CSTFRS 2021

ESSLLI 2021 Workshop on Computing Semantics with Types, Frames and Related Structures

Proceedings of the Workshop

26–27 July, 2021

Utrecht, The Netherlands (online)

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 978-1-959429-37-1

Preface

The series of workshops on Computing Semantics with Types, Frames and Related Structures (CSTFRS) is intended as a forum for people interested in structured representations of linguistic information, especially from a computational perspective. A first edition of the workshop took place in Gothenburg as part of IWCS 2019. The second edition, whose proceedings are presented here, was hosted by the 32nd European Summer School in Logic, Language and Information (ESSLLI 2021).

Structured representations play a central role in the study of natural language semantics, especially in cognitively oriented approaches in the tradition of Fillmore, Jackendoff, and Langacker. Formal semantics in the Montague-tradition, on the other hand, is less concerned with the structure of representations but with logical expressions, truth conditions, and model-theoretic interpretations. In recent years, however, there has been a growing body of research which aims to integrate structured entities into formal semantic accounts. Important developments in this field are the uses of rich type systems and frame-based representations in lexical and compositional semantics. A key feature of these approaches is that semantic representations can themselves be used to compute semantic content and have yielded a way of combining compositional and lexical semantics by providing a single representational system for different modalities.

The topics of this workshop cover both foundational issues (e.g. developments in rich type theoretical semantics) and applications of theories that employ structured representations to specific linguistic phenomena.

We would like to thank the organizers of ESSLLI 2021 for hosting our workshop online.

Stergios Chatzikyriakidis
Rainer Osswald

Program Chairs & Organization

Stergios Chatzikyriakidis, University of Gothenburg (Sweden)
Rainer Osswald, Heinrich Heine University Düsseldorf (Germany)

Program Committee

Daisuke Bekki, Ochanomizu University (Japan)
Robin Cooper, University of Gothenburg (Sweden)
Jonathan Ginzburg, Université Paris-Diderot (France)
Eleni Gregoromichelaki, University of Gothenburg (Sweden)
Justyna Grudzińska, University of Warsaw (Poland)
Laura Kallmeyer, Heinrich Heine University Düsseldorf (Germany)
Yusuke Kubota, National Institute for Japanese Language and Linguistics (Japan)
Zhaohui Luo, Royal Holloway – University of London (UK)
Bruno Mery, Université de Bordeaux (France)
Richard Moot, Université de Montpellier & LIRMM (France)
Valeria de Paiva, Nuance Communications & University of Birmingham (UK)
Sylvain Pogodalla, INRIA Nancy – Grand Est, LORIA Lab (France)
Christian Retoré, Université de Montpellier & LIRMM (France)
Peter Sutton, Heinrich Heine University Düsseldorf (Germany)
Henk Zeevat, Heinrich Heine University Düsseldorf (Germany)

Invited Speakers

Jean-Philippe Bernardy, University of Gothenburg (Sweden)
Robin Cooper, University of Gothenburg (Sweden)
Justyna Grudzińska, University of Warsaw (Poland)

Table of Contents

Workshop Program

Monday, July 26

09:45–11:15	*Modalities for Functional Programming, Logic and Semantics* Jean-Philippe Bernardy (Invited Speaker)
11:30–12:10	*On the Dual Interpretation of Nouns as Types and Predicates in Semantic Type Theories* William Babonnaud
12:15–12:55	*Analytical, Symbolic and First-Order Reasoning within Neural Architectures* Samuel Ryb & Marten van Schijndel
14:00–14:40	*Donkey Anaphora: Type-Theoretic Semantics with Both Strong and Weak Sums* Zhaohui Luo
14:45–15:30	*General discussion*

Tuesday, July 27

09:45–11:15	*So what's all this structure good for? Some uses of record types in TTR* Robin Cooper (Invited Speaker)
11:30–12:10	*Improving DRS Parsing with Separately Predicted Semantic Roles* Tatiana Bladier, Gosse Minnema, Rik van Noord & Kilian Evang
12:15–12:55	*Semantic Learning in a Probabilistic Type Theory with Records* Staffan Larsson, Jean-Philippe Bernardy & Robin Cooper
14:00–15:30	*Dependent Types in Action: An Investigation into a Connection between Preposition Senses and the Logic of Scope* Justyna Grudzinska (Invited Speaker)
16:00–16:40	*Feature Structures in the Wild: A Case Study in Mixing Traditional Linguistic Knowledge Representation with Neural Language Models* Gerald Penn & Ken Shi
16:45–17:30	*General discussion and concluding remarks*

Invited Papers

So what's all this structure good for? Some uses of record types in TTR

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

Abstract

TTR, a type theory with records, makes use of structured semantic objects such as record types. In this paper we will first explore in what sense this represents an increase in structure over what we normally find in a classical Montague style semantics. We will then mention some of the various uses of record types relating them to Σ-types in type theory, discourse representation structures, frames, feature structures and various kinds of mental states.

Finally, we will consider how the introduction of such structured semantic objects compares with the kind of proof theoretic approaches common in type theoretical approaches to linguistic semantics on the one hand and on the other hand linguistic theories which introduce a level of logical form or discourse representation structure as a mediation between natural language syntax and model theory. I will try to argue that, in general terms, all these approaches introduce similar notions of structure but that there are advantages to introducing the structure in terms of semantic objects.

1 Structured objects in TTR

In this paper we will address and reflect on the use of structured objects in TTR, a Type Theory with Records (Cooper, 2012; Cooper and Ginzburg, 2015; Cooper, in prep). TTR is inspired by Martin-Löf type theory (Martin-Löf, 1984; Nordström et al., 1990) and is a rich type theory in the sense that it does not contain just basic ontological types as in simple type theory such as Montague's e ("entities") and t ("truth-values") and function types defined on these but types of objects like *Tree* and events like *boy-hugs-dog*. Central to such a type theory is the notion of *judging* an object or situation/event, a, to be of a type, T. In symbols this is expressed as (1).

(1) $a : T$

Types in TTR can be structured objects. One example of this is *ptypes*, types where a predicate is used to construct a type from appropriate arguments to the predicate. Suppose that b is some particular boy and that d is some particular dog. We use (2) to represent the type of situations in which b hugs d.

(2) hug(b,d)

The notation in (2) is thought of as representing a *labelled set* (the graph of a function whose domain is a designated set of labels and whose range is $\{\text{hug}, b, d\}$). Thus we encode it as the set of ordered pairs in (3).

(3) $\{\langle\text{pred}, \text{hug}\rangle, \langle\text{arg1}, b\rangle, \langle\text{arg2}, d\rangle\}$

Note that the labelled set in (3) is NOT the set of witnesses of the type. Thus the structure of the type is not given by the set of its witnesses and the type is not identified by the set of its witnesses. If s is a witness for the type 'hug(b,d)', that means that s stands in the *of-type relation* to the set (3).

This gives types the ontological status of mathematical objects (such as sets) whatever you think that is. One view is that mathematical objects are part of the basic furniture of the world. Another view is that they are mental constructs imposed on physical reality. We could also place them in Frege's (1918/1919) third realm. Yet another view, akin to the relational view of meaning of Barwise and Perry (1983) and more recently the relational interpretation of quantum theory (Rovelli, 2021) is that they are inherent in the relation between an observer and the world. Whatever view we take, they should be no more (and probably no less) mysterious than sets. It would probably be a mistake to think of individuals or events as being less mysterious. The boy and the dog in our example do not correspond to any unique collection of physical particles since the physical reality which we identify as these individuals is constantly changing. At best, we can say that these individuals represent strings of events that cohere in a certain way. But what kind of coherence of events counts as identifying an individual seems to depend on the nature of the observer (*cf.* the discussion of schemes of individuation by Barwise, 1989, Chapters 10 and 11 and, for a recent view on how we construct reality from the neuroscience perspective, Seth, 2021).

The type in (2) is the type of situations where a particular boy, b, hugs a particular dog, d. We use *record types* to represent a more general record type, *boy-hugs-dog*, the type of situations were some boy hugs some dog. Such a record type is given in (4).

(4) $$\left[\begin{array}{lcl} \text{x} & : & \textit{Ind} \\ \text{c}_{\text{boy}} & : & \text{boy(x)} \\ \text{y} & : & \textit{Ind} \\ \text{c}_{\text{dog}} & : & \text{dog(y)} \\ \text{e} & : & \text{hug(x,y)} \end{array}\right]$$

A record type is a finite set of fields, each consisting of a label and a type (or a dependent type together with a sequence of paths in the type, as we will explain below). The witnesses of record types are *records*. A record of the type (4) could be of the form (5)

(5) $$\left[\begin{array}{lcl} \text{x} & = & \text{sam} \\ \text{c}_{\text{boy}} & = & s_1 \\ \text{y} & = & \text{fido} \\ \text{c}_{\text{dog}} & = & s_2 \\ \text{e} & = & s_3 \\ \ldots & & \end{array}\right]$$

where the conditions in (6) hold.

(6) sam : *Ind*
s_1 : boy(sam)
fido : *Ind*
s_2 : dog(fido)
s_3 : hug(sam, fido)

Records are also finite sets of fields. For a record, r, to be a witness for a record type, T, for each of the fields in T, there must be a field in r with the same label where the value in the record field is of the type in the field of the record type. Note that this allows for there to be more fields in the record with labels not occurring in the type. (This is represented by the '. . . ' in (5).)

In TTR, (4) is a convenient notation for (7) where the dependent fields are spelt out as containing a pair consisting of a dependent type (a function which returns a type) and a sequence of paths within the type (the paths where arguments to the dependent type are to be found in the record being tested against the type).

(7) $$\left[\begin{array}{l} \text{x}:\textit{Ind} \\ \text{c}_{\text{boy}}:\langle\lambda v{:}\textit{Ind}\ .\ \text{boy}(v), \langle\text{x}\rangle\rangle \\ \text{y}:\textit{Ind} \\ \text{c}_{\text{dog}}:\langle\lambda v{:}\textit{Ind}\ .\ \text{dog}(v), \langle\text{x}\rangle\rangle \\ \text{e}:\langle\lambda v_1{:}\textit{Ind}\ .\ \lambda v_2{:}\textit{Ind}\ .\ \text{hug}(v_1, v_2), \langle\text{x}, \text{y}\rangle\rangle \end{array}\right]$$

Record types in TTR are labelled sets like ptypes, though using a different stock of labels to ptypes. This means that a record type whose graphical display is of the form (8a) is actually the set of ordered pairs (8b).[1]

(8) a. $$\left[\begin{array}{lcl} \ell_1 & : & T_1 \\ \ell_2 & : & T_2 \\ \ldots & & \\ \ell_n & : & T_n \end{array}\right]$$

b. $\{\langle\ell_1, T_1\rangle, \langle\ell_2, T_2\rangle, \ldots, \langle\ell_n, T_n\rangle\}$

Record types are, then, another kind of structured object in TTR.

Types play an important role in type theories not only as types of objects or situations but also in that they can be used to model propositions. See Wadler (2015) for a discussion of the history of this idea which has a number of sources. The "propositions as types" dictum is of central importance in Martin-Löf type theories. A type, T, as a proposition is said to be "true" just in case there is something, a, such that $a : T$, that is, T has a witness. As propositions associated with linguistic interpretation are normally taken to be record types in TTR, it follows that we treat propositions as structured objects.

An unconsidered first reaction to this fact is that this makes propositions in TTR very different from the kind of unstructured propositions that one finds in Montague semantics. However, this claim is not true given that we have based our claim of structure on the fact that record types are sets of ordered pairs. For Montague, a proposition is a function from world-time pairs to truth values. That is, a proposition is a set of ordered pairs since functions are modelled as sets of ordered pairs. For us, a proposition can be a record type and, as we have seen, a record type is a set of ordered pairs. So

[1]This is actually a slight simplification. See the discussion of flavours in labelled sets in Cooper (in prep).

what is it that makes us think that TTR's record types are structured objects whereas Montague's propositions are not?

TTR's record types differ from Montague's propositions in two main respects. The first is size. For Montague, the functions modelling propositions are infinite, defined on an uncountable domain, the set of world-time pairs. Record types, on the other hand, are finite functions defined on a (normally small) finite set of labels. The other respect in which they differ is content. For Montague, the proposition *a boy hugs a dog* contains nothing corresponding to *boy*, *dog* or *hugs*. These are used, of course, in defining what function from possible worlds and times to truth values is the proposition but you cannot look at the resulting function and determine what was used to define it. Record types, on the other hand, contain all the elements that were used to build them up in their various fields, so, in a rather trivial sense, you *can* look at the record type and determine what was used to define it. For Montague, then, the process of compositional interpretation involves loss of information, a kind of "catastrophic forgetting" to borrow a term for a rather different (but nevertheless not entirely unrelated) problem from the literature on neural networks. It is precisely this difference which leads us to think of Montague's propositions as unstructured and record types as structured.

2 Some uses of structured types

In this section we will briefly review some of the uses to which structured types have been put in TTR.

2.1 Types as models of propositions

On p. 2, we introduced the "propositions as types" dictum and explained its importance in type theoretic approaches to semantics. One opportunity this offers, depending on how you define your type theory, is a direct way of dealing with what is often referred to as *hyperintensionality* in the possible worlds approach to natural language semantics. In TTR we allow distinct types to have the same witnesses. Consider the ptypes represented in (9).

(9) a. buy(kim,syntactic_structures,sam)
    b. sell(sam,syntactic_structures,kim)

(9a) represents the type of situations where Kim buys Chomsky's 1957 book *Syntactic Structures* from Sam and (9b) represents the type of situations where Sam sells *Syntactic Structures* to Kim. TTR allows us to characterize type systems in which (10) holds.

(10) $s : \text{buy}(a, b, c) \leftrightarrow s : \text{sell}(c, b, a)$

(10) is a restriction on type systems in the same way in which "meaning postulates" in Montague's semantics are restrictions on the possible worlds to which we should direct our attention. In the type theory version, however, we have two distinct types (which can be used as propositions) which have exactly the same witnesses.

This provides us with a good reason to think of propositions as types rather than as sets of possible worlds (or in Montague's version sets of world time pairs). In a possible worlds theory the expression 'buy(a,b,c)' intuitively represents the set of possible worlds in which a buys b from c. But we have no independent way of characterizing which worlds those are apart from saying that they are the worlds in which 'buy(a,b,c)' is true. We cannot say what it is that the worlds have in common which makes them worlds in which the expression is true. Types, on the other hand, provide a theory of what situations (or possible worlds) might have in common and then expressions are related to the types. From the perspective of possible world semantics, types provide an "inside-out semantics" where you start from the commonalities (the types) and reason about what objects might be witnesses for the types, rather than starting from a set of possible worlds that have something in common but failing to provide a theory of what the commonality is (apart from saying that a certain sentence is true in all the worlds in the set). Possibly, if you have a theory of situation types, you might try to recover a possible worlds theory by looking at the sets of witnesses of the types. But in order to do this you would need to figure out a way of characterizing situation sufficiently large to count as a world. Normally, in a type theory (or a situation theory, for that matter) there is nothing corresponding to a largest situation. For example, records are finite sets of fields in TTR and can be used to model situations, but for any record it is possible to create a new record by adding an additional field.

2.2 Subtyping

Using structured record types allows us to introduce a lattice-like notion of subtyping which is supervenient on this structure. For example, any situation in which a boy hugs a dog is a situation in which there is a boy. This can be expressed as in (11).

$$(11)\quad \begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{c}_{\text{boy}} & : & \text{boy(x)} \\ \text{y} & : & \mathit{Ind} \\ \text{c}_{\text{dog}} & : & \text{dog(y)} \\ \text{e} & : & \text{hug(x,y)} \end{bmatrix} \sqsubseteq \begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{c}_{\text{boy}} & : & \text{boy(x)} \end{bmatrix}$$

Here the supertype contains a subset of the fields in the subtype. Exactly which subsets of fields are available as supertypes is regulated by the dependencies among the fields. If we remove a field on which another field depends we have to remove the dependent field as well.

2.3 Σ-types

It is often said in type theory that record types are really just a variant notation for Σ-types. Intuitively Σ-types correspond to existential quantification. A Σ-type $(\Sigma x : A)B(\!(x)\!)$ (where we use the notation $B(\!(x)\!)$ to indicate that B depends on x) is the type of ordered pairs $\langle a, b\rangle$ where $a : A$ and $b : B(\!(a)\!)$, corresponding to "There is an A, a, such that $B(\!(a)\!)$". For example, if A is *Dog* and B is *Bark*, the Σ-type $(\Sigma a : \mathit{Dog})\mathit{Bark}(\!(a)\!)$ can be construed as the type of situation in which some dog barks.

We can illustrate the relationship between record types and Σ-types by looking at our running example of a record type corresponding to "some boy hugs some dog" in (12).

$$(12)\quad \begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{c}_{\text{boy}} & : & \text{boy(x)} \\ \text{y} & : & \mathit{Ind} \\ \text{c}_{\text{dog}} & : & \text{dog(y)} \\ \text{e} & : & \text{hug(x,y)} \end{bmatrix} \approx$$

$$\left\{ \begin{array}{l} (\Sigma x\!:\!\mathit{Ind})(\Sigma c_b\!:\!\text{boy}(x))(\Sigma y\!:\!\mathit{Ind})(\Sigma c_d\!:\!\text{dog}(y))\text{hug}(x,y) \\ (\Sigma y\!:\!\mathit{Ind})(\Sigma c_d\!:\!\text{dog}(y))(\Sigma x\!:\!\mathit{Ind})(\Sigma c_b\!:\!\text{boy}(x))\text{hug}(x,y) \\ \ldots \end{array}\right.$$

Note that as TTR record types are sets of fields there are several Σ-types which intuitively correspond to a single record type. We have represented just two of them in (12). Intuitively, all of these Σ-types are equivalent since they all correspond to "a boy hugs a dog". Note, however, that this equivalence is not directly derivable from the characterization of Σ-types. These Σ-types do not have a witness in common, for example. In TTR we use record types in place of Σ-types.

2.4 Discourse representation structures

The labels in record types can play the same role as discourse referents in discourse representation structures (Kamp and Reyle, 1993) in that they can be used to account for anaphora. In particular, the labels associated with situation types seem intuitively related to the discourse referents in segmented discourse representation theory (SDRT, Asher and Lascarides, 2003). Here we will just give an example in (13) of how our running example of a record type can be seen as corresponding to a discourse representation structure.

$$(13)\quad \begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{c}_{\text{boy}} & : & \text{boy(x)} \\ \text{y} & : & \mathit{Ind} \\ \text{c}_{\text{dog}} & : & \text{dog(y)} \\ \text{e} & : & \text{hug(x,y)} \end{bmatrix} \approx$$

$$\begin{array}{|l|}\hline \quad \text{x c}_b \text{ y c}_d \text{ e} \\ \hline \\ \text{c}_b : \begin{array}{|c|}\hline \phantom{\text{boy(x)}} \\ \hline \text{boy(x)} \\ \hline \end{array} \\ \\ \text{c}_d : \begin{array}{|c|}\hline \phantom{\text{dog(y)}} \\ \hline \text{dog(y)} \\ \hline \end{array} \\ \\ \text{e} : \begin{array}{|c|}\hline \phantom{\text{hug(x,y)}} \\ \hline \text{hug(x,y)} \\ \hline \end{array} \\ \\ \hline \end{array}$$

2.5 Dialogue gameboards/information states

Ginzburg (2012) used TTR record types to represent dialogue gameboards which keep track of the current dialogue state (questions under discussion among other things) according to a dialogue participant. To illustrate this kind of use of record types we present in (14) a preliminary version of the type *InfoState* from Cooper (in prep, Chap. 2) based on the characterization of dialogue gameboards as information states in Larsson (2002).

$$(14)\quad \begin{bmatrix} \text{private:} & \begin{bmatrix}\text{agenda:list}(\mathit{RecType})\end{bmatrix} \\ \text{shared:} & \begin{bmatrix}\text{latest-utterance:}\mathit{Sign}^* \\ \text{commitments:}\mathit{RecType}\end{bmatrix} \end{bmatrix}$$

Here the information state is divided into two fields: private and shared information, that is private to the particular dialogue participant and shared with the other dialogue participant(s). Among the private information is an agenda, a list of record types, corresponding to types that the dialogue participant plans to realize in upcoming turns in the dialogue.

Among the shared information is the latest utterance, a string of signs in the sense of head driven phrase structure grammar (see 2.6) and commitments, a record type representing what has been agreed on between the dialogue participants for the sake of the dialogue so far.

2.6 Feature structures

Record types have also been used in the role of feature structures as used in head-driven phrase structure (HPSG, Sag et al., 2003). To illustrate this, we give, in (15), a version of the basic type *Sign* from Cooper (in prep, Chap. 3).

(15) $\sigma : \textit{Sign}$ iff $\sigma : \begin{bmatrix} \text{s-event} & : & \textit{SEvent} \\ \text{syn} & : & \textit{Syn} \\ \text{cont} & : & \textit{Cont} \end{bmatrix}$

where *Syn* is the type in (16).

(16) $\sigma : \textit{Sign}$ iff $\sigma : \begin{bmatrix} \text{s-event} & : & \textit{SEvent} \\ \text{syn} & : & \textit{Syn} \\ \text{cont} & : & \textit{Cont} \end{bmatrix}$

Here a sign consists of two fields correponding to the two parts of a Saussurean sign (de Saussure, 1916), speech event and content, and an additional field for syntax, providing a category and a string of signs as daughters. *Sign* is defined as a basic type whose witnesses are witnesses of the record type given rather than directly as that record type. This is because signs can contain other signs (the daughters) and the witnesses of the type *Sign* need to be defined inductively rather than have the type itself be a non-well founded object.

2.7 Frames

Record types have also been used to represent frames in the sense of FrameNet (Ruppenhofer et al., 2006) and in the sense of Barsalou (1992)) and the more recent developments in frame theory represented in Löbner et al. (2021). For a discussion of frames in TTR see Cooper (2016) and Cooper (in prep, Chap. 5).

2.8 Types as models of concepts, memories and beliefs

We have talked of types as models of propositions. It is natural also to use types to model various kinds of "mental objects". If we think of concepts as types, then we can say that the concept is instantiated if there is a witness for the type. If we think of a memory as a type, then that memory is correct if there is (or was) a witness for the type. It is natural to think of beliefs as propositions and therefore, in our terms, types. A belief is true, then, if there is a witness for the type. Such a view demands that we think of how types could be represented in the brain. Some preliminary suggestions are made by Cooper (2019), arguing that we should consider types as implemented in the brain by patterns of neural activation (rather than as neural architecture). Thinking of mental objects as types, gives the types a dual nature. They can be used to classify the world as in the standard view of types but now in addition they can be used to classify mental states in terms of which types are represented in the mental states.

Let us consider how this idea might look in a little more detail. (For a fuller discussion see Cooper, in prep, Chap. 6.) We treat long term memory (or beliefs) as a type: a type of how the world would be if your memory is correct. An agent's, a, long term memory, $T_{\text{ltm}(a)}$, might be characterized as in (17).

(17) $T_{\text{ltm}(a)} =$
$$\begin{bmatrix} \text{id}_1 & : & \begin{bmatrix} \text{x}:\textit{Ind} \\ \text{e}:\text{boy(x)} \end{bmatrix} \\ \text{id}_2 & : & \begin{bmatrix} \text{x}:\textit{Ind} \\ \text{e}:\text{dog(x)} \end{bmatrix} \\ \text{id}_3 & : & \begin{bmatrix} \text{e}:\text{hug}(\Uparrow\text{id}_1.\text{x}, \Uparrow\text{id}_2.\text{x}) \end{bmatrix} \\ \dots & & \end{bmatrix}$$

Here we have only specified the first three fields of what would probably be a very large type. The '$\Uparrow$'s in the field labelled 'id$_3$' are notational sugar to indicate that the path being referred to is not within the immediate record type in which the path notation occurs but one record type higher up.

Thinking of memory as a record type in this way is similar to thinking of memory as a large mental file (Recanati, 2012) with many subfiles. On this view, the basic intuition about belief, is that a believes T (a type functioning as a proposition), that is, the type 'believe(a, T)' is witnessed ("true"), just in case $T_{\text{ltm}(a)} \sqsubseteq T$. This says that a believes T just in case any way the world could be that is consistent with a's memory would be of type T.

However, this basic intuition is not technically quite sufficient to do the job that we need. Suppose that T is our running example of a record type, repeated in (18).

(18) $\begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c_{boy}} & : & \mathrm{boy(x)} \\ \mathrm{y} & : & \mathit{Ind} \\ \mathrm{c_{dog}} & : & \mathrm{dog(y)} \\ \mathrm{e} & : & \mathrm{hug(x,y)} \end{bmatrix}$

Intuitively, we would want to say that a *does* believe T if T is (18) and a's long term memory is (17). But (17) is *not* a subtype of (18). In fact the two types have no witnesses in common because the labelling required by the two types is different. However, if we *relabel* (18) as indicated in (19) the appropriate subtype relation will hold.

(19) $\mathrm{x} \rightsquigarrow \mathrm{id_1.x}$, $\mathrm{c_{boy}} \rightsquigarrow \mathrm{id_1.e}$, $\mathrm{y} \rightsquigarrow \mathrm{id_2.x}$, $\mathrm{c_{dog}} \rightsquigarrow \mathrm{id_2.e}$, $\mathrm{e} \rightsquigarrow \mathrm{id_3.e}$

The relabelling here replaces paths in a type with new paths and thus is able to significantly alter the way in which types are structured, as in this case.

Thus we refine our characterization of what it means for a to believe T by saying that the type 'believe(a,T)' is witnessed just in case there is a relabelling, η, of T such that $T_{\mathrm{ltm}(a)} \sqsubseteq \eta(T)$. An important consequence of this is that if 'believe(a,T)' is witnessed, so is 'believe(a,T')' where T' is a relabelling of T. That is, labelling is irrelevant for identifying beliefs, even though the labelling is important for other purposes such as identifying anaphora DRT-style.

3 Semantic objects *vs* languages

Let us remind ourselves of the central goal of Montague's original semantic project. It was to show that there is in principle no difference between natural languages and formal languages defined by logicians in respect of the fact that it is possible to provide a model theoretic semantics defined on the syntax of natural languages (without first having to translate them into a formal language and characterize a model theoretic semantics for that). This is made explicit in the paper 'English as a Formal Language' (EFL) included in Montague (1974). In 'Universal Grammar' (UG) (also included in Montague, 1974) Montague present a rigorous framework showing how we can use a formal language to represent model theoretic objects and guarantee that translating natural language into this formal language induces a model theoretic semantics defined on the natural language syntax. He does this by composing a homomorphism from the natural language syntactic algebra to the formal language syntactic algebra with a homomorphism from the formal language syntactic algebra to an algebra of semantic objects. This composed homomorphism is from the NL syntactic algebra to the algebra of semantic objects. 'The Proper Treatment of Quanitification in Ordinary English' (PTQ) in (Montague, 1974) is an instance of the framework in UG. Translation into intensional logic is used to induce a model theoretic semantics defined directly on the syntax of English. This makes the presentation much easier to read than the explicit direct interpretation of English syntax into model theory in EFL. The direct interpretation of natural language syntax into the model theory is essential to Montague's original claim that natural languages are formal languages.

TTR introduces structured objects (in the sense that we have discussed) into the realm of semantic objects which play the role of Montague's model theoretic objects and eschews an intermediate language between the natural language syntax and the semantic objects. In this sense TTR adheres to Montague's original project as we have presented it here. There is, however, something puzzling about introducing structured semantic objects in this way: they begin to take on the kind of structure you might expect in syntactic expressions of a formal language. An example, of this is ptypes such as 'hug(b,d)' as discussed earlier. In TTR we also have conjunctive types ($T_1 \wedge T_2$), disjunctive types ($T_1 \vee T_2$) and negative types ($\neg T$). While record types do not have the kind of structure we normally see in a standard logic they do nevertheless have similar structures to those of feature matrices used in syntactic and phonological description.

The construction and inference operations we need to describe and relate structured objects like this seem to take on the syntactic nature of corresponding operations used in proof theory. TTR is not alone in this. It always seems to happen when structured objects are introduced into the semantic domain. Examples from the past are situation semantics (Barwise and Perry, 1983) and logical atomism (Russell, 1918, 1924). Given the normal assumption that model theory models aspects of the world, many find it problematic that the world takes on the structure of a language in this way and for this reason, perhaps, think that a traditional possible worlds semantics is more realistic — despite the intractability of possible worlds and the problems with intensionality.

There are alternatives to introducing structured objects among the objects in the semantic domain. One of these is to take a radical proof theoretic approach to semantics. According to this we think of semantic theory as providing a mapping from natural language to a proof theoretic language. There may, or may not, be a model theory associated with this language. If there is a model theory it is more concerned with the metalogical study of the proof theoretic language rather than a central component of the semantic theory for natural language. Semantics is seen as primarily involving a correct account of inference rather than associating natural language expressions with the right kind of semantic objects. Perhaps most of the work on a Martin-Löf style type theoretic approach to natural language semantics takes some version of this approach. While inference is undeniably central to semantics and lies at the heart of the motivation for the semantic objects associated with natural language expressions in a model theoretic approach, such a purely proof theoretic approach to inference appears to give up any hope of building a semantic theory which is related to how we perceive and interact with the world.

A second alternative is to introduce an intermediate semantic representation language between natural language syntax and the model theory. An example of this is the use of a discourse representation structure (DRS) language in the early versions of Discourse Representation Theory (Kamp and Reyle, 1993). The use of a Chomskyan logical form together with a formal semantics is another example of this strategy. The point of such theories is that the intermediate representation language introduces structure which is not present in the natural language but which appears to be necessary to facilitate semantic interpretation. Such an intermediate language, therefore, does not follow the discipline set out in Montague's UG and is thus not eliminable in the way that Montague's intensional logic is in PTQ. In effect the argument is that the intermediate language is a necessary part of the theory precisely because it does not meet the conditions inlvolving homomorphisms which is set up in UG. The claim that an intermediate language is necessary is, of course, interesting and non-trivial, but we should be clear that it is abandoning a central tenet of Montague's original project, namely that there is no significant difference between natural languages and formal languages in that they can both have a model theory defined recursively on their syntactic structure. It seems like we cannot give a semantics for natural language after all — we first have to translate it to another language which is suitable for model theoretic interpretation.

Here is a question for both of these alternatives: if we have to translate natural language into a formal language, L, in order to give a semantics for the natural language, why is it that we have not evolved to speak L rather than the natural language? Perhaps we can see Montague's insistance on giving a semantics directly on the structure of the natural language as marking him out as an early pioneer of natural logic (van Benthem, 1986, Chapter 6) albeit from a model theoretic rather than a proof theoretic perspective.

Why is it, then, that if we incorporate the kind of structured objects we need into our semantic universe, then these objects appear to take on aspects of structure similar to that of a language? I would like to turn this question around. Perhaps it is not that the objects take on aspects of structure of the language but rather that the language takes on aspects of the structure in terms of which we perceive the world. In TTR we think of the types as providing structured relations between objects in the world, independently of language. This view seems not unrelated to the relational interpretation of quantum theory (Rovelli, 2021) in which the world is structured in terms of observable relations. It seems attractive to say that our languages in certain respects reflect the structure of the world as we perceive it.

This raises the question as to whether the difference between a proof theoretic approach and a model theoretic approach with structured objects is one of philosophical perspective rather than a matter of empirical analysis. One might think that the interesting questions lie not so much in whether you choose a model theoretic or a proof theoretic approach but rather in which kinds of structure you need in order to achieve an adequate account of inference in natural language. This seems to be a reasonable claim.[2] However, there are some reasons which seem to make working with semantic objects preferable to working with expressions in a language.

One reason is the very general one that using semantic objects helps us not lose sight of the fact that the project involves accounting for interaction

[2]And one that I have made a number of times in an ongoing conversation with Ruth Kempson over the past thirty years or so.

with the world, for example, that we need to be talking about the individuation of objects in the real world in addition to making sure that certain expressions stand in appropriate inferential relations. Another reason is that using semantic objects keeps us honest about the exact nature of the structure we are proposing. It can sometimes be easy to write down expressions without being precise about the nature of the structure they encode, for example, whether variables are being used as variables over objects or variables over variables or whether the absence of parentheses is a notational abbreviation or an indication that parentheses are not present in the expression.

A more substantial reason perhaps is that using semantic languages can impose unnecessary or unwanted linguistic structure without us realizing that this is happening. We will take record types as an example of this. Consider a record type as in (20).

(20) $\left[\begin{array}{lll} \ell_1 & : & T_1 \\ \ell_2 & : & T_2 \end{array}\right]$

In general in the type theory community this record type would be notated as (21).

(21) $\{\ell_1 : T_1; \ell_2 : T_2\}$

As an object it is natural to think of this record type as a *set*, (22), as we have done in this paper.

(22) $\{\langle \ell_1, T\rangle, \langle \ell_2, T_2\rangle\}$

If the record type is a set of fields, then the order of the fields does not matter. On the other hand, if we think of the record type as an expression in a language, then it is natural to think of the fields as ordered. This means that there are two expression record types corresponding the one object record type as in (23).

(23) a. $\{\ell_1 : T_1; \ell_2 : T_2\}$
b. $\{\ell_2 : T_2; \ell_1 : T_1\}$

In general, then, thinking of the record types as expressions leads us into a considerable (and possibly undesirable) multiplication in the number of available record types. An argument for the ordering when thinking of record types as expressions in a language is that if T_2 depends on the field $\ell_1 : T_1$ then the ℓ_2-field must be ordered after the ℓ_1-field. This is a convention which is standardly followed in proof-theoretically based approaches to type theory. But it is just a convention on the order in which things are written down. Consider the two alternative conventions represented in (24).

(24) a. "Let n be a natural number. Consider $\mathrm{succ}(n)\ldots$"
b. "Consider $\mathrm{succ}(n)$, where n is a natural number..."

When we think of the type as a set it becomes clear that the relevant order involves the order in which the fields are added to the set in constructing it. We can only add a dependent field to a record type which already contains the field on which it depends. This is made clear in the definition of dependent record types given in Cooper (2012, in prep). This does not require us to think of the record type itself as an ordered set.

This might seem like a rather abstract discussion which does not seem to have significant consequences for actual semantic analysis. But note that this discussion points to the difference between record types and Σ-types discussed in 2.3, where it did have consequences for the inferences we can make.

4 Conclusion

In this paper we have discussed what it means to be a structured semantic object and the uses to which structured semantic objects can be put in TTR. In the previous section we discussed the relationship between using structured semantic objects and expressions in a language and suggested that we are presented with a three-way choice in building a semantic theory:

- importing proof theoretic techniques into the model theory
- going entirely proof theoretic
- have an intermediate language between natural language syntax and model theory (and thereby giving up on Montague's project of providing a semantics directly for natural languages)

I have indicated my preference for the first option. Whatever your choice, it does seem that some kind of structured objects or representations are necessary in order to be able to give an adequate semantics for natural languages.

Acknowledgments

The research reported in this paper was supported by a grant from the Swedish Research Council (VR project 2014-39) for the establishment of the Centre

for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg.

References

Nicholas Asher and Alex Lascarides. 2003. *Logics of conversation*. Cambridge University Press.

Lawrence W. Barsalou. 1992. Frames, concepts, and conceptual fields. In A. Lehrer and E. F. Kittay, editors, *Frames, fields, and contrasts: New essays in semantic and lexical organization*, pages 21–74. Lawrence Erlbaum Associates, Hillsdale, NJ.

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

Jon Barwise and John Perry. 1983. *Situations and Attitudes*. Bradford Books. MIT Press, Cambridge, Mass.

Johan van Benthem. 1986. *Essays in Logical Semantics*. Springer Netherlands.

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, pages 271–323. Elsevier BV. General editors: Dov M. Gabbay, Paul Thagard and John Woods.

Robin Cooper. 2016. Frames as Records. In Annie Foret, Glyn Morrill, Reinhard Muskens, Rainer Osswald, and Sylvain Pogodalla, editors, *Formal Grammar: 20th and 21st International Conferences FG 2015, Barcelona, Spain, August 2015, Revised Selected Papers FG 2016, Bozen, Italy, August 2016, Proceedings*, number 9804 in LNCS, pages 3–18. Springer.

Robin Cooper. 2019. Representing Types as Neural Events. *Journal of Logic, Language and Information*, 28(2):131–155.

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 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.

Gottlob Frege. 1918/1919. Der Gedanke. Eine logische Untersuchung. *Beiträge zur Philosophie des deutschen Idealismus*, 1:58–77.

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

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic*. Kluwer, Dordrecht.

Staffan Larsson. 2002. *Issue-based Dialogue Management*. Ph.D. thesis, University of Gothenburg.

Sebastian Löbner, Thomas Gamerschlag, Tobias Kalenscher, Markus Schrenk, and Henk Zeevat, editors. 2021. *Concepts, Frames and Cascades in Semantics, Cognition and Ontology*. Springer International Publishing.

Per Martin-Löf. 1984. *Intuitionistic Type Theory*. Bibliopolis, Naples.

Richard Montague. 1974. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven. Ed. and with an introduction by Richmond H. Thomason.

Bengt Nordström, Kent Petersson, and Jan M. Smith. 1990. *Programming in Martin-Löf's Type Theory*, volume 7 of *International Series of Monographs on Computer Science*. Clarendon Press, Oxford.

François Recanati. 2012. *Mental Files*. Oxford University Press.

Carlo Rovelli. 2021. *Helgoland*. Penguin Books Ltd (UK).

Josef Ruppenhofer, Michael Ellsworth, Miriam R.L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2006. FrameNet II: Extended Theory and Practice. Available from the FrameNet website.

Bertrand Russell. 1918. The Philosophy of Logical Atomism. *The Monist*. Reprinted in (Russell, 1956).

Bertrand Russell. 1924. Logical Atomism. In J. H. Muirhead, editor, *Contemporary British Philosophy*. Routledge. Reprinted in (Russell, 1956).

Bertrand Russell. 1956. *Logic and Knowledge: Essays 1901–1950*. George Allen & Unwin. Edited by Robert C. Marsh.

Ivan A. Sag, Thomas Wasow, and Emily M. Bender. 2003. *Syntactic Theory: A Formal Introduction*, 2nd edition. CSLI Publications, Stanford.

Ferdinand de Saussure. 1916. *Cours de linguistique générale*. Payot, Lausanne and Paris. edited by Charles Bally and Albert Séchehaye.

Anil Seth. 2021. *Being You: a New Science of Consciousness*. Faber and Faber.

Philip Wadler. 2015. Propositions as Types. *Communications of the ACM*, 58(12):75–84.

Contributed Papers

On the Dual Interpretation of Nouns as Types and Predicates in Semantic Type Theories

William Babonnaud
Loria, Université de Lorraine, CNRS, Inria Nancy Grand-Est, Nancy, France
william.babonnaud@loria.fr

Abstract

In order to mediate the debate on whether common nouns are better interpreted as types or as predicates in type-theoretical semantic frameworks, the present paper shows that type theories whose models in category theory are *toposes* have access to a property drawing a one-to-one correspondence between first-order predicates and base types, thus enabling more flexibility for common noun interpretation. Using this flexibility and linguistic arguments based on negative predications, a subsequent proposal is made to interpret nouns as predicates with refined argument types.

1 Introduction

Any formal semantic framework can be characterised by its underlying type theory. One of the smallest possible theories for this purpose is Church's simple theory of types, built on two base types *e* and *t* for entities and truth values and an arrow constructor to create function types, which is characteristic of the grammar of Montague (1973). Yet the integration of lexical semantics into formal frameworks, motivated by the recognition of complex phenomena such as polysemy, selection restrictions and transfers of meaning (see e.g. Nunberg, 1979, 1995; Cruse, 1986), resulted in the use of richer type theories whose typical organisation consists in a subdivision of the type *e* into a hierarchy of *subtypes*, ordered by a subtyping relation, which intuitively enables the categorisation of entities according to some of their properties. In practice however, no consensus has been reached on what types should inhabit such a hierarchy. As listed by Retoré (2014), the collections proposed over the years vary from a set of around ten base types to a large system of one type for each common noun in the language.

This last position, initially introduced by Ranta (1994), have been more recently updated and defended in the works of Luo and Chatzikyriakidis (Luo, 2012a; Chatzikyriakidis and Luo, 2017), who argue that this so-called *CNs-as-types* approach provides a straightforward and efficient interpretation of common nouns (CNs). Under this view, the predication of a noun, e.g. *man*, on some entity x is represented as the typing judgement $x : man$. It is thus opposed to the more traditional and dominant *CNs-as-predicates* approach, which is directly inherited from Montague grammars and consists in interpreting *man* as a predicate **man** : $e \to t$, in such a way that the same predication as above is interpreted as the logical formula $\mathbf{man}(x)$. Those alternative possibilities are actually parts of a larger conceptual shift from classical predicate calculus to systems based on Modern Type Theories (MTTs), among which appear Martin-Löf's type theory (Martin-Löf, 1984) and Luo's unified theory of dependent types (UTT) (Luo, 1994; Luo et al., 2013)[1].

Representing CNs as types rather than predicates follows the idea of distinguishing between nouns on the one side, and adjectives and verbs on the other. Types are thus interpreted as collections of entities, which then represents ranges of significance for propositional functions in a Russellian sense (see sect. 2.12 of Ranta, 1994). But what if such types could be defined as ranges of other functions? As suggested by Retoré (2014), some theories may indeed allow a dual interpretation, that is, a correspondence between types jugdements such as x : *man* and truth on some predicate $\mathbf{man}(x)$. If Retoré is right, it could give access to a large amount of intermediate positions between CNs-as-types and CNs-as-predicates. This opens the debate on which way of interpreting CNs is the best-suited for natural language semantics, if any one is. Answering this question will have consequences on

[1]The various features that distinguish MTTs from Montague semantics will not be discussed here, see (Ranta, 1994) and (Luo, 2012a) for details.

the composition of the type hierarchy, as it varies in size and expressiveness depending on the choosen interpretation for CNs.

The present paper aims at mediating the debate between types and predicates by showing that, under the right settings, Retoré's suggestion—call it *CNs-as-both*—is an unavoidable property of some type theories, including UTT in its extension with coercive subtyping (Luo et al., 2013). As a result, the way of interpreting CNs and the type hierarchy are not constrained by the type theory itself, but would be a matter of lexicon design. We start defending this idea in Sect. 2 by highlighting the fact that the CNs-as-predicate approach stays relevant even in presence of subtyping, therefore nuancing Luo's position in (Luo, 2012b; Chatzikyriakidis and Luo, 2017). Then, in Sect. 3 we discuss the properties of UTT in light of its model in category theory, and establish that it has access to the CNs-as-both property. Finally, linguistic arguments to support an intermediate position between CNs-as-types and CNs-as-predicates, through the analysis of negation in sentences, is provided in Sect. 4.

2 Accomodating Predicates with Subtyping

In order to show that intermediate positions between types and predicates for CN interpretation are suitable, it is first necessary to examine the suitability of the CNs-as-predicate approach in the kind of type theories we are interested in. Recall that the type theories considered here are assumed to use a type hierarchy ordered by a subtyping relation. We will further assume the existence of a type constructor $\bullet$ to account for inherent polysemy (Pustejovsky, 1995), such that if a and b are base types, then the *dot type* $a \bullet b$ is considered a subtype of both a and b[2]. Hence, a predicate-based approach must be usable in presence of subtyping and dot types.

Yet its bad interaction with subtyping is one of the main criticisms addressed against the CNs-as-predicates approach, as demonstrated by Luo (2012b): consider the two types *phys* and *info* and the two words *book* and *heavy*, whose semantic interpretations would be a predicate $[\![\text{book}]\!] = \textbf{book} : \mathit{phys} \bullet \mathit{info} \to t$ and a second-order predicate $[\![\text{heavy}]\!] : (\mathit{phys} \to t) \to (\mathit{phys} \to t)$. Then, the application of *heavy* on *book* shall raise on the semantic side the condition that *phys* must be a subtype of $\mathit{phys} \bullet \mathit{info}$, which is the converse of the natural subtyping relation. Thus a simple application of an adjective to a CN in such a setting seems to be a rather complex task indeed. Luo concludes therefore that Montagovian grammars with their traditional CNs-as-predicates approach are unusable to deal with subtyping.

If the conclusion is right for Montagovian grammars, it has however to be nuanced when considering other frameworks of Montagovian inspiration, such as the Type Composition Logic (TCL) of Asher (2011) or the Montagovian Generative Lexicon (MGL) proposed by Retoré and his colleagues (Bassac et al., 2010; Retoré, 2014), as they often include additional strategies to overcome this subtyping problem in compositionality. In particular, we shall state that the demonstration above relies upon two hidden assumptions: (i) that direct application with subtyping is the only available operation for term composition, and (ii) that adjectives are necessarily interpreted as second-order predicates. The next paragraphs illustrate how new compositional strategies challenge these assumptions and enable the construction of subtyping-compliant frameworks.

Let us start with the functorial strategy proposed by Asher (2011) for TCL in cases where a dot type is involved, for instance when applying *heavy* to *book* as above. The intended meaning of the dot type $\mathit{phys} \bullet \mathit{info}$ is to represent two different aspects of *book*: one where the book is seen as a physical instance with a cover and pages, and another one involving only its informational contents. When combining *book* with *heavy*, we intend the latter to select only the physical aspect of the former, which licenses in a semantic framework a transformation of either $[\![\text{book}]\!]$ or $[\![\text{heavy}]\!]$—depending on the presuppositions to account for. TCL integrates these transformations as functors F and G which override subtyping constraints at application time by sending $[\![\text{heavy}]\!]$ to $F([\![\text{heavy}]\!]) : (\mathit{phys} \bullet \mathit{info} \to t) \to (\mathit{phys} \bullet \mathit{info} \to t)$ and **book** to $G(\textbf{book}) : \mathit{phys} \to t$ respectively, thus enabling a more flexible account of term composition which still respects subtyping since the functors are restricted for use with dot types only[3].

[2]Note that this type constructor is definable in UTT with coercive subtyping as explained in (Luo, 2010).

[3]Actually, dot types are treated differently than subtyping in Asher's framework because the projections from the dot type to its aspects are not necessarily assimilable to subtyping relations from a theoretical point of view, see chap. 5 of (Asher, 2011) for discussion.

The implementation of such a strategy clearly rules out assumption (i), as it provides an alternative way of dealing with compositionality in particular cases. Another kind of compositional strategy, to be found e.g. in MGL, uses type polymorphism to get similar flexibility. We shall go further on this path by examining how polymorphism enables us to dismiss assumption (ii) as well. This assumption arises from the Montagovian tradition of seeing adjectives as noun modifiers, whence the second-order predicate interpretation given above. The semantic interpretation of *heavy*, for instance, is then obtained by a term of the following shape:

$$[\![\text{heavy}]\!] = \lambda P\, x.\, \mathbf{heavy}(x) \wedge P(x) \qquad (1)$$

It involves a predicate **heavy** : $phys \to t$, which has the same type structure than the noun argument, and is embedded in a higher-order term with a logical conjunction to get the noun-modifier behaviour. Compared to other syntactical categories such as nouns or verbs, this interpretation of adjectives is singularly complex, not forgetting the difficulties it raises when interacting with subtyping. The semantic separation between nouns and adjectives is motivated by linguistic evidence that only nouns seem to bring support for quantification and counting, and the CNs-as-types approach as introduced by Ranta (1994) follows this idea. However, many grammarians and linguists have also pointed out the similitudes between these syntactical categories, thus supporting the possibility of treating nouns (or more accurately substantives) and adjectives as a continuum, where they would ultimately be distinguished by their predispositions with regard to linguistic functions and meaning specialisation (Jespersen, 1924; Guillaume, 1973; Gardelle, 2007).

If this continuum view is correct, it challenges the modern perception of adjectives as noun modifiers, not directly at the syntactic level where it stays relevant, but at the semantic one. Under the CNs-as-predicate approach, nouns are predicated of entities, and so could be adjectives. As a consequence, we could imagine a framework where the semantic interpretations of nouns and adjectives are of the same type shape. In the case of *heavy*, this means moving from the interpretation given in (1) above to its predicate component **heavy** : $phys \to t$ as the new term for $[\![\text{heavy}]\!]$. Composition would then be performed thanks to another polymorphic term, for instance in MGL or in the framework proposed by Babonnaud and de Groote (2020), which extends Montagovian grammars with records for modeling dot types, bounded polymorphism for composition, and a coercion-inference algorithm to account for subtyping. Assuming the integration of the latter framework in a syntax-semantic interface, the application of *heavy* to *book* would require to treat the deep syntactic relation between the adjective and the noun as the bounded polymorphic operator given in (2):

$$\begin{aligned}&\Lambda\alpha.\lambda P\, Q\, x.P(x) \wedge Q(x) :\\ &\quad \forall\alpha < e.(\alpha \to t) \to (\alpha \to t) \to (\alpha \to t) \qquad (2)\end{aligned}$$

The application of this operator to both $[\![\text{heavy}]\!]$ and $[\![\text{book}]\!]$ would trigger the inference algorithm, which would obtain a solution to the type constraints by unifying α with $phys \bullet info$ and introducing the coercion $c : phys \bullet info < phys$ to accomodate the variable to the predicate **heavy**. Thus, the resulting term would be $\lambda x.\mathbf{heavy}(c(x)) \wedge \mathbf{book}(x) : phys \bullet info \to t$.[4] Here again, the framework seems to be able to deal with subtyping while applying the CNs-as-predicates approach.

It should be noticed from the discussion in the two previous paragraphs that dismissing the assumption (ii) necessarily entails dismissing (i) as well, because traditional Montagovian grammars cannot support directly the combination of adjectives and nouns if they have the same type shape. Yet, the examples discussed above show that the addition of properties and features in frameworks of Montagovian inspiration enables them to accommodate the use of subtyping relation and dot types with the use of predicates. As a matter of fact, there is enough theoretical support to cope with the difficulties between subtyping and the CNs-as-predicates approach, which means that those difficulties cannot be an argument to definitively rule out the predicate view.

3 Theoretical Support to the Duality of Types and Predicates

3.1 On the Necessity of CNs-as-both

We now turn to the central question of this paper: is there any theoretical reason to choose one way

[4]It may be surprising in this approach that the resulting type is $phys \bullet info \to t$ and not $phys \to t$, since the former is not a subtype of the latter. This is not a problem for composition if we assume that other higher-order predicates also work with bounded polymorphic operators, as done by Babonnaud and de Groote (2020). Moreover, we may see such a type as coherent with the idea that *heavy book* still provides the possibility of copredication constructions.

of interpreting CNs rather than the other? Both CNs-as-types and CNs-as-predicates approaches have practical drawbacks—difficulties for modeling negation for the former, complex interaction with subtyping for the latter—and none of them is characteristic of a particular type theory: the Dependent Type Semantics of Bekki (2014) is an example of framework based on a MTT but interpreting CNs as predicates, and conversely it is not hard to conceive a Montagovian-style framework using types for CNs. The possibilities offered by the CNs-as-both property may enable the reconciliation of these views under a unified setting. In this section, we shall highlight that there actually are theoretical considerations supporting such a property in many frameworks.

Recall that having the CNs-as-both property means that for any entity x, we have for instance the typing judgement $x : \mathit{man}$ if and only if $\mathbf{man}(x)$ is true. Yet Chatzikyriakidis and Luo (2017) point out that such a definition in some settings (including simple type theory) may threaten the decidability of type checking. The reason lies in the "if" part of the equivalence: if one assert that the composition $\mathbf{man}(x)$ is well-typed, it results in $x : e$, and proving $x : \mathit{man}$ requires to prove the truth of the predications; but the truth of logical formulae is generally undecidable. However, Chatzikyriakidis and Luo exploit themselves the decidable direction of this duality—from types to predicates—for modeling negation: they introduce what they call a *predicational form* of typing judgement, that is, for any type, say *man*, a corresponding predicate $p_{\mathit{man}} : \mathit{man} \to t$ such that if $x : \mathit{man}$ then $p_{\mathit{man}}(x) = \texttt{true}$, where $\texttt{true}$ is a tautological proposition[5]. In this case, the problem of type checking is avoided by the definition, since the well-typedness of $p_{\mathit{man}}(x)$ necessarily entails $x : \mathit{man}$.

The very fact that such a predicational form is needed even in a framework following the CNs-as-types approach is indicative of the practical interest that the CNs-as-both property could have. Yet we do not want to just define for each type a corresponding predicate which is true if and only if its argument is of the right type because it could threaten the decidability of some parts of a semantic analysis. Moreover, if we want such a property to be usable in a MTT-based framework, it would be better to provide a constructivist account of this duality. To establish such a result, we propose here to take a step forward in the path of type theory abstraction by studying which model UTT with coercive subtyping can receive in category theory.

[5] For clarity and other reasons which will become clearer later in this section, we amalgamate the classical type of truth values and the intuitionistic type of propositions under the same notation *t*.

3.2 A Categorical Perspective on UTT

To the best of the author's knowledge, computational linguistics have only made a sparse use of category theory and its results compared to other domains of computer science[6]. Through discussing the acceptability of the CNs-as-both property, we shall also illustrate how the categorical interpretation of type theories may help to design a framework suitable for natural language semantics. However, as the present paper cannot bring a full account of the definitions and properties useful to build a categorical model of UTT, we will stick to the fundamentals of category theory, and the curious reader is invited to consult other sources such as (Goldblatt, 1979; MacLane and Moerdijk, 1992; Johnstone, 2002) for more definitions and results.

A *category* is a collection of *objects* and, for each pair A, B of objects, a set of *morphisms* from A to B, obeying two additional laws. The convenient notation $f : A \to B$ is used to express the fact that f is a morphism from A to B; A and B are then respectively called *domain* and *codomain* of f. The additional laws are the following: first, there is an operation of composition on morphisms which sends $f : A \to B$ and $g : B \to C$ to the morphism $g \circ f : A \to C$ and is associative; and second, there is for any object A a morphism $id_A : A \to A$, called identity of A, which is neutral for right and left compositions. Two other categorical notions will be used in the rest of this paper: in any category, a *terminal object* is an object 1 such that for any object A there is a unique morphism $1_A : A \to 1$, and a *monomorphism* is a monorphism $f : A \to B$ such that for any object X and pair of morphisms $g, h : X \to A$, the equality $f \circ g = f \circ h$ implies $g = h$. We will use the notation $f : A \rightarrowtail B$ to indicate that f is a monomorphism from A to B. For illustration, a well-known example of category is **Set**, whose objects are all the possible sets and whose morphisms are functions between them, with the common definitions of composition and identity function. Furthermore

[6] Notable exceptions include (La Palme Reyes et al., 1994; Coecke et al., 2010; Asher, 2011).

the terminal objects of **Set** are the singletons, and its monomorphisms are exactly the injective functions[7].

A type theory may be interpreted as a category whose objects are types and morphisms are mappings between the corresponding types, enriched with additional properties as counterparts to each constructor or feature of the type system. Thus, a type a will correspond to an object A, a term of type $a \to b$ will be represented as a morphism $A \to B$ and, as a special case, an entity $x : a$ will correspond to a morphism $1 \to A$, assuming that a terminal object exists in the category[8]. Notable correspondences hold between cartesian closed categories and typed λ-calculus (Lambek, 1980), and between locally cartesian closed categories and Martin-Löf type theories (Seely, 1984)[9]. As for Luo's UTT with coercive subtyping, it can be captured by another kind of category called *topos*. Indeed, as explained by Luo (1994), the underlying logic of UTT is a higher-order one, and Lambek and Scott (1986) showed that the categories generated by such type theories are precisely toposes. Toposes already emerged in Asher's (2011) categorical model for TCL as suitable (and even necessary) for interpreting dot types, and Babonnaud (2019) further argues that toposes could be the best categorical models to interpret on a unified basis a large variety of semantic frameworks with subtyping.

The definition of a topos includes several key properties that will not be exhaustively listed here[10]. For our purposes, it is enough to know that the relevance of toposes for semantic models comes from the existence in these categories of a particular object Ω called *subobject classifier*, which can be interpreted as the categorical counterpart for the type t of truth values or propositions—depending on the underlying logic of the chosen type system. The subobject classifier is characterised by a specific morphism $\top : 1 \to \Omega$ which represent the value *true* (or the tautological proposition $\texttt{true}$), and a universal property which binds it to the existence of monomorphisms which, as argued by Babonnaud (2019), can interpret subtyping relations. This property is formally embodied into the following Ω-axiom (Goldblatt, 1979):

Ω-axiom. *For every monomorphism $f : B \rightarrowtail A$ there is a unique morphism $\chi_f : A \to \Omega$ such that:*

(i) $\chi_f \circ f = \top \circ 1_B$; and

(ii) for any object C and morphism $g : C \to A$ such that $\chi_f \circ g = \top \circ 1_C$, there is a unique morphism $h : C \to B$ such that $f \circ h = g$.[11]

This axiom, conjointly with the property of toposes to have all finite limits (see Goldblatt, 1979; MacLane and Moerdijk, 1992; Johnstone, 2002), have an important consequence once transposed in a semantic type system. Assume a topos $\mathcal{T}$ with a distinguished object E corresponding to e. A first-order predicate such as **man** : $e \to t$ is interpreted as a morphism $man : E \to \Omega$. The properties of $\mathcal{T}$ ensure that there exists an object M along with a monomorphism $c : M \rightarrowtail E$[12] such that $man \circ c = \top \circ 1_M$, that is, there exists a type *man* in the system such that $x : man$ entails $\mathbf{man}(c(x)) = \texttt{true}$. Conversely, one can also prove that if $y : e$ is such that $\mathbf{man}(y) = \texttt{true}$, then there is an $x : man$ such that $y = c(x)$[13]. As a result, the Ω-axiom is a theoretical support in toposes for the duality between predicates and types, and as the categorical model generated by UTT is a topos, we conclude that this axiom and its consequences are also part of UTT.

3.3 Translation of Topos Properties into UTT

To know that the Ω-axiom exists in UTT is one thing, but understanding how this axiom manifests itself is another one which shall be clarified here. But before exploring in more details its practical consequences in the theory, let us give a few words about the predicational form of typing judgement proposed by Chatzikyriakidis and Luo (2017). It is

[7]Note that not all categories may have terminal objects and monomorphisms. Moreover, this example shows that terminal objects are defined up to isomophism, so that the notation 1 and reference to *the* terminal object of a category are valid by misuse of language.

[8]The type-theoretical counterpart of the terminal object 1 is generally the unit type.

[9]The reader may consult (Bell, 2012) among others for history and details on categories and their equivalence with type theories.

[10]While not directly linked to the present discussion, it may be worth noticing at least that toposes have all the properties of the kinds of categories mentioned above, that is, cartesian closed and locally cartesian closed categories. As a consequence, a topos is also acceptable as a model for simple type and Martin-Löf's theories.

[11]This also entails $1_B \circ h = 1_C$, but it is already true by definition of the terminal object. Category theorists shall recognise in this property the defintion of B as the *pullback* of χ_f and $\top$.

[12]The object M is then said to be a *subobject* of E, whence the name of *subobject classifier* given to Ω.

[13]It is so because variables of type e are interpreted as morphisms $1 \to E$. The result then comes from direct application of property (ii) of the Ω-axiom

not hard to see that the corresponding morphism of p_a in the topos model $\mathcal{T}$ of UTT is $\top \circ 1_A : A \to \Omega$: not only this morphism has the expected domain and codomain, but if we take a morphism $x : 1 \to A$ corresponding to an entity of type a, we can also prove that $\top \circ 1_A \circ x = \top$, which is the desired property[14]. Then we notice that this morphism $\top \circ 1_A$ is exactly the kind of morphism that appears in the right-hand side of the equalities in the Ω-axiom. As a consequence, an interpretation of the axiom is the following: if we are given a subtyping coercion $c : a \leq b$, then we can find a predicate $\chi_c : b \to t$ such that $\chi_c \circ c = p_a$.

But what kind of predicate would be χ_c? Can we define it in a constructivist way? Given that UTT, as shown by Luo (1994), allows for building higher-order propositions, we can answer by the affirmative. Assume a constant $c : a \to b$ corresponding to the coercive subtyping $a \leq b$. Then, pose the following:

$$\chi_c := \lambda y : b.\, \exists x : a.\, (c(x) = y) : b \to t \qquad (3)$$

The existentially quantified part of this term is a definable proposition in UTT. Thus χ_c is defined as the "characteristic" of a in b, such that χ_c is true on y if and only if y is in the image of the coercion function c. This shows how the conversion from type to predicate works in such type theories. As for the reverse direction of the type-predicate duality, if we are given any predicate $P : b \to t$, then following Seely (1984) the corresponding type may be defined as:

$$a := \Sigma x : b.\, (P(x) = \mathtt{true}) \qquad (4)$$

where Σ denotes a dependent sum[15]. In other words, a as defined in (4) is the type of pairs (x, p) where $x : b$ and p is a proof that $P(x)$ is true. As declared by Luo (2010), the first projection π_1 of such a sum is indeed a subtyping coercion, that is, $\pi_1 : a \leq b$ as required.

The formulations in (3) and (4) are sound in the sense that if $c : a \leq b$ is a coercion then we have a proof of $x : a$ if and only if we have a proof of $x : \Sigma y : b.(\chi_c(y) = \mathtt{true})$, and if $P : b \to t$ is a predicate and $\pi : (\Sigma x : b.P(x) = \mathtt{true}) \to b$ is the corresponding subtype coercion, we have a proof of $P(x) = \mathtt{true}$ if and only if we have a proof of $\chi_\pi(x) = \mathtt{true}$. Hence there is a way to move from types to predicates: for instance, if $x : \mathit{man}$ and $c : \mathit{man} \leq e$, it suffices to define $\mathbf{man} = \lambda x.\exists y.(c(y) = x) : e \to t$ to have a corresponding predicate with $\mathbf{man}(x) = \mathtt{true}$.

We should however point out that the categorical model presented in this paper, as well as the properties described, do not suffice to ensure that the corresponding type theory has key properties such as normalisation and decidability, nor that its implementation would be easier[16]. Moreover, as Chatzikyriakidis and Luo (2017) warned, we still have a threat to type-checking decidability when trying to move from predicates to types since proving $P(x) = \mathtt{true}$ in a MTT may be as hard as proving the truth of $P(x)$ in classical predicate calculus. To overcome this difficulty, a solution could be to adapt the type system so that the types of predicate arguments are themselves subtypes of e in order to make type checking more precise, as shall be discussed in the next section.

4 A Linguistic Perspective on Base Types

4.1 Type Theory and the Lexicon

Let us start with the following observation: if a is the type interpretation of some CN, then any b such that $a \leq b \leq e$ defines a possible predicational interpretation of this CN, the extreme cases being respectively Chatzikyriakidis and Luo's predicational form $p_a : a \to t$, and regular Montagovian predicates $e \to t$. A topos type theory therefore provides a full range of predicate interpretations for CNs which only depends on the types we accept in the hierarchy between their direct type interpretations and the greatest base type e. Besides, we shall notice that the Ω-axiom applies to *any* predicate, which may include for instance adjectives and intransitive verbs: we may then obtain some unexpected types such as the type *heavy* of heavy entities. Hence we may end up with an unreasonable amount of types and a large variety of possible interpretations for predicates—on top of the type checking difficulty.

[14]The complete proof runs as follows: by associativity of the composition, $(\top \circ 1_A) \circ x = \top \circ (1_A \circ x)$, and $1_A \circ x$ is a morphism $1 \to 1$. Yet, by definition of 1, there can be only one morphism $1 \to 1$, which is its identity id_1. Hence, $\top \circ (1_A \circ x) = \top \circ id_1 = \top$ because the identity is neutral for composition.

[15]As mentioned in footnote 11, this works because by Ω-axiom the object A corresponding to a is the pullback of $P : B \to \Omega$ and $\top$.

[16]As an anonymous rewiever rightly highlighted, the categorical interpretation of dependent types of Seely (1984) suffered from coherence issues that may be treated by a careful work on models, has discussed e.g. in (Curien et al., 2014).

We claim that the key solution for this issue is a proper definition of the *lexicon*, whose idea follows the lines of Pustejovsky and Batiukova (2019): a mapping between linguistic lemmas and their semantic representation, along with all relevant data for compositionality. It manifested earlier in this paper as the bracket application $[\![.]\!]$ sending a word to its typed interpretation. Facing the profusion of possible interpretations of a word in our type theory, a carefully-designed lexicon behaves as a filter which selects one (sometimes two) interpretation to be used. Thus we are not committed to use all the theoretically possible types and predicates, but only a proper subset thereof—without ruling out the properties of the type system. In other words, the lexicon may provide the basis to define a decidable fragment of a type theory by restriction to the terms using the constants it carries, without need to apply the type-predicate duality anymore within that fragment.

This puts a new highlight on the question of what types inhabit the hierarchy: the problem does not lie in the type theory proper, but rather at the interface between language and semantic representation, that is, in the lexicon. The main concern of lexicon design is therefore to choose types and predicates that are relevant for distinguishing between straightforward cases of semantic composition and other phenomena such as coercions, and this choice should obey the following criterion: types must be precise enough to distinguish the various cases of composition, but not too precise if this precision is not relevant for compositional matters. As such, we intend to separate the notion of type from the notion of meaning to the extent that the involvement of types in a semantic analysis is narrowed to the compositional behaviour of words, abstracting upon the other dissimilarities they may have.

4.2 Types in Compositionality and Negative Predications

To explain this latter idea, consider for instance the words *cat* and *dog* and their corresponding interpretations under the CNs-as-types approach. What does distinguish them from a compositional point of view? As a matter of fact, both words are very similar to that extent: lots of predicates which are meaningful on one are also meaningful on the other, as to be seen with physical descriptions, actions or even moods. Only a few words seem to resist such an analysis, among which the verbs corresponding to their respective cries, *meow* and *bark*. Yet even the compositional power of these predicates w.r.t. *cat* and *dog* is questionable: how meaningless is it to apply e.g. *meow* to *dog*? We are prone to think that a meowing dog is an absurdity, and that *meow* should be only employed with cats, hence a typing restriction **meow** : $cat \to t$. Let however constrast this view with the sentences in (1) below:

(1) a. Dogs do not meow but bark.
  b. #Tables do not meow but bark.

The sentence (1a) is obviously true, while (1b) is anomalous. Now, if we forget about the second verb in both sentences, we have to admit that the resulting ones should still differ in meaning, since tables and dogs fail to meow for different reasons. Moreover, reversing the verbs would turn (1a) to falsity but would keep (1b) anomalous, which hints that a meaningful predication of *meow* to *dog* could be possible. As a consequence, we shall recognise that the negative predications in (1a) and (1b) have different underlying logics.

Actually, as discussed by Horn (1989), many philosophers have recognised at least two forms of negation which, following the terminology of Sommers (1965, 1971), will be called here *negation* and *denial*. The subtle difference between them can be illustrated by the sentences in (2), where it is integrated in the distinction between the use of *not* for negation versus the prefix *un-* for denial, and results in a divergence in semantic acceptability:

(2) a. Triangles are not intelligent.
  b. #Triangles are unintelligent.

Their distinction is a matter of application level and presupposition: denial applies to predicates so that in a meaningful predication either the predicate or its denial is true on the argument, while negation applies directly to propositions and do not obey this excluded middle clause. Thus (2b) is anomalous because triangles can be neither intelligent nor unintelligent, hence a meaningless predication. By constrast, (2a) is acceptable under the reading which states precisely that *intelligent* (or its denial) cannot apply to triangles.

However, for predicates like *bark* and *meow* both negation and denial are rendered by the particle *not*, leading to an ambiguity in the negative predications in (1). Yet those sentences show that these predicates belong to the same "meaning scale" where they contrast each other, similarly to the opposition between *intelligent* and *unintelligent*. We may reasonably assume that all predicates from such a meaning scale are meaningful on the same arguments; from which we may conclude that (1a) contains a denial, while (1b) contains a pure negation. As a result, *meow* applies meaningfully to *dog* and meaninglessly to *table*, a distinction capturable with semantic types by extending the span of *meow* from *cat* to a greater argument type which includes *dog* and *cat*, but excludes *table*. If the type *animate* fits such a role, then we end up with a new interpretation $[\![\text{meow}]\!] = \textbf{meow} : \textit{animate} \to t$ in the lexicon.

The analysis above, if correct, should be reproduceable on any predicate which seems to distinguish *cat* and *dog* from a compositional point of view, including the nouns themselves in their predicative use if we accept that sentences in the sort of *"cats are not dogs"* are actually denials. From this generalisation, we conclude that the types *cat* and *dog* are not needed in our lexicon, since all predicates are blind to the distinction they introduce w.r.t. compositionality. Their interpretations, instead of relying on those types, would use the supertype *animate* in predicates **cat** : $\textit{animate} \to t$ and **dog** : $\textit{animate} \to t$, in such a way that any entity determined to be a dog or a cat would receive by type-checking the type *animate* in question, which is sufficient for further predications. Any other kind of difference between cats and dogs is not a matter of compositionality anymore, and should rather be accounted for in deeper semantic or pragmatic analyses.

4.3 Interpreting Negation in Type Theories

The previous discussion raises the issue of interpreting negation in semantic type theories. Chatzikyriakidis and Luo (2017) propose a polymorphic operator NOT rather than the usual connective $\neg$ to ensure consistency with their predicational forms[17], since well-typedness of $\neg p_{man}(x)$ enforces the contradictory condition $x : \textit{man}$ whereas $\text{NOT}(p_{man}, x)$ does not. However, this interpretation is not sufficient to distinguish negation from denial because predicational forms are too restrictive: in particular, $\text{NOT}(p_{dog}, x)$ has the same meaning regardless of x being of type *cat* or *table*.

Taking a broader predicate interpretation for *dog* may solve this problem as well-typedness of $\neg\textbf{dog}(x)$ holds for $x : \textit{animate}$, and $\textit{cat} < \textit{animate}$. We may thus interpret denials using $\neg$, and keep the operator NOT for negations, so that $\text{NOT}(\textbf{dog}, x)$ would mean that the type of x is uncompatible with the argument type of **dog**. It appears then that NOT enables us to transpose the type-theoretic property of type incompatibility into a logical formula. Another option for negation could be to use the most general predicate $\textbf{dog}' : e \to t$ while using **dog** only for denials, replacing $\text{NOT}(\textbf{dog}, x)$ by $\neg\textbf{dog}'(k(x))$ with k an adequate coercion to e, but that would require an additional mechanism to introduce such a predicate when needed, for we cannot be committed to have this general interpretation available in the lexicon.

5 Conclusion

A first observation about the interpretation of CNs is the fact that neither types nor predicates seem to offer a greater practical advantage: in both cases, adding little theoretical machinery into the framework enables their interaction with subtyping and dot types in a fairly straightforward way. In the case of predicates however, this interaction comes at the cost of reconsidering composition, as it generally requires new mechanisms that go beyond direct application. Nevertheless, such a revision in compositionality may be an unavoidable step towards better interpretations of complex semantic phenomena anyways.

Yet the main observation of the present paper is the fact that any type theory with enough assumptions can actually model both views of interpreting CNs in a equivalent way, by establishing a bijective correspondence between predicates of type $e \to t$ and subtypes of e. Theories with this property include MTTs such as Martin-Löf type theory and Luo's UTT, and further considerations show that many other type-theoretical frameworks can be extended to get access to this property as well. The discussion in Sect. 3 shows how abstracting type theories through the perspective of category theory helps in identifying and establishing their key properties. The present paper particularly highlighted the correspondence between the expected duality of interpretations and a general property of the categorical class of *toposes*.

[17] In UTT, the operator NOT has the polymorphic type $\Pi\alpha : \text{CN}.(\alpha \to t) \to (e \to t)$ (by assimilating propositions to t and objects to e), where CN is the type universe of CN types. Note that this universe seems to correspond conceptually to the collections of subtypes of e. As a consequence, types of the form $\Pi\alpha : \text{CN}.\tau$ and of the bounded polymorphic form $\forall\alpha < e.\tau$ from (Babonnaud and de Groote, 2020), as used in Sect. 2, may be seen as equivalent.

This duality gives access to intermediate choices of interpretation for CNs which blur the lines between the strict applications of each view. In Sect. 4, we proposed an interpretation of CNs as predicates with refined argument types, whose choice relies on their compositional abilities with other predicates from the language, and the possible arguments thereof. This refinement requires to determine on which entities it is meaningful to assert or deny the given predicate, thus excluding the entities on which the predication is absurd. In negative sentences, this amounts to be able to dissociate pure negation and denials. Generalising such a reasoning on a natural language should eventually lead to the construction of a lexicon using a proper sub-hierarchy of the possible types, each type corresponding to some cluster of CNs with the same compositional behaviour.

References

Nicholas Asher. 2011. *Lexical Meaning in Context: A Web of Words*. Cambridge University Press.

William Babonnaud. 2019. A topos-based approach to building language ontologies. In *Formal Grammar. 24th International Conference, FG 2019, Riga, Latvia, August 11, 2019, Proceedings*, pages 18–34, Berlin. Springer.

William Babonnaud and Philippe de Groote. 2020. Lexical selection, coercion, and record types. In *LENLS17: Logic & Engineering of Natural Language Semantics, Online, November 15-17, 2020*.

Christian Bassac, Bruno Mery, and Christian Retoré. 2010. Towards a type-theoretical account of lexical semantics. *Journal of Logic, Language, and Information*, 19(2):229–245.

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

John L. Bell. 2012. Types, sets, and categories. In Dov M. Gabbay, Akihiro Kanamori, and John Woods, editors, *Sets and Extensions in the Twentieth Century*, volume 6 of *Handbook of the History of Logic*, pages 633–687. North-Holland Publishings.

Stergios Chatzikyriakidis and Zhaohui Luo. 2017. On the interpretation of common nouns: Types versus predicates. In Stergios Chatzikyriakidis and Zhaohui Luo, editors, *Modern Perspectives in Type-Theoretical Semantics*, volume 98 of *Studies in Linguistics and Philosophy*, pages 43–70. Springer.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36:345–384.

David A. Cruse. 1986. *Lexical Semantics*. Cambridge University Press.

Pierre-Louis Curien, Richard Garner, and Martin Hofmann. 2014. Revisiting the categorical interpretation of dependent type theory. *Theoretical Computer Science*, 546:99–119.

Laure Gardelle. 2007. Adjectifs dits 'substantivés' et noms composés : quel continuum entre adjectif et nom ? In *Journée d'études concours – Université Jean Moulin Lyon III*, Lyon, France.

Robert Goldblatt. 1979. *Topoi: The Categorial Analysis of Logic*, volume 98 of *Studies in logic and the foundations of mathematics*. North-Holland Publishings.

Gustave Guillaume. 1973. *Principes de linguistique théorique*. Les Presses de l'Université Laval, Québec.

Laurence R. Horn. 1989. *A Natural History of Negation*. The University of Chicago Press.

Otto Jespersen. 1924. *The Philosophy of Grammar*. George Allen & Unwin Ltd., London.

Peter T. Johnstone. 2002. *Sketches of an Elephant: A Topos Theory Compendium*. Oxford University Press.

Marie La Palme Reyes, John Macnamara, and Gonzalos E. Reyes. 1994. Reference, kinds and predicates. In John Macnamara and Gonzalo E. Reyes, editors, *The Logical Foundations of Cognition*, volume 4 of *Vancouver Studies in Cognitive Science*, pages 91–143. Oxford University Press.

Joachim Lambek. 1980. From λ-calculus to cartesian closed categories. In J. Roger Hindley and Jonathan P. Seldin, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 375–402. Academic Press, London.

Joachim Lambek and Philip J. Scott. 1986. *Introduction to Higher Order Categorical Logic*. Cambridge University Press.

Zhaohui Luo. 1994. *Computation and Reasoning: A Type Theory for Computer Science*. Oxford University Press.

Zhaohui Luo. 2010. Type-theoretical semantics with coercive subtyping. In *Proceedings of SALT 20*, pages 38–56.

Zhaohui Luo. 2012a. Common nouns as types. In *Logical Aspects of Computational Linguistics. 7th International Conference, LACL 2012, Nantes, France, July 2-4, 2012, Proceedings*, pages 173–185, Berlin. Springer.

Zhaohui Luo. 2012b. Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy*, 35(6):491–513.

Zhaohui Luo, Sergei Soloviev, and Tao Xue. 2013. Coercive subtyping: Theory and implementation. *Information and Computation*, 223:18–42.

Saunders MacLane and Ieke Moerdijk. 1992. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer, New York. Corr. 2nd edition 1994.

Per Martin-Löf. 1984. *Intuitionistic type theory: Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980*. Bibliopolis, Napoli.

Richard Montague. 1973. The proper treatment of quantification in ordinary english. In Patrick Suppes, Julius Moravcsik, and Jaakko Hintikka, editors, *Approaches to Natural Language*, pages 221–242. Reidel, Dordrecht.

Geoffrey Nunberg. 1979. The non-uniqueness of semantic solutions: Polysemy. *Linguistics and Philosophy*, 3(2):143–184.

Geoffrey Nunberg. 1995. Transfers of meaning. *Journal of Semantics*, 12(2):109–132.

James Pustejovsky. 1995. *The Generative Lexicon*. MIT Press, Cambridge.

James Pustejovsky and Olga Batiukova. 2019. *The Lexicon*. Cambridge University Press.

Aarne Ranta. 1994. *Type-Theoretical Grammar*. Oxford University Press.

Christian Retoré. 2014. The montagovian generative lexicon ΛTy_n: a type theoretical framework for natural language semantics. In *Proceedings of the 19th International Conference on Types for Proofs and Programs*, volume 26 of *LIPICS*, pages 202–229.

Robert A. G. Seely. 1984. Locally cartesian closed categories and type theory. *Mathematical Proceedings of the Cambridge Philosophical Society*, 95(1):33–48.

Fred Sommers. 1965. Predicability. In Max Black, editor, *Philosophy in America*, pages 262–281. Cornell University Press, Ithaca.

Fred Sommers. 1971. Structural ontology. *Philosophia*, 1:21–42.

Improving DRS Parsing with Separately Predicted Semantic Roles

Tatiana Bladier[1], **Gosse Minnema**[2], **Rik van Noord**[2], **Kilian Evang**[1]
(1) University of Düsseldorf, Germany
(2) University of Groningen, the Netherlands
{tatiana.bladier, evang}@hhu.de
{g.f.minnema, r.i.k.van.noord}@rug.nl

Abstract

This paper addresses Semantic Role Labeling (SRL) within the context of English Discourse Representation Structure (DRS) parsing. In particular, we investigate whether semantic roles predicted by a near-state-of-the-art SRL model can be used to improve the outputs of modern end-to-end neural DRS parsers using a rule-based post-processing algorithm. We compare two methods of generating training data for the SRL model from the Parallel Meaning Bank, one DRS-based and one CCG-based. We also compare two different post-processing algorithms. Our results vary across different DRS parsers, but overall we find a small to moderate improvement of up to 0.5 F1 on the final DRSs. We find a small but consistent advantage of DRS-based over CCG-based training data generation, and of token-based over concept-based post-processing, where applicable.

1 Introduction

With the increasing availability of multi-layered semantically annotated corpora, semantic parsing today is typically approached as an end-to-end task of predicting a meaning representation in one go, including information on word senses, predicate-argument structure, scope, semantic roles, and more. Since each of these layers is complex in its own right, it might be beneficial to rely on multiple specialized components to separately predict individual semantic layers, and to combine their output. In this paper, we focus on separately predicting semantic roles in the context of Discourse Representation Structure (DRS) parsing.

DRSs are meaning representations grounded in Discourse Representation Theory (Kamp and Reyle, 1993). We use the English part of the Parallel Meaning Bank (PMB; Abzianidze et al., 2017), which contains sentences annotated with DRSs. Figure 1 shows an example. Events (e.g., `e1`) are related to their participants (e.g., `x1`, `x2`) via semantic roles (e.g., `Theme`, `Destination`) from the VerbNet/LIRICS inventory (Bonial et al., 2011). Semantic roles are a crucial aspect of meaning since they encode how each entity participates in an event (Fillmore, 1968).

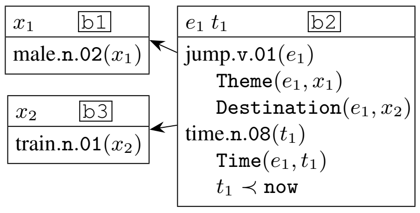


Figure 1: DRS for *He jumped into the train* (source: PMB, document `00/2759`)

Semantic role labelling (SRL) is typically approached as a task of labeling tokens or parse tree edges with predicate/role labels, independently of other aspects of meaning (e.g., Li et al., 2019, 2020b; Shi et al., 2020; Marcheggiani and Titov, 2020; Li et al., 2020a). Conversely, DRS parsers such as Evang (2019); Fancellu et al. (2020); van Noord et al. (2020); Liu et al. (2021) do not have dedicated SRL modules but predict a complete meaning representation of which roles are one part. In this paper, we explore the possibility of combining semantic parsers with a dedicated SRL system. The main research question we seek to answer is: can we in this way obtain DRSs with more accurate semantic roles?

Our approach is summarized in Figure 2: we first convert the PMB training data into a standard SRL annotation format (§2) in order to train a near-state-of-the-art SRL system on it (§3). At test time, we merge the output of DRS parsers with that of the SRL system using a rule-based post-processing algorithm (§4), aiming to produce a more accurate final DRS. We experiment with applying our procedure on top of several recent DRS parsing systems, and find that, albeit with some caveats, our procedure leads to overall better scores (§5). [1]

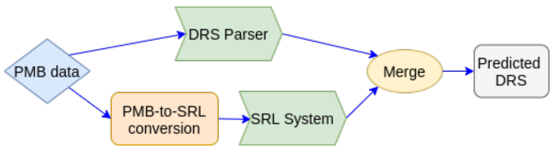


Figure 2: System overview

2 DRS-to-SRL Conversion

Before we can train an SRL system, we first need to convert semantic role annotations in the PMB to a more standard SRL format. Two characteristics of the PMB make this a non-trivial task. First, role annotations in the PMB are *predicate-based*, meaning that roles are carried by predicates instead of by arguments, as in standard SRL systems. Table 1 illustrates this: in standard SRL, the Theme role would be marked on *he*. Instead, in the PMB, the role is annotated on *jumped*, the predicate assigning the role; in a later step, the DRS parser makes sure that the role is associated to the discourse referent introduced by "He". Second, prepositional and adverbial roles (e.g. *into the train*, *slowly*) are treated differently from "core" semantic roles: they are carried by the preposition or adverb itself, instead of by the verbal predicate they are associated to.

Token	*He*	*jumped*	*into*	*the*	*train*
PMB		Theme	Destination		
SRL: head	Theme	PRED			Destination
SRL: span	Theme	PRED	{ ←	Destination	→ }

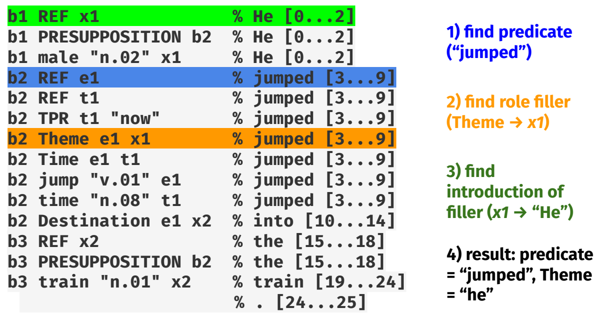
Table 1: PMB-style versus standard SRL annotations.

We experiment with two approaches for converting PMB role labels to a standard SRL format:

2.1 DRS-based conversion

Here, predicates and fillers for semantic roles are found via DRSs, which in the training data are *anchored*, i.e., most clauses are aligned to exactly one token. We extract predicate-role-filler triples such as ⟨jumped,Theme,he⟩ from the anchored DRSs by looking for role clauses such as `b2 Theme e1 x1` and then finding the clause introducing the filler (`b1 REF x1`, anchored to *He*), and the clause introducing the event (`b2 REF e1`, anchored to *jumped*). The process is illustrated in Figure 3.[2]

Disadvantages of this approach are 1) that it only yields the heads of the fillers, not full spans, and 2) that in some cases, the 'deep' semantic structure of the DRS does not directly match the surface realisations of the semantic roles we want to find. One example of the latter problem is found in sentences such as "She saw herself", where a DRS-based approach would return "She" as the Stimulus role, instead of "herself", which is the surface filler of this role but does not introduce a discourse referent of its own.



```
b1 REF x1                % He [0...2]
b1 PRESUPPOSITION b2     % He [0...2]
b1 male "n.02" x1        % He [0...2]
b2 REF e1                % jumped [3...9]
b2 REF t1                % jumped [3...9]
b2 TPR t1 "now"          % jumped [3...9]
b2 Theme e1 x1           % jumped [3...9]
b2 Time e1 t1            % jumped [3...9]
b2 jump "v.01" e1        % jumped [3...9]
b2 time "n.08" t1        % jumped [3...9]
b2 Destination e1 x2     % into [10...14]
b3 REF x2                % the [15...18]
b3 PRESUPPOSITION b2     % the [15...18]
b3 train "n.01" x2       % train [19...24]
                         % . [24...25]
```

1) find predicate ("jumped")
2) find role filler (Theme → *x1*)
3) find introduction of filler (*x1* → "He")
4) result: predicate = "jumped", Theme = "he"

Figure 3: Example of DRS-based conversion.

2.2 CCG-based conversion

The second approach aims at overcoming both limitations of the DRS-based approach by making use of the CCG derivations in the PMB. Here, predicates and fillers for semantic roles are found via the CCG (Categorial Combinatorial Grammar, Steedman 2000) syntax trees and predicate-based role annotations in the PMB.

Main conversion process First, we transform the CCG trees using the `pmb_ccg_to_term` module in the `LangPro` package (Abzianidze, 2017), removing directionality of the combinatory rules and reducing the number of possible combinators, which simplifies tree traversing. In particular, long-distance dependencies (such as *wh*-movement) are handled using the λ-operator, which introduces a relationship between two variables at different points in the tree. An example of this kind of tree is given in Figure 4.

[1]Code and data at https://github.com/TaniaBladier/DRS_Parsing_with_SRL

[2]The DRS in *clause notation* in Figure 3 is equivalent to the one in *box notation* in Figure 1, but additionally shows the alignment with tokens in the sentence.

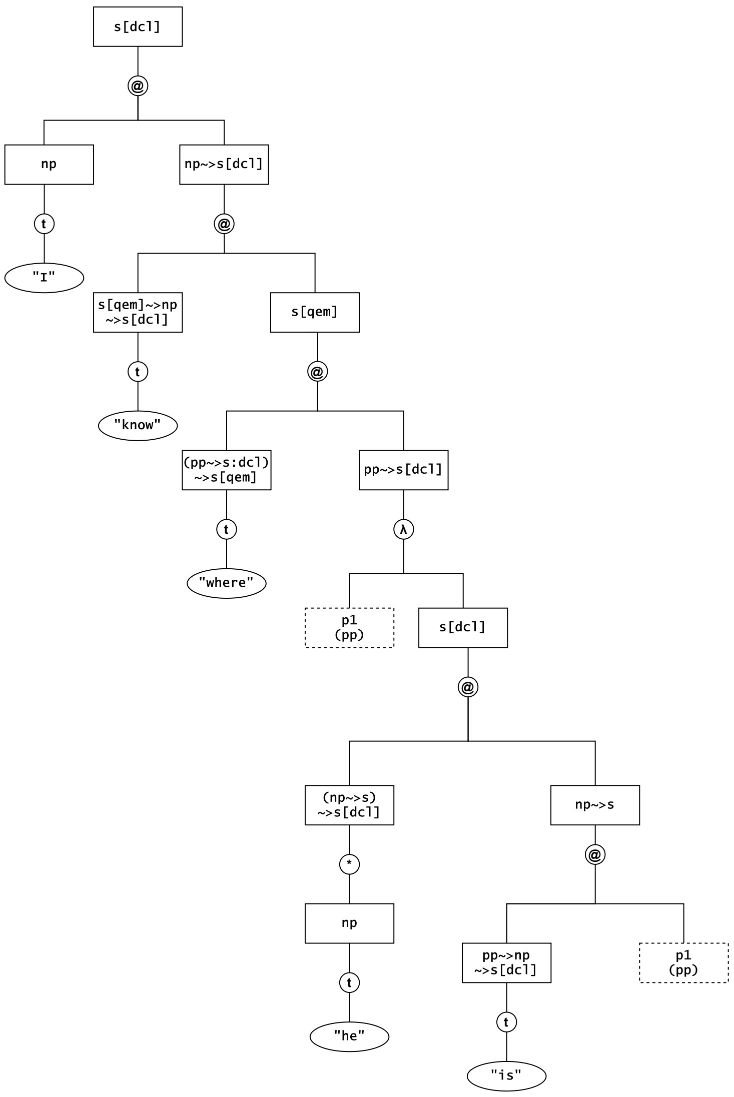


Figure 4: Simplified CCG tree with examples of all combinators (@: simple functional application; λ: variable introduction;, $*$: type-raising). Solid rectangles are types, circles are operators, dotted rectangles are lambda variables, and ovals are lexical nodes. `s[dcl]` means 'declarative sentence'; `s[qem]` means 'embedded question'.

Next, we deploy our role span extraction algorithm, which traverses the simplified tree and tries to match the semantic roles annotated on each predicate to the constituents filling these roles. Figure 5 displays a high-level overview of this process, showing how CCG arguments get mapped to constituents in the tree. This process is explained in more detail in Figure 6.

Given a simplified tree, we extract each predicate's syntactic roles from its CCG type signature and match them with the annotated semantic roles. For example, suppose *jump* has the type signature NP→S[3] and the role annotation `[Theme]`, then it has a single NP syntactic role, corresponding to a Theme semantic role. Then, we move upwards through the syntax tree, checking the type signature at every step; whenever we detect that a role has been filled, we process the constituent that was

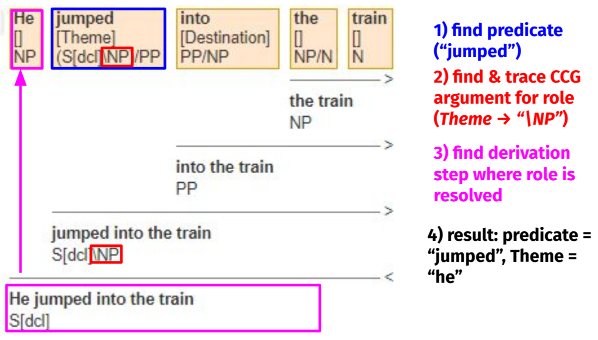

He
[]
NP
jumped
[Theme]
(S[dcl]\NP)/PP
into
[Destination]
PP/NP
the
[]
NP/N
train
[]
N
>
the train
NP
>
into the train
PP
>
jumped into the train
S[dcl]\NP
<
He jumped into the train
S[dcl]
1) find predicate ("jumped")
2) find & trace CCG argument for role (Theme → "\NP")
3) find derivation step where role is resolved
4) result: predicate = "jumped", Theme = "he"

Figure 5: Example of CCG-based conversion.

[3]The original CCG category would be S\NP, which we simplify into the direction-agnostic NP→S.

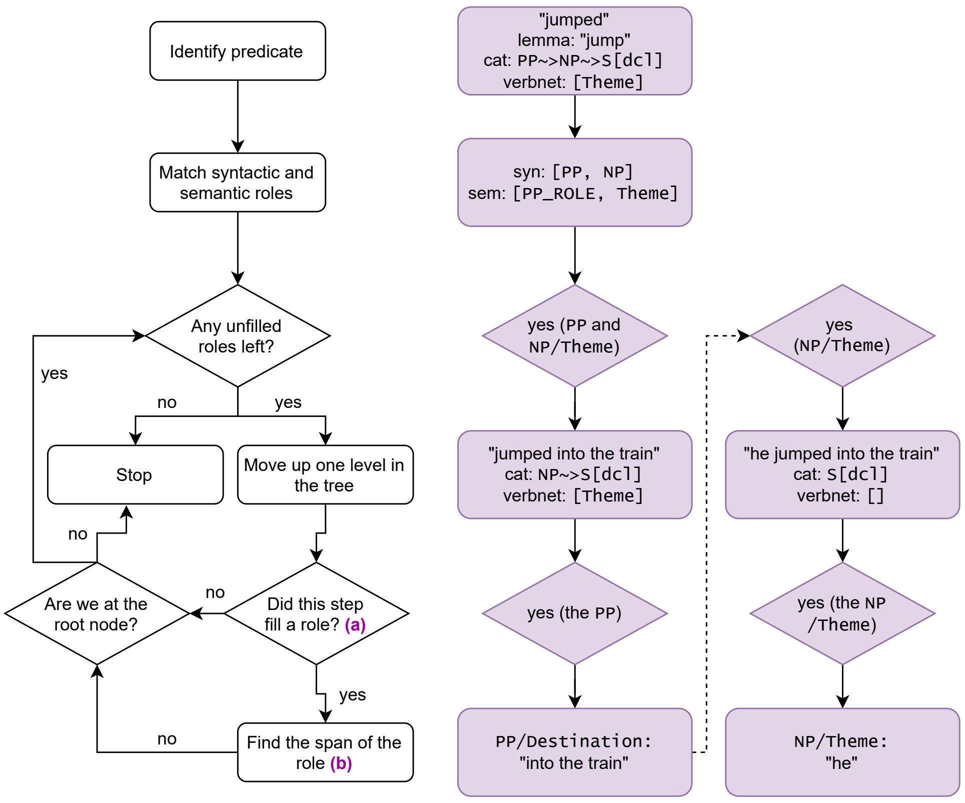

Identify predicate
Match syntactic and semantic roles
Any unfilled roles left?
yes
no
yes
Stop
Move up one level in the tree
no
Are we at the root node?
no
Did this step fill a role? (a)
yes
no
Find the span of the role (b)
"jumped" lemma: "jump" cat: PP~>NP~>S[dcl] verbnet: [Theme]
syn: [PP, NP] sem: [PP_ROLE, Theme]
yes (PP and NP/Theme)
yes (NP/Theme)
"jumped into the train" cat: NP~>S[dcl] verbnet: [Theme]
"he jumped into the train" cat: S[dcl] verbnet: []
yes (the PP)
yes (the NP /Theme)
PP/Destination: "into the train"
NP/Theme: "he"

Figure 6: Flow chart of the main CCG-based conversion process. Algorithmic steps in white, example in purple.

merged at that point of the tree as the filler of the corresponding semantic role. This process is repeated until we have found a filler for every role, or until we reach the top of the tree.[4]

Detecting merged constituents A crucial step of our process (step *(a)* in Figure 6) is detecting, given a particular node in the tree, whether a role has been resolved at that node. In many cases, this is straightforward; for example, in the sentence in Figures 5 and 6, we can see that *he* fills the NP/Theme role of *jump* at the point where *he* is combined with *jumped into the train* through simple functional application, changing the type signature from NP→S to S. In other cases, more complicated rules are needed, for example when dealing with to-clauses (*She wants me to leave*), where, on combining *wants me* with *to leave*, the type signature of *to leave* changes from NP→S[to] to NP→S[dcl]. In such cases, at first glance, it appears as if not much has changed except a change of clause type (from a *to*-clause to a declarative sentence), whereas in fact, *me* has filled the subject NP of *leave*, and a new NP argument (the subject NP of *wants*) has been added. We have developed a set of heuristics that cover all such difficult cases occurring in the gold annotations in the PMB. While we believe that this amounts to a wide general coverage, it is likely that there exist other constructions that our algorithm does not (yet) cover.

Once it has been defined that a role is resolved at a given node in the tree, the next crucial step (step *b* in Figure 6) is to find the correct role span within the constituent that was combined. In many cases (like *he* in *he jumped*), the entire constituent is the role filler, but in other cases (like *wants me* in *She wants me to leave*), only a part of the constituent (*me*) is the role filler that we are looking for. To find this constituent, we designed a separate algorithm that moves down the tree starting from the merged constituent, until an argument with the correct type is found.

PP and adverbial roles Semantic roles carried by PP constituents (e.g. *into the train*) or by adverbial phrases (e.g. *quickly*) pose an additional chal-

[4]In some cases, e.g. *wh*-questions, it is possible that some roles remain unfilled.

lenge, since, in the PMB annotation framework, these roles are annotated on the syntactic head of the PP or adverbial phrase (e.g. *into* in *into the train*) rather than on the verb that they combine with. In cases where the PP is a syntactic argument of the verb (as in *jump into the train*), we solve this by first adding a placeholder role (see the `PP_role` at the top of Figure 6) corresponding to the verb's PP argument, and then replacing this by the semantic role carried by the PP at the point where it is combined with the predicate. In cases where a PP or adverb is an adjunct (e.g. with type signature $\mathtt{S}\rightarrow\mathtt{S}$ or $(\mathtt{NP}\rightarrow\mathtt{S})\rightarrow(\mathtt{NP}\rightarrow\mathtt{S})$), we add the semantic roles introduced by the adjunct to the predicates in the constituent that is modified (e.g., *quickly* modifies *he ran* in *he ran quickly*. To ensure that adjuncts get the right scope, we added a constraint to our algorithm that forbids adding adjunct roles to predicates if doing so would cross a clause boundary; e.g., *loudly* in *he loudly said he was going to leave* can modify *said* but not *leave*.

Span-to-head conversion As a final step, to make the outputs of the CCG-based algorithm comparable to those of the DRS-based algorithm, we add a final step that converts the extracted role spans to their semantic heads. This algorithm consists of a set of (recursive) rules defining what the head of each type of phrase is. For example, $H(\text{the old woman}) = H(\text{old woman}) = H(\text{woman}) = \text{woman}$, where H is a function applying the appropriate rule for a given phrase type and returning the 'head part' of the phrase. There are many possible phrase types, but in general, the head of an NP is a noun, the head of a VP is a verb, the head of a PP is an NP, and the head of a sentence is the VP.

2.3 Comparing the approaches

Comparing the outputs of both conversion approaches, we find that 68% of documents match exactly, and 82% differ by at most one role. This shows that both approaches show significant differences worth further investigating. The differences mainly concern structural mismatches between syntax and semantics. For example, in sentences with co-referential NPs, CCG-based conversion gives more intuitive results than DRS-based conversion: in *she handed him*$_1$ *the money that she owed him*$_2$, DRS-based conversion treats the two *hims* as the same entity and assigns the Beneficiary role of *owe* to *him*$_1$, whereas CCG-based conversion correctly assigns it to *him*$_2$. Similarly, with reflexives, in *she saw herself*, DRS-based conversion is unable to assign any role to *herself*, since this word does not introduce a new discourse referent but refers back to *she*. The syntax-driven CCG-based conversion also allows for a better resolution of *hearer* and *speaker* discourse participants in such sentences as *I don't remember your name*.

On the other hand, CCG-based conversion has difficulties dealing with light verb constructions where the semantics of the main verb and the light verb interact. For instance, in *he had his wallet stolen*, the relationship between *he* and *stolen* is not detected. Finally, more heuristics will need to be added to CCG-based conversion to cover all adjunct semantic roles due to the way that these are annotated in the PMB, e.g. *by*-clauses in passive sentences. Also, the CCG-based conversion needs additional rules to distinguish between the semantic and syntactic head in such constructions as *all of the town* or *a kilo of plums*.

3 SRL Predictions

We predict semantic roles using the graph-based end-to-end coreference resolution system by He et al. (2018). This syntax-agnostic SRL model jointly predicts predicates, role fillers, and role labels. The SRL system builds contextualized representations for spans of arguments and predicate tokens based on BiLSTM outputs. The argument spans and predicates are predicted independently of each other and the aggressive beam pruning is used to discard the least probable combinations of predicate and argument spans. The output of the system is a graph, which lists predicted SRL roles as edges and the associated text spans as nodes. The SRL graph is predicted directly over text spans. Unlike He et al., we do not predict the full spans of semantic roles, but only syntactic heads of the semantic role spans, since the DRSs in the PMB do not contain information about full spans of arguments.[5] We experiment with GloVe (Pennington et al., 2014) and ELMo (Peters et al., 2018) embeddings to train the SRL system.[6]

We use the gold section of the English PMB data (release 3.0.0) to train and test the SRL system, which contains a train, dev, and test split of

[5]The full spans of semantic arguments can be reconstructed from head spans using syntactic information from dependency graphs (Gliosca and Amsili, 2019).

[6]The hyper-parameters are given in the appendix.

6 620, 885, and 898 documents, respectively. The SRL system is trained on the output of both DRS-to-SRL conversion tools separately. We include only verbal predicates and exclude the predicate *be* due to its inconsistent annotation in the PMB.

4 Merging DRS and SRL Predictions

As baseline DRS parsers without external SRL prediction, we use DRS parsers for which the output is publicly available: the transition-based compositional parser of Evang (2019) and three neural sequence-to-sequence models: the character-level model of van Noord et al. (2018b), an extension of this model that uses linguistic features (van Noord et al., 2019) and the best BERT-based model of van Noord et al. (2020). We refer to these models with E19, N18, N19, and N20.

We propose two methods for merging DRS and SRL output: a *token-based* method for parsers that are lexically anchored (each clause maps to one token), such as E19, and a *concept-based* method for parsers for which this is not the case (N18, N19, N20). Both methods only aim to *replace* roles in the DRS; no new full clauses are inserted.

Token-based merging When the SRL system predicts a predicate-role-filler tuple such as ⟨jumped,Theme,he⟩, we look for a corresponding role prediction in the parser output. A corresponding prediction is a role clause such as `b2 Agent e1 x1`, where the event discourse referent (`e1`) and the filler discourse referent (`x1`) are introduced by the corresponding tokens, i.e., *jumped*, and *he*, respectively. We say that a referent is introduced by a token if the token is anchored to a concept clause for that referent, such as `b2 jump "v.01" e1` or `b1 male "n.02" x1`. In this example, the DRS parser predicted a different role (Agent) than the SRL system (Theme), so we replace the former with the latter.

Concept-based merging Concept-based merging works similarly but does not rely on clauses being anchored to tokens. Instead, concept clauses are *matched* to tokens using corpus-level alignment and lemmatization. We say that a concept clause (e.g., `b1 male "n.02" x1`) *matches* a token (e.g., *he*) if it is observed anchored to the same word anywhere in the full PMB training data (bronze, silver, and gold). We also say that a concept clause (e.g., `b2 jump "v.01" e1`) matches a token (e.g., *jumped*) if there is a string match between the concept and the lemma[7] of the token (*jump*).

Restrictions In order to avoid some incorrect role replacements, we impose the following heuristics to restrict replacement: a role r is not replaced with r' if 1) r is one of the special roles `Time` and `Name`, 2) r' was predicted by the SRL system with $< 50\%$ precision, 3) r' already exists in the same box as r. For concept-based merging, the general concepts `person`, `be` and `entity` are never matched with any input tokens.

5 Experiments and Discussion

The main results of our experiments are shown in Table 2. Overall, we see small but consistent improvements for all parsers, except for N20, the most recent system. It seems that once the parser reaches a certain accuracy it is not straightforward to improve the scores by using an imperfect external system. This is also reflected by the number of replaced roles, which goes down as the parsers get better. Comparing the two conversion methods, we find that DRS-based conversion leads to higher scores. The difference with CCG-based conversion is small, though consistent between setups. In a sense, this is unsurprising given that DRS is also our target representation format. Furthermore, we found that using ELMo outperformed GloVe; while this is unsurprising, it supports the intuition that using a higher quality SRL system leads to more improvement. In other words, any development on the SRL parsing side is likely to lead to better performance on DRS parsing as well. Comparing token-based to concept-based merging on the output of the E19 parser (the only one where it is applicable), it makes more replacements and results in slightly higher accuracy, suggesting an advantage in terms of recall over concept-based merging.

Room for improvement As can be seen in Table 2, SRL performance seems to be a bottleneck; hence, using future, higher-quality SRL systems might also lead to better overall performance of our method. In particular, due to the merging step in our pipeline system, missing roles in SRL predictions are less costly than wrong predictions. Hence, we expect that SRL systems that are optimized for precision rather than for F-score will be more suited for use in our task. Furthermore, we

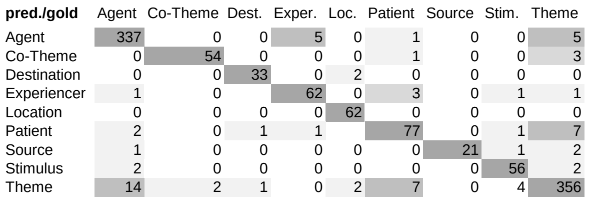
[7]We use spaCy (Honnibal et al., 2020) for this.

Experiments	SRL		E19-tok		E19		N18		N19		N20	
	dev	test	dev	test	dev	test	dev	test	dev	test	dev	test
Baseline	–	–	81.4 (0)	81.4 (0)	81.4 (0)	81.4 (0)	84.3 (0)	84.9 (0)	86.8 (0)	88.7 (0)	88.4 (0)	89.3 (0)
DRS conv.: upper	100	100	+1.5 (154)	+1.3 (144)	+1.3 (124)	1.2 (124)	+0.9 (92)	+1.2 (132)	+0.9 (88)	+1.1 (117)	+0.5 (51)	+0.7 (76)
CCG conv.: upper	100	100	+1.2 (145)	+1.2 (134)	+1.2 (115)	1.1 (118)	+0.9 (89)	+1.2 (129)	+0.8 (80)	+1.1 (114)	+0.5 (50)	+0.8 (78)
DRS conv. + GloVe	79.7	81.6	+0.3 (129)	+0.3 (113)	+0.4 (97)	+0.2 (102)	+0.2 (68)	+0.4 (92)	+0.1 (64)	+0.2 (90)	-0.2 (57)	-0.1 (70)
DRS conv. + ELMo	**85.8**	86.3	**+0.5** (128)	**+0.4** (120)	**+0.5** (104)	**+0.4** (110)	**+0.3** (73)	**+0.5** (107)	**+0.2** (74)	**+0.3** (104)	-0.1 (55)	0.0 (69)
CCG conv. + GloVe	80.7	83.0	+0.3 (129)	+0.3 (117)	+0.3 (107)	+0.2 (108)	+0.1 (96)	+0.4 (102)	0.0 (93)	+0.1 (103)	-0.2 (73)	-0.1 (74)
CCG conv. + ELMo	85.2	**87.0**	+0.4 (118)	**+0.4** (109)	+0.4 (99)	+0.3 (103)	+0.2 (81)	+0.4 (104)	+0.1 (73)	+0.2 (102)	-0.2 (63)	0.0 (66)

Table 2: Experiment results, including F-scores and number of replaced roles (in brackets). The F-scores are calculated using Counter (van Noord et al., 2018a). Scores for N19 and N20 are averaged over 5 runs. E19-tok uses token-based merging, E19 uses concept-based merging like the rest.

expect that further improvements in the conversion algorithms will lead to better overall performance.

Error analysis We identified four sources of errors in the SRL predictions. The data show an imbalanced role distribution towards the roles Theme and Agent, which take up 52% of all annotations out of 32 semantic roles. This leads to overprediction of these roles by the SRL-labeler. Indeed, for N20 we find that these roles have an insertion precision of $< 50\%$, or in other words, they were more often wrongly inserted than that they correctly replaced a non-matching role. Figure 7 shows the confusion matrix for the most frequent semantic roles.



pred./gold	Agent	Co-Theme	Dest.	Exper.	Loc.	Patient	Source	Stim.	Theme
Agent	337	0	0	5	0	1	0	0	5
Co-Theme	0	54	0	0	0	1	0	0	3
Destination	0	0	33	0	2	0	0	0	0
Experiencer	1	0		62	0	3	0	1	1
Location	0	0	0	0	62	0	0	0	0
Patient	2	0	1	1		77	0	1	7
Source	1	0	0	0	0	0	21	1	2
Stimulus	2	0	0	0	0	0	0	56	2
Theme	14	2	1	0	2	7	0	4	356

Figure 7: Confusion matrix for semantic labeling errors, showing the numbers of predicted labels for the most frequent labels.

The role Theme and Agent are also frequently predicted extra in cases where no semantic role should be predicted. For example, the pronoun *her* in the sentence *she ate her dinner* is erroneously assigned the role Agent. Semantic roles of prepositional phrases also lead to prediction errors. For example, the phrase *the field of biology* in the sentence *He is working in the field of biology* is wrongly recognized as Location instead of Theme. Another cause of prediction errors are possessive determiners which are wrongly predicted as role fillers. For example, both *her* and *dinner* are predicted as Patient in the following sentence: *She ate her dinner*. Also, no semantic roles are predicted by the SRL-labeler if the head word has no vector embedding due to a special character, for example like *post∼office*. Due to the merging step in our pipeline, the erroneously missing semantic roles in SRL predictions do not lead to a drop of parsing performance and also do not improve it.

6 Conclusions and Future Work

We have presented experiments on using externally predicted semantic roles to improve the output of four recent DRS parsers. We saw that there is considerable room for improvement and our method fills it – but not fully, especially as parsers get more accurate. We conclude that our approach is useful especially with parsers such as E19 which do not reach state-of-the-art accuracy but may have other advantages such as smaller models or lexical anchoring. An advantage of our approach is that it is very flexible: it can be applied on top of any DRS parsing model without having to alter or retrain the model itself. This means that our method, or an improved version of it, could also be applied to future DRS parsers, possibly with completely different architectures. In future work we intend to experiment with enhancing the SRL system using syntactic input from CCG-based supertags and also try out other SRL systems. We also plan to experiment with prediction of nominal and adjectival predicates along with their semantic roles. We also intend to reconstruct and predict full spans of semantic roles. Moreover, we plan to carry out parsing experiments with further languages in the PMB, including Dutch, German, and Italian, as our method should be universally applicable. Finally, it would be interesting to improve the SRL predictions by enforcing coherence of predicted predicates and corresponding semantic roles.

Acknowledgements

We would like to thank two anonymous reviewers for their valuable comments. The work presented in this paper has been partially funded by the European Research Council, within the ERC grant TreeGraSP[8].

References

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

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247, Valencia, Spain. Association for Computational Linguistics.

Claire Bonial, William Corvey, Martha Palmer, Volha V Petukhova, and Harry Bunt. 2011. A hierarchical unification of lirics and verbnet semantic roles. In *2011 IEEE Fifth International Conference on Semantic Computing*, pages 483–489. IEEE.

Kilian Evang. 2019. Transition-based DRS parsing using stack-LSTMs. In *Proceedings of the IWCS Shared Task on Semantic Parsing*, Gothenburg, Sweden. Association for Computational Linguistics.

Federico Fancellu, Ákos Kádár, Ran Zhang, and Afsaneh Fazly. 2020. Accurate polyglot semantic parsing with DAG grammars. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3567–3580, Online. Association for Computational Linguistics.

C.J. Fillmore. 1968. The case for case. In E. Bach and R. Harms, editors, *Universals in Linguistic Theory*. Holt, Rinehart and Winston, New York.

Quentin Gliosca and Pascal Amsili. 2019. Résolution de coréférence basée sur les têtes. In *Actes de la conférence TALN 2019 (articles courts)*, Toulouse.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, Melbourne, Australia. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic*. Studies in Linguistics and Philosophy. Kluwer, Dordrecht, Boston, London.

Tao Li, Parth Anand Jawale, Martha Palmer, and Vivek Srikumar. 2020a. Structured tuning for semantic role labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8402–8412, Online. Association for Computational Linguistics.

Zuchao Li, Shexia He, Hai Zhao, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2019. Dependency or span, end-to-end uniform semantic role labeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6730–6737.

Zuchao Li, Hai Zhao, Rui Wang, and Kevin Parnow. 2020b. High-order semantic role labeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1134–1151, Online. Association for Computational Linguistics.

Jiangming Liu, Shay B. Cohen, Mirella Lapata, and Johan Bos. 2021. Universal Discourse Representation Structure Parsing. *Computational Linguistics*, pages 1–33.

Diego Marcheggiani and Ivan Titov. 2020. Graph convolutions over constituent trees for syntax-aware semantic role labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3915–3928, Online. Association for Computational Linguistics.

Rik van Noord, Lasha Abzianidze, Hessel Haagsma, and Johan Bos. 2018a. Evaluating scoped meaning representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Rik van Noord, Lasha Abzianidze, Antonio Toral, and Johan Bos. 2018b. Exploring neural methods for parsing discourse representation structures. *Transactions of the Association for Computational Linguistics*, 6:619–633.

Rik van Noord, Antonio Toral, and Johan Bos. 2019. Linguistic information in neural semantic parsing with multiple encoders. In *Proceedings of the 13th International Conference on Computational Semantics - Short Papers*, pages 24–31, Gothenburg, Sweden. Association for Computational Linguistics.

Rik van Noord, Antonio Toral, and Johan Bos. 2020. Character-level representations improve DRS-based semantic parsing Even in the age of BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4587–4603, Online. Association for Computational Linguistics.

[8]https://treegrasp.phil.hhu.de/

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.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Tianze Shi, Igor Malioutov, and Ozan Irsoy. 2020. Semantic role labeling as syntactic dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7551–7571, Online. Association for Computational Linguistics.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Appendix

Layer	**Hyper-parameters**	**Value**
Characters CNN	numb. of filters	50
Bi-LSTM	state size # layers	200 3
Words embedding	vector dim.	300
Char. embedding	dimension	8
	batch size	40
Dropout	dropout rate	0.5
	Max. gradient norm	5.0
	Optimizer	Adam
	Learning rate	0.001
	Decay rate	0.999
	Decay frequency	100

Hyper-parameters of the SRL system.

Semantic Learning in a Probabilistic Type Theory with Records

Staffan Larsson **Jean-Philippe Bernardy** **Robin Cooper**
CLASP, Dept. of Philosophy, Linguistics and Theory of Science
University of Gothenburg, Box 200, SE 40530 Sweden
sl@ling.gu.se jean-philippe.bernardy@gu.se cooper@ling.gu.se

Abstract

We propose a probabilistic account of semantic learning from interaction formulated in terms of probabilistic type theory with records, building on Cooper et al. (2014, 2015); Larsson and Cooper (2021). Starting from a probabilistic type theoretic formulations of naive Bayes classifiers, we illustrate our account of semantic learning with a simple language game (the fruit recognition game).

1 Introduction

A probabilistic type theory was presented by Cooper et al. (2014) and Cooper et al. (2015), which extends Cooper's Type Theory with Records (TTR, Cooper, 2012a; Cooper and Ginzburg, 2015; Cooper, in prep). This theory, Probabilistic Type Theory with Records (ProbTTR) assigns probability values, rather than Boolean truth-values, to type judgements.

TTR has been used previously for natural language semantics (see, for example, Cooper (2005) and Cooper (2012a)), 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)). We believe that a probabilistic version of TTR could be useful in all these domains.

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. It is the latter goal that we pursue here.

In this paper we build on the account of probabilistic inference and classification in ProbTTR introduced by Larsson and Cooper (2021). There, a ProbTTR version of a random variable, not present in the work of Cooper et al. (2015), was introduced. It was also shown how probabilistic classification of perceptual evidence can be combined with probabilistic reasoning. By proposing a Bayesian account of semantic learning formulated in terms of probabilistic type theory, we connect probabilistic semantic learning to the modeling of perceptual meaning as classifiers.

In the following, we first provide a brief overview of TTR and Probabilistic TTR. Section 3 reviews the account of semantic classification presented by Larsson and Cooper (2021). Section 4 details a frequentist account of semantic learning in ProbTTR, and Section 5 provides an example of semantic learning. Section 6 concludes and discusses related and future work.

2 Background

This section reviews the background needed to follow the rest of the paper: TTR, Probabilistic TTR fundamentals, and Bayes nets and Naive Bayes classifiers.

2.1 TTR: A brief introduction

We will be formulating our account in a Type Theory with Records (TTR). We can here only give a brief and partial introduction to TTR; see also Cooper (2005), Cooper (2012b) and Cooper (in prep). To begin with, $s : T$ is a judgment that some s is of type T. One *basic type* in TTR is *Ind*, the type of an individual; another basic type is *Real*, the type of real numbers.

Next, we introduce *records* and *record types*. If $a_1 : T_1, a_2 : T_2(a_1), \ldots, a_n :$

$T_n(a_1, a_2, \ldots, a_{n-1})$, where $T(a_1, \ldots, a_n)$ represents a type T which depends on the objects $a_1, \ldots, a_n$, the record to the left in Figure 1 is of the record type to the right.

In Figure 1, $\ell_1, \ldots \ell_n$ are *labels* which can be used elsewhere to refer to the values associated with them. A sample record and record type is shown in Figure 2.

Types constructed with predicates may be *dependent*. This is represented by the fact that arguments to the predicate may be represented by labels used on the left of the ':' elsewhere in the record type. In Figure 2, the type of c_{man} is dependent on ref (as is c_{run}).

If r is a record and ℓ is a label in r, we can use a *path* $r.\ell$ to refer to the value of ℓ in r. Similarly, if T is a record type and ℓ is a label in T, $T.\ell$ refers to the type of ℓ in T. Records (and record types) can be nested, so that the value of a label is itself a record (or record type). As can be seen in Figure 2, types can be constructed from predicates, e.g., "run" or "man". Such types are called *ptypes* and can intuitively be thought of as types of situations. Such types of situations can be construed as propositions, following the "propositions as types" principle.

2.2 Probabilistic TTR fundamentals

The core of ProbTTR is the notion of a probabilistic judgement, where a situation s is judged to be of a type T with some probability.

(1) $p(s : T) = r$, where $r \in$ [0,1]

Such a judgement expresses a subjective probability in that it encodes an agent's take on the likelihood that a situation is of that type.

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 (2).

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

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.

(3) $p(\varphi$.sit:φ.sit-type)= φ.prob

We assume that agents track observed situations and their types, modelled as probabilistic Austinian propositions.

We use $p(T_1||T_2)$ to represent the probability that any situation s is of type T_1, given that s is of type T_2. Note that $p(T_1||T_2)$, is different from $p(T_1|T_2)$, the probability of there being something of type T_1 given that there is something of type T_2. We can refer to the former as the *bound variable* (or perhaps *universal*) conditional probability[1], and the latter as the *existential* conditional probability.

2.3 Bayesian nets and the Naive Bayes classifier

A Bayesian Network is a Directed Acyclic Graph (DAG). 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 for its random variables. Bayesian Networks provide graphical models for probabilistic learning and inference (Pearl (1990); Halpern (2003)).

A standard Naive Bayes model is a special case of a Bayesian network. More precisely, it 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 3 illustrates the relation between evidence types and class types in a Naive Bayes classifier.

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

(4)

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

where c is the value of C, e_i is the value of E_i $(1 \leq i \leq n)$ and

(5) $p(c \mid e_1, \ldots, e_n) =$

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

2.4 Random variables in TTR

Larsson and Cooper (2021) introduce a type theoretic counterpart of a random variable in Bayesian

[1]In Bayesian jargon, such conditional probabilities are often referred to as *likelihoods*.

$$\begin{bmatrix} \ell_1 & = & a_1 \\ \ell_2 & = & a_2 \\ \dots & & \\ \ell_n & = & a_n \\ \dots & & \end{bmatrix} : \begin{bmatrix} \ell_1 & : & T_1 \\ \ell_2 & : & T_2(l_1) \\ \dots & & \\ \ell_n & : & T_n(\ell_1, l_2, \dots, l_{n-1}) \end{bmatrix}$$

Figure 1: Schema of record and record type

$$\begin{bmatrix} \text{ref} & = & \text{obj}_{123} \\ \text{c}_{\text{man}} & = & \text{prf}_{\text{man}} \\ \text{c}_{\text{run}} & = & \text{prf}_{\text{run}} \end{bmatrix} : \begin{bmatrix} \text{ref} & : & \text{Ind} \\ \text{c}_{\text{man}} & : & \text{man(ref)} \\ \text{c}_{\text{run}} & : & \text{run(ref)} \end{bmatrix}$$

Figure 2: Sample record and record type

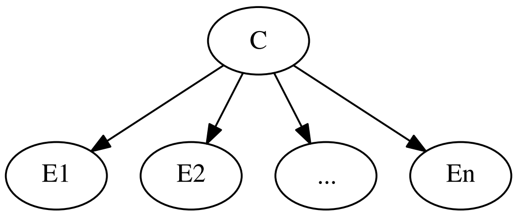


Figure 3: Evidence and Class in a Naive Bayes classifier

inference. To represent a single (discrete) random variable with a range of possible (mutually exclusive) values, ProbTTR uses a *variable type* V whose range is a set of *value types* $\mathfrak{R}(V) = \{A_1, \dots, A_n\}$ such that the following conditions hold.

(6) a. All value types for a variable type V are subtypes of V, formally $A_j \sqsubseteq V$ for $1 \leq j \leq n$

b. All value types for a given variable type V are disjoint, formally $A_j \perp A_i$ for all i, j such that $1 \leq i \neq j \leq n$

c. The probability of a situation s being of a variable type V is either 0 or 1, which is also the sum of the probabilities of s being of any of the variable value types , formally for any s, $p(s : V) \in \{0, 1\}$ and $p(s : V) = \sum_{T \in \mathfrak{R}(V)} p(s : T)$

(6)(c) encodes a conceptual difference between the probability that something has a property (such as colour, $p(s\text{:Colour})$), and the probability that it has a certain value of a variable (e.g. $p(s\text{:Green})$). If the probability distribution over different values (colours) sums to 1, then the probability that the object in question has a colour is 1. The probability that an object has colour is either 0 or 1. We assume that certain ontological/conceptual type judgements of the form "physical objects have colour" are categorical, and so have Boolean values.

2.5 Representing probability distributions

For a situation s, a probability distribution over the m value types $A_j \in \mathfrak{R}(\mathbb{A}), 1 \leq j \leq m$ belonging to a variable type $\mathbb{A}$ can be written (as above) as a set of probabilistic Austinian propositions, e.g.

$$(7)\quad \{ \begin{bmatrix} \text{sit} = s \\ \text{sit-type} = A_j \\ \text{prob} = p(s : A_j) \end{bmatrix} \mid A_j \in \mathfrak{R}(\mathbb{A}) \}$$

However, we will also have use for a vector representation of probability distributions, which is also more compact. If we assume $\mathfrak{R}(\mathbb{A})$ is an ordered set $\{A_1, \dots A_m\}$, we can define probability distribution $d_{\mathbb{A}}(s)$:

(8) $d_{\mathbb{A}}(s) = \langle p_1, \dots, p_m \rangle$ where $p_j = p(s : A_j)$ for $A_j \in \mathfrak{R}(\mathbb{A}), 1 \leq i \leq m$

2.6 A ProbTTR Naive Bayes classifier

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

(9) a. a collection of n evidence variable types $\mathbb{E}_1^\kappa, \dots, \mathbb{E}_n^\kappa$

b. n associated sets of evidence value types $\mathfrak{R}(\mathbb{E}_1^\kappa), \dots, \mathfrak{R}(\mathbb{E}_n^\kappa)$

c. a class variable type $\mathbb{C}^\kappa$, e.g. *Fruit*, 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.

Larsson and Cooper (2021) define a ProbTTR Bayes classifier κ as a function from a situation s (of the meet type of the evidence variable types $\mathbb{E}_1^\kappa, \ldots, \mathbb{E}_n^\kappa$) 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 $\mathbb{E}_1^\kappa, \ldots, \mathbb{E}_n^\kappa$. Formally, a ProbTTR NaiveBayes classifier is a function

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

such that if[2] $s : \mathbb{E}_1^\kappa \wedge \ldots \wedge \mathbb{E}_n^\kappa$, then

(11) $$\kappa(s)=\{\left[\begin{array}{l} \text{sit} = s \\ \text{sit-type} = C \\ \text{prob} = p^\kappa(s : C) \end{array}\right] \mid C \in \mathfrak{R}(\mathbb{C}^\kappa)\}$$

2.7 The fruit recognition game

Larsson and Cooper (2021) illustrate semantic classification using a Naive Bayes classifier in ProbTTR using the *fruit recognition game*. Later in this paper, we will build on this example to illustrate mentor-driven semantic learning.

In this game a teacher shows fruits to a learning agent. The agent makes a guess, the teacher provides the correct answer, and the agent learns from these observations.

We will use shorthands *Apple* and *Pear* for the types corresponding to an object being an apple or a pear, respectively[3]. Furthermore, we will assume that the objects in the fruit recognition game have one of two shapes (a-shape or p-shape, corresponding to types *Ashape* and *Pshape*) and one of two colours (green or red, corresponding to types *Green* and *Red*).

[2]Recall that for any s, $p(s : V) \in \{0, 1\}$ for any variable type V. Therefore, any type judgement regarding a variable type, such as that involved in the classifier function, can be regarded as categorical.

[3]For details, see Larsson and Cooper (2021).

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 4 shows the evidence and class types of the fruit recognition game in a simple Bayesian Network.

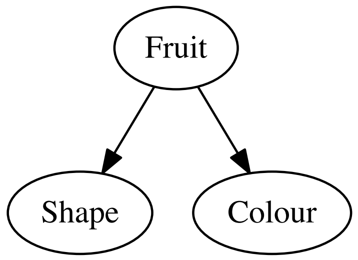


Figure 4: Bayesian Network for the fruit recognition game

For a situation s the classifier FruitC(s) returns a probability distribution over the value types in $\mathfrak{R}(\textit{Fruit})$.

(12) FruitC(s) =
$$\{\left[\begin{array}{l} \text{sit} = s \\ \text{sit-type} = F \\ \text{prob} = p^{\text{FruitC}}(s : F) \end{array}\right] \mid F \in \mathfrak{R}(\textit{Fruit})\}$$

3 Semantic classification using conditional probabilities

In this section, we follow Larsson and Cooper (2021) in showing how semantic classification (i.e., estimating a probability distribution over class value types) works under the assumption that we can compute conditional probabilities $p(C_j || E_1 \ldots E_n)$ of a class value types C_j given evidence value types $E_1 \ldots E_n$.

In general, for $C_j \in \mathfrak{R}(\mathbb{C}^\kappa)$, we have

(13) $p^\kappa(s : C_j) =$
$$\sum_{\substack{E_1 \in \mathfrak{R}(\mathbb{E}_1^\kappa) \\ \vdots \\ E_n \in \mathfrak{R}(\mathbb{E}_n^\kappa)}} p^\kappa(C_j || E_1 \ldots E_n) p(s : E_1) \ldots p(s : E_n)$$

Correspondingly, in the fruit recognition game, for each $F \in \mathfrak{R}(\textit{Fruit})$ we have

(14) $p^{\mathit{FruitC}}(s : F) =$

$$\sum_{\substack{L \in \Re(\mathit{Col}) \\ S \in \Re(\mathit{Shape})}} p(F||L \wedge S)p(s : L)p(s : S)$$

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

(15) $p^{\mathrm{FruitC}}(s{:}\mathit{Apple}) =$
$p(\mathit{Apple}||\mathit{Green} \wedge \mathit{Ashape})p(s{:}\mathit{Green})p(s{:}\mathit{Ashape}) +$
$p(\mathit{Apple}||\mathit{Green} \wedge \mathit{Pshape})p(s{:}\mathit{Green})p(s{:}\mathit{Pshape}) +$
$p(\mathit{Apple}||\mathit{Red} \wedge \mathit{Ashape})p(s{:}\mathit{Red})p(s{:}\mathit{Ashape}) +$
$p(\mathit{Apple}||\mathit{Red} \wedge \mathit{Pshape})p(s{:}\mathit{Red})p(s{:}\mathit{Pshape})$

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 (16) illustrate a joint probability distribution for the Bayesian Network in Figure 4.

(16)

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.

The non-conditional probabilities in (15) are derived from the agents' take on the particular situation being classified; let us call it s_5.

(17)

$T =$	*Ashape*	*Pshape*	*Green*	*Red*
$\mathrm{p}(\mathrm{s}_5{:}T)$	0.90	0.10	0.80	0.20

This means we have e.g.

(18) $d_{\mathit{Shape}}(s_5) = \langle 0.90, 0.10\rangle$

Larsson and Cooper (2021) 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.

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

(19) $p^{\mathrm{FruitC}}(\mathrm{s}_5{:}\,\mathit{Apple}) =$
$0.93 * 0.80 * 0.90 + 0.63 * 0.80 * 0.10 +$
$0.56 * 0.20 * 0.90 + 0.13 * 0.20 * 0.10 =$
$0.67 + 0.05 + 0.10 + 0.00 =$
0.82

4 Frequentist semantic learning

For the model of semantic classification that uses conditional probabilities, a central question is of course how to estimate conditional probabilities, of the form $p(C||E_1 \wedge \ldots \wedge E_n)$ (where $C \in \Re(\mathbb{C}), E_i \in \Re(\mathbb{E}_i), 1 \leq i \leq n$). Using Bayes rule and marginalising over the class value types, we get for a Naive Bayes classifier:

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

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

For all combinations of evidence value types $E_1, \ldots, E_n$ and class value types C, we need (a) the conditional probability of the evidence value types given the class value type, $p(E_i||C)$, and (b) the prior of the class value type, $p(C')$.

4.1 Computing conditional probabilities

Following a frequentist[4] methodology, conditional probabilities can be estimated by counting previous instances of C and E_i:

$$p(E_i|C) = \frac{|E_i\&C|}{|C|}$$

This relies on previous judgements being categorical rather than probabilistic. However, it appears reasonable to assume that agents sometimes make non-categorical judgements, assigning a probability other than 0 or 1 to a situation being of a certain type, and we want to explore the idea of using

[4]Regarding the tension between Bayesian and frequentist modelling, one might argue that no practically useful model is purely Bayesian, since the moment that you introduce data, you will extract frequencies from it. While theoretical models may be purely bayesian, in all machine learning models there is an element of frequentism. However, being too naively frequentist yields models which generalise poorly. To take an extreme example, a purely frequentist 5-gram model will assign 0-probability to any 5-gram which does not occur at all in the data, which is clearly wrong. Bayesian reasoning is ultimately a mathematical recipe to construct models which explicitly capture the dependencies between various random variables (hidden or not). In the paper we show two ways to improve the model, with varying levels of complexity and robustness.

such non-categorical past judgements as a basis for future (probabilistic) judgements.

Cooper et al. (2015) sketch a solution with a frequentist flavour (but also with some differences to regular frequentist learning acccounts), 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 $\mathfrak{J}$.

So the history of judgements $\mathfrak{J}$ does not contain definite judgements, but rather probabilistic ones. How are these probabilities to be understood? We assume that each such probability corresponds to the judging agent's estimate of the probability that a member of the linguistic community would judge s to be of type T. That is, we assume that agents make the (semantic) judgements that they estimate that other agents would also make (on average). This can be intuitively justified by the assumption that agents they take language (including meanings) to be public (shared in a community). Hence, each probabilistic judgement in the history can be considered to correspond to a large number N of independent categorical judgements.

How do we motivate this? After all, language is categorical in nature at least insofar as a speaker makes or does not make an utterance U to describe some situation s, thus categorising s as (categorically) correctly described by U. However, the categorical nature of language does not imply that agents cannot entertain non-categorical judgements, only that once they speak their judgements, they become categorical[5]. When it comes to computing the probabilities needed for probabilistic classifiers, this means that $round(p(s : C)p(s : E_i)N)$ of them are considered to be of type $(C \wedge E_i)$. (The motivation for rounding to integers using the $round$ function is that if we talk about discrete events, there must be an integer number of them.) On this basis, we can compute likelihoods and probabilities as a ratio of the frequencies of occurrences, summed over all judgements in the history:

[5]We are here ignoring for the moment some complications, including that hearers may assign probabilities to speakers having made an utterance U based on perceptual, semantic and pragmatic confidences. We hope to return to these points in future work.

(21)

$$p(E_i||C) = \lim_{N\to\infty} \frac{\sum_{j\in\mathfrak{J},j.\text{sit}=s} round(p(s:C)p(s:E_i)N)}{\sum_{j\in\mathfrak{J},j.\text{sit}=s} round(p(s:C)N)} = \frac{\sum_{j\in\mathfrak{J},j.\text{sit}=s} p(s:C)p(s:E_i)}{\sum_{j\in\mathfrak{J},j.\text{sit}=s} p(s:C)}$$

Formula (21) tells us that we can consider probabilities in the history of judgments as fractions of events; and this is justified by interpreting them as fractions of language-community speakers making the corresponding categorical judgement. In this sense, we are providing a frequentist interpretation of epistemic probability[6]

One might ask regarding (21), is it possible to multiply the probabilities associated with variables that may be dependent, without taking into account their conditional probabilities? Yes, in this case, it is. We are multiplying probabilistic judgements that have already been made rather than hypothetical judgements[7].

For the purposes of this paper, we will assume that the probabilities needed are indeed encoded directly in $\mathfrak{J}$, but of course in general this might

[6]Assuming we have N situations in total, some integer number will be classified as each category. We can then sum these integer numbers. We can get a probability by taking the limit of the ratio with N tending to infinity.

[7]This can perhaps be better understood by analogy to counting in standard probability theory. Suppose that we have a corpus of English sentences where all nouns are annotated for part of speech, and for whether the noun has a subject/object role (or neither). We estimate the conditional probability that a noun is a subject by counting the number of nouns that are also subjects, and divide this sum by the total number of nouns.

$$p(\text{Subject}|\text{Noun}) = \frac{|\text{Subject\&Noun}|}{|\text{Noun}|}$$

Categorical judgements can be regarded as probabilistic judgments with probability 1.0, so that judging a word w to be noun is to judge the probability of w being a nount to be 1.0. Assuming a word w has ben judged as being a subject and a noun, we can describe this probabilistically as p(w is Subject)=1.0, p(w is Noun)=1.0, and we conclude that p(w is Subject&Noun)=p(w is Subject)p(w is Noun)=1.0*1.0=1.0 without involving p(Subject|Noun), the conditional probability of something being a subject given that it is a noun (which is in fact what we are trying to compute). In doing so, we are *not* taking the probability that a word is a subject to be independent of its being a noun. In fact, we are assuming the opposite. We are trying to compute the conditional probability of a word being a subject given that it is a noun. But when counting a word w as being both a subject and a noun, we do not invoke this conditional probability as part of the enumeration. To do so it would be both circular and unnecessary, as we have already judged w to be a subject, and to be a noun.

not be the case. Cooper et al. (2015) explains how probabilities of complex types (such as meet types, join types, function types and record types) can be computed from simpler types.

4.2 Computing priors

In addition to conditional probabilities, (20) requires the prior probabilities of the class value types $C \in \mathfrak{R}(\mathbb{C})$. We use $\mathfrak{p}_{\mathfrak{J}}(T)$ to denote the prior probability that an arbitrary situation is of type T given $\mathfrak{J}$. However, it is important to note that the prior probability for a value type A is not the same as the probability $p(A)$ that there is something of type A. Rather, it is the probability that some arbitrary situation s (of which we have no other relevant information) is of type A. To see this, imagine that $p(s_1 : A) = 0.8$ and $p(s_2 : A) = 0.2$, and that there are no judgements concerning other situations in $\mathfrak{J}$. In this case, $p(A)$, the probability that there is something of type A[8], is $0.8 + 0.2 - (0.8 * 0.2) = 0.84$. However, the prior probability that an arbitrary situation is of type A, $\mathfrak{p}_{\mathfrak{J}}(A)$, is $(0.8 + 0.2)/2 = 0.5$.

Following this, we define the prior probability of an arbitrary situation being of a type T, $\mathfrak{p}_{\mathfrak{J}}(T)$, thus:

(22)
$$\mathfrak{p}_{\mathfrak{J}}(T) = \frac{\sum_{j \in \mathfrak{J}_T} j.\text{prob}}{\text{P}(\mathfrak{J})} \text{ if P}(\mathfrak{J}) > 0\text{, otherwise } 0$$

where $\mathfrak{J}_T$ is the set of all judgements concerning T:

(23)
$$\mathfrak{J}_T = \{j \mid j \in \mathfrak{J}, j.\text{sit-type} = T\}$$

and $\text{P}(\mathfrak{J})$ is the cardinality of situations in $\mathfrak{J}$, i.e. the total number of situations in $\mathfrak{J}$[9]:

(24) $\text{P}(\mathfrak{J}) = |\{s | \exists j \in \mathfrak{J}, j.\text{sit} = s\}|$

Accordingly, we replace (20) with (25), where $p(C)$ is replaced with $\mathfrak{p}_{\mathfrak{J}}(C)$:

(25) $\hat{p}^{\kappa}(C||E_1 \wedge \ldots \wedge E_n) =$
$$\frac{\mathfrak{p}_{\mathfrak{J}}(C)p(E_1||C)\ldots p(E_n||C)}{\sum_{C' \in \mathfrak{R}(\mathbb{C}^{\kappa})} \mathfrak{p}_{\mathfrak{J}}(C')p(E_1||C')\ldots p(E_n||C')}$$

[8]See Cooper et al. (2015) for details.

[9]This replaces an earlier definition in Cooper et al. (2015).

5 Example: frequentist semantic learning in the fruit recognition game

The conditional probabilities in (16) are generated from $\mathfrak{J}$ by a learning component. Let's assume that $\mathfrak{J}$ is as in Figure 5, based on previous rounds of the game.

The recorded judgements concerning the types *Apple* and *Pear* are here assumed to be derived not only from the agent's own perception of the fruits in question, but also (and perhaps primarily) from a tutor's explicit judgements, possibly in combination with an estimation of the likelihood that the teacher is competent at judging apples and pears under whatever conditions (light etc.) held at the time of judgement.

In our example, $p(F||L \wedge S)$ comes from previous experience as encoded in $\mathfrak{J}$. We estimate this probability with Bayes' rule, as in (26).

(26) $p(F||L \wedge S) =$
$$\frac{\mathfrak{p}_{\mathfrak{J}}(F)p(L||F)p(S||F)}{\sum_{F' \in \mathfrak{R}(\textit{Fruit})} \mathfrak{p}_{\mathfrak{J}}(F')p(L||F')p(S||F')}$$

To compute this we need the following for all $F \in \{\textit{Apple}, \textit{Pear}\}$:

(27) a. for all $L \in \{\textit{Green}, \textit{Red}\}, p(L||F)$

b. for all $S \in \{\textit{Ashape}, \textit{Pshape}\}, p(S||F)$

c. $\mathfrak{p}_{\mathfrak{J}}(F)$

We use (21) to compute conditional probabilities, so that for example

(28) $p(\textit{Green}||\textit{Apple}) =$
$$\frac{\sum_{j \in \mathfrak{J}, j.\text{sit}=s} p(s : \textit{Apple})p(s : \textit{Green})}{\sum_{j \in \mathfrak{J}, j.\text{sit}=s} p(s : \textit{Apple})} =$$
$$\frac{0.9 * 1.0 + 0.7 * 0.5 + 1.0 * 0.9 + 0.0 * 0.1}{1.0 + 0.5 + 0.9 + 0.1} = 0.86$$

We also use (22) to compute priors, so that for example

(29)
$$\mathfrak{p}_{\mathfrak{J}}(\textit{Apple}) = \frac{\sum_{j \in \mathfrak{J}, j.\text{sit}=s} p(s : \textit{Apple})}{\text{P}(\mathfrak{J})} =$$
$$\frac{1.0 + 0.5 + 0.9 + 0.1}{4} = \frac{2.50}{4} = 0.63$$

j.p	*j*.sit-type∈ $\mathfrak{R}$(*Fruit*)		*j*.sit-type∈ $\mathfrak{R}$(*Col*)		*j*.sit-type∈ $\mathfrak{R}$(*Shape*)	
j.sit	*Apple*	*Pear*	*Green*	*Red*	*Ashape*	*Pshape*
s_1	1.0	0.0	0.9	0.1	0.7	0.3
s_2	0.5	0.5	0.7	0.3	0.6	0.4
s_3	0.9	0.1	1.0	0.0	1.0	0.0
s_4	0.1	0.9	0.0	1.0	0.0	1.0

Figure 5: Conditional probabilities in the fruit recognition game

Based on this, we compute the conditional probabilities shown in (16) and used in classification in Section 2.7, for example

(30) $p(\mathit{Apple}||\mathit{Green} \wedge \mathit{Ashape}) =$

$$\frac{\mathfrak{p}_{\mathfrak{J}}(\mathit{Apple})p(\mathit{Green}||\mathit{Apple})p(\mathit{Ashape}||\mathit{Apple})}{\sum_{F\in\{\mathit{Apple},\mathit{Pear}\}}\mathfrak{p}_{\mathfrak{J}}(F)p(\mathit{Green}||F)p(\mathit{Ashape}||F)} =$$

$$\frac{0.41}{0.41+0.03} = 0.93$$

Based on the judgement above in (19), our agent may venture the guess that the fruit in question in s_5 is an apple, to which the tutor may respond "Very good!". This in turn could trigger extending $\mathfrak{J}$ to include probabilistic judgements concerning the classification of s_5 as being of types *Apple*, *Pear*, *Green*, *Red*, *Ashape* and *Pshape*, to be used in future rounds of the game.

6 Conclusion

Cooper et al. (2014) and Cooper et al. (2015), and more recently Larsson and Cooper (2021), 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. We have proposed a Bayesian account of semantic learning formulated in terms of ProbTTR, thereby connecting probabilistic semantic learning to other phenomena studied in TTR and ProbTTR, including the modeling of perceptual meaning as classifiers (Larsson, 2013; Larsson and Cooper, 2021).

Our treatment of learning relies on the idea that an agent keeps a record of their previous judgements concerning the likelihood of a classification and sums the probabilities of these judgements. The agent computes conditional probabilities and priors for current judgements on the basis of this record. We have illustrated this view of learning with the fruit recognition game. This simplified example provides a sketch of how an agent can acquire a set of predicates through mentor vetted (supervised) classifier learning.

With respect to semantic learning, this paper follows in the general footsteps of van Eijck and Lappin (2012), who propose a probabilistic theory of language semantics which includes a sketch of semantic learning. It appears that our model is an instance of the strategies outlined by van Eijck and Lappin. Where they only sketch a strategy, we have shown in detail how learning from examples can be modelled.

As part of the Rational Speech Act Theory, Goodman and Lassiter (2015); Lassiter and Goodman (2017) provide an account of semantic update of an agent's view of the world, which can possibly be regarded as a form of semantic learning. Even though they apply it to a single event, their account can be generalised to several events in a natural way. Indeed, Bernardy et al. (2018, 2019) have implemented such a generalisation. What sets the present work apart, in addition to being formulated in ProbTTR, is that each individual event is not categorical, but itself probabilistic. We have achieved this by incorporating elements of frequentist thinking in an otherwise Bayesian account. Conversely, the approaches previously mentioned manage to remain in a purely Bayesian framework, but they do not generalise to probabilistic events in a straightforward manner.

Future work includes exploring and adapting other learning methods to ProbTTR, including a linear transformation model and related neural network and deep learning models, and continuing to apply ProbTTR to a variety of problems in natural language semantics.

Acknowledgements

This work was supported by grant 2014-39 from the Swedish Research Council (VR) for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg.

References

Jean-Philippe Bernardy, Rasmus Blanck, Stergios Chatzikyriakidis, and Shalom Lappin. 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.

Jean-Philippe Bernardy, Rasmus Blanck, Stergios Chatzikyriakidis, Shalom Lappin, and Aleksandre Maskharashvili. 2019. 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.

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. 2012a. 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. 2012b. 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.

N. Goodman and D. Lassiter. 2015. Probabilistic semantics and pragmatics: Uncertainty in language and thought. In S. Lappin and C. Fox, editors, *The Handbook of Contemporary Semantic Theory, Second Edition*, pages 655–686. Wiley-Blackwell, Malden, 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 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.

Staffan Larsson and Robin Cooper. 2021. Bayesian classification and inference in a probabilistic type theory with records. In *Proceedings of NALOMA 2021*.

Daniel Lassiter and Noah D. Goodman. 2017. Adjectival vagueness in a Bayesian model of interpretation. *Synthese*, 194(10):3801–3836.

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

Jan van Eijck and Shalom Lappin. 2012. Probabilistic semantics for natural language. In Z. Christoff, P. Galeazzi, N. Gierasimszuk, A. Marcoci, and S. Smets, editors, *Logic and Interactive Rationality (LIRA), Volume 2*, pages 17–35. ILLC, University of Amsterdam.

Donkey Anaphora: Type-Theoretic Semantics with Both Strong and Weak Sums

Zhaohui Luo
Royal Holloway, University of London
`zhaohui.luo@hotmail.co.uk`

Abstract

Donkey sentences are among the challenging examples that present a difficult problem in compositional logical semantics and their semantic treatment is one of the early applications of dependent type theory to linguistic semantics, where the strong sum Σ, rather than weak sums (as given by traditional existential quantifiers), is used for existential quantification. However, it is known that this method is inadequate because it fails to deal with counting properly. In this paper, we propose to consider the semantics of donkey sentences in a type theory with both strong and weak sums and show that, with both sum operators, donkey sentences can be given adequate semantic interpretations which, in particular, take care of counting properly.

1 Introduction

Donkey sentences, as first studied by Geach (1962) and exemplified in (1), where an anaphoric expression refers to an existentially quantified entity, are among the challenging examples that present a difficult problem in compositional logical semantics.

(1) Every farmer who owns a donkey beats it.

Their studies (and that of trans-sentential anaphora) have led to the development of dynamic semantics such as DRT (Kamp, 1981) and DPL (Groenendijk and Stokhof, 1991) which, however, require one to consider substantial changes of the underlying logical systems.[1] (For a recent summary of the dynamic approach to donkey anaphora, see Brasoveanu and Dotlacil (2021).)

[1]For example, DPL (Groenendijk and Stokhof, 1991) is a rather non-standard logical system: among other things, it is non-monotonic and the notion of dynamic entailment fails to be reflexive or transitive.

In the mid-80s, as one of the early applications of dependent type theory in logical semantics, researchers such as Mönnich (1985) and Sundholm (1986) have proposed to use Martin-Löf's type theory (Martin-Löf, 1984) to deal with donkey anaphora, where Σ-types are employed to represent existentially quantified formulas. Σ-types $\Sigma x{:}A.P(x)$ are also called *strong sums*, as opposed to the traditional existentially quantified formulas $\exists x{:}A.P(x)$ which are called *weak sums*, because from an object of the strong sum, one can obtain its witness, by means of a projection operation, while this is not possible for the weak sum. It is because of the availability of witness projection that an anaphoric reference can be obtained from an object of a Σ-type, while this is not possible for an existential quantification in the traditional case (and hence the problem in the first place). However, it is known that this approach of using Σ-types to deal with donkey anaphora suffers from a problem of counting (Sundholm, 1989; Tanaka, 2015) and fails to provide us an adequate solution. (See §2 for more details.)

In this paper, we contend that the problem of the above type-theoretical approach has come from a double role played by Σ, as an existential quantifier, on the one hand, and as a structural mechanism to represent collections of objects, on the other. These two roles should be separate and played by different type constructors. But in traditional logics (first-order logic or simple type theory) or in Martin-Löf's type theory, only either $\exists$ or Σ exists, not both, and therefore there is no way to consider such a separation. We show that, in a type theory with both strong and weak sums, donkey sentences can be given adequate semantics in which counting is taken into proper account.

Our proposal is also linked to the research on different readings of donkey sentences and, in particular, the strong and weak readings as studied by Chierchia (1990) and others. Also, donkey anaphora are closely related to (and, for some researchers, they are examples of) the so-called E-type anaphora, as first studied by Evans (1977, 1980), which may be interpreted by means of descriptions (see, for example, Nouwen (2021) for a recent discussion). It is not surprising that Σ-types are essentially useful in semantic interpretations of donkey sentences since they have close links to descriptions (Martin-Löf, 1984; Carlström, 2005; Mineshima, 2013) and we shall give some brief discussions about this.

Combining strong and weak sums in type theory is a subtle matter that needs us to tread carefully, for otherwise we may easily slip into problems such as inconsistency. We shall discuss this briefly as well.

This is a short version of a paper we plan to write. In this paper, in particular, we shall focus on telling a complete story of this new treatment of donkey anaphora with both strong and weak sums, but shall be brief about or completely omit some related respects.

2 Strong and Weak Sums in Type Theory

In this section, we explain the concepts of weak sums (for example, traditional existential quantifiers) and strong sums (Σ-types) and, using the notion of the cardinality of a finite type, illustrate the counting problem when using only Σ-types to interpret donkey sentences.

Weak sums (existential quantifiers). Under the Curry-Howard propositions-as-types principle (Curry and Feys, 1958; Howard, 1980), traditional existentially quantified formulas are examples of weak sum types of the form $\exists x.P(x)$. In first-order logic, depending on whether it is intuitionistic or classical, the existential quantifier can be introduced directly or defined by means of the universal quantifier together with negation, respectively. In higher-order logic (or simple type theory) as used in Montague's semantics, where there is an impredicative type **t** of all formulas, it can be either directly introduced or defined by means of the universal quantifier as in (2).

(2) $\exists x.P(x) = \forall X{:}\mathbf{t}.\ (\forall x.(P(x) \Rightarrow X)) \Rightarrow X.$

It is known that, given a proof of $\exists x.P(x)$, although one knows that there is an entity such that P holds, in the logical calculus one cannot find out which entity it is. It is because of this that an anaphoric reference to an existentially quantified entity becomes problematic. For example, in a traditional compositional semantics, the donkey sentence (1) would obtain (3) as its interpretation, which is not a well-formed formula since the variable y in $beat(x, y)$ is out of the scope of the existential quantifier.

(3) (#) $\forall x.\ [farmer(x)\ \&$
$\exists y.(donkey(y)\ \&\ own(x, y))]$
$\Rightarrow beat(x, y)$

This illustrates the original problem in interpreting donkey sentences, as mentioned at the beginning of Introduction.

Strong sums (Σ-types). Σ is a dependent type constructor. If A is a type and B is a family of types that depend on objects of type A, then $\Sigma x{:}A.B(x)$ is a type, consisting of pairs (a, b) such that a is of type A and b is of type $B(a)$. Σ-types are associated with the projection operators π_1 and π_2 so that, for (a, b) of type $\Sigma x{:}A.B(x)$, $\pi_1(a, b) = a$ and $\pi_2(a, b) = b$. Formally, Σ-types are governed by the inference rules in Appendix A.

Besides being useful mechanisms to organise structures in various applications, Σ-types may also play other roles. For example, in Martin-Löf's type theory, Σ also plays the role of existential quantifier in its logic[2]. Therefore, for instance, the donkey sentence (1) can be interpreted as (4), in which F_Σ, as defined in (5), is the type intended to represent the collection of donkey-owning farmers, where F and D are the types that interpret farmer and donkey, respectively.[3]

(4) $\forall z : F_\Sigma.\ beat(\pi_1(z), \pi_1(\pi_2(z)))$

(5) $F_\Sigma = \Sigma x{:}F\ \Sigma y{:}D.\ own(x, y)$

Σ-types are strong in the sense that from a proof of $\Sigma x{:}A.P(x)$ one can preform the first projection operation to obtain the witness of this 'existentially' quantified formula and it is because of this, if Σ is used as existential quantifier, one can project out its witness from a proof term of the Σ-type, even

[2]This is concerned with intuitionistic philosophy – a strongly minded intuitionist may believe that the witness of a proven existentially quantified formula can be obtained internally in a logical calculus. We omit further discussions here.

[3]In formal semantics based on modern type theories, CNs such as 'farmer' and 'donkey' are interpreted as types (rather than predicates). This was first proposed by Mönnich (1985) and Sundholm (1986) and further elaborated in (Ranta, 1994; Luo, 2012).

outside its scope (the terms $\pi_1(z)$ and $\pi_1(\pi_2(z))$ in (4) are such examples).

The type F_Σ above contains two occurrences of Σ and they play two different roles: the first acts as a structural mechanism to represent the collection of the farmers who own donkeys and the second as the existential quantifier to say that there exists a donkey owned by the farmer concerned. As we shall see below, using Σ to play this double role is problematic. In particular, F_Σ is in fact representing a collection whose cardinality (the number of its objects) is different from that of the collection of donkey-owning farmers and, therefore, the semantic interpretation (4) of (1) is inadequate (Sundholm, 1989; Tanaka et al., 2015).

Counting and cardinality of finite types. When a type A is finite in the sense that it has finitely many objects, it is possible to define its cardinality $|A|$ as the number of its objects. Formally, a type is finite if, for some n, it is isomorphic to $Fin(n)$, the type with exactly n objects – see Appendix B. For example, the cardinality of a finite Σ-type is the number of pairs in the type.

The problem of counting can be illustrated by considering the sentence in (6),[4] where the quantifier Every in (1) is replaced by Most. Its formal semantics by means of Σ-types in Martin-Löf's type theory is given in (7), which can be seen obtained by replacing $\forall$ by the quantifier $Most_S$, which is defined by Sundholm (1989) (S in $Most_S$ for Sundholm) so that, for a finite type A, $Most_S\ x{:}A.P(x)$ is true if, and only if, more than half of the objects in A satisfy P.

(6) Most farmers who own a donkey beat it.

(7) $Most_S\ z : F_\Sigma.\ beat(\pi_1(z), \pi_1(\pi_2(z)))$

Let us now consider the cardinality of F_Σ, as defined in (5). Because of the second Σ in F_Σ, $|F_\Sigma|$ is not that of the collection of donkey-owning farmers; instead, to calculate $|F_\Sigma|$, we'd have to count every triple (x, y, p) of farmers x, donkeys y and proofs p that x owns y. For example, if there are ten farmers, one of whom owns twenty donkeys and beats all of them, and the other nine own one donkey each and do not beat their donkeys. Then, $|F_\Sigma| \geq 29$ (it is an inequality because, if farmer x owns donkey y, there may be more than one proof that x owns y), but the number of farmers who do not beat their donkeys is 9. Therefore, the above semantics (7) of (6) would be true in such a case, which is obviously incorrect.[5]

[4]Thanks to Justyna Grudzińska for a discussion about this example.

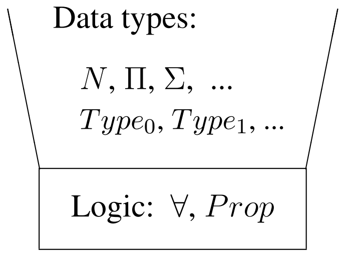


Figure 1: The type structure in UTT.

3 Donkey Anaphora: a Type-Theoretical Solution with Both Σ and $\exists$

In this section, we shall first introduce a dependent type theory UTT (Luo, 1994), which has both strong and weak sums, and then show how donkey sentences like (1) and (6) can be interpreted type-theoretically, giving adequate treatments for different readings and taking care of counting in a proper way as well.

3.1 UTT: an impredicative type theory

The type structure of UTT (Unifying Theory of dependent Types) (Luo, 1994) consists of two parts: the world of data types and that of logical propositions (see Fig. 1). It contains various types such as dependent product types (Π-types), strong sum types (Σ-types), the type N of natural numbers, the universes $Type_i$, and many other types. UTT also contains an impredicative type universe $Prop$ of logical propositions which provide means to describe the logical properties of objects of any type (see Appendix C). Formally, UTT can be considered as the combination of Martin-Löf's (intensional) type theory (Martin-Löf, 1975; Nordström et al., 1990) with Coquand-Huet's Calculus of Constructions (Coquand and Huet, 1988). In computer science, type theories such as UTT have been implemented in theorem proving systems (called proof assistants) for formalisation of mathematics and verification of programs, and recently, they have been used for formal reasoning

[5]This is similar to the 'proportion problem' when one uses DRT to interpret such donkey sentences, where one counts farmer-donkey pairs rather than the donkey-owning farmers. See Kanazawa (1994) and Brasoveanu and Dotlacil (2021), among others, for discussions.

based on linguistic semantics (see, for example, (Chatzikyriakidis and Luo, 2016)).[6]

Note that UTT contains both strong sums $\Sigma x{:}A.B(x)$ (Σ-types, as 'data types') and weak sums $\exists x{:}A.P(x)$ (existentially quantified types, as logical propositions), and this is essential when considering semantic interpretations of donkey sentences in §3.2 below.

Logic and proof irrelevance. In UTT, a type is a logical proposition if it is of type $Prop$. The type universe $Prop$ is impredicative and, therefore, the other logical operators can be defined by means of the operator $\forall$ for universal quantification[7]. For example, the conjunction operator and the existential quantifier $\exists$ can be defined as in (8) and (9), respectively, and the definitions of the other operators can be found in Appendix C.

(8) $P \wedge Q = \forall X : Prop.\,(P \Rightarrow Q \Rightarrow X) \Rightarrow X$

(9) $\exists x : A.P(x)$
$\quad = \forall X{:}Prop.(\forall x : A.(P(x) \Rightarrow X)) \Rightarrow X$

The principle of *proof irrelevance* says that any two proofs of the same logical proposition should be the same. For instance, it implies that, for farmer x and donkey y, any two proof terms of the proposition $own(x, y)$ should be the same. It has been shown that, when employing a type theory for natural language semantics, proof irrelevance should be enforced (Luo, 2012, 2019). Note that, because in UTT there is a clear distinction between logical propositions and other types (the former being those of type $Prop$), it is straightforward to introduce proof irrelevance by means of the following rule (Werner, 2008; Luo, 2012):

$$\frac{P : Prop \quad p : P \quad q : P}{p = q : P}$$

Intuitively, it says that, if P is a logical proposition and if p and q are proof terms of P, then p and q are the same. In particular, according to the above rule, every proposition of type $Prop$ is either an empty type or a singleton type. In terms of cardinality, we have $|P| \leq 1$ for every $P : Prop$ and, therefore, if A is finite and $Q : A \to Prop$ is a predicate over A, then we have

(10) $|\Sigma x{:}A.Q(x)| \leq |A|$.

3.2 Semantics of donkey anaphora in UTT

When a type theory has both strong and weak sums (Σ-types and $\exists$-propositions as in UTT), together with proof irrelevance, there is a new way to semantically interpret donkey sentences, which takes care of counting adequately. We'll use the example (6), which is repeated as (11) below, to explain.

(11) Most farmers who own a donkey beat it.

In §2, we have shown that, because in Martin-Löf's type theory Σ is used to play a double role, the semantic interpretation (7) of (11) is inadequate because it gets counting wrong. In that definition, we have used quantifier $Most_S$ defined in Martin-Löf's type theory and, here, we can define a semantic interpretation of the quantifier most in UTT in a similar fashion as in (Sundholm, 1989) but with a crucial difference: instead of Σ, we shall use $\exists$ as defined in (9) as the existential quantifier and, intuitively, for a finite A, $Most\ x{:}A.P(x)$ also means that more than half of the objects in A satisfy P. Note that, $Most_S\ x{:}A.P(x)$ is a non-propositional type, but $Most\ x{:}A.P(x)$ is a logical proposition of type $Prop$. (See Appendix D for details.)

Having defined $Most$ in UTT, we can now interpret the donkey sentence (11) as (12), in which $F_\exists$ is defined in (13):

(12) $Most\ z : F_\exists.$
$\quad \forall y' : \Sigma y{:}D.own(\pi_1(z), y).\, beat(\pi_1(z), \pi_1(y'))$

(13) $F_\exists = \Sigma x{:}F.\ \exists y{:}D.own(x, y)$

Note that $|\exists y{:}D.own(x, y)| \leq 1$, that is, if $\exists y{:}D.own(x, y)$ is true, the cardinality of the proposition is 1. Therefore, the type $F_\exists$ correctly represents the collection of donkey-owning farmers, as intended, and the above semantics (12) is adequate and, in particular, it deals with counting correctly.

Researchers have studied different readings (in particular, strong and weak readings) of donkey anaphora, as studied by Chierchia (1990, 1992) and

[6]There are several proof assistants based on type theories including Agda (Agda, 2008) based on Martin-Löf's type theory, Coq (Coq, 2010) implementing the type theory pCIC, and Lego/Plastic (Luo and Pollack, 1992; Callaghan and Luo, 2001) implementing UTT. It may be worth remarking that pCIC, implemented in the Coq proof assistant, is very similar to UTT – this is especially the case after Coq's universe Set became predicative in 2004 (it was impredicative in earlier versions).

[7]The fact that other logical operators can be defined in higher-order logical systems by means of universal quantifier was discovered in the 60s by Prawitz (1965) (and several others, independently) and, this is the same in an impredicative type theory.

others. For instance, the strong and weak readings of (11) are (14) and (15), respectively:

(14) Most farmers who own a donkey beat *the donkeys they own.*

(15) Most farmers who own a donkey beat *some donkeys they own.*

The above interpretation (12) of (11) is a strong one, interpreting (14) directly: most donkey-owning farmers beat *all* donkeys they own. A weaker interpretation of its weak reading (15) would be (16), obtained from (12) by changing $\forall$ into $\exists$:

(16) $Most\ z : F_{\exists}.$
$\exists y' : \Sigma y{:}D.own(\pi_1(z), y).\ beat(\pi_1(z), \pi_1(y'))$

People have also considered more sophisticated examples where donkey anaphora are involved in various ways. For example, (17) is one of them, taken from Brasoveanu's thesis (Brasoveanu, 2007), in which the readings for the donkey anaphora are different ('a TV' having a strong reading and 'a credit card' a weak one). Its type-theoretical semantics with both strong and weak sums is given in (18).

(17) Every person who buys a TV and has a credit card uses it to pay for it.

(18) $\forall z : \Sigma x{:}Person.\ \exists y_1{:}TV.\ buy(x, y_1)$
$\qquad\qquad \wedge\ \exists y_2{:}Card.\ own(x, y_2)$
$\forall y : \Sigma y_1{:}TV.\ buy(\pi_1(z), y_1)$
$\exists y' : \Sigma y_2{:}Card.\ own(\pi_1(z), y_2).$
$\quad pay(\pi_1(z), \pi_1(y), \pi_1(y'))$

One may change the quantifier Every in (17) into Most (and make other minor changes in the sentence) and, in that case, we can use the quantifier $Most$ defined in UTT to interpret the sentence and the resulting interpretations take care of counting correctly as well.

3.3 E-type anaphora

Here, we discuss, albeit rather briefly, the so-called E-type anaphora to which donkey anaphora are closed related (and, for some researchers, donkey anaphora are examples of E-type anaphora).[8] E-type anaphora are first studied by Evans (1977, 1980), and further discussed by many, including (Heim and Kratzer, 1998) among others. They can be interpreted by means of descriptions (Russell, 1905, 1919) (see, for example, Nouwen (2021) for a recent discussion). An example, due to Evans, is (19). Note that the pronoun 'they' in (19) is not bound by 'Few' for otherwise the meaning is incorrect. A common conceptual answer, proposed by Evans (1977, 1980), is that these pronouns are *descriptive* in that they can be paraphrased by means of descriptions as exemplified in (20) that paraphrases (19).

(19) Few congressmen admire Kennedy, and they are very junior.

(20) Few congressmen admire Kennedy, and *the congressmen that do admire Kennedy* are very junior.

As pointed out by Martin-Löf (1984), strong sum types (Σ-types) are related to descriptions, because he regards them as logical propositions as well. If you think that $\Sigma x{:}A.B(x)$ as the existentially quantified formula, it is strong and therefore its first projection operator π_1 gives us an internal means of obtaining the witness from a proof of the existentally quantified formula. As explained in §2, this is stronger than the traditional existential operator $\exists$ for which such a projection operator does not exist, and it is exactly because of this that Σ offers a form of description, as pointed out by Martin-Löf (1984) and further studied by Carlström (2005) and Mineshima (2013). For example, the E-type example (19) may be interpreted as (21), either in Martin-Löf's type theory or in UTT, where we assume that the quantifier Few has been defined:

(21) $Few\ x{:}C.\ admire(x, K)$
$\qquad \wedge \forall z{:}[\Sigma x{:}C.admire(x, K)].junior(\pi_1(z))$

However, it should be made clear that Σ-types are not the same as traditional existentially quantified formulas and, therefore, it is unclear how far one may go to analyse E-type anaphora by means of Σ-types. Actually, it would not go very far since, as analysed above, using Σ as existential quantifier does cause problems such as counting, which will show up in context of E-type anaphora as well.

4 Combining Strong and Weak Sums

It is worth mentioning that combining the strong sum (Σ) and the weak sum ($\exists$) in type theory is a subtle matter and, if not careful, it is easy to get into problems. It would be interesting to note that

[8] Here, I use the term 'E-type' for a kind of anaphora, rather than an approach to solving anaphora ('the E-type approach' as people often put it).

UTT does not have 'Σ-propositions' because the so-called 'large Σ-propositions' would lead to inconsistency and the so-called 'small Σ-propositions' would make the weak sum types become strong.[9] Consider, for example, to add large Σ-types into the impredicative universe $Prop$ by adding the following rule (together with those for its introduction and projections that we omit):

$$(*) \qquad \frac{A\ type \quad P : A \to Prop}{\Sigma x{:}A.P(x) : Prop}$$

It turns out that such Σ-propositions cannot be consistently added – if they were added using the above rule $(*)$ (and related ones), the resulting type theory would be inconsistent in the sense that even the false proposition would become provable (Hook and Howe, 1986; Luo, 1994).

One may want to add Σ-propositions (so-called small Σ-types) by a rule like the following, this time restricting A to be a proposition of type $Prop$:

$$\frac{A : Prop \quad P : A \to Prop}{\Sigma x{:}A.P(x) : Prop}$$

Although the resulting type theory may be consistent[10], there is another problem: the addition of such small strong sum as propositions in $Prop$ would make the weak sum proposition $\exists x{:}A.P(x)$ become strong (rather unexpectedly!) in the sense that there is now an internal function in the type theory that, from a proof of $\exists x{:}A.P(x)$, returns an object $a : A$ such that $P(a)$ holds. That would mean that the traditional existential quantifier is not weak anymore – such a side effect is of course problematic and would make the above interpretation method we have proposed fail to deal with counting correctly.

Therefore, neither of the above large or small Σ is a viable possibility and, put in another way, the approach taken in UTT seems to be the only viable approach in combining strong and weak sums.

5 Concluding Remarks

As a concluding remark, we point out that, in this paper, we have studied a completely proof-theoretic approach. This is rather different from the model-theoretic approaches that have been considered in the literature (see, for example, Brasoveanu and Dotlacil (2021)). Among other things, this has the advantage of clearer treatment, on the one hand, and enables the use of proof assistants in reasoning, on the other.

Acknowledgement. I am very grateful to the anonymous reviewers for their helpful remarks.

[9]These situations are discussed in (Luo, 1994), from which the interested reader may obtain more information.

[10]This consistency is a folklore – most researchers, including the author, believe that it is the case, although the author has not seen a proof of it.

References

Agda. 2008. The Agda proof assistant (v2). URL: `http://appserv.cs.chalmers.se/users/ulfn/wiki/agda.php`

A. Brasoveanu. 2007. *Structured Nominal and Modal Reference*. Ph.D. thesis, The State University of New Jersey.

A. Brasoveanu and J. Dotlacil. 2021. Donkey anaphora: farmers and bishops. *In D. Gutzmann et al. (eds), The Wiley Blackwell Companion to Semantics*.

P. Callaghan and Z. Luo. 2001. An implementation of LF with coercive subtyping and universes. *Journal of Automated Reasoning*, 27(1):3–27.

J. Carlström. 2005. Interpreting descriptions in intensional type theory. *The Journal of Symbolic Logic*, 70(2).

S. Chatzikyriakidis and Z. Luo. 2016. Proof assistants for natural language semantics. In *International Conference on Logical Aspects of Computational Linguistics*, pages 85–98. Springer.

G. Chierchia. 1990. Anaphora and dynamic logic. *ITLI Publication Series for Logic, Semantics and Philosophy of Language, LP90-07*.

G. Chierchia. 1992. Anaphora and dynamic binding. *Linguistics and Philosophy*, 15.

Coq. 2010. *The Coq Proof Assistant Reference Manual (Version 8.3), INRIA*.

T. Coquand and G. Huet. 1988. The calculus of constructions. *Information and Computation*, 76(2-3):95–120.

H. Curry and R. Feys. 1958. *Combinatory Logic*, volume 1. North Holland Publishing Company.

G. Evans. 1977. Pronouns, quantifiers and relative clauses. *Canadian Journal of Philosophy*, 7.

G. Evans. 1980. Pronouns. *Linguistic Inquiry*, 11(2).

P. Geach. 1962. *Reference and Generality: An Examination of Some Medieval and Modern Theories*. Cornell University Press.

J. Groenendijk and M. Stokhof. 1991. Dynamic predicate logic. *Linguistics and Philosophy*, pages 39–100.

I. Heim and A. Kratzer. 1998. *Semantics in Generative Grammar*. Blackwell.

J. Hook and D. Howe. 1986. Impredicative strong existential equivalent to Type:Type. Technical Report TR86-760, Cornell University.

W. Howard. 1980. The formulae-as-types notion of construction. In *To H. B. Curry: Essays on Combinatory Logic*, pages 479–490. Academic Press. (Notes written and distributed in 1969.).

H. Kamp. 1981. A theory of truth and semantic representation. *In J. Groenendijk et al (eds.) Formal Methods in the Study of Language*, pages 189–222.

M. Kanazawa. 1994. Weak vs. strong readings of donkey sentences and monotonicity inference in a dynamic setting. *Linguistics and Philosophy*, 17(2).

Z. Luo. 1994. *Computation and Reasoning: A Type Theory for Computer Science*. Oxford University Press.

Z. Luo. 2012. Common nouns as types. In *Logical Aspects of Computational Linguistics (LACL'2012). LNCS 7351*.

Z. Luo. 2019. Proof irrelevance in type-theoretical semantics. *Logic and Algorithms in Computational Linguistics 2018 (LACompLing2018), Studies in Computational Intelligence (SCI)*, pages 1–15. Springer.

Z. Luo and R. Pollack. 1992. LEGO Proof Development System: User's Manual. LFCS Report ECS-LFCS-92-211, Dept of Computer Science, Univ of Edinburgh.

P. Martin-Löf. 1975. An intuitionistic theory of types: predicative part. In *Logic Colloquium'73*.

P. Martin-Löf. 1984. *Intuitionistic Type Theory*. Bibliopolis.

K. Mineshima. 2013. *Aspects of Inference in Natural Language*. Ph.D. thesis, Keio University.

U. Mönnich. 1985. *Untersuchungen zu einer konstruktiven Semantik fur ein Fragment des Englischen*. Habilitation. University of Tübingen.

B. Nordström, K. Petersson, and J. Smith. 1990. *Programming in Martin-Löf's Type Theory: An Introduction*. Oxford University Press.

R. Nouwen. 2021. E-type pronouns: congressmen, sheep and paychecks. *In D. Gutzmann et al. (eds), The Wiley Blackwell Companion to Semantics*.

D. Prawitz. 1965. *Natural Deduction, a Proof-Theoretic Study*. Lmqvist and Wiksell.

A. Ranta. 1994. *Type-Theoretical Grammar*. Oxford University Press.

B. Russell. 1905. On denoting. *Mind*, 14(56).

B. Russell. 1919. *Introduction to Mathematical Philosophy*. George Allen and Unwin.

G. Sundholm. 1986. Proof theory and meaning. In *Handbook of philosophical logic*, pages 471–506. Springer.

G. Sundholm. 1989. Constructive generalized quantifiers. *Synthese*, 79(1):1–12.

R. Tanaka. 2015. Generalized quantifiers in dependent type semantics. Talk given at Ohio State University.

Ribeka Tanaka, Koji Mineshima, and Daisuke Bekki. 2015. Factivity and presupposition in dependent type semantics. In *Proceedings of TyTLeS, ESSLLI2015*.

B. Werner. 2008. On the strength of proof-irrelevant type theories. *Logical Methods in Computer Science*, 4(3).

A Rules for Σ-types

$$\frac{A\ type \quad x{:}A \vdash B\ type}{\Sigma x{:}A.B\ type}$$

$$\frac{a : A \quad b : [a/x]B \quad x{:}A \vdash B\ type}{(a,b) : \Sigma x{:}A.B}$$

$$\frac{p : \Sigma x{:}A.B}{\pi_1(p) : A} \qquad \frac{p : \Sigma x{:}A.B}{\pi_2(p) : [\pi_1(p)/x]B}$$

$$\frac{a : A \quad b : [a/x]B}{\pi_1(a,b) = a : A} \qquad \frac{a : A \quad b : [a/x]B}{\pi_2(a,b) = b : [a/x]B}$$

B Cardinality of Finite Types

We give the formal definition of finite types. It will use the auxiliary type $Fin(n)$ for which we define first.

The type $Fin(n)$, indexed by $n : N$ with N being the type of natural numbers, consists of exactly n objects and can be specified by means of with the following introduction rules (we omit their elimination and computation rules):

$$\frac{n : N}{zero(n) : Fin(n+1)}$$

$$\frac{n : N \quad i : Fin(n)}{succ(n,i) : Fin(n+1)}$$

The cardinality of a finite type A, notation $|A|$, is defined to be n if, and only if, A is isomorphic to $Fin(n)$, that is, in the type theory concerned, there is a bijective function between A and $Fin(n)$. In particular, $|Fin(n)| = n$, since the identity function over $Fin(n)$ is bijective.

C Logic in UTT

The logic in UTT[11] consists of the impredicative universe $Prop$, specified by the following rules:

$$\frac{}{Prop\ type} \qquad \frac{P : Prop}{P\ type}$$

and the operator $\forall$ for universal quantification, specified by

$$\frac{A\ type \quad x{:}A \vdash P : Prop}{\forall x{:}A.P : Prop}$$

$$\frac{x{:}A \vdash b : P \quad x{:}A \vdash P : Prop}{\lambda x{:}A.b : \forall x{:}A.P}$$

$$\frac{f : \forall x{:}A.P \quad a : A}{f(a) : [a/x]P}$$

$$\frac{x{:}A \vdash b : P \quad a : A}{(\lambda x{:}A.b)(a) = [a/x]b : [a/x]P}$$

In UTT, other logical operators can be defined by means of $\forall$ and here are some definitions (see, for example, §5.1 of (Luo, 1994)):

$$\begin{array}{rcl}
P \Rightarrow Q & = & \forall x : P.\ Q \\
\textbf{true} & = & \forall X : Prop.\ X \Rightarrow X \\
\textbf{false} & = & \forall X : Prop.\ X \\
P \wedge Q & = & \forall X : Prop.\ (P \Rightarrow Q \Rightarrow X) \Rightarrow X \\
P \vee Q & = & \forall X : Prop. \\
 & & \quad (P \Rightarrow X) \Rightarrow (Q \Rightarrow X) \Rightarrow X \\
\neg P & = & P \Rightarrow \textbf{false} \\
\exists x : A.P(x) & = & \forall X : Prop. \\
 & & \quad (\forall x : A.(P(x) \Rightarrow X)) \Rightarrow X \\
(a =_A b) & = & \forall P : A \to Prop.\ P(a) \Rightarrow P(b)
\end{array}$$

[11] One can find its definition in §9.2.1 of (Luo, 1994), where it is specified in terms of the logical framework LF.

D *Most* in UTT

Let A be a finite type with $|A| = n_A$, $P : A \to Prop$ a predicate over A, and $Fin(n)$ the types with n objects defined in Appendix B. Then, in UTT, the logical proposition $Most\ x{:}A.P(x)$ of type $Prop$ is defined as follows, where $inj(f)$ is a proposition expressing that f is an injective function:

$$\begin{array}{l}
Most\ x{:}A.P(x) \\
\quad = \quad \exists k : N.\ (k \geq \lfloor n_A/2 \rfloor + 1) \\
\qquad \wedge\ \exists f{:}Fin(k) \to A. \\
\qquad\qquad inj(f) \wedge \forall x{:}Fin(k).P(f(x))
\end{array}$$

Feature Structures in the Wild: A Case Study in Mixing Traditional Linguistic Knowledge Representation with Neural Language Models

Gerald Penn and Ken Shi
Department of Computer Science
University of Toronto
{gpenn,kenshi}@cs.toronto.edu

Abstract

This paper briefly presents an evaluation of three models: a domain-specific one based upon typed feature structures, a neural language model, and a mixture of the two, on an unseen but in-domain corpus of user queries in the context of a dialogue classification task. We find that the mixture performs the best, which opens the door to a potentially new application of neural language models. A further examination of the domain- We also consider the inner workings of the domain-specific model in more detail, as well as how it came into being, from an ethnographic perspective. This has changed our perspective on the potential role of structured representations in the future of dialogue systems, and suggests that formal research in this area may have a new role to play in validating and coordinating *ad hoc* dialogue systems development.

1 Introduction

While contemporary NLP research marvels at how closely a simple neural language model can come to a coherent conversation partner in dialogue tasks, it nevertheless remains true that language models, neural or otherwise, are difficult to adapt in a manner that keeps both the responses constructive and the number of dialogue turns to a minimum in settings where users expect a rapid and successful conclusion to their information-seeking interactions.

Over the past year, we have worked with an industrial partner, iNAGO, Inc., a specialist in conversational agents in domains such as product information and control, navigation and automotive driver assistance, to find ways in which recent developments in dialogue systems could improve their products. Focussing on dialogue act classification at the outset, we did indeed find a way to make a simple but important improvement, which is described below, but what struck us as equally salient is just how well their system works to begin with, relative to research systems currently in circulation.

A subsequent investigation of just how their system works has revealed some novel simplifications of concepts that should be very familiar territory to this audience: typed feature structures, user modelling and semantic distances defined through a combination of lattice-theoretic calculations on epistemic networks and similarity coefficients. The novelty arises to a great extent because the developers at iNAGO were mostly unfamiliar with research publications on these topics, and so the resemblance of their proposed solution to a variety of representational strategies that have been used in the dialogue research community over the last 30 years is in itself noteworthy.

But the reason that our improvement works, we believe, stems from the complementarity of this dialogue classifier and the language-modelling-based approach that we combined it with. This complementarity needs to be investigated in more detail in a wider range of domains and across languages with a wider distribution of resource availabilities (we have experimented only with English-language systems), but it opens the door to a possible application of neural language models that has hitherto not been considered, possibly because of misbegotten claims of their cognitive plausibility, which are in turn more suggestive of their use exclusively as drop-in replacements. Domain-specific models are known to have problems with coverage, particularly outside their domains. Large-scale neural language models do not have this problem, even if their within-domain performance is somewhat lackluster by comparison. Even the simple combination of the two that we tried appears to address the weaknesses of both of these approaches in isolation.

We will begin with a discussion of the general approach to mixing the results of these two approaches to dialogue act classification, and then return to how iNAGO's system computes its own results.

2 Re-ranking

Re-ranking is a simple method for combining discriminative and generative models that takes the top answers from the generative model, in order of preference, and merely changes the order of preference using information from the discriminative model. The top answers from the generative model, which serve as the inputs to the re-ranker, are often known as *candidates*. In our case, these candidates are generated by iNAGO's system in response to a user-provided query. It should be noted that in this particular application, the user-provided query is also made available to the discriminative model, which is not always the case in re-ranking.

3 Task and Models

We evaluated three models: iNAGO's classifier, without re-ranking, the responses of a BERT-based dialogue act classifier, and the result of mixing the two models, which we shall refer to as the *Mixed model*.

With all three models, the task is to classify the transcription of a query spoken by an automobile driver according to several hundred predetermined classes of query that the system is capable of answering, based upon information about the vehicle, the state of the vehicle at the time of the query, and other information from map resources, etc., as needed. The result is a list of classes, in decreasing order of their confidence scores. Higher confidence answers have lower ordinate rank, i.e., the best answer, of rank 1, is the class with the highest confidence. The presumption is that the answer corresponding to the class with the highest confidence in the database of predetermined classes would be returned to the user when this model is used.

Our mixture method crucially relies upon the availability of these confidence scores.

The BERT-based classifier uses the pre-trained model distributed with the original paper (Devlin et al., 2019), and adds the three levels of embeddings (Figure 1) into a single vector that represents an entire string of input. Queries are classified by computing the cosine similarity of the vector for the query with the vectors for each of a list of sentences, one for each class in the database, that characterizes a prototypical question for that class, very much as an FAQ list would. Again the classes are ranked by this similarity score.

The Mixed model takes evidence from both iNAGO's model and the BERT model into consideration. It does so by treating iNAGO's confidence score, c_i, as a mixture parameter, and computes the sequence:

$$m_i = c_i \cdot a_i + (1 - c_i) \cdot b_i$$

for each candidate class, where a_i is the rank assigned by iNAGO's model, and b_i is the rank assigned by the BERT model. The new ranking of the candidates is then given by sorting the candidates in decreasing order of m_i.

4 Data

The queries that were used in our experiments were automatically generated using the method of Zheng (forthcoming) from documentation provided by a Tier-1 auto manufacturer. The documentation was only provided in April, 2021, and were thus not available either to the present authors or to iNAGO during the development of its model. The corpus consists of 232 queries.

Ground-truth answers were not made available for any of the queries by the manufacturer, but iNAGO manually mapped the corresponding answers generated by Zheng (forthcoming) to the most suitable class within their database of prototype questions and answers. iNAGO provided us with their model's rankings and confidence scores, as well as a complete list of the 410 classes and prototypes from the database that their model could refer to. As a result, we are capable of computing scores that evaluate these models in a dialogue turn classification task, but not overall measures of dialogue quality such as number of turns to completion, or the percentage of accomplishment of a stated user goal.

Among the 232 queries, 20 of them were *unrecalled*, meaning that an appropriate class was available within iNAGO's database (it was for all 232), but was not in iNAGO's model's candidate list. The existence of these instances precludes the use of an average precision score directly, as is standard in query-reranking approaches to internet search or information retrieval systems, on any

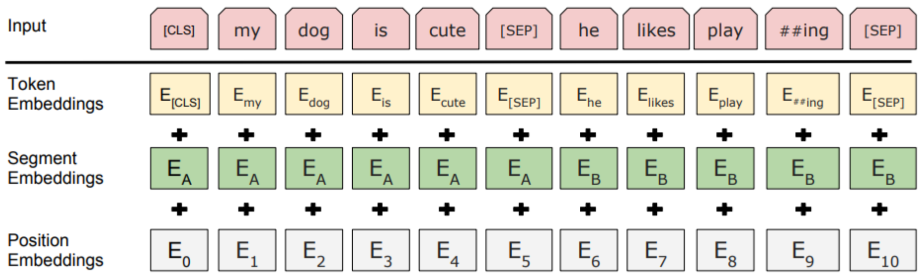

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 1: A graphical depiction of the computation of BERT embeddings (Devlin et al., 2019).

of our three models except BERT, which always produces a score.

Below, we report our evaluation of the three models on two query samples from this dataset: *Without Unrecalled Cases*, in which only the 212 queries for which the correct class label was recalled at any rank are used, and *With Unrecalled Cases*, which considers all 232. For the latter sample, for the purposes of computing the Mixed model's ranking of the 20 unrecalled queries, the manually annotated class is appended at the bottom of the list of candidate classes for iNAGO's model, with a confidence of zero.

5 Evaluation Scores

We used three different scores to evaluate each model. All can be regarded as derivative measures of performance, although they have applications to further exploratory data analysis, such as through visualization.

We compute the *mean*, *variance* and *median* of the rank of the ground truth category in each candidate list. In the case where confidences can be interpreted as probabilities, this corresponds to a data likelihood score.

We also compute the *mean reciprocal rank (MRR)*, which is a variant of mean average precision. The formula of MRR is:

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i}$$

where N is either 212 or 232 (see Section 4) and $rank_i$ denotes the rank of item i in a list. MRR is a classification accuracy measure that bestows partial credit for answers of rank greater than 1, according to a hyperbolic curve.

Finally, we compute the *top-1 accuracy* of the model. Here, we simply consider the percentage of cases where the model assigned the top rank (1) to the manually annotated class.

6 Model Evaluations

The evaluation scores are given in Tables 1–2. See also Figures 2–4 for the counts of the manually annotated label's rank (the highest was 75) without consideration of unrecalled cases, and Figures 5–8 for counts including unrecalled cases. The generally hyperbolic shape of those distributions compels us to compute the logarithms of the counts at each rank and fit those logarithms to a line with slope B using least-squares regression, having coefficient of determination, R^2.

7 Discussion of Results

There are two points that can be clearly ascertained. The first is that iNAGO's model is to be credited for its generally better performance on these queries, which were unseen during the development of that model, but mostly in-domain. BERT is widely regarded as not an easy model to beat, and iNAGO's model did beat it soundly in both conditions across all measures. As the histograms show, iNAGO's model and the Mixed model are also both generally sharper around the top rank than BERT.

That a linear combination of two independent, unbiased estimators should exist that lowers variance is to be expected. On the other hand, we did not determine the mixture parameter by directly optimizing variance or covariance. Note also that the iNAGO model's variance was already low, when it was able to locate the correct class label at any rank. This suggests that the mixed model's improvement to the iNAGO model was primarily through its greater coverage.

The second point is that iNAGO's and BERT's performances are complementary enough to be of

	Mean (Var)	Median	MRR	T1	$-B$	R^2
iNAGO	1.594 (2.489)	**1.0**	**0.854**	**77.83%**	0.866	0.741
BERT	3.675 (16.145)	2.0	0.575	41.04	0.187	0.701
Mixed	**1.552** (2.903)	**1.0**	0.851	75.00	**1.031**	0.933

Table 1: Performance on Dialogue Classification Task, not including unrecalled cases.

	Mean (Var)	Median	MRR	T1	$-B$	R^2
iNAGO	2.263 (10.766)	**1.0**	0.798	73.28%	0.266	0.407
BERT	3.560 (15.382)	2.0	0.590	43.10	0.259	0.567
Mixed	**1.629** (3.239)	**1.0**	**0.841**	**74.14**	**1.544**	0.835
BERT (unrecalled cases only)	2.350 (6.029)	1.0	0.748	65.00	0.360	0.481

Table 2: Performance on Dialogue Classification Task, including unrecalled cases.

mutual benefit to each other. This is particularly true when we consider the cases unrecalled by iNAGO's model on their own, where BERT's performance is so good that the Mixed model's performance on all 232 cases has a mean rank of less than 2.

With a corpus of this size, fine-tuning BERT was beside the point, and so this experiment was conducted in a zero-shot setting. On the other hand, the BERT model that assigned ranks within the mixed model was also the base model.[1] Corpora of this size are not uncommon to dialogue system designers, and so this is an ecologically valid setting.

8 How Did They Do It?

The better performance of iNAGO's model naturally compelled us to ask how it works. The answer is surprising in just how unsurprising it is. It begins with a round of slot/filler labelling inside the query or candidate prototype using a sequential labeller, very much as one finds in the ATIS NLU task (Niu and Penn, 2019). Three linear passes over the annotated string with very small cascades of between one and three rules lead to the iterative construction of a data structure that is essentially identical to a typed feature structure (Carpenter, 1992). The signature of the formalism contains 17 features and a type hierarchy consisting of around 40 000 types, although all but about 4 000 of those types are proper nouns that designate types of cuisine, landmarks, titles of songs, etc. The feature structures represent a combination of propositional content and user intention, the latter being classifiable into 11 discrete types.

[1]An anonymous reviewer suggested that we attempt to fine-tune the BERT model on this dataset, in spite of obvious concerns about the generalization bound on a set of this size. Indeed, performance was worse with respect to every measure after fine-tuning.

8.1 Rules

The first cascade of rules looks only for evidence that the input is or is not a continuation of an earlier dialogue, and then classifies the input by user intention. The second cascade fills in or refines the value types of features that have been determined to exist either (1) by the intention type, (2) by the presence of a particular slot filler in the labelled input sequence or (3) by previous dialogue turns in the case of a continuation. The "filling" is monotonic and is consistent with the type-inferencing rules of Carpenter (1992) that are used to compute what he terms most general satisfiers of expressions from a Rounds-Kasper-style attributed description language.

The third cascade modifies the feature structure non-monotonically if it matches a template defined by one of its rules. This stage essentially handles exceptions that could not be accommodated by the second stage. Each rule in this cascade handles one exception each. Templates can detect:

1. whether the value at a feature path has been refined by the second cascade as a result of the current input,
2. whether a feature value bears a subtype of a given type,
3. whether a feature value is exactly a given type, and
4. whether the value of one of a finite number of extra-logical variables is equal to a given constant,

5. closed under conjunction and disjunction.

The extra-logical variables are set by the state of the automobile. Impressively, however, there are only two rules/exceptions in the third cascade.

8.2 Similarity

Given a pair of these feature structures, one for a query and one for a candidate, their similarity is determined through a modified form of a weighted Jaccard index acting upon a set-theoretic reduction of the two structures. In this reduction, both feature structures are reduced to sets of feature paths terminating in a value that consists of a type and no other substructures. The actual lengths of the feature paths are irrelevant to the similarity score, but serve to identify like values between the two feature structures that can be compared. Given $|K|$ such paths, on which at least one of feature structures F and G define a value, let us call F_k (resp. G_k) the value of F (resp. G) on path $k \in K$, where it is defined, and the most general type, $\perp$ (in the orientation of Carpenter (1992) — many others would call it $\top$), elsewhere.

While each value is merely a type with no appropriate features, that type is situated within a type hierarchy. This graph of subtyping relations is assumed by Carpenter (1992) to be a meet semilattice for convenience, as it is here. Let $h(\tau)$ be the height of type τ, where the height of a type is taken to be the length of the longest chain from $\perp$ to that type. The A-similarity of F and G is then definable as:

$$A(F, G) = \frac{\Sigma_k w_k \cdot h(F_k \sqcap G_k)}{\Sigma_k w_k \cdot \max(h(F_k), h(G_k))},$$

where $\sigma \sqcap \tau$ is the meet of types σ and τ. It is this A-similarity that is returned as iNAGO's confidence score. Note that its range is $[0, 1]$ when $\Sigma_k w_k = 1$ and that high values are attained through a combination of (1) there being many (vs. few) paths on which both the query and a candidate have values defined, and (2) those values having very high (vs. low) meets. The meets are maximally high when both $F_k = G_k$ and F_k takes on a very high/specific value. iNAGO determined the weights w_k through *ad hoc* experimentation on labelled training queries that had been obtained from a different source.

The use of the height of a meet, or the depth of a least common superconcept (LCS) in the parlance of lexical semanticists, dates back to the *conceptual similarity* score of Wu and Palmer (1994), although there it is used as a normalizer on the length of the walk from F_k to G_k via their LCS in the semilattice. The walk lengths from either F_k or G_k to $F_k \sqcap G_k$ are not taken into account in A-similarity.

The iNAGO model ranks the prototypes of its classes by their A-similarity to each query, subject to two thresholds. First, no more than the top 75 classes can be returned. Second, no class with an A-similarity of less than 0.1 can be returned. These thresholds were set empirically. Only one query returned a list that was truncated at 75. The median length of ranked class labels was 11.

8.3 Nomenclature

It is clear from the nomenclature used in company-internal documentation that the developers of this system had not read Carpenter (1992), nor anything else about typed feature logic, or feature-based grammar development. Types are referred to as "entities," features as "roles," feature paths as "criteria," typed feature structures as "interpretations," the type hierarchy as a "criteria set," and chains of types in the hierarchy as "paths." Their knowledge of these structured representational devices seems to factor exclusively through the same early-1980s research on programming language theory and inclusional polymorphism that was so influential on both typed feature logic and linguistic formalisms such as HPSG (particularly the earlier, pre-1994 versions of it; Pollard and Sag, 1987), on the one hand, and modern, object-oriented, imperative programming language constructs, on the other.

As a result, we see no evidence for any sort of deeper convergence or objective suitability of the formalism for dialogue analysis that iNAGO happened upon. Instead, we claim that this manner of structured representation had become, and arguably remains, the *de facto* strategy for reasoning about language and dialogue among university-educated software engineers. The real question may therefore be not how they did it, but why they would ever have done anything else.

9 Conclusion

This paper briefly presented an evaluation of three models, a domain-specific one based upon typed feature structures, a neural language model and a mixture of the two, on an unseen but in-domain corpus of user queries. Our first recommendation is therefore that mixtures of semantically rich, conventional dialogue classifiers with neural language models should be investigated further, as our results suggest that they can produce the best combination of classifier accuracies and coverage.

We then considered the domain-specific model in more detail. While it is probable that the approach taken by this model would not scale up well to very large domains on its own, to say nothing of domain-independent dialogue modelling, it is indeed difficult to fathom why this manner of reasoning about dialogue should simply go away. Software developers, it appears, need no particular formal instruction in order to create them, perhaps apart from some standardization of their terminologies. Domain-specific approaches very apparently can still achieve higher levels of performance than what black-box semantic embeddings are currently capable of.

Our second recommendation is therefore the same as our first recommendation: we *really* should, as a community, encourage this sort of model combination as a means of enabling and enhancing what software developers are already doing without our permission. It not only improves the accuracy of the systems they are building, but may provide a low-cost means of relaxing domain restrictions. Our third recommendation is that those engaged in the formal study of structured representations should develop their unique capacity to provide the means for validating and coordinating domain-specific dialogue systems that spring up "in the wild," which will allow us to harness a very large pool of unspecialized talent to advance the state of the art in this field.

References

B. Carpenter. 1992. *The Logic of Typed Feature Structures: With Applications to Unification Grammars, Logic Programs and Constraint Resolution*. Cambridge Tracts in Theoretical Computer Science. CUP.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL*, pages 4171–4186.

J. Niu and G. Penn. 2019. Rationally reappraising ATIS-based dialogue systems. In *Proc. 57th ACL*, pages 5503–5507.

C. Pollard and I. Sag. 1987. *Information-based Syntax and Semantics*. Number 13 in CSLI Lecture Notes. CSLI Publications.

Z. Wu and M. Palmer. 1994. Verb semantics and lexical selection. In *Proc. 32nd ACL*, pages 133–138.

Chenxing Zheng. forthcoming. Self-guided question generation with transformers. Master's thesis, York University.

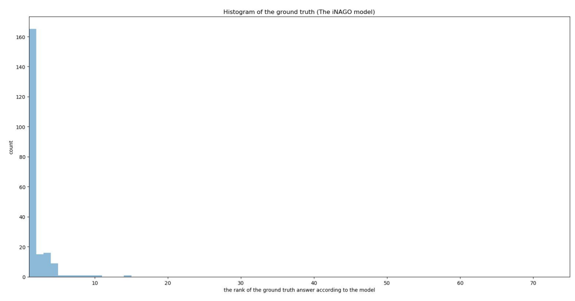


Figure 2: Count of ground truth's rank (iNAGO's model), without unrecalled cases.

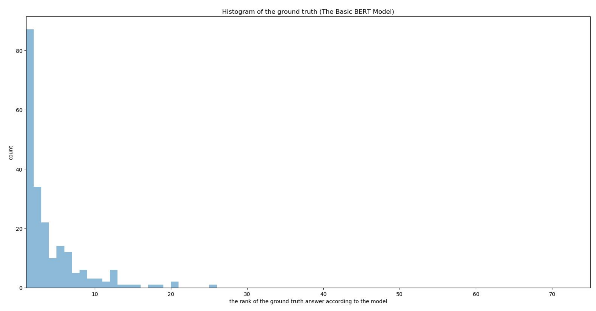


Figure 3: Count of ground truth's rank (BERT model), without unrecalled cases.

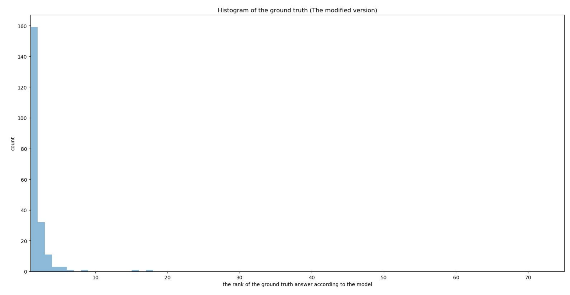


Figure 4: Count of ground truth's rank (Mixed model), without unrecalled cases.

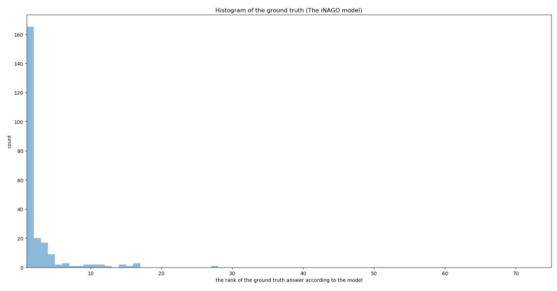


Figure 5: Count of ground truth's rank (iNAGO's model), with unrecalled cases.

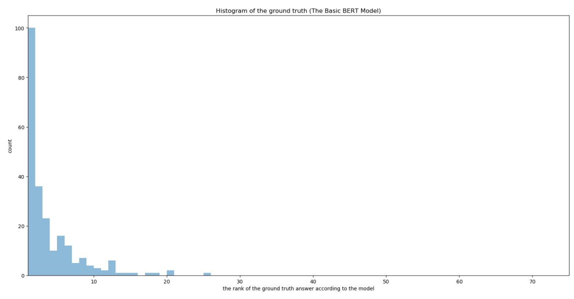


Figure 6: Count of ground truth's rank (BERT model), with unrecalled cases.

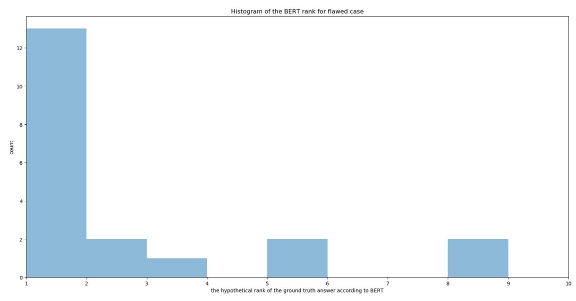


Figure 7: Count of ground truth's rank (BERT model), unrecalled cases only.

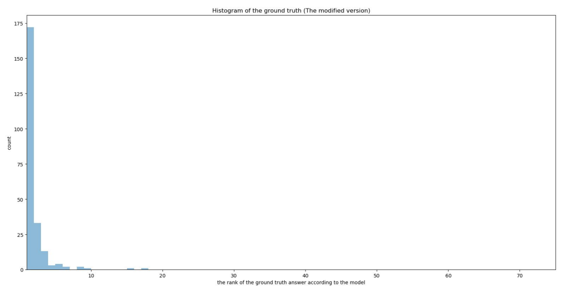


Figure 8: Count of ground truth's rank (Mixed model), with unrecalled cases.

Analytical, Symbolic and First-Order Reasoning within Neural Architectures

Samuel Ryb
Cornell University
shr59@cornell.edu

Marten van Schijndel
Cornell University
mv443@cornell.edu

Abstract

While prior work shows that pre-trained language models (PLMs) primarily emulate knowledge of entailment relations using surface heuristics, this paper examines whether PLMs learn aspects of symbolic and first-order logic relations as a side effect of learning word prediction. We introduce Logic and Knowledge Natural Language Inference (LAKNLI), a new NLI task, and we probe two different PLMs: one fine-tuned on NLI tasks and the other without NLI fine-tuning. Results show that PLMs are sometimes able to use logical knowledge for word prediction, yet they still rely heavily on heuristics. We also examine the conditions under which PLMs succeed and fail at utilizing logical relations.

1 Introduction

State-of-the-art language models often rely on surface level heuristics (McCoy et al., 2019). This is problematic when the heuristics make incorrect predictions involving logical properties (e.g., models may predict that *Either Alice knows Bob or Carl knows Claire* implies *Bob or Carl knows Claire*). This paper examines whether language models, can, in addition to surface level heuristics, infer symbolic and first-order logic relations from textual data. We introduce LAKNLI (Logic and Knowledge Natural Language Inference), a new probing dataset which assesses whether language models can reason by using logical patterns to predict entailment relationships (without being explicitly trained on them). We also examine the conditions under which BERT (Devlin et al., 2019), a widely used pre-trained language model, succeeds and fails at utilizing these logical relations.

2 LAKNLI

2.1 Overview of the Task and Dataset

In order to probe pre-trained language models (PLMs) to examine their symbolic and first-order logic reasoning abilities, we create a new probing task and dataset: LAKNLI (Logic and Knowledge Natural Language Inference). When solving LAKNLI, a language model needs to exploit the logical connective in the premise to predict whether a logical entailment exists between it and the hypothesis.

The dataset is divided according to 7 logical connectives (such as *and*, *or*, etc; full list given in Appendix A). 20 premises are attributed to each logical connective, where each premise is followed by 4 different hypotheses:

- **Premise (P)**: Some statement which is assumed as true. The premise is structured according to one of the deductive schemas given in Appendix A. **Example:** Alice got home by 2PM and met Bob then.
- **Direct Deduction (DD)** The (word-for-word) logical deduction, subject to one of the seven deductive schemas, which logically follows from the premise. A model should always judge a DD hypothesis to be entailed from the premise even if it solely relies on one of our distractor heuristics (LO or SS; defined below). **Example:** Alice got home by 2PM. Alice met Bob at 2PM.
- **Lexical Overlap (LO)** Some (possibly nonsense) bag-of-words reiterated from the premise. The hypothesis does not logically follow from the premise (Parikh et al., 2016). **Example:** Alice met home by Bob.
- **Subsequence Overlap (SS)** A random sequence of consecutive words reiterated from

the premise. The hypothesis does not logically follow from the premise (otherwise SS and DD would be indistinguishable). **Example:** 2PM and met Bob.

- **Knowledge (K)** A hypothesis which significantly restricts lexical and subsequence overlap, yet still logically follows from the premise. A model will only judge K to be entailed by the premise if the training text statistics encode some of the logical subtleties of natural language. **Example:** Someone saw someone else in the afternoon.

Premise-Hypothesis (P-H)	**Entailment**
Premise - Direct Deduction (P-DD)	✓
Premise - Lexical Overlap (P-LO)	✗
Premise - Subsequence (P-SS)	✗
Premise - Knowledge (P-K)	✓

Table 1: The entailment relations that a non-heuristic based statistical learner should predict when probed on LAKNLI.

Note that LO and SS hypotheses do not logically contradict their corresponding premises. Rather, it is not possible to derive a logical entailment relation between an LO or SS hypothesis and the corresponding premise.[1]

2.2 Distinguishing LAKNLI From Other NLI Tasks and Datasets

While other resources provide related benefits to LAKNLI, the structure of LAKNLI differs from existing NLI tasks and datasets. HANS (McCoy et al., 2019) contains LO and SS sentences which are grammatically correct, while in LAKNLI there is no requirement for the LO and SS sentences to be grammatical nor make sense. In LAKNLI, grammatical correctness is an additional heuristic that models can use to determine the entailment relation between a premise and its hypothesis. If a model fails to classify LO and SS hypotheses as being non-entailed from their premise, one should question the model's ability to accurately encode formal properties of syntax.

SuperGLUE (Wang et al., 2019) is another resource which enables probing of a model's sensitivity to logical relations. However, SuperGLUE is not as narrowly targeted for the probing of logical information as LAKNLI. In SuperGLUE, logical connectives can appear in the premise for some items and in the hypothesis for other items. In LAKNLI, the main logical connective always occurs in the premise. LAKNLI also attempts to avoid any sentence ambiguity in terms of the main logical connective and in terms of referent binding.

Lastly, the construction and use of knowledge sentences (K) which minimize the utility of LO and SS heuristics are unique to LAKNLI and, to our knowledge, have not previously been used within NLI tasks and datasets.

3 Probing BERT architectures

In this paper, we focus on probing the BERT (Devlin et al., 2019) architecture's facility with logical relations. Given BERT's extensive pre-training and success on many NLP tasks, the model can provide an example of the kinds of analyses that can be done with LAKNLI as well as providing a solid baseline measure for other how well neural language models in general would perform on LAKNLI. That is, if BERT can solve LAKNLI, other neural language models may also be able to solve this task. However, if BERT fails on this task, perhaps other neural language models will not be able to solve this task. We first use a BERT-NLI model, which is fine-tuned on SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2018), to see whether BERT has the ability to use symbolic logic to infer entailment even when fine-tuned on the task. We then probe a non-fine tuned BERT-base model on LAKNLI, to examine its capacity to reason about and deduce textual inferences correctly without explicit NLI training.

4 BERT-NLI

4.1 Preprocessing

Given a premise and a hypothesis, BERT-NLI outputs the corresponding logical relations: *entailment*, *non-entailment*, and *neutral*. We passed all of the P-{DD,LO,SS,K} sentence pairs from LAKNLI through BERT-NLI, but we coded neutral outputs as non-entailment.

Since exact lexical and syntactic overlap occurs between P and DD, any NLI-competent model should mark the relationship of all P-DD sentence pairs as entailment. However, BERT-NLI did not mark all P-DD pairs as entailment (see Appendix B), indicating a total failure to process

[1] In this paper, we use 'P-X' to indicate a premise-hypothesis pair. We use '{X,Y}_Z' to indicate that a model was trained on hypotheses of types X and Y and tested on hypotheses of type Z.

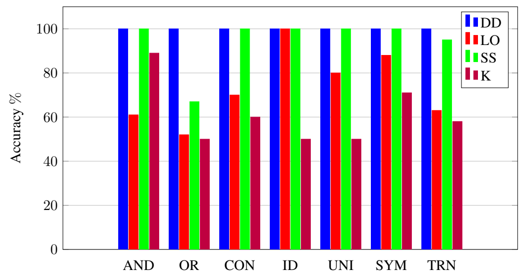


Figure 1: BERT-NLI entailment relation prediction accuracy when evaluated on a subset (492 P-H sentence pairs) of LAKNLI.

Connective	**Accuracy**
AND (conjunction)	89%
OR (disjunction)	50%
CON (conditional)	60%
ID (identity)	50%
UNI (universal)	50%
SYM (symmetry)	71%
TRN (transitivity)	58%

Table 2: BERT-NLI's accuracy in predicting the entailment relations of LAKNLI's P-K sentences.

those items. Therefore, we removed all premises and associated hypotheses in cases where P-DD was predicted as non-entailment. This resulted in removing 68 premise-hypothesis sets and preserving 492 premise-hypothesis sets.

4.2 Results and Discussion

The results (see Figure 1) parallel those of McCoy and Linzen (2018) and indicate that BERT-NLI primarily encodes entailment relations according to subsequence overlap. BERT-NLI has the most success at correctly predicting the entailment relation between AND P-K sentences, achieving an accuracy of 89% (see Table 2). Above chance performance suggests that BERT-NLI encodes some of the logical relations included in LAKNLI (AND, CON, SYM, TRN).

5 BERT-base (uncased)

Since BERT-NLI only outputs one of three NLI categories, we used BERT-base to conduct a more thorough error analysis of the kinds of logical relations that can be inferred from text statistics.

We used a support-vector machine to probe BERT. The [CLS] token in BERT encodes the overall meaning of each sentence, so we used each layer's encoding of the [CLS] token as the input to our SVM. As only 560 P-H pairs are available in LAKNLI, we used 5-fold cross-validation to train 3 different SVM probes (see Table 3).

Probe Name	**Trained On**	**Tested On**
{DD,LO}_LO	P-DD, P-LO	P-LO
{DD,SS}_SS	P-DD, P-SS	P-SS
{DD,K}_K	P-DD, P-K	P-K

Table 3: Trained probes and their descriptions.

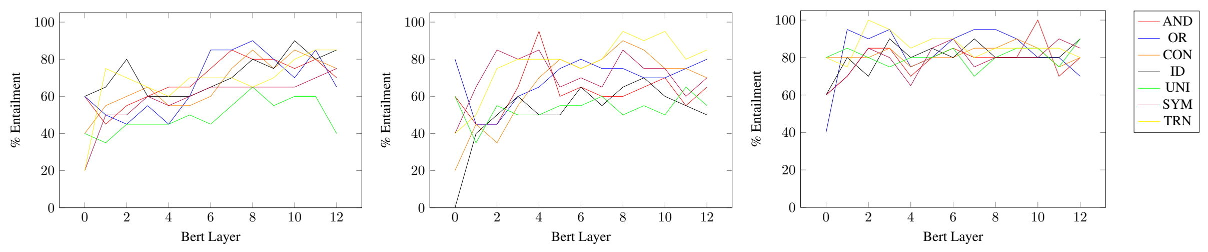
Remark Consider a probe of the form {A,B}_C, where A, B and C are some hypothesis types from LAKNLI. If C=A or C=B (i.e. the test sentences are of type A or B), the probe should predict 1 (an entailment relation), otherwise, the probe should predict 0 (a non-entailment relation). For example, consider the {DD,LO}_LO probe, which was trained on P-DD and P-LO sentence pairs. Since it was tested on P-LO sentence pairs, the probe should output 1 for all of those items. If either P-SS or P-K sentences (the item classes which were not observed during probe training) are used during testing of a {DD,LO} probe, it should output 0 for all of those items.

We trained all probes on DD hypotheses in addition to another set of hypotheses, as DD hypotheses have a common property with all LO, SS and K hypotheses. That is, DD hypotheses include lexical and syntactic overlap with P (like LO and SS hypotheses), yet are still logically entailed from P (like K sentences, which do not include overlap). Since P-K pairs contain minimal lexical overlap, training probes on only P-K pairs could make the probe negatively correlate lexical overlap with entailment. That is, the probe could learn that lexical overlap indicates non-entailment and vice-versa. Our aim in training the probe on P-DD as well as on P-K was to push the probe to identify more generalizable knowledge within BERT (i.e. lexical overlap can produce entailment under some conditions).

5.1 Experiment #1

Training probes on contextualized embeddings can yield high test accuracy even when embeddings do not necessarily encode relevant information (Hewitt and Liang, 2019). Therefore, we defined two tasks:

- **The linguistic task** tracked whether BERT solved LAKNLI using logical relations. For this task we used the {DD,K}_K probe.



(a) {DD,LO}_LO (b) {DD,SS}_SS (c) {DD,K}_K

Figure 2: Percent of test items classified as containing an entailment relation across the linguistic and control task probes.

- **The control task** tracked whether BERT solved LAKNLI using surface heuristics. For this task we used the {DD,LO}_LO and {DD,SS}_SS probes.

Each of these probes was trained on the P-H [CLS] embeddings from each of the 13 layers of the model,[2] and we tracked the flow of information throughout each layer of BERT. Figure 2c shows the number of linguistic task items classified as having an entailment relation while Figures 2a and 2b show the percent of control task items classified as having an entailment relation, for each logical connective, across each layer of BERT-base.

5.1.1 Results and Discussion

All of the probes classified several items as exhibiting entailment across all non-embedding layers for both the linguistic and control tasks (see Figure 2). These results replicate previous findings showing that BERT relies on surface level heuristics (McCoy and Linzen, 2018; McCoy et al., 2019), but we found that information about logical relations were decodable from BERT's internal representations as well.

In order to determine whether BERT's contextualized embeddings encoded the semantics of logical connectives or whether the trained probes learned the logical connectives separately from BERT, we measured the *selectivity* of the probing task (Hewitt and Liang, 2019). However, we modified the original definition to fit our experiment:

Definition 1. *selectivity = percent entailment of {DD,K}_K probe on $layer_i$ - max(percent entailment of {DD,LO}_LO probe on $layer_i$, percent entailment of {DD,SS}_SS probe on $layer_i$).*

A positive selectivity score represents the degree to which the probe tracked entailment relations using logic rather than surface heuristics, while the absolute value of a negative selectivity score represents the degree to which the probe tracked entailment relations using surface heuristics rather than logic (see Figure 3 and Appendix C).

[2] We denote the embedding layer as *layer 0*.

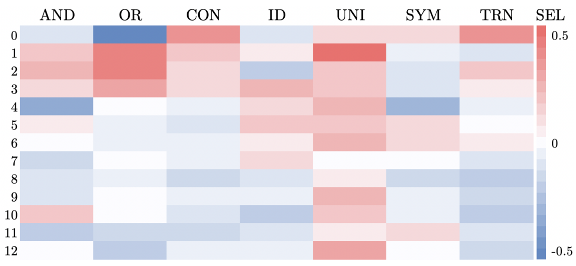


Figure 3: Probe selectivity computed by Definition 1.

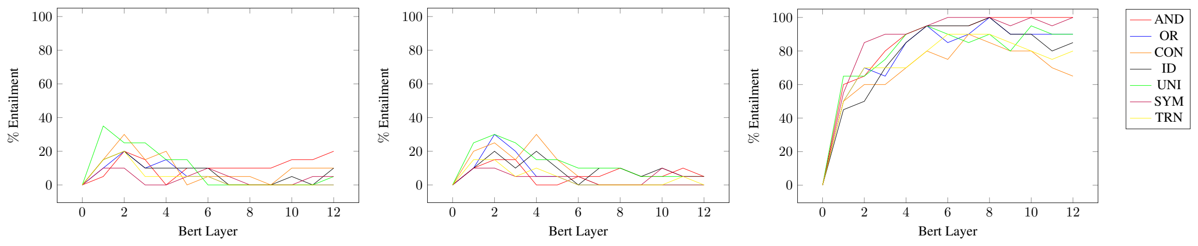
A selectivity score of 1 would indicate that BERT solved LAKNLI using *only* logical understanding and knowledge. While this was not what we observed, our results still confirm that BERT was able to track entailment relations and encode some symbolic and first-order reasoning properties when solving LAKNLI (represented by a positive selectively score, with some exceptions where the selectivity scores are negative).

Our results offer a counterpoint to McCoy et al. (2019) who claimed that BERT primarily encodes entailment relations according to surface heuristics. We confirmed that BERT uses surface heuristics but sometimes also encodes knowledge and logical reasoning, though it was trained solely on text data.

5.2 Experiment #2

In order to examine how BERT distinguishes between knowledge and surface heuristic hypotheses, we trained three more probes (see Table 4). The percent of test items the probes classified as entailment can be seen in Figure 4.

The goal of this analysis was to identify the specific features encoded by BERT that can distinguish the hypotheses from one another. Lexical overlap is one obvious difference, which we controlled for



(a) {K}_LO (b) {K}_SS (c) {K}_K

Figure 4: Percent of test sentence pairs that are classified as having an entailment relation when trained on K sentences and tested on LO, SS and K.

Probe Name	**Trained On**	**Tested On**
{K}_LO	P-K	P-LO
{K}_SS	P-K	P-SS
{K}_K	P-K	P-K

Table 4: Trained probes and their descriptions

in the previous analysis by including DD hypotheses in each probe training set. By removing DD hypotheses from the probe training sets in this analysis, we anticipated that the probes would learn a negative correlation between lexical overlap and logical entailment. However, we also expected that the probes would help us identify other encoded features that distinguish between knowledge-based properties and surface layer heuristics.

5.2.1 Results and Discussion

The {K}_K probe (Figure 4c) performed substantially above chance after layer 4. The high percentage of items classified as containing an entailment relation in the intermediate and upper layers suggests that BERT was able to exploit logical connectives to correctly determine entailment relations. This result is consistent with previous observations that BERT's intermediate and upper layers can encode semantic meaning (Jawahar et al., 2019), though it may also indicate the availability of input features at higher layers of BERT (we explore this more at the end of this section).

The {K}_LO (Figure 4a) and {K}_SS (Figure 4b) probes should have achieved 0 percent entailment classification across all layers, for each logical connective. When trained on P-K pairs, the probe should not have been able to deduce an entailment relation between the P-LO and P-SS sentence pairs. While only a small percentage of items were classified as containing entailment relations in layers 6-12, the lower 6 layers still classified 30% of test items as entailment relations. That the probes marked entailment relations between premises and (sometimes ungrammatical) hypotheses suggests that BERT does not use syntax and grammar as a heuristic to encode logical entailment relations. While BERT has the ability to handle subject-verb agreement, learn rich syntactic features from middle layers and encode the most information regarding linear word order in lower layers (Goldberg, 2019; Jawahar et al., 2019; Rogers et al., 2020) our results should cause one to question BERT's understanding of natural language grammar properties.

In order to determine how much more BERT was able to distinguish knowledge hypotheses (K) from their corresponding lexical and subsequence overlap hypotheses (LO, SS), we again used selectivity (although modified slightly from Definition 1):

Definition 2. *selectivity = percent entailment classification of {K}_K probe on layer*$_i$ *- max(percent entailment classification of {K}_LO probe on layer*$_i$ *, percent entailment classification of {K}_SS probe on layer*$_i$*).*

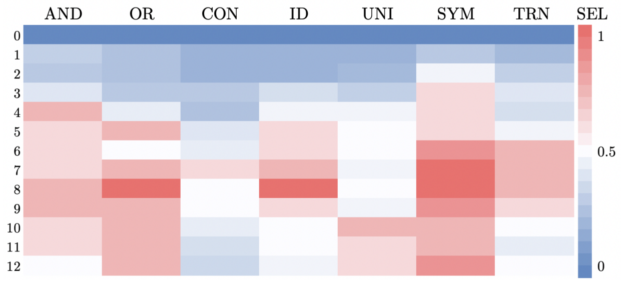


Figure 5: Probe selectivity scores computed by Definition 2.

Per Figure 5, the probe was best able to distinguish between knowledge and surface heuristics within layers 5 - 10 (see Figure 5 and Appendix C). This result aligns with our previous selectivity results (Figure 3 and Figure 8), which indicated that the probe was able to decode logical relations from BERT's representations mainly in the intermediate (and some upper) levels. We hypothesize that this was most likely due to BERT encoding some properties of syntactic structures primarily in lower layers (used for LO and SS sentences) and encoding semantic and pragmatic information in intermediate and upper layers (used for K sentences).

As noted earlier, the high selectivity scores may have been due to a lack of lexical overlap between the premise and knowledge hypotheses. To explore the influence of lexical overlap on the probe, we trained an additional {K}_DD probe (see Figure 6).

Probe Name	Trained On	Tested On
{K}_DD	P-K	P-DD

Table 5: Trained probe and its description.

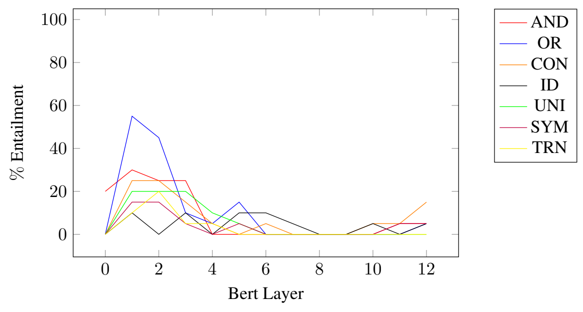


Figure 6: Percent of test items classified as containing an entailment relation by a {K}_DD probe.

The {K}_DD probe should have achieved near 100 percent entailment classification across all layers. However, as seen in Figure 6, this did not occur, which indicates that a probe trained solely on K sentences will learn entailment based on the *lack* of surface level heuristics. However, a probe trained on both DD and K sentences (that is, a probe trained equally on sentences with and without surface level overlap), can produce more generalizable entailment predictions, independent of lexical overlap, as seen in Figure 2c.

Therefore, one potential solution to ensure accurate probing results is to train probes on data that contains a balance of surface features and knowledge features. Furthermore, adding contradictory knowledge hypotheses to LAKNLI would enable researchers to calculate precision and recall scores, which would give a better indication as to whether a probe solves LAKNLI using logical relations.

6 Error Analyses

We next conducted qualitative analyses, where, when applicable, we categorized P-K sentence pairs from LAKNLI according to an error type. All analyses used data from the {K}_K probe in Section 5.2.[3] Below are the error types we used, followed by their descriptions and a sample P-K sentence pair:

- **Visual Reasoning (VR)** Sentences which require an analytic understanding of phrases related to spatial reasoning, such as *left of, in front of, behind, between* etc. **Example:** *P: If Alice is next to Bob, Bob is to the right of Carl. Alice is next to Bob → K: Carl is to the left of someone.*
- **Common Knowledge (CK)** Sentences which should be generally understood without any specialized knowledge. **Example:** *P: Alice is friends with Bob → K: Alice and Bob both like each other.*
- **World Knowledge (WK)** Sentences which require general knowledge about entities (such as animals, geographic locations, etc.) in the real world. **Example:** *P: Every boy who likes Alice is in New York City. Bob likes Alice → K: Bob is in North America.*

	AND	OR	CON	ID	UNI	SYM	TRN	TOTAL
VR	4	3	2	1	2	0	7	19
CK	7	3	8	8	9	14	10	59
WK	5	3	5	7	6	1	2	29

Table 6: Number of P-K sentence pairs tagged according to an error type within LAKNLI.

Visual Reasoning Prior visual commonsense reasoning (VCR) tasks such as NLVR (Suhr et al., 2017) have some similarities to LAKNLI, although we only probed PLMs on textual data. Our goal was to examine PLMs' inferential understanding of object relations through the use of non-visually grounded language and analytic consequences (e.g., X is to the right of Y $\iff$ Y is to the left of X).

Sentences which are tagged as VR within LAKNLI can be further classified according to the reasoning phrase involved. Such reasoning phrases and their frequencies are: *next to* (5), *right of / left of* (9), *in front of / behind* (5), *on top of / below* (3), *north of / south of* (1).

[3]Within this section the phrase *the probe* refers to the {K}_K probe.

The probe seemed to struggle with understanding analytic consequences including *right of/left of* (e.g., X is to the right of Y $\iff$ Y is to the left of X), *on top of/below* (e.g., X is on top of Y $\iff$ Y is below X), and *in front of/behind* (e.g., X is in front of Y $\iff$ Y is behind X) in lower layers. However, from layer 4 and above, the degree of error substantially decreased, with only 1 error in layers 4, 5, 6 and 8 and 2 errors in layer 7. This result suggests that BERT was able to perform best at a visual reasoning task between layers 4-8, which aligns with the results from Section 5.2.

The majority of incorrect entailment predictions in the upper layers (9-12) required understanding the relationship between left and right. Even in upper layers, which are supposedly more pragmatically advanced (Tenney et al., 2019), BERT was not fully able to understand such spatial implications.

In terms of logical connectives, it is possible that BERT also encodes the semantics of the biconditional deductive schema (which is not one of the seven deductive schemas included in LAKNLI) from layer 4 upwards, since the probe correctly predicted the majority of visual reasoning phrases which involved analytic consequences. However, due to the small size of LAKNLI, and since BERT did not encode the pragmatic relationship between left and right, future work should probe BERT on larger visual reasoning datasets with an emphasis on analytic consequences.

Common Knowledge Common knowledge reasoning tasks require language models to understand general scenarios that humans intuitively understand (Mostafazadeh et al., 2016; Zellers et al., 2018; LoBue and Yates, 2011).

For the error analysis, we further subcategorized CK P-K sentence pairs into the following types: *Description* (sentences which include a general description about an individual or circumstance), *General Action* (sentences which involve an individual doing an action), *Spatial Relation* (sentences with phrases that impose a spatial relation between at least two entities yet do not require a language model to solve a visual reasoning task), *Time* (sentences which refer to specific and/or general times of the day).

While many errors occurred with no particular pattern in the lower layers (1-3), BERT seems able to encode information regarding *spatial relations* in these lower layers, particularly understanding phrases such as *in proximity, near, adjacent, is in*. BERT seemed to encode general descriptions from layer 6 and above, with the probe classifying 100% of the test items as entailment relations within layers 7-11.

The probe also struggled to correctly predict the entailment relations of some *general action* P-K sentence pairs, particularly in lower layers 1-4 and in upper layers 9, 11-12 despite classifying a large number of items as entailment relations in the intermediate (and one upper) layers 6-8, 10. The probe incorrectly predicted the entailment relations of the following two general action P-K sentence pairs (a) *P: Alice is at home. If Bob walks to the park then Alice walks to the park. Bob walks to the park.* $\rightarrow$ *K: Alice leaves her house*, (b) *P: Alice is at home. Alice walks to the park with Bob. Bob walks to the park with Carl.* $\rightarrow$ *K: Alice leaves her house* in layers 1-4, 11, 12 and 1-5, 9-12, respectively. Since these were the only P-K sentence pairs in which the displacement of an agent from one location to another was apparent, it is plausible that BERT fails to understand such a relation. This hypothesis is supported by Forbes et al. (2019) who highlighted BERT's struggle to reason about objects and properties in the physical world.

BERT's understanding of *time* was inconsistent, with low entailment accuracy in the higher layers of the model. The probe incorrectly predicted the entailment relations of two P-K sentence pairs in layers 7, 9-12 and layers 9-10, which required the model to understand that eating at 2PM is associated with lunch time. These initial results correlate with those of (Han et al., 2019) that BERT embeddings do not achieve high accuracy at temporal relation extraction tasks. While their testing sentences included phrases which were indicative of temporal relations, the sentences in LAKNLI make use of numerical symbols to denote time (e.g., *eats lunch at* ***2****PM*). Therefore, it may be that the probe incorrectly predicted some entailment relations due to BERT struggling to encode numerical symbols (Wallace et al., 2019).

World Knowledge For this analysis, world knowledge P-K sentence pairs were further subcategorized into three types: *Location* (sentences which include relations between cities, countries and continents), *Eco-Systems* (sentences which require an understanding of the basic facts of nature), *Languages* (sentences which require an understanding of the common facts about languages and where they are typically spoken).

The probe correctly predicted the entailment relations of *language* P-K sentence pairs such as *P: If Alice studies French, Bob studies Italian and no other languages. Alice studies French and English and Spanish* $\rightarrow$ *K: Bob has knowledge of the language spoken in Venice* throughout all of BERT's layers, suggesting that BERT encodes where certain languages are commonly spoken. This was supported by the probe correctly predicting the entailment relation of *P: Alice studies either English or French and Bob either studies English or Spanish. Neither Alice nor Bob study English* $\rightarrow$ *K: Alice and Bob both learn a Romance language* in all layers, which indicates that perhaps BERT knows that French and Spanish are both considered Romance languages. This suggests that BERT may be encoding properties of set-theoretic membership (e.g., Spanish $\in$ Romance languages, French $\in$ Romance languages).

Half of the *eco-system* P-K sentence pairs required understanding common features of sea creatures, such as their abilities to swim or live in water. The probe correctly predicted the entailment relations of all those P-K sentence pairs in layers 1-8, yet struggled in upper layers (9, 11-12). This result aligns with recent work by Singh et al. (2020) which stressed that many of the intermediate layers contain knowledge-based information that is not included in the final layer.

7 Conclusions

Our work shows that BERT encodes some symbolic and first-order logic relations after training on only textual information. Despite lexical overlap having a large effect, we find that SVM probes of BERT trained on LAKNLI's knowledge sentences still achieve high NLI accuracy. It is therefore possible that the text statistics encode enough about symbolic logic relations for BERT to use these relations to solve NLI tasks. However, since text-trained models are likely unable to effectively learn reference (Merrill et al., 2021), future work must be done to determine the extent of symbolic understanding possible in models that lack reference. We hope that the LAKNLI dataset can help further investigate this question.

Acknowledgements

We thank the reviewers for their feedback on this work. Furthermore, we thank John Whitman and Claire Cardie for earlier feedback and suggestions pertaining to this work.

References

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*, pages 632–642, Lisbon, Portugal. 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.

Maxwell Forbes, Ari Holtzman, and Yejin Choi. 2019. Do neural language representations learn physical commonsense? *CoRR*, abs/1908.02899.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *CoRR*, abs/1901.05287.

Rujun Han, Mengyue Liang, Bashar Alhafni, and Nanyun Peng. 2019. Contextualized word embeddings enhanced event temporal relation extraction for story understanding. *CoRR*, abs/1904.11942.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. *CoRR*, abs/1909.03368.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Peter LoBue and Alexander Yates. 2011. Types of common-sense knowledge needed for recognizing textual entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 329–334, Portland, Oregon, USA. Association for Computational Linguistics.

R. Thomas McCoy and Tal Linzen. 2018. Non-entailed subsequences as a challenge for natural language inference. *CoRR*, abs/1811.12112.

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

Will Merrill, Yoav Goldberg, Roy Schwartz, and Noah A. Smith. 2021. Provable limitations of acquiring meaning from ungrounded form:what will future language models understand? *TACL*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

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.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works.

Jaspreet Singh, Jonas Wallat, and Avishek Anand. 2020. BERTnesia: Investigating the capture and forgetting of knowledge in BERT. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 174–183, Online. Association for Computational Linguistics.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of natural language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 217–223, Vancouver, Canada. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307–5315, Hong Kong, China. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *CoRR*, abs/1905.00537.

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, New Orleans, Louisiana. Association for Computational Linguistics.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.

A LAKNLI's Logical Deductive Schema Templates

The seven logical connectives and their deductive schemas are defined below in the Fitch format. This demonstrates (1) the crossover between zeroth/first-order logic and equivalence relations within natural language (the main logical connective is bolded within P) and (2) the relationship between premises and the 4 different hypotheses (DD, LO, SS, K) subject to each logical connective:

And (conjunction) Elim (AND)

1	$A \wedge B$	
2	A	$\wedge Elim(1)$
3	B	$\wedge Elim(1)$

Sample AND Sentences

P *Alice is friends with Bob* ***and*** *Alice is not friends with Carl.*

DD *Alice is friends with Bob. Alice is not friends with Carl.*

LO *Bob is friends with Carl.*

SS *Bob and Alice is not friends with Carl.*

K *Alice knows two people.*

Or (disjunction) Elim (OR)

1	$A \vee B$	
2	$\neg A$	
3	$\quad A$	
4	$\quad \bot$	$\bot Intro(2,3)$
5	$\quad B$	$\bot Elim(4)$
6	$\quad B$	
7	$\quad B$	$Reit(6)$
8	B	$Or - Elim(1, 3-5, 6-7)$

Sample OR Sentences
P *Alice is at home.* ***Either*** *Alice walks to the park* ***or*** *Bob walks to the park. Alice does not walk to the park.*
DD *Bob walks to the park.*
LO *Home walks to the park.*
SS *to the park or Bob walks.*
K *A man does not remain in his current state.*

Conditional Elim (CON)

1	$A \rightarrow B$	
2	A	
3	B	$\rightarrow Elim(1,2)$

Sample CON Sentences
P ***If*** *Alice attends the party Bob attends the party. Alice attends the party.*
DD *Bob attends the party.*
LO *Attends Alice the party.*
SS *The party Bob attends.*
K *Two people attend an event.*

Identity Elim (ID)

1	$b = f(a)$	
2	$P(f(a))$	
3	$P(b)$	$= Elim(1,2)$

Sample ID Sentences
P *Bob* **is** *Alice's uncle. Alice's uncle* **is** *in New York City.*
DD *Bob is in New York City.*
LO *New York City is Alice's uncle.*
SS *Uncle is in New York City.*
K *Somebody is in North America.*

Universal Elim (UNI)

1	$\forall x(P(x,a) \rightarrow Q(x))$	
2	$P(b,a)$	
3	$P(b,a) \rightarrow Q(b)$	$\forall Elim(1)$
4	$Q(b)$	$\rightarrow Elim(2,3)$

Sample UNI Sentences
P ***Every*** *boy who likes Alice is in New York City. Bob likes Alice.*
DD *Bob is in New York City.*
LO *New York City is Alice.*
SS *Alice is in New.*
K *Somebody is in North America.*

Symmetry (SYM)

1	$A \sim B$
2	$B \sim A$

Sample SYM Sentences
P *Alice's party outfit* ***is similar*** *to Bob's shirt.*
DD *Bob's shirt is similar to Alice's party outfit.*
LO *Bob's outfit is similar to Alice's party.*
SS *Party outfit is similar to Bob's.*
K *Two people have comparable clothing.*

Transitivity (TRN)

1	$A \sim B$
2	$B \sim C$
3	$A \sim C$

Sample TRN Sentences
P *Alice's party* ***is north of*** *Bob's party. Bob's party* ***is north of*** *Carl's party.*
DD *Alice's party is north of Carl's party.*
LO *Alice's party is Bob's party.*
SS *Bob's party is north.*
K *Carl's event is located in a southern location compared to Alice's event.*

We avoided using negation ($\neg$), except when necessary (e.g. in the OR deductive schema), as it is often used as a heuristic by PLMs to infer a non-entailment relation between a premise and a hypothesis (McCoy and Linzen, 2018). However, considering that negation changes the semantics of universal quantifiers (e.g. $\neg\forall x P(x) \iff \exists x \neg P(x)$), conditionals (e.g. Modus Tollens $P \rightarrow Q, \neg Q \therefore \neg P$) and conjunctions/dijunctions ($\neg(A \wedge B) \iff \neg A \vee \neg B$) amongst other logical connectives, we leave it for future work to determine a more sophisticated approach for probing PLM's abilities to understand first-order equivalences involving negation.

B BERT-NLI Preprocessing results

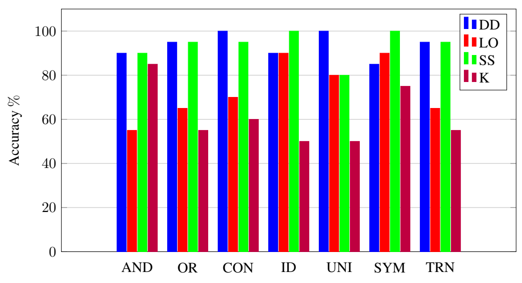
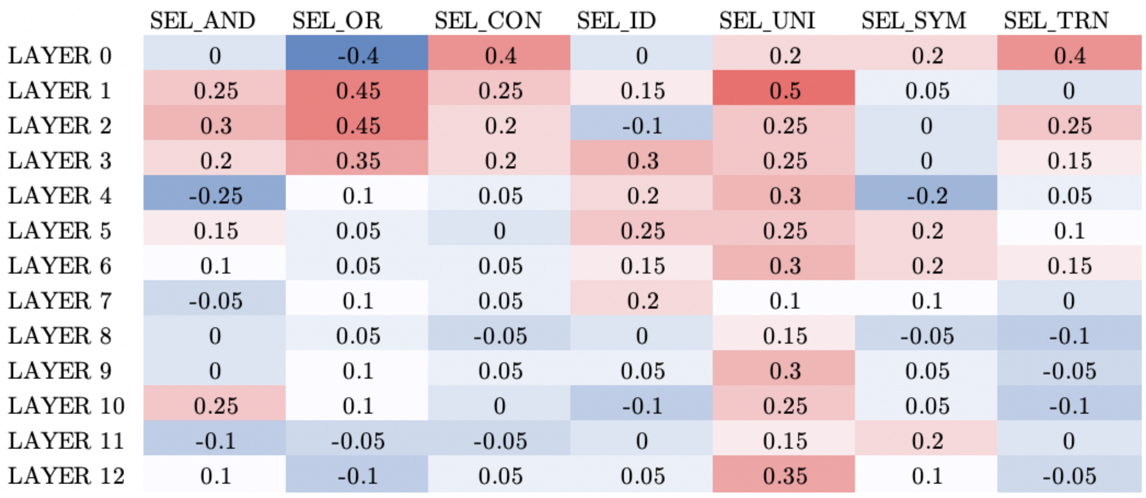


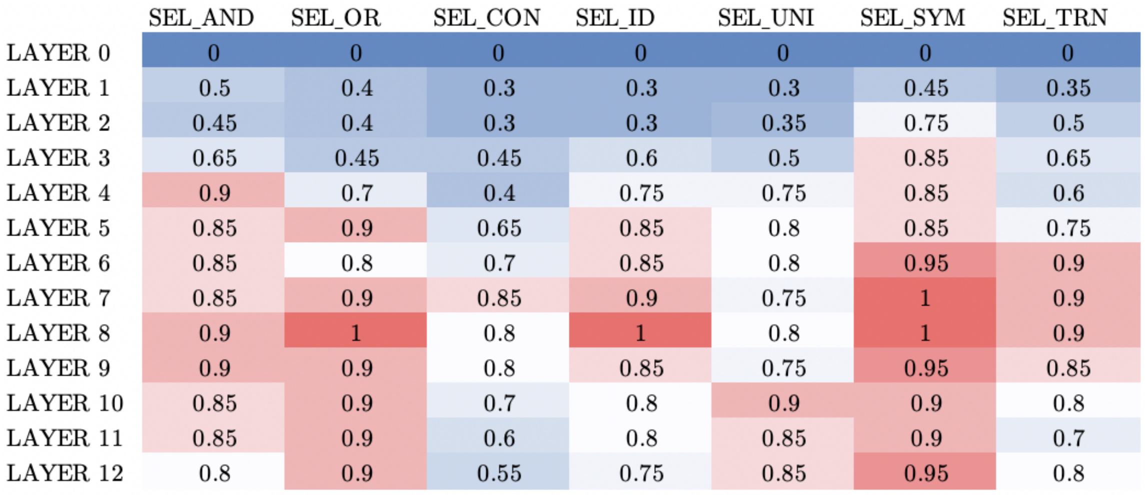
Figure 7: BERT-NLI entailment relation prediction accuracy across all 560 P-H sentence pairs from LAKNLI.

C Selectivity Scores



	SEL_AND	SEL_OR	SEL_CON	SEL_ID	SEL_UNI	SEL_SYM	SEL_TRN
LAYER 0	0	-0.4	0.4	0	0.2	0.2	0.4
LAYER 1	0.25	0.45	0.25	0.15	0.5	0.05	0
LAYER 2	0.3	0.45	0.2	-0.1	0.25	0	0.25
LAYER 3	0.2	0.35	0.2	0.3	0.25	0	0.15
LAYER 4	-0.25	0.1	0.05	0.2	0.3	-0.2	0.05
LAYER 5	0.15	0.05	0	0.25	0.25	0.2	0.1
LAYER 6	0.1	0.05	0.05	0.15	0.3	0.2	0.15
LAYER 7	-0.05	0.1	0.05	0.2	0.1	0.1	0
LAYER 8	0	0.05	-0.05	0	0.15	-0.05	-0.1
LAYER 9	0	0.1	0.05	0.05	0.3	0.05	-0.05
LAYER 10	0.25	0.1	0	-0.1	0.25	0.05	-0.1
LAYER 11	-0.1	-0.05	-0.05	0	0.15	0.2	0
LAYER 12	0.1	-0.1	0.05	0.05	0.35	0.1	-0.05

Figure 8: Probe selectivity scores computed by Definition 1.



	SEL_AND	SEL_OR	SEL_CON	SEL_ID	SEL_UNI	SEL_SYM	SEL_TRN
LAYER 0	0	0	0	0	0	0	0
LAYER 1	0.5	0.4	0.3	0.3	0.3	0.45	0.35
LAYER 2	0.45	0.4	0.3	0.3	0.35	0.75	0.5
LAYER 3	0.65	0.45	0.45	0.6	0.5	0.85	0.65
LAYER 4	0.9	0.7	0.4	0.75	0.75	0.85	0.6
LAYER 5	0.85	0.9	0.65	0.85	0.8	0.85	0.75
LAYER 6	0.85	0.8	0.7	0.85	0.8	0.95	0.9
LAYER 7	0.85	0.9	0.85	0.9	0.75	1	0.9
LAYER 8	0.9	1	0.8	1	0.8	1	0.9
LAYER 9	0.9	0.9	0.8	0.85	0.75	0.95	0.85
LAYER 10	0.85	0.9	0.7	0.8	0.9	0.9	0.8
LAYER 11	0.85	0.9	0.6	0.8	0.85	0.9	0.7
LAYER 12	0.8	0.9	0.55	0.75	0.85	0.95	0.8

Figure 9: Probe selectivity scores computed by Definition 2.

Author Index