government and Binding: The Pisa Lectures*. Number 9 in Studies in Generative Grammar. Foris Publications, Dordrecht.

Cooper, R., Crouch, D., Eijck, J. V., Fox, C., Genabith, J. V., Jaspars, J., Kamp, H., Milward, D., Pinkal, M., Poesio, M., Pulman, S., Briscoe, T., Maier, H., and Konrad, K. (1996). Using the Framework. Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Cooper, R., Dobnik, S., Lappin, S., and Larsson, S. (2015). Probabilistic Type Theory and Natural Language Semantics. In *Linguistic Issues in Language Technology, Volume 10, 2015*. CSLI Publications.

Dagan, I., Glickman, O., and Magnini, B. (2006). The PASCAL Recognising Textual Entailment Challenge. In Quiñonero-Candela, J., Dagan, I., Magnini, B., and d'Alché Buc, F., editors, *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, Lecture Notes in Computer Science, pages 177–190, Berlin, Heidelberg. Springer.

Domingos, P. and Lowd, D. (2009). Markov Logic: An Interface Layer for Artificial Intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–155. Publisher: Morgan & Claypool Publishers.

Emerson, G. (2020). Linguists who use probabilistic models love them: Quantification in functional distributional semantics. In *Proceedings of the Probability and Meaning Conference (PaM 2020)*, pages 41–52, Gothenburg. Association for Computational Linguistics.

Emerson, G. and Copestake, A. (2017a). Semantic Composition via Probabilistic Model Theory. In *IWCS 2017 - 12th International Conference on Computational Semantics - Long papers*.

Emerson, G. and Copestake, A. (2017b). Variational Inference for Logical Inference. *arXiv:1709.00224 [cs]*. arXiv: 1709.00224.

Goodman, N. D. and Frank, M. C. (2016). Pragmatic Language Interpretation as Probabilistic Inference. *Trends in Cognitive Sciences*, 20(11):818–829.

Goodman, N. D. and Lassiter, D. (2015). Probabilistic Semantics and Pragmatics: Uncertainty in Language and Thought. In Lappin, S. and Fox, C., editors, *The Handbook of Contemporary Semantic Theory*, pages 655–686. John Wiley & Sons, Ltd.

Goodman, N. D. and Stuhlmüller, A. (2013). Knowledge and Implicature: Modeling Language Understanding as Social Cognition. *Topics in Cognitive Science*, 5(1):173–184.

Grice, H. P. (1975). Logic and Conversation. In Cole, P. and Morgan, J. L., editors, *Syntax and Semantics*, volume 3, Speech Acts, pages 41–58. Academic Press, New York.

Lappin, S. and Leass, H. J. (1994). An Algorithm for Pronominal Anaphora Resolution. *Computational Linguistics*, 20(4):535–561.

Lassiter, D. and Goodman, N. D. (2013). Context, scale structure, and statistics in the interpretation of positive-form adjectives. *Semantics and Linguistic Theory*, 23(0):587–610. Number: 0.

Lassiter, D. and Goodman, N. D. (2017). Adjectival vagueness in a Bayesian model of interpretation. *Synthese*, 194(10):3801–3836.

Marr, D. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. MIT Press, Cambridge.

Mineshima, K., Martínez-Gómez, P., Miyao, Y., and Bekki, D. (2015). Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2061, Lisbon, Portugal. Association for Computational Linguistics.

Montague, R. (1973). The Proper Treatment of Quantification in Ordinary English. In Hintikka, K. J. J., Moravcsik, J. M. E., and Suppes, P., editors, *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, Synthese Library, pages 221–242. Springer Netherlands, Dordrecht.

Orita, N., Vornov, E., Feldman, N., and Daumé III, H. (2015). Why discourse affects speakers' choice of referring expressions. In *Proceedings of the 53rd*

Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1639–1649, Beijing, China. Association for Computational Linguistics.

Raina, R., Ng, A. Y., and Manning, C. D. (2005). Robust textual inference via learning and abductive reasoning. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 3*, AAAI'05, pages 1099–1105, Pittsburgh, Pennsylvania. AAAI Press.

Ranta, A. (2011). *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford.

Reinhart, T. M. (1976). *The syntactic domain of anaphora*. Thesis, Massachusetts Institute of Technology. Accepted: 2005-08-02T20:38:55Z.

Sutton, P. R. (2018). Probabilistic Approaches to Vagueness and Semantic Competency. *Erkenntnis*, 83(4):711–740.

Sutton, P. R. and Filip, H. (2016). Vagueness, Overlap, and Countability. *Proceedings of Sinn und Bedeutung*, 20:730–747.

van Eijck, J. and Lappin, S. (2012). Probabilistic semantics for natural language. In Christoff, Z., Galeazzi, P., Gierasimczuk, N., Marcoci, A., and Smets, S., editors, *Logic and interactive rationality (LIRA)*, volume 2, pages 17–35. Citeseer.

van Eijck, J. and Unger, C. (2010). *Computational Semantics with Functional Programming*. Cambridge University Press, Cambridge.

Xiang, M., Dai, Z., and Wang, S. (2021a). When Parsing and interpretation misalign: a case of wh-scope ambiguity resolution in Mandarin. Under review.

Xiang, M., Kennedy, C., Weijie, X., and Leffel, T. (2021b). Pragmatic Reasoning and Semantic Convention: A Case Study on Gradable Adjectives. Under review.

A (Mostly) Symbolic System for Monotonic Inference with Unscoped Episodic Logical Forms

Gene Louis Kim[1], Mandar Juvekar[2], Junis Ekmekciu[3], Viet Duong[4], and Lenhart Schubert[5]
University of Rochester
Department of Computer Science
{gkim21[1],schubert[5]}@cs.rochester.edu
{mjuvekar[2],jekmekci[3],vduong[4]}@u.rochester.edu

Abstract

We implement the formalization of natural logic-like monotonic inference using Unscoped Episodic Logical Forms (ULFs) by Kim et al. (2020). We demonstrate this system's capacity to handle a variety of challenging semantic phenomena using the FraCaS dataset (Cooper et al., 1996).These results give empirical evidence for prior claims that ULF is an appropriate representation to mediate natural logic-like inferences.[1]

1 Introduction

A monotone function between partially ordered sets either preserves or inverts the ordering of argument values. More precisely, a function f is said to be upward monotone if $x \leq y$ implies $f(x) \leq f(y)$. Similarly, f is said to be downward monotone if $x \leq y$ implies $f(x) \geq f(y)$. If neither of these hold, f is said to be non-monotone. When used in the context of subset relations and entailment, monotonicity can be a tool for making natural language inferences. For instance, consider the second example in fig. 1. *Never* is downward monotone in entailment, since it flips the entailment ordering of (1) *I had a girlfriend taller than me before* entails (2) *I had a girlfriend before* to (2) *I **never** had a girlfriend before* entails (1) *I **never** had a girlfriend taller than me before*. Natural logic is an approach to generating natural language inferences based on syntactic structure and knowledge of the semantic properties of the lexical items and local constructions (Van Benthem et al., 1986; Sánchez-Valencia, 1991). An important fragment of natural logic is monotonicity calculus which operates using syntactic structure and the knowledge of polarity inducing elements and monotonicity relationships. Figure 1 shows the three basic cases of monotonicity inference, upward, downward, and non- monotone contexts leading to different entailment conditions.

Up (FraCaS)	*Some delegates (finished the survey on time)*▲ ⇒ *Some delegates finished the survey*
Down (MED)	*I never had a (girlfriend)*▼ *before* ⇒ *I never had a girlfriend taller than me before*
Non- (MED)	*Exactly 12 aliens read (magazines)*■ ⇏ *Exactly 12 aliens read (news magazines)*■

Figure 1: Upward (...)▲, downward (...)▼, and non-monotone (...)■ examples from the FraCaS and MED datasets.

Episodic Logic (EL) is an extended first-order logic designed to closely match the form and expressivity of natural language (Schubert, 2000). Unscoped Logical Form (ULF) is an underspecified form of EL. ULF completely specifies the semantic type structure of EL, but leaves scope, anaphora, and word sense unresolved (Kim and Schubert, 2019a). Kim and Schubert (2019b) proposed that ULF is suitable for five classes of inferences, namely monotonic inferences, inferences based on clause-taking verbs, inferences based on counterfactuals, inferences from questions, and inferences from requests. Kim et al. (2019) experimentally demonstrated the capacity of ULF to generate all of those classes of inferences except monotonic inference. Kim et al. (2020) presented a proof-based formalism for natural logic-like monotonic inference for ULF. They established a correspondence between their formalism and the natural logic treatment of Sánchez Valencia (1991), and showed that the formalism was capable of handling foundational natural logic inferences from the prior literature. We present an implementation of Kim et al.'s (2020) monotonic inference formalism and give empirical evidence for the feasibility of using ULFs as a basis for making natural logic-like

[1]The code is made available at https://github.com/genelkim/ulf-fracas.

Proceedings of the 1st and 2nd Workshops on Natural Logic Meets Machine Learning (NALOMA), pages 71–80

inferences. Our system achieves a high precision on monotonicity problems using a small number of sound inference rules on a variety of inference cases. We thereby complete the work of Kim et al. (2019) in experimentally demonstrating that ULF is in fact capable of handling the five kinds of inference outlined by Kim and Schubert (2019b).

2 Background

Kim et al. (2019) demonstrated the capacity to use ULFs to generate inferences from clause-taking verbs, counterfactuals, questions, and requests while focusing on discourse-contexts that regularly give rise to these phenomena. They generated forward inferences from manually annotated ULFs using symbolic meta-axioms generalized to handle syntactic idiosyncrasies and achieved reasonable precision on a multi-genre dataset. Our work seeks to complement this by generating Natural Logic-like inferences from ULFs. Furthermore, we start our inferences from English using a symbolic transducer from English constituency parses and expand the scope of inferences to enable automatic evaluation on pre-constructed datasets.

2.1 Theoretical Inference Method

Kim et al. (2020) present a proof-based inference method which uses ULF as the base semantic representation. Polarities are computed respective to specific scopings of ULFs—in the form of scoped logical forms (SLFs)—then propagated back to the ULFs to enable inferences that are contingent on the polarity context. This method includes inference rules for ULFs that correspond directly to inference rules in Sánchez-Valencia's (1991) formulation of Natural Logic. The most notable inference rules are

Monotonicity (UMI)

$$\frac{\phi[P1^{\blacktriangle}],\ ((\text{every.d}\ P1)\ (\text{be.v}\ (=\ (\text{a.d}\ P2))))}{\phi[P2]}\,,$$

$$\frac{\phi[P2^{\blacktriangledown}],\ ((\text{every.d}\ P1)\ (\text{be.v}\ (=\ (\text{a.d}\ P2))))}{\phi[P1]}$$

Conversion (UCI)

$$\frac{((d1\ P)\ (\text{be.v}\ (=\ (d2\ Q))))}{((d1\ Q)\ (\text{be.v}\ (=\ (d2\ P))))}\quad \text{where } d1 \in \{\text{some.d, a.d, no.d}\} \text{ and } d2 \in \{\text{some.d, a.d}\}.$$

Polarity contexts that are necessitated by operators present in the formulas are omitted for clarity, e.g., every.d imposes a negative polarity on its restrictor and a positive polarity on its body. The remaining inference rules are *Polarity Marking* and *Negation Introduction/Elimination*.

Below is an simple inference example from the FraCaS dataset—the actual output of our system—which demonstrates a simple use of the UMI inference rule.[2] This example also shows some differences between our system and the original theoretical method presented by Kim et al. (2020). Namely, our UMI rules generalize to variants of *every A is a B* (in this case *all As are Bs*), our initial polarity marking method circumvents the need for SLFs (Sections 3.3 and 3.4), and we haves rules to generate monotonicity relations from intersective predicate modification (Section 3.4).

Inference Example (FraCaS Problem 24)

1. ((many.d (plur delegate.n)) — Assumption
   ((past obtain.v)
   (k (interesting.a (plur result.n)))
   (adv-a (from.p (the.d survey.n)))))
2. ((all.d (interesting.a (plur result.n))) — Inter. modifier relation, 1.
   ((pres be.v) (= (k (plur result.n)))))
3. ((many.d (plur delegate.n)$^{\blacksquare}$) — Pol marking 1.
   ((past obtain.v)
   (k (interesting.a (plur result.n))$^{\blacktriangle}$)
   (adv-a (from.p (the.d survey.n)))))
4. ((many.d (plur delegate.n)) — UMI 2.,3.
   ((past obtain.v) (k (plur result.n))
   (adv-a (from.p (the.d survey.n)))))

For the syntactic conventions of ULF, such as the type-designating suffixes '.d', '.v', and '.n', see the descriptions provided by Kim and Schubert (2019b) or Kim et al. (2020).

2.2 Automated Monotonicity Inference

Building computational approaches to natural logic inference—distinct from general natural language inference—is an active area of research (Angeli and Manning, 2014; Tian et al., 2014; Mineshima et al., 2015; Abzianidze, 2016; Hu et al., 2019; Haruta et al., 2020). In order to evaluate our monotonicity-specific inference system fairly, we focus on the FraCaS dataset (Cooper et al., 1996) which carefully presents monotonicity-based entailments, for evaluation, and aim to show competence on monotonicity, rather than state-of-the-art (SOTA) performance. In our experiments (Section 5) we compare against a few notable systems that were previously evaluated on the same parts of the FraCaS dataset: Mineshima et al. (2015), Abzianidze (2016), Hu et al. (2019), and Haruta et al. (2020).

[2] Irrelevant polarity marking symbols are omitted for brevity and clarity.

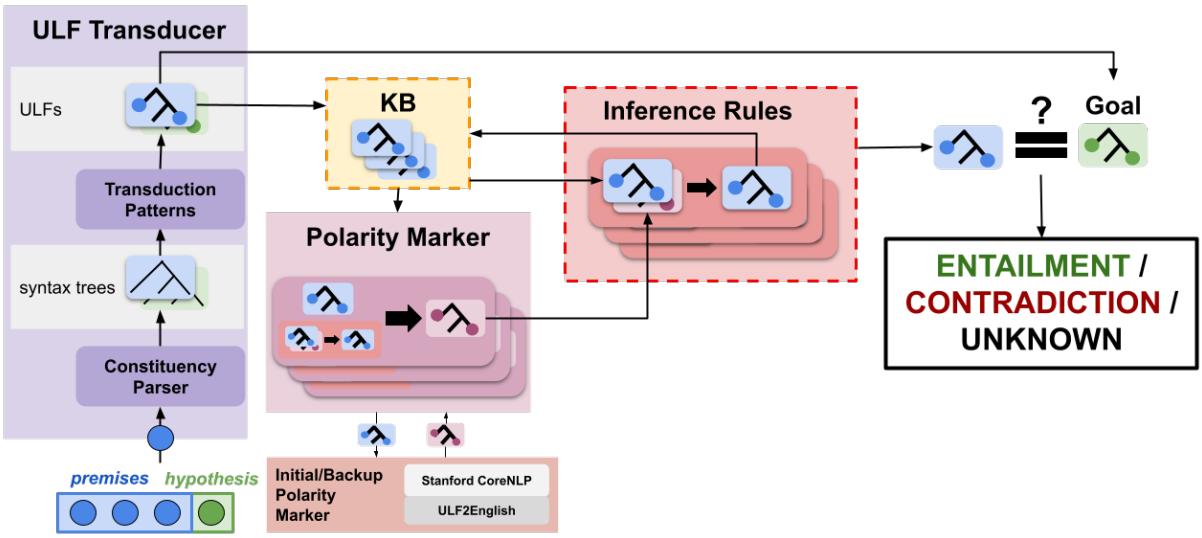

Figure 2: A diagram of the inference system component dependencies.

Mineshima et al. (2015) and Abzianidze (2016) extend first-order lambda logical forms with higher-order terms (e.g most, many, half of, etc.) and augment first-order inference with rules geared towards those terms. Haruta et al. (2020) achieve SOTA performance by employing degree and event semantics to approximate key higher-order logic features presented in different linguistic phenomena. Hu et al. (2019) differs from the others by running directly on the natural language text, with a combinatory categorical grammar (CCG)-based monotonicity labeling system.

Our approach most resembles Hu et al.'s (2019) system because our logical form closely matches the form and expressiveness of natural language, which enables monotonic reasoning using a relatively compact set of inference rules and we also use an auxiliary representation to obtain monotonicity labels. Our goal is not to achieve the SOTA performance on this dataset; rather the SOTA system results are provided to contextualize our system's performance with respect to the wider research efforts on this dataset.

3 System Description

Our inference system starts with a set of premise sentences and a hypothesis sentence in English which are automatically converted to ULF and then used to determine an *entailment*, *contradiction*, or *unknown* relationship between the premises and the hypothesis through a forward inference search from the premises. The inference process is modeled after the theoretical framework described by Kim et al. (2020) which uses SLFs for identifying the polarity operator scopes and computing the global polarity context of each sub-expression. These polarities are then mapped back to the corresponding ULFs which are used as the basis for the inference rules.

We simplify Kim et al.'s (2020) framework in two ways. First, we do not include the scoping possibilities in the proof process. That is, we compute a single SLF for each ULF and assume that it is the correct scoping. Second, we introduce variations of the monotonicity and conversion inference rules that correspond to ULF macros and specific syntax. These reduce two steps of inference (expanding the macro and then applying the inference rule) to a single step. Both of these simplifications are introduced to reduce the search space and speed up the inference process.

Here we describe an example of the second simplification. We directly extract the monotonicity relation from the complex expression containing the nominal predicate with its premodifiers and postmodifiers (i.e., all but the determiner or kind-forming operator of the term derived from a noun phrase). In postnominal modification, the operands of the n+preds macro—combining a nominal predicate with postmodifying predicates—are used directly for inference, without expansion of the macro construct to a conjunctive lambda predicate. For example, for a postmodified noun, as in the phrase *a dog in the park*, we directly extract the entailments *every dog in the park is a dog* and *every dog in the park is in the park*, rather than first

converting the phrase to *something that is a dog and that is in the park* and indirectly computing the rules via explicit predicate intersection. For an intersective prenominal modifier, as in the phrase *a happy dog*, we directly extract the entailments *every happy dog is a dog* and *every happy dog is happy*, rather than first converting the phrase to *something that is happy and that is a dog* first.

The inference system has the following high-level components:

- a heuristic-based inference search function
- a constituency parse to ULF tree transducer
- a global polarity marking function
- inference rules with polarity propagators
- external knowledge resources

Figure 2 shows a diagram of the component dependencies. While most of the inference system is symbolic, the initial constituency parses and initial polarity marking—used for ULF transduction and scope selection, respectively—are computed using NN and ML methods. Furthermore, the ML-based polarity marking is used when the symbolic polarity propagation methods fail or take too long.

3.1 Search Process

Our inference process is guided by a simple heuristic forward search. Algorithm 1 describes this process in detail. In order to retain completeness while using fast and naive heuristic functions, the search process alternates between heuristic guided search and breadth-first search every several steps. This is a generic search process, where h is a heuristic function which estimates the distance from some formula, x, to the goal formula, ϵ is a small positive number which is used to gives preference formulas reached earlier in the search process in cases of ties, and c is the number of search steps in a row that the search process uses heuristic search before switching to BGS and vice-versa. Section 4 specifies the values of these parameters that we use in our experiments.

3.2 ULF Transducer

The ULF transducer converts constituency parses into ULFs with a series of simple correspondences from the phrase structure and POS tags to ULF expressions. This technique is the same as those used in the initial stages of prior transduction-based EL parsers (Schubert, 2002; Schubert and Tong, 2003; Gordon and Schubert, 2010; Schubert, 2014), but modified for Kim and Schubert's (2019b) modern

Algorithm 1 Heuristic search. Inference rules map a set ULF premises to a set of ULF inferences.

Inputs: Φ, a set of premises; ψ, a goal ULF; h, a heuristic function; M, a search depth limit.
Outputs: The entailment classification.
Global Constants: U, a list of unary rules; B, a list of binary rules; ε, a small positive number; c, a step count for search method change.
Procedure:
Initialize $n \leftarrow 0$, KB $\leftarrow \Phi$.
Initialize $Q_{\text{h}} \leftarrow$ empty priority queue.
Initialize $Q_{\text{bfs}} \leftarrow$ empty basic queue.
Initialize $Q \leftarrow Q_{\text{h}}$.
Initialize $Q_{\text{other}} \leftarrow Q_{\text{bfs}}$.
loop
  If $n > M$ or $Q = \varnothing$, **return** UNKNOWN.
  If $\psi \in$ KB, **return** ENTAILMENT.
  If $\neg\psi \in$ KB, **return** CONTRADICTION.
  $\nu \leftarrow Q.\text{pop}()$.
  $t_{\text{unary}} \leftarrow U \times \nu$.
  $t_{\text{binary}} \leftarrow B \times ((\nu \times \nu) \cup (\nu \times \text{KB}) \cup (\text{KB} \times \nu))$.
  Push all results x of computing the tuples in t_{unary} and t_{binary} that are not contained in KB to Q_{h} with key $h(x) + n\varepsilon$ and Q_{bfs}.
  KB $\leftarrow$ KB $\cup\, \nu$.
  $n \leftarrow n + 1$.
  if $n \mod c = 0$ **then**
    tmp $\leftarrow Q$.
    $Q \leftarrow Q_{\text{other}}$.
    $Q_{\text{other}} \leftarrow$ tmp.
  end if
end loop

ULF specification. Some transduction rules add type assumptions that are not necessarily true, but are unlikely to affect the monotonicity inferences. For example, ULF makes a semantic distinction between event modifiers (e.g. *today*) and proposition modifiers (e.g. *surprisingly*) which is not relevant for monotonicity inferences. If the parser fails to eliminate one of these options, it assumes that it is an event modifier.[3]

We use the Berkeley neural parser (Kitaev and Klein, 2018) to get the constituency trees.[4] A neural network-based ULF parser has recently become

[3]The transduction rules are written in a combination of the tree-to-tree transduction language (Purtee and Schubert, 2012) and a simplified variant.

[4]The version 0.2.0 release and the `benepar_en3` model available at `https://github.com/nikitakit/self-attentive-parser/`.

available (Kim et al., 2021), but we opted not to use it because sentences in monotonicity datasets tend to be fairly short and follow written English syntax. Kim et al.'s (2021) parser is more robust to language length and variety. However, for our evaluation datasets we found a symbolic transduction to be more reliable. Additionally, our symbolic transductions have more predictable and regular errors. This allows monotonicity inferences to succeed even with minor errors.

3.3 Polarity Marking

We delegate the initial polarity marking problem to a component of the Natlog and NaturalLi systems (MacCartney and Manning, 2008; Angeli and Manning, 2014) which runs over raw English text.[5] We then align the polarities of each token to the corresponding ULF sub-expression. Rather than using the actual English premises and hypothesis we use the output of the ULF2English system (Kim et al., 2019) so that we can use its subroutines to assist in subexpression alignment.

This alignment is then used to select the scoping by finding the possible SLF that minimizes the number of polarity discrepancies between the NatLog polarity labels and the labels inferred from the scoping and a manually curated list of negative polarity operators. The inference rules propagate the polarities so this is only performed on the input sentences (Section 3.4). During the inference process, this polarity marking is reserved as a fallback in cases where polarity propagation via inference rules fails or takes too long.

Possible SLFs are computed by generating every possible scope configuration while accounting for island constraints. We roughly model scope island constraints with the following rule: *Scoping operators cannot scope outside of ancestors that are ULF type-shifters*. This rule handles complex modifiers (which are shifted from predicates to modifiers) and reified clausal complements (e.g., *I believe that everyone thinks.*) and is implemented trivially with the ULF type system. This is an approximation of the full range of actual island constraints, which come in various classes and with nuances that are still under active investigation in linguistics research. Our rule tends to be stricter than actual scope island constraints leading to some losses in expressive capacity, such as exceptions to commonly accepted island constraints (Barker, 2021) and the *de dicto* / *de re* distinction for clausally-embedded indefinite quantifiers (Donnellan, 1966; Burge, 1977).[6] However, this is only a limitation of our implementation of scoping and polarity propagation. A more nuanced treatment of available scopings can be accommodated by the underlying theoretical inference framework (Kim et al., 2020).

3.4 Inference Rules

All of our inference rules fall under one of four categories.

1. *Monotonicity Substitution*

   This is the core monotonicity inference. Given the premise *Every A is a B*, *B* is substituted for *A* in positive polarity contexts and *A* is substituted for *B* in negative polarity contexts. In order to reduce the proof lengths, we suppress ULF macro expansion rules and extract monotonicity relations directly from macro instances.

2. *Conversion*

   Some A is a B $\Leftrightarrow$ *Some B is an A*

3. *Conservativity*

   δ *As are Bs* $\Leftrightarrow$ δ *As are As that/who are Bs*, where δ is a determiner. This is a category of inferences in the FraCaS dataset and a commonly used inference step for introducing and eliminating relative clauses in simple quantified expressions.

4. *Equivalences*

   This includes equivalent determiner substitutions (e.g., *Every dog is happy* $\Leftrightarrow$ *All dogs are happy*) and predicate synonym substitutions (e.g., *I saw the accident* $\Leftrightarrow$ *I witnessed the accident*).

We have 9 total inference rules when accounting for specializations for macros—though some of these inference rules themselves include several distinct transduction patterns to account for minor syntactic variations.

In order to identify whether a modification is intersective, we use the non-subsective adjective list by Nayak et al. (2014) expanded to words in the WordNet (Miller, 1995) synsets.

[5] This is available through the Natural Logic component of Stanford CoreNLP.

[6] For example, the referential reading of *someone* in the sentence *I know that someone lied* is not available if the indefinite quantifier is not allowed to take wide scope over the sentence.

Polarity Propagation For computational efficiency, each inference rule has a corresponding polarity propagation function. The polarity propagation function takes the premise ULF formulas, their polarity markings, and the conclusion and computes the polarity marking of the conclusion.

As a concrete example, consider the polarity propagation function for the UMI inference rule with the premises and conclusions based on FraCaS problem 24 described in section 2.1. The premises are steps 1 (*many delegates obtained interesting results from the survery*) and 2 (*all interesting results are results*) in the inference example and the conclusion is step 4 (*many delegates obtained results from the survey*). The polarity marking[7] for step 1 is step 3 of the proof and the polarity marking for step 2 is (all.d (interesting.a (plur result.n))▼ ((pres be.v) (= (k (plur result.n)▲)))).

The propagation function identifies that (plur result.n)▲ is the polarized version of the subexpression that substituted for (interesting.a (plur result.n)) in the step 1 premise. Thus, most polarity markings are transferred over from the step 1 polarity marking except the marking for (plur result.n) in the substituted subexpression. This leads to the following polarity marking of the conclusion.[8]

((many.d (plur delegate.n)■)
((past obtain.v) (k (plur result.n)▲)
(adv-a (from.p (the.d survey.n)))))

Most of these propagation functions can be implemented efficiently without accessing the corresponding SLFs because the inference context eliminates the possibility of polarity operators interacting outside of the localized expression substitution due to scope island constraints (Fodor and Sag, 1982; Park, 1995; Ruys and Winter, 2011; Barker, 2015). For example, the conversion rule substitutes two nominal predicates for each other in sentences with the main verb *be*, an indefinitely quantified subject, and a nominal subject complement. In this case, any quantifier embedded within either nominal predicate is constrained by the Complex NP Constraint (Ross, 1967).

A notable exception is the monotonicity substitution of determiners: the polarity propagation function must have access to the SLFs and cannot be implemented as efficiently because the new determiner may induce different polarities in its restrictor and body than the replaced determiner.[9] Properly computing the global polarity from this requires access to the quantifier scopes.

4 Experimental Setup

In our experiments we allow a maximum of 50 inference steps and use a leaf label F1 heuristic (LL-F1) which alternates with breadth-first search (BFS) every 5 inference steps. LL-F1 computes the F1 score between the leaf labels of the new formula and the goal formula, ignoring order, but preserving repetitions. This is turned into a cost ranging 0-to-1 by subtracting it from 1.

The FraCaS dataset is a set of entailment questions related to specific semantic phenomena that were curated by semanticists (Cooper et al., 1996). It contains 346 problems, of which 12 do not have well-defined answers. We focus on the most relevant section of the FraCaS dataset, section 1: Generalized Quantifiers (GQs). This is also the largest section, making up almost a quarter of the dataset. Due to the small size of the FraCaS dataset and the challenging phenomena it contains, prior research has trained and tested models on the same problems, focusing on the capacity of their systems to perform such inferences, rather than their competence in learning and generalizing to a larger scale. This aligns nicely with our goal to demonstrate the capacity to use ULFs as the basis for monotonic inferences, rather than present a system to compete with the state-of-the-art on entailment tasks.

5 Results

Our experiments show that our system is able to precisely cover a variety of semantic phenomena and constructions, but, as expected from a demonstration system, does not achieve the robustness of SOTA entailment systems.

Table 2 shows the confusion matrix of our system on the FraCaS dataset. Our system shows very high precision (it is never incorrect when it makes a definitive conclusion—not UNK) because of the soundness of our inference rules. While our system fails to correctly identify any contradictions, this is not an inherent limitation of the system. It was simply the case that parser errors led to the inability to match the inferred negated formula with the hypothesis in the 5 problems that have contradiction labels.

[7] Omitting irrelevant polarities.

[8] Again, omitting irrelevant polarities.

[9] For example, in positive contexts, *the* may be replaced with *a*, as in, *I saw the dog* ⇒ *I saw a dog*. *The* imposes a flat entailment context on its restrictor whereas *a* imposes a positive entailment context which warrants a fresh computation of the global polarity markings.

Section	Accuracy %													
	Single-premise				Multi-premise				Overall					
	BL	Ours	MN	LP	BL	Ours	MN	LP	BL	Ours	MN	LP	HU	HR
1 GQs	45	73	82	93	57	67	73	93	50	70	78	93	88	99

Table 1: FraCaS performance of our system (Ours) compared against a majority class (ENT) baseline (BL) and several notable RTE systems: MN (Mineshima et al., 2015), LP (Abzianidze, 2016), HU (Hu et al., 2019), and HR (Haruta et al., 2020). Hu et al. (2019) and Haruta et al. (2020) only report the overall accuracy of their systems.

Gold\Pred.	ENT	CON	UNK
ENT	**22**	0	15
CON	0	**0**	5
UNK	0	0	**32**

Table 2: Confusion matrix on the FraCaS dataset.

In Table 1, the accuracy of our system is compared to the majority class baseline and other natural logic systems that focus on monotonicity and FraCaS inferences. According to the table, a variety of methods prove effective at monotonic reasoning over a variety of linguistic phenomena. LP and HR perform notably well and both rely on CCG parses for obtaining the representation and theorem provers for managing inferences. Although our system falls short of the performance of SOTA systems on FraCaS, we still perform noticeably better than the majority class baseline. Investigating the error cases of our system makes clear that the shortfalls of our system are not inherent in the theoretical approach—rather they are due to syntactic and inference cases that were not addressed in this exploratory inference system.

The polarity propagation system used the fallback system (the polarity marking component of the Natlog system) in 42 out of the 3,109 (1.3%) total polarity propagation calls made in the GQs section of the FraCaS evaluation.

5.1 Qualitative Analysis

Figure 3 shows three distinct success and three distinct failure cases of our system. First looking at the successes, example 18 is a multi-premise entailment problem which requires conservativity inference and multiple UMIapplications in both positive and negative contexts. Example 59 has two distinct components—first the determiner *a few* must be generalized to *at least a few*, second *female* must be recognized as an intersective modifier and removed to generalize the nominal predicate in positive polarity context. Example 60 again has the intersective modifier *female*, but must not trigger an inference because of the negative polarity context.

Now taking a look at the failures, example 25 requires the recognition of *in major national newspapers* as an adjunct that may be dropped for a more general meaning. Our system parses the premise incorrectly—specifically "results published in major national newspapers" is parsed as a single kind-of-event[10] argument rather than an argument and an adjunct. This is can be addressed by an improvement of the ULF parser, e.g., an expansion of the verb subcategorization frames known by the ULF transduction rules. Example 48 requires the introduction of the phrase *a lot of* in negative polarity context, since it acts as a specializing modifier. Our system does not recognize *a lot of* as specializing modifier and this sort of multi-word idiosyncratic syntactic construction for a specializing modifier needs to be addressed specifically in the grammar. Finally, example 76 is a reversal of the intersective modifier *female* that was in the successful example 59. Because we use a forward inference framework, the proof-system does not have access to the modifier *female*. This could be handled by extraction of necessary intersective monotonicity rules from the hypothesis or more generally keeping a lexicon of intersective modifiers—though, the latter approach would be less efficient if implemented naively.

6 Conclusion

We have presented a simple implementation of forward monotonic inference starting with English sentences and using ULFs as the representational basis. Our system shows a high degree of precision on a variety of monotonicity phenomena,

[10] A kind-of-event is a type in the domain of discourse in EL semantics corresponding to generic events. For example, in the sentence "The news reporting on a missing kitten was unexpected", *unexpected* is a predicate over the kind-of-event "The news reporting on a missing kitten". This is distinct from similar EL types of events, which are particular instances, and propositions, which are statements that may be true or false.

Successes

ID	Correct inference
18	*Every European has the right to live in Europe*; *Every European is a person*; *Every (person who has the right to live in Europe)▼ can travel freely within Europe* ⇒ *Every European can travel freely within Europe*
59	*(A few)▲ (female committee members)▲ are from Scandinavia* ⇒ *At least a few committee members are from Scandinavia*
60	*Few (female committee members)▼ are from southern Europe* ⇏ *Few committee members are from southern Europe*

Failures

ID	Correct inference
25	*Several delegates (got the results published in major national newspapers)▲* ⇒ *Several delegates got the results published*
48	*At most ten commissioners spend (time)▼ at home* ⇒ *At most ten commissioners spend a lot of time at home*
76	*Few (committee members)▼ are from southern Europe* ⇒ *Few female committee members are from southern Europe*

Figure 3: Several examples of inference successes and failures. The relevant polarity contexts for the final (or last two if not overlapping) inference step is marked with (...)▲ or (...)▼ and the relevant spans are underlined in the premises and hypothesis. Our inference system predicated UNK for each of the failure examples.

empirically confirming the final class of inferences that Kim and Schubert (2019b) proposed would be supported by ULF alongside a suite of pragmatics-oriented inference capabilities of ULF described in Kim et al. (2019). The present effort is a feasibility demonstration and further engineering, expanding the coverage, is needed to create a system competitive with the state-of-the-art.

The specifics of our demonstration system and the results point to many possible avenues of improvement. Beyond direct improvements to the ULF parser, operator scoping, and inference rules to cover more constructions, the proof-search process can be expanded to explicitly include alternate parsing and scoping choices thereby enabling proper exploration of ambiguous constructions. For example, each particular English-to-ULF parse and each scoping choice leading to a distinct SLF can be formulated as an inference rule that can be explored. The inference rules can also be made more flexible by implementing the RI-1 and RI-2 rules that Kim et al. (2020) describe as a generalization of UMI. The direct access to syntactic structure from ULF leaves room for a much more sophisticated treatment of linguistic constraints (notably island constraints as discussed in section 3.4) and the logical type structure makes ULF theoretically capable of inferences from disjunctive conclusions; e.g., *Alice has a dog or a cat*, given that *Alice has a furry pet* and *Furry pets are either dogs or cats*. Finally, in the vein of merging ML/DL and symbolic approaches, ULF can be reliably translated back into English (Kim et al., 2019) so that ML/DL approaches that work over raw English text can be accessed and used in conjunction with the symbolic rules. In fact, the polarity marking component of our system (section 3.3) is precisely an example of such a bridging of methods.

7 Acknowledgments

This work was supported by NSF EAGER grant NSF IIS-1908595, DARPA CwC subcontract W911NF-15-1-0542, and a Sproull Graduate Fellowship from the University of Rochester. We are grateful to the anonymous reviewers for their helpful feedback.

References

Lasha Abzianidze. 2016. Natural solution to fracas entailment problems. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 64–74.

Gabor Angeli and Christopher D. Manning. 2014. NaturalLI: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 534–545, Doha, Qatar. Association for Computational Linguistics.

Chris Barker. 2015. Scope. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantics*, 2 edition, chapter 2, pages 40–76. Wiley Blackwell.

Chris Barker. 2021. Rethinking scope islands. *Linguistic Inquiry*, pages 1–55.

Tyler Burge. 1977. Belief de re. *The Journal of Philosophy*, 74(6):338–362.

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

Keith S Donnellan. 1966. Reference and definite descriptions. *The philosophical review*, 75(3):281–304.

J. Fodor and I. Sag. 1982. Referential and quantificational indefinites. *Linguistics and Philosophy*, 5:355–398.

Jonathan Gordon and Lenhart Schubert. 2010. Quantificational sharpening of commonsense knowledge. In *Proceedings of the AAAI 2010 Fall Symposium on Commonsense Knowledge*.

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, and Larry Moss. 2019. Natural language inference with monotonicity. In *Proceedings of the 13th International Conference on Computational Semantics - Short Papers*, pages 8–15, Gothenburg, Sweden. Association for Computational Linguistics.

Gene Kim, Benjamin Kane, Viet Duong, Muskaan Mendiratta, Graeme McGuire, Sophie Sackstein, Georgiy Platonov, and Lenhart Schubert. 2019. Generating discourse inferences from unscoped episodic logical formulas. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 56–65, Florence, Italy. Association for Computational Linguistics.

Gene Kim and Lenhart Schubert. 2019a. A type-coherent, expressive representation as an initial step to language understanding. In *Proceedings of the 13th International Conference on Computational Semantics*, Gothenburg, Sweden. Association for Computational Linguistics.

Gene Louis Kim, Viet Duong, Xin Lu, and Lenhart Schubert. 2021. A transition-based parser for unscoped episodic logical forms.

Gene Louis Kim, Mandar Juvekar, and Lenhart Schubert. 2020. Monotonic inference for underspecified episodic logic. In *Proceedings of the Workshop Natural Logic Meets Machine Learning*.

Gene Louis Kim and Lenhart Schubert. 2019b. A type-coherent, expressive representation as an initial step to language understanding. In *Proceedings of the 13th International Conference on Computational Semantics - Long Papers*, pages 13–30, Gothenburg, Sweden. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 521–528, Manchester, UK. Coling 2008 Organizing Committee.

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 the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2061.

Neha Nayak, Mark Kowarsky, Gabor Angeli, and Christopher D Manning. 2014. A dictionary of non-subsective adjectives. Technical Report CSTR 2014-04, Department of Computer Science, Stanford University.

Jong C. Park. 1995. Quantifier scope and constituency. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 205–212, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Adam Purtee and Lenhart Schubert. 2012. TTT: A tree transduction language for syntactic and semantic processing. In *Proceedings of the Workshop on Applications of Tree Automata Techniques in Natural Language Processing*, ATANLP '12, pages 21–30, Stroudsburg, PA, USA. Association for Computational Linguistics.

John Robert Ross. 1967. *Constraints on Variables in Syntax*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Eddy G Ruys and Yoad Winter. 2011. Quantifier scope in formal linguistics. In *Handbook of philosophical logic*, pages 159–225. Springer.

Victor Sánchez Valencia. 1991. Categorial grammar and natural logic. ILTI Prepublication: Logic, Philosophy and Linguistics (LP) Series.

Victor Sánchez-Valencia. 1991. *Studies on Natural Logic and Categorial Grammar*. Ph.D. thesis, University of Amsterdam.

Lenhart Schubert. 2002. Can we derive general world knowledge from texts? In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 94–97, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lenhart Schubert. 2014. From treebank parses to episodic logic and commonsense inference. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 55–60, Baltimore, MD. Association for Computational Linguistics.

Lenhart Schubert and Matthew Tong. 2003. Extracting and evaluating general world knowledge from the brown corpus. In *Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning*, pages 7–13.

Lenhart K. Schubert. 2000. The situations we talk about. In Jack Minker, editor, *Logic-based Artificial Intelligence*, pages 407–439. Kluwer Academic Publishers, Norwell, MA, USA.

Ran Tian, Yusuke Miyao, and Takuya Matsuzaki. 2014. Logical inference on dependency-based compositional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 79–89, Baltimore, Maryland. Association for Computational Linguistics.

Johan Van Benthem et al. 1986. *Essays in Logical Semantics*. Springer.

Author Index