Formal techniques for Neuro-Symbolic Modeling Lecture 5

Kyle Richardson and Vivek Srikumar

This lecture

Tasks = contracts

We want models that do more than what the data says

Learning from ~~examples~~ Knowledge

Relaxing logic and using relaxed logic to learn

A worked example

Three case studies

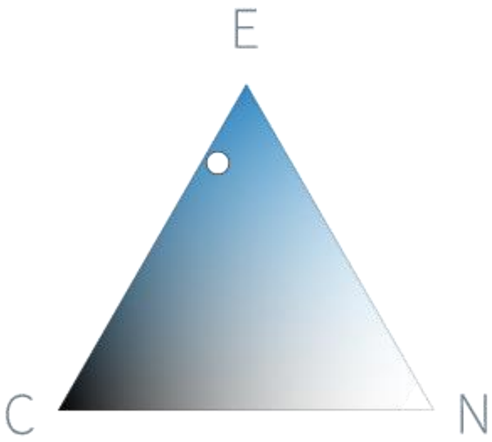
Tasks = contracts

We want models that do more than what the data says

Example 1: Natural language inference

Premise Before it moved to Chicago, aerospace manufacturer Boeing was the largest company in Seattle.

Hypothesis Boeing is a Chicago-based aerospace manufacturer.



Judgment	Probability
Entailment	75.6%
Contradiction	19.9%
Neutral	4.5%

It is quite likely that the premise entails the hypothesis.

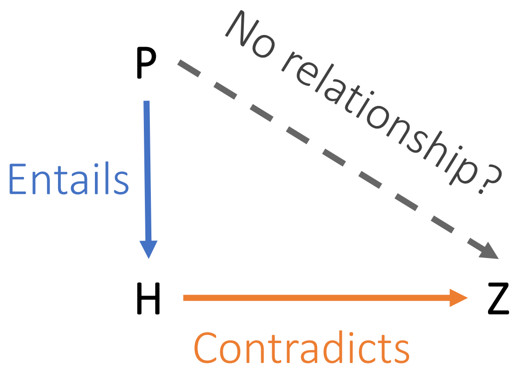
https://demo.allennlp.org/textual-entailment/

Can neural networks understand text?

P John is on a train to Berlin.

H John is traveling to Berlin.

Z John is having lunch in Berlin.



The same system cannot simultaneously hold these three beliefs!

Violates this invariant

```
If P entails H and H contradicts Z
```

Can neural networks use such "theory" in the form of invariant knowledge?

A BERT-based model that gets ~90% on benchmark data violates this invariant on 46% of a large collection of sentence triples.

Tasks* define predicates

Example: The natural language inference task defines three predicates called `Entail(P,H)`, `Contradict(P,H)` and `Neutral(P,H)`

P	John is on a train to Berlin.	→	`Entail(P, H)` ¬`Contradict(P,H)` ¬`Neutral(P,H)`
H	John is traveling to Berlin.		

Labeled datasets show examples of these predicates

Models try to find the best fitting predicates given their arguments

* Tasks that require labeling examples or parts of them

Model behavior as constraints

Expected behavior: *"If a sentence P entails a sentence H, and H entails the sentence Z, then P entails Z"*

$$\forall \text{ sentences } P, H, Z, \qquad \texttt{Entail}(P,H) \land \texttt{Entail}(H,Z) \rightarrow \texttt{Entail}(P,Z)$$

(Four such valid transitivity constraints exist)

Expected behavior: *"The contradict predicate is symmetric."*

$$\forall \text{ sentences } P, H, \qquad \texttt{Contradict}(P,H) \leftrightarrow \texttt{Contradict}(H,P)$$

A Logic-Driven Framework for Consistency of Neural Models. *Li, Gupta, Mehta and Srikumar*. EMNLP, 2019.

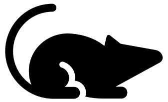
Example 2: Semantic Role Labeling (SRL)

Who did what to whom, where, when, why?

The mouse ate the cheese when the farmer left the kitchen.

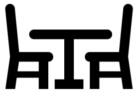
Arg0 Arg1 Arg0 M-T Arg1

These *semantic roles* are defined by the PropBank data (Palmer et al)



	ate
Arg0	The mouse
Arg1	the cheese
ArgM-TMP	when the farmer left the kitchen

	left
Arg0	the farmer
Arg1	the kitchen



Semantic Role Labeling: The contract

- **Input**: A sentence
- **Output**: *Structured* semantic frames for all verbs

Expected behavior: Outputs should satisfy certain constraints

- Core arguments (e.g. `Arg0`, `Arg1`) cannot repeat…
  …but modifiers (e.g. `ArgM-TMP`) can

- Certain arguments (called references, e.g. `R-Arg0`) can appear only if the corresponding referent argument exists (here, `Arg0`)

These symbolic constraints come from the task definition and linguistic assumptions

If labels satisfy symbolic properties…

…when and how do we inject this knowledge into the modeling and prediction process?

Can we do so using the existing gradient-based machinery for neural networks?

Learning from ~~examples~~ knowledge

Relaxing logic and using relaxed logic to learn

Where can knowledge be involved?

Model design

Loss function design & training

Enforce congruent predictions

Augmenting neural networks with logic
(ACL 2019)

Logic-based loss design
(EMNLP 2019,
ACL 2020, IJCAI 2021)

Structured learning & prediction

Model counting covered in lecture 3

Covered in lecture 4

This section of the tutorial

Neural network land vs. Logic land

Neural Networks

✓Differentiable compute, easy to use

✗ Hard to supervise except via labeled examples

First-order logic

✗ Not differentiable, hard to use with today’s best infrastructure

✓ Expressive and easy to state for domain experts

What we want: **Best of both**!

Three challenges facing logic in neural network land

1. Bridging predicates in rules with neural networks

2. Making logic differentiable

3. Using differentiable logic

Predicates in neural networks

All neural networks expose *interfaces* in the form of nodes that have externally defined meaning

Recall: Labels are predicates

P John is on a train to Berlin.

H John is traveling to Berlin.

`Entail`

BERT

`P,H`

= `Entail(P, H)`

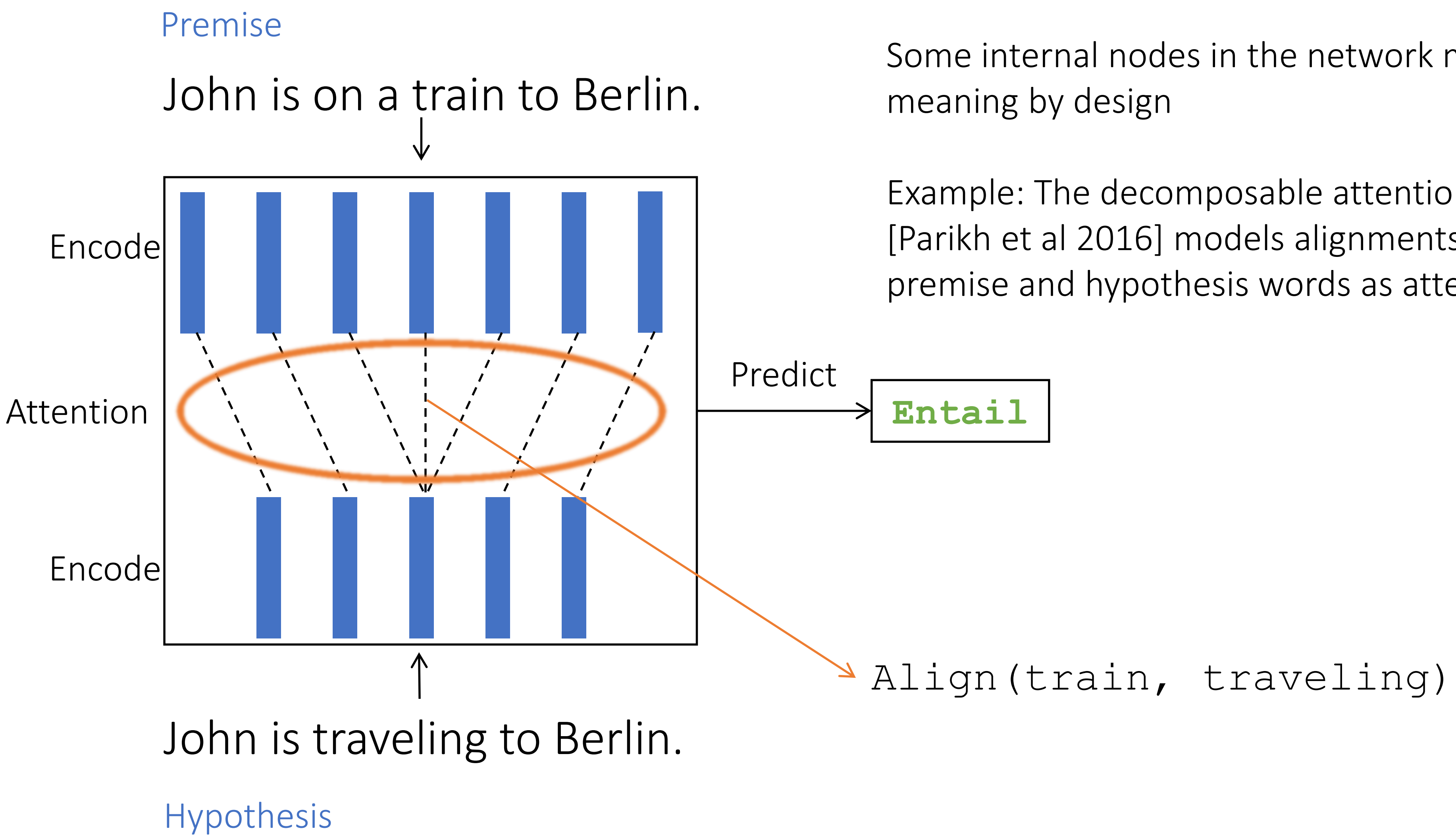
Labeled datasets are formal specifications

(P1, H1, Entail)
(P2, H2, Contradict)
(P3, H3, Neutral)
(P4, H4, Neutral)

=

$\texttt{Entail}(P1, H1)$
$\land\, \texttt{Contradict}(P2, H2)$
$\land\, \texttt{Neutral}(P3, H3)$
$\land\, \texttt{Neutral}(P4, H4)$

Predicates *within* neural networks



Some internal nodes in the network may have meaning by design

Example: The decomposable attention model [Parikh et al 2016] models alignments between premise and hypothesis words as attention

Parikh, Täckström, Das, and Uszkoreit. "A Decomposable Attention Model for Natural Language Inference." In *EMNLP 2016*

Named neurons

Nodes in a computation graph that have *externally* defined meaning

Named neurons can be:

- Any output nodes in the network
- Inputs to the network and their deterministic properties
- Sometimes, internal nodes that have defined behavior

Named neurons give us the vocabulary for writing rules

Three challenges facing logic in neural network land

1. Bridging predicates in rules with neural networks?

   Answer: *Named neurons*, nodes in a computation graph that have *externally* defined meaning

2. Making logic differentiable?

3. Using differentiable logic?

Relaxing Boolean operators

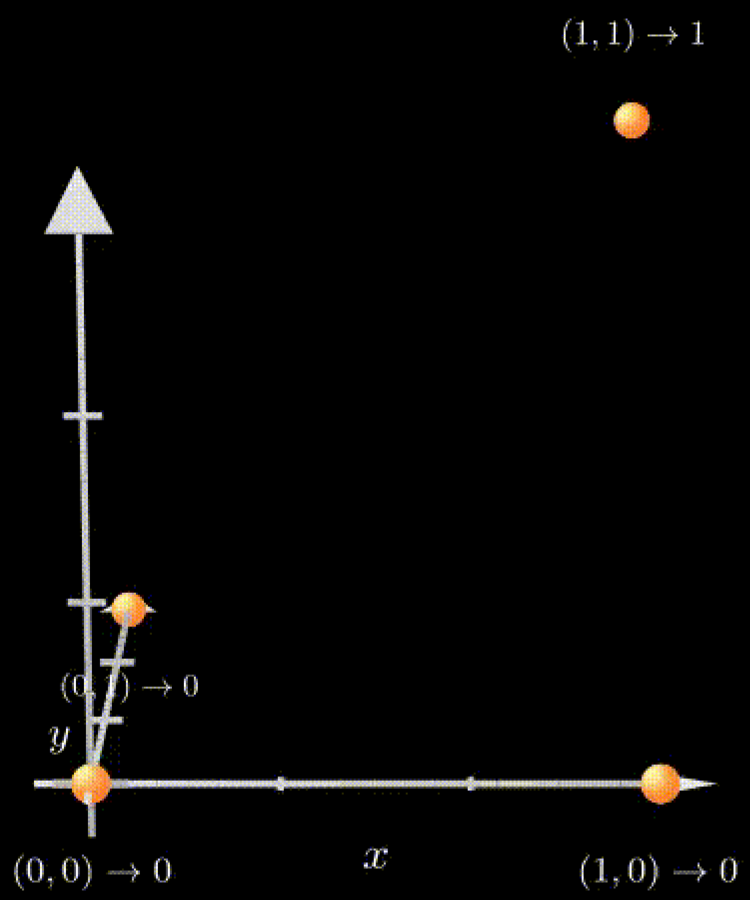
Triangular norms provide systematic relaxations of logic

Some are continuous and sub-differentiable

Inputs, outputs
live in {0,1}

	Boolean logic
Not	$\neg A$
And	$A \wedge B$
Or	$A \vee B$
Implies	$A \rightarrow B$

Klement, Erich Peter, Radko Mesiar, and Endre Pap. *Triangular norms*. Vol. 8. 2013.


$(1, 1) \to 1$
$(0, 1) \to 0$
y
$(0, 0) \to 0$
x
$(1, 0) \to 0$

Relaxing Boolean operators

Triangular norms provide systematic relaxations of logic

Some are continuous and sub-differentiable

	Boolean logic (Inputs, outputs live in {0,1})	Product (Inputs, outputs live in [0,1])	Gödel	Łukasiewicz
Not	$\neg A$	$1-a$	$1-a$	$1-a$
And	$A \wedge B$	ab	$\min(a,b)$	$\max(0,a+b-1)$
Or	$A \vee B$	$a+b-ab$	$\max(a,b)$	$\min(1,a+b)$
Implies	$A \to B$	$\min\left(1,\frac{b}{a}\right)$	$\begin{cases} 1 & \text{if } b > a \\ b & \text{else} \end{cases}$	$\min(1,1-a+b)$

Klement, Erich Peter, Radko Mesiar, and Endre Pap. *Triangular norms*. Vol. 8. 2013.

Three challenges facing logic in neural network land

1. Bridging predicates in rules with neural networks?

   Answer: *Named neurons*, nodes in a computation graph that have *externally* defined meaning

2. Making logic differentiable?

   Answer: Use a *t-norm relaxation*

3. Using differentiable logic?

What logic can do for neural networks?

Introduce inductive bias by…

- …changing network architecture

  to networks that prefer satisfying the constraints

- …by regularizing learning

  to penalize models that violate the constraints

What relaxed logic can do for neural networks?

Introduce inductive bias by…

- …changing network architecture

  to networks that prefer satisfying the constraints

- …by regularizing learning

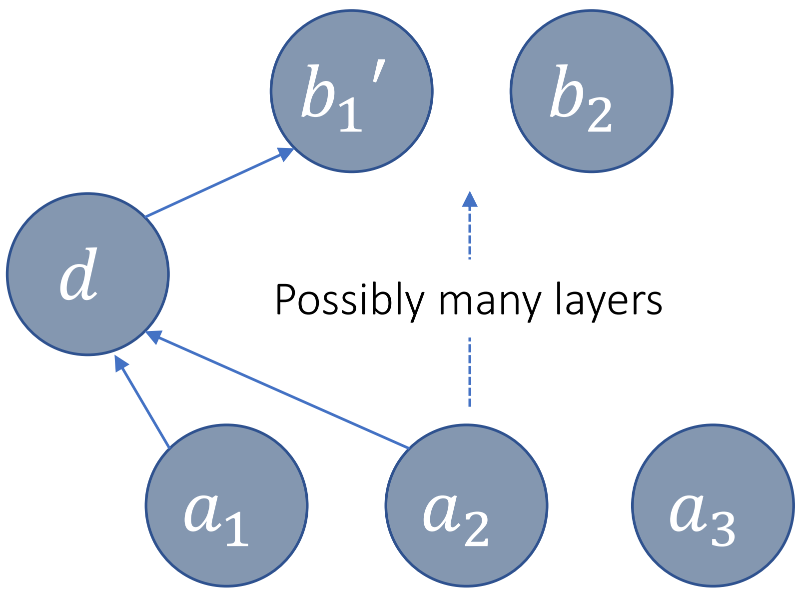
  to penalize models that violate the constraints

Augmenting Neural Networks with First-order Logic. *Li and Srikumar*. ACL 2019.

Augmenting models: An example

$b_1{}' = \sigma(\mathbf{w}^T\mathbf{x} + \rho\, d(a_1, a_2))$

$\cancel{b_1 = \sigma(\mathbf{w}^T\mathbf{x})}$



$$A_1 \wedge A_2 \rightarrow B_1$$

Step 1: LHS in Łukasiewicz logic

$$d(a_1, a_2) = \max(0, a_1 + a_2 - 1)$$

Step 2: Define constrained node $b_1{}'$

Step 3: Replace original b_1 with $b_1{}'$

No additional trainable parameters introduced

Hyperparameter ρ controls how strongly the constraint is enforced

What logic can do for neural networks?

Introduce inductive bias by…

- …changing network architecture

  to networks that prefer satisfying the constraints

- …by regularizing learning

  to penalize models that violate the constraints

A Logic-Driven Framework for Consistency of Neural Models. *Li, Gupta, Mehta and Srikumar*. EMNLP, 2019.
Structured Tuning for Semantic Role Labeling. *Li, Jawale, Palmer and Srikumar*. ACL, 2020.
Evaluating Relaxations of Logic for Neural Networks: A Comprehensive Study. *Medina-Grespan, Gupta, Srikumar*. IJCAI 2021.
Logic-driven Indirect Supervision: An Application to Crisis Counseling. Medina-Grespan et al, ACL 2023

Unifying data & knowledge

Labeled data = propositions about examples

Knowledge can be written as rules

- e.g. $\forall\ P, H, Z,\quad \text{Entail}(P, H) \wedge \text{Entail}(H, Z) \rightarrow \text{Entail}(P, Z)$
- Universally quantified

Labeled examples and constraints are, together, a collection of rules of the form

$$\forall x, L(x) \rightarrow R(x)$$

Encouraging consistency of models

$$\forall x, L(x) \rightarrow R(x)$$

Labeled data + knowledge

Learning goal: Prefer models that set all the rules of this form to be true

Or alternatively: Find models maximize a t-norm relaxation

Inconsistency losses

Use any neural model, any library and any optimizer
Product t-norm + labeled examples gives cross entropy loss

Worked example

Multiclass classification + product t-norm = cross-entropy loss

Multiclass classification

The setting: Suppose we have a dataset consisting of n examples $x_1, x_2, \cdots, x_n$ (e.g. sentences, documents, text, etc) whose labels are $Y_1, Y_2, \cdots, Y_n$

We seek to train a model that learns how to label new examples.

We can write the data as

$$Y_1(x_1) \wedge Y_2(x_2) \wedge \cdots \wedge Y_n(x_n)$$

Or equivalently

$$\bigwedge_{i=1}^{n} \top \rightarrow Y_i(x_i)$$

How to relax logical formulas for learning models

1. Pick a t-norm

2. Replace predicates with model probabilities (unless they can be deterministically computed)

3. Recursively replace the Boolean operations

4. (For numerical reasons: Take the logarithm of the final expression)

Relaxing using the product t-norm

$$\bigwedge_{i=1}^{n} \top \rightarrow Y_i(x_i)$$

Not syntactically valid yet

$$\bigwedge_{i=1}^{n} 1 \rightarrow P(Y_i \mid x_i)$$

Not syntactically valid yet

$$\bigwedge_{i=1}^{n} \min\left(1, \frac{P(Y_i \mid x_i)}{1}\right)$$

$$\bigwedge_{i=1}^{n} P(Y_i \mid x_i)$$

The final relaxation

$$\prod_{i=1}^{n} P(Y_i \mid x_i)$$

The Rules

$Y_i(x_i)$ becomes $P(Y_i \mid x_i)$

$\top$ becomes 1

$A \rightarrow B$ becomes $\min\left(1, \frac{b}{a}\right)$

$A \wedge B$ becomes ab

Back to multiclass classification

The dataset: $\bigwedge_{i=1}^{n} \top \rightarrow Y_i(x_i)$ → Its relaxation: $\prod_{i=1}^{n} P(Y_i \mid x_i)$

Our goal: to find a model that tries to make this formula hold → A more reachable goal: to make its relaxation more probable

Or equivalently, find a model that maximizes the logarithm of the relaxation

$$\sum_{i=1}^{n} \log P(Y_i \mid x_i)$$

Or equivalently, find a model that minimizes the negative logarithm of the relaxation

$$\sum_{i=1}^{n} -\log P(Y_i \mid x_i)$$

This is the popular cross-entropy loss

Case studies

Natural Language Inference

SNLI dataset, decomposable attention model [Parikh et al 2016]

Two constraints (written in logic):

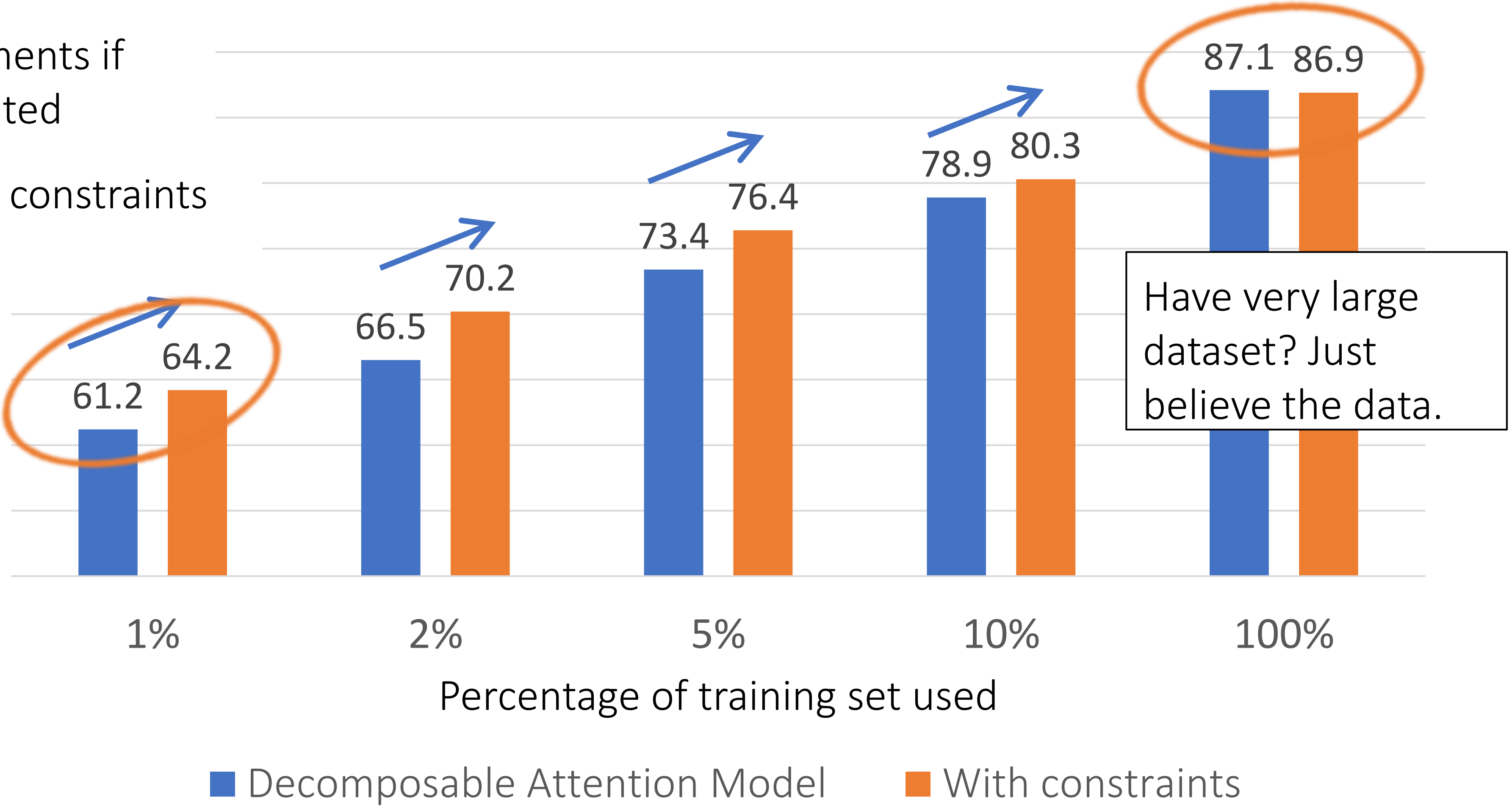
1. If two words are related, they should be aligned
2. If no content word in the hypothesis is aligned,
   then the label cannot be Entail

Results: Natural Language Inference

1. Constraints help

2. Larger improvements if training data is limited

3. With 0.5M data, constraints don't seem to help



Inconsistency of natural language inference

BERT based models for SNLI & MultiNLI datasets

Two kinds of regularizers from constraints:

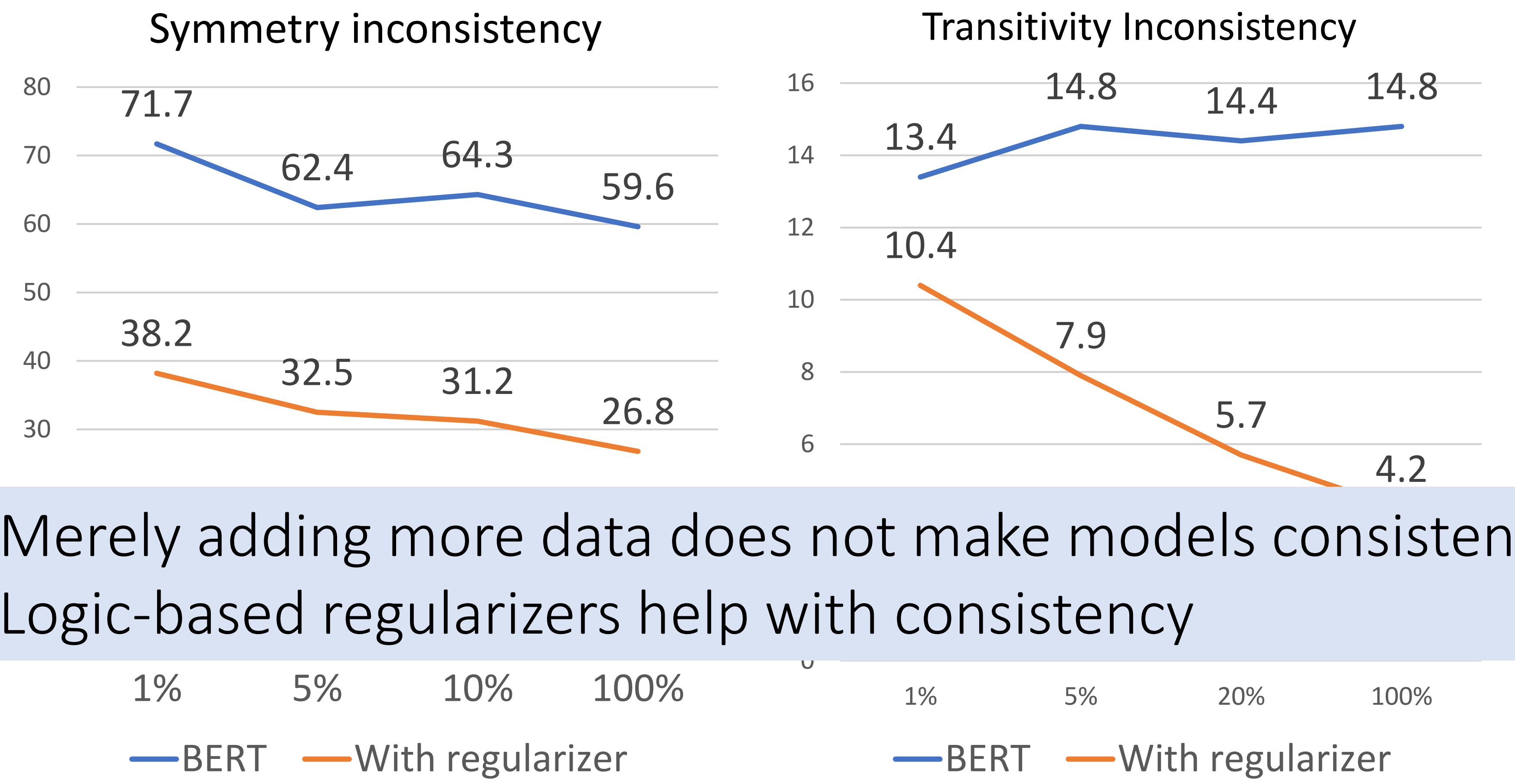
1. Symmetry constraint:
$$\forall P, H, \text{Contradict}(P, H) \leftrightarrow \text{Contradict}(H, P)$$

2. Four transitivity constraints of the form
$$\text{Entail}(P, H) \wedge \text{Entail}(H, Z) \rightarrow \text{Entail}(P, Z)$$

A Logic-Driven Framework for Consistency of Neural Models. *Li, Gupta, Mehta and Srikumar*. EMNLP, 2019.

Results: Inconsistency of natural language inference

Inconsistency represents violation of constraints. Lower is better.

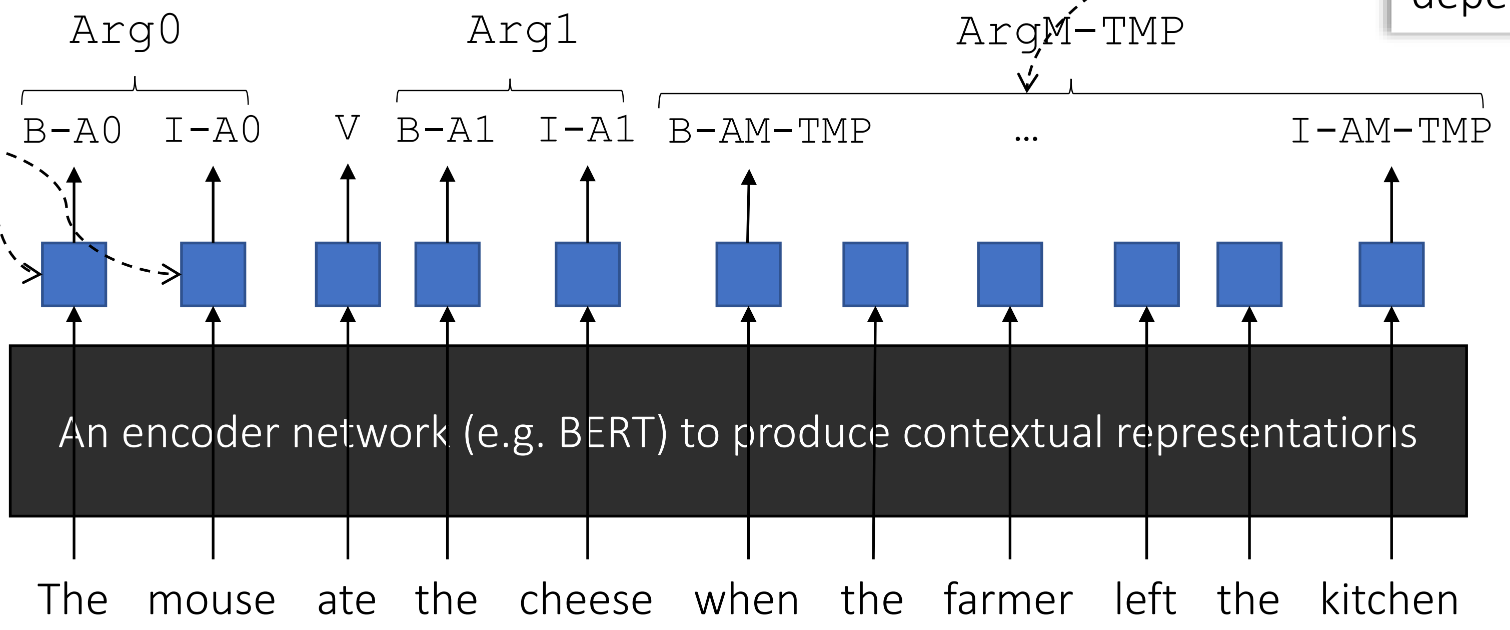


1. Merely adding more data does not make models consistent
2. Logic-based regularizers help with consistency

A common design of neural SRL models

There is usually a CRF layer across these label predictions to introduce sequential dependencies between labels

A BIO classifier over the token and verb (a small neural network)



Each predicate treated separately. Here we see the predictions for the verb *ate.*

A different commonly seen design involves span-based predictions instead of word-level ones.

A representative model: With RoBERTa embeddings, on Wall Street Journal data, we can get ~88% label F-scores

Constraints in SRL: Unique Core Roles

Each core argument can occur at most once in the output for a verb

For any verb u, and a word i

for any core argument X (i.e. one of `A0`, `A1`, `A2`, `A3`, `A4`, `A5`)

If a model labels the i^{th} word as the beginning of a label X

Then, for any other word j

e model cannot predict that it is the beginning of the same label

1. Compile into a differentiable expression using a t-norm
2. Minimize the negative of the expression as part of training

Structured Tuning for Semantic Role Labeling. *Li, Jawale, Palmer and Srikumar*. ACL, 2020.

Constraints in SRL: Unique Core Roles

$$\forall\, u, i \in s, X \in \mathcal{A}_{core},$$
$$B_X(u,i) \rightarrow \bigwedge_{\substack{j \in s \\ j \neq i}} \neg B_X(u,j)$$

This is mapped via combination of product and Gödel t-norms to

$$\sum_{u,i,X} \max\left(0, \log B_X(u,i) - \min_{\substack{j \in S \\ j \neq i}} \log\big(1 - B_X(u,j)\big)\right)$$

Model probabilities for this label

Whenever the model assigns high probabilities to invalid outputs, the loss will be high.

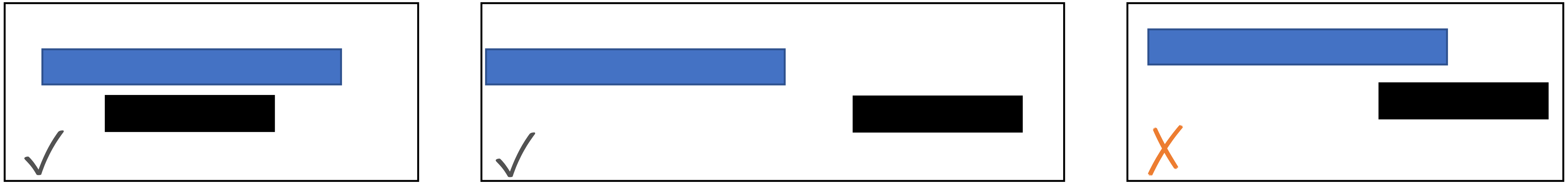
(Conversion details in the paper)

Other constraints (informally)

Both compiled to losses

The exclusively overlapping role constraint:

- In any sentence, an argument for a predicate can either be contained in, or fully outside, the argument for any predicate



The frame core role constraint

- A verb can have only those core arguments that are defined in PropBank

Scenario 1: The low data regime

- Train with 3% data with and without constraints

CoNLL 05: 1.1k examples
CoNLL 12: 2.7k examples

- Constraints greatly improve precision in the low data regime over the strong RoBERTa baseline

CoNLL 05: 70.48 → 72.6
CoNLL 12: 74.79 → 76.31

F-scores also improve (paper has details)

- Constraint violations reduced, especially for unique core roles and the frame constraints

Scenario 2: More training data

- Train with the full CoNLL 05 data

- Surprisingly still better in terms of precision, recall and f-scores, though the margin is lower
  - Strong out of domain performance on Brown corpus data

- Constraint violations reduced for unique core roles and the frame constraints
  - The unconstrained model doesn't seem to violate the exclusive overlap constraint!

CoNLL 05: 35k examples, 91k propositions

Test f-score: 87.85 → 88.03
Brown f-score: 78.64 → 79.80

Scenario 3: Even more training data

- Train with the full CoNLL 12 data

CoNLL 12: 90k examples,
253k propositions

- Constrained and unconstrained models are comparable

Test f-score: 86.47 → 86.61

If you have a lot of data, it is okay to believe the data

Knowledge via soft logic helps neural models

Successful experiments across many different tasks

- Natural language inference
- Question answering
- Text chunking
- Semantic role labeling
- Joint digit recognition and numerical operations over them
- Information extraction
- Dialogue labeling

General flavor of results

1. When we have less data, knowledge gives better statistical models
2. We can “inject” invariances into learned systems…

   …which are sometimes not learned, even with lots of data

Check out pylon: **A PyTorch Framework for Learning with Constraints** (https://pylon-lib.github.io/)