Outline

- Interpretability introduction
- AllenNLP Interpret
  - Framework overview
  - Saliency maps
  - Adversarial attacks
- Probing
  - Overview
  - Using control tasks
  - Experiments
A philosophical shift

Hand-engineered models:
- Lots of components
- Easy to intuit what each component is learning

New concept:
- One model
- Black-box: hard to tell what the model is learning
Interpretability background

- Problems of annotation artifacts, biases and adversarial attacks

What does it mean for a model to be interpretable?

- No mathematical definition of interpretability.
  - "Interpretability is the degree to which a human can understand the cause of a decision." (*)
  - "Interpretability is the degree to which a human can consistently predict the model's result" (**) 

- Interpret on the level of predictions
  - Why did the model make certain predictions?

- Interpret on the level of components
  - How does the model make predictions?

(**) Kim, Been, Rajiv Khanna, and Oluwasanmi O. Koyejo. “Examples are not enough, learn to criticize! Criticism for interpretability.” Advances in Neural Information Processing Systems (2016)
Interpretability is becoming more popular

Google trends result for ‘explainable AI’
Outline

- Interpretability introduction
- AllenNLP Interpret
  - Framework overview
  - Saliency maps
  - Adversarial attacks
- Probing
  - Overview
  - Using control tasks
  - Experiments

Open source toolkit with methods to analyze model performance and reasoning

Turning supervised tasks into tools for interpreting representations
AllenNLP Interpret: A Framework for Explaining Predictions of NLP Models

Eric Wallace¹  Jens Tuyls²  Junlin Wang²  Sanjay Subramanian¹
Matt Gardner¹  Sameer Singh²
¹ Allen Institute for Artificial Intelligence  ² University of California, Irvine
ericw@allenai.org, sameer@uci.edu
# AllenNLP

AllenNLP is an open-source NLP research library, built on PyTorch.

## Package Overview

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allennlp</td>
<td>an open-source NLP research library, built on PyTorch</td>
</tr>
<tr>
<td>allennlp.commands</td>
<td>functionality for a CLI and web service</td>
</tr>
<tr>
<td>allennlp.data</td>
<td>a data processing module for loading datasets and encoding strings as integers for representation in matrices</td>
</tr>
<tr>
<td>allennlp.models</td>
<td>a collection of state-of-the-art models</td>
</tr>
<tr>
<td>allennlp.modules</td>
<td>a collection of PyTorch modules for use with text</td>
</tr>
<tr>
<td>allennlp.nn</td>
<td>tensor utility functions, such as initializers and activation functions</td>
</tr>
<tr>
<td>allennlp.training</td>
<td>functionality for training models</td>
</tr>
</tbody>
</table>
Introduction

Why did my model make this prediction?

Most of the open-source interpretation before either were narrow focused or analyzed existing models
Introduction

*Why did my model make this prediction?*

Most of the open-source interpretation before either were narrow focused or analyzed existing models

*Allen NLP Interpret*: allows to apply existing interpretation methods to *new models*, as well as develop *new interpretation methods*
AllenNLP Interpret Toolkit

- Existing interpretation techniques implemented
- Interactive visualizations
- APIs to develop new interpretation methods
AllenNLP Interpret Toolkit

- Existing interpretation techniques implemented
- AllenNLP Interpret
- Interactive visualizations
- APIs to develop new interpretation methods
  - For example APIs to obtain input gradients
AllenNLP Interpret Toolkit

- Existing interpretation techniques implemented
- Interactive visualizations
- APIs to develop new interpretation methods
- HTML and JavaScript components that are available for clear visualization
Available Interpretations
Available Interpretations

- Saliency based
- Adversarial attacks
Saliency maps

Identifying the importance of the input tokens using gradients

Attempt to highlight regions which model was “looking at” when making decisions

Saliency maps implemented in AllenNLP Interpret

<table>
<thead>
<tr>
<th>Method</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla Gradient</td>
<td>Simonyan et al, 2014</td>
</tr>
<tr>
<td>Integrated Gradient</td>
<td>Sundararajan et al, 2017</td>
</tr>
<tr>
<td>SmoothGrad</td>
<td>Smilkov et al, 2017</td>
</tr>
</tbody>
</table>
Vanilla Gradient

Gradient of the loss with respect to each token

- Derivative of the input is found by backpropagation on the trained model
- Saliency map for each token

Vanilla Gradient

Gradient of the loss with respect to each token

Example for BERT

Simple Gradients Visualization

See saliency map interpretations generated by visualizing the gradient.

Sentence:

[CLS] This [MASK] for a gradient calculation [SEP]

Visualizing the top 3 most important words.

Integrated Gradients

An intuitive understanding:

- Construct a sequence of entries, interpolating from baseline to the actual entry
- Average the gradients

Integrated Gradients

Define a baseline $x'$, which is an input absent of information. Word importance is determined by integrating the gradient along the path from this baseline to the original input.

\[
\phi_i^{IG}(f, x, x') = \left( x_i - x'_i \right) \times \int_{\alpha=0}^{1} \frac{\delta f(x' + \alpha(x - x'))}{\delta x_i} \, d\alpha
\]

- Avoids problems with local gradients being saturated

Integrated Gradients

Define a baseline \( x' \), which is an input absent of information. Word importance is determined by integrating the gradient along the path from this baseline to the original input.

\[
\phi_i^{IG}(f, x, x') = \left( x_i - x'_i \right) \times \int_{\alpha=0}^{1} \frac{\delta f(x' + \alpha(x - x'))}{\delta x_i} d\alpha
\]

From baseline to input...

- Avoids problems with local gradients being saturated

Integrated Gradients

In practice a discrete sum approximation is used, with a scale parameter.

Integrated Gradients

Define a baseline $x'$, which is an input absent of information. Word importance is determined by integrating the gradient along the path from this baseline to the original input.

SmoothGrad

Average the gradient over many noisy versions of the input by adding Gaussian noise to embeddings and taking averages

SmoothGrad Visualization

See saliency map interpretations generated using SmoothGrad.

Sentence:

[CLS] This is an [MASK] for a gradient calculation. [SEP]

Visualizing the top 3 most important words.

SmoothGrad

Average the gradient over many noisy versions of the input by adding Gaussian noise to embeddings and taking averages

Saliency maps: comparison

Sentence:

Hand [MASK] is the act of cleaning one's hands for the purpose of removing soil, grease, microorganisms, or other unwanted substances.

Mask 1 Predictions:
- 71.5% cleaning
- 25.3% washing
- 0.6% wash
- 0.5% wiping
- 0.4% removal
Saliency maps: comparison

Sentence:
Hand [MASK] is the act of cleaning one's hands for the purpose of removing soil, grease, microorganisms, or other unwanted substances.

Vanilla gradient
[CLS] Hand [MASK] is the act of cleaning one's hands for the purpose of removing soil, grease, microorganisms, or other unwanted substances. [SEP]

Integrated gradient
[CLS] Hand [MASK] is the act of cleaning one's hands for the purpose of removing soil, grease, microorganisms, or other unwanted substances. [SEP]

SmoothGrad
[CLS] Hand [MASK] is the act of cleaning one's hands for the purpose of removing soil, grease, microorganisms, or other unwanted substances. [SEP]
Adversarial Attacks

**HotFlip**


uses the gradient to swap out words from the input in order to change the model’s prediction

**Input Reduction**


remove as many words as possible from the input without changing a model’s prediction
HotFlip Attack

Use gradients to estimate an individual change that would have the greatest effect, followed by a beam search to find an optimal manipulation strategy.

---

South Africa’s historic Soweto township marks its 100th birthday on Tuesday in a mood of optimism.

57% World

South Africa’s historic Soweto township marks its 100th birthday on Tuesday in a mood of optimism.

95% Sci/Tech

Chancellor Gordon Brown has sought to quell speculation over who should run the Labour Party and turned the attack on the opposition Conservatives.

75% World

Chancellor Gordon Brown has sought to quell speculation over who should run the Labour Party and turned the attack on the opposition Conservatives.

94% Business
HotFlip Attack

Example from AllenNLP paper:

**Original Input:** an interesting story about two lovers, I would recommend it to anyone!

**Flipped Input:** an interesting story about two lovers, I would recommend it to inadequate!

**Prediction changed to:** Negative

From online demo:

**Original Input:** [CLS] This is an [MASK] of a Hot ##F ##lip attack for the presentation. [SEP]

**Flipped Input:** [CLS] design is an [MASK] of the Hot ##F ##lip render for the kernel. [SEP]

**Prediction changed to:** extension
Input Reduction

Answer
Keanu Reeves, Laurence Fishburne, Carrie-Anne Moss, Hugo Weaving, and Joe Pantoliano

Passage Context
The Matrix is a 1999 science fiction action film written and directed by The Wachowskis, starring Keanu Reeves, Laurence Fishburne, Carrie-Anne Moss, Hugo Weaving, and Joe Pantoliano. It depicts a dystopian future in which reality as perceived by most humans is actually a simulated reality called "the Matrix", created by sentient machines to subdue the human population, while their bodies' heat and electrical activity are used as an energy source. Computer programmer "Neo" learns this truth and is drawn into a rebellion against the machines, which involves other people who have been freed from the "dream world."

Question
Who stars in The Matrix?

Original Input: Who stars in The Matrix?
Reduced Input: stars Matrix
Input reduction

Textual entailment example (decomposable attention combined with ELMo model, trained on SNLI dataset)

Premise: Two women are wandering along the shore drinking iced tea.

Original Input: Two women are sitting on a blanket near some rocks talking about politics

Reduced Input: Two women are sitting on a blanket near some rocks talking about politics

Result: contradiction for both original and reduced inputs
Adversarial attacks and interpretability

Adversarial attacks can help to diagnose model vulnerabilities

Training using adversarial examples could provide more interpretable saliency maps *

Alternatively, can use interpretability to detect adversarial attacks **

---

Quick demo
Available Models in AllenNLP Interpret

- **Reading Comprehension**
  - NAQANet, BiDAF
- **Masked Language Modeling**
  - BERT, RoBERTa and more
- **Text Classification and Textual Entailment**
  - BiLSTM and self-attention classifiers
- **Named Entity Recognition (NER) and Coreference Resolution**
System Overview
Conclusions - what is AllenNLP Interpret

Open-source flexible toolkit that facilitates model analysis

- Convenient framework that can run on custom models
- Existing methods as well as ability to add custom analysis methods
- Ready-to-use visualization toolkit
- Interesting demo online to play with in your free time
Conclusions: saliency maps and adversarial attacks

Saliency maps allow to assess the importance of input tokens for prediction using gradients

- Convenient tool to gain insights into the model
- Different methods might produce different results

Adversarial attacks demonstrate network misbehaviors and can be used to produce more interpretable results (or alternatively interpretable models should be less susceptible to adversarial attacks)
Probing
Probing is a hot new interpretability technique developed to understand large models.
A philosophical shift

Hand-engineered models:
- Lots of components
- Easy to intuit what each component is learning

Unsupervised representations
- One model, one component
- Black-box: hard to tell what the model is learning
Hypothesis: Deep learners encode linguistic properties in their intermediate representations.

How to test this hypothesis?
Formal definitions

Sentence 1: The cat ran quickly.

Part-of-speech: DT NN VBD RB

a sentence: \( x_{1:T} = \{x_1, x_2, \ldots, x_T\} \)

intermediate representations: \( h_{1:T} \)

output labels: \( y_{1:T} \)

a task: \( f(x_{1:T}) = y_{1:T} \)

a probe: \( f_\theta(h_{1:T}) = \hat{y}_{1:T} \)
Probing

\[ f_\theta(h_{1:T}) = \hat{y}_{1:T} \]

Probes are **supervised** models.

A probe consumes the representation of another model to perform a task.

Argument: Good performance by a probe on some task → the upstream model encodes linguistic information about that task.

Say we study BERT embeddings. If a probe consumes BERT embeddings and performs well on POS tagging, we say BERT implicitly encodes parts of speech.
Probing

- Term coined by Guillaume Alain & Yoshua Bengio
- "Understanding intermediate layers using linear classifier probes." ICLR 2017
- Used linear classifiers on ResNets
- "We suggest that the reader think of those probes as thermometers used to measure the temperature simultaneously at many different locations."
Why this analogy makes sense

- Probes have access to all intermediate layers
- Probes map the representations to a continuous space
- Using probes, we get a reading for one value.
Why do we care about probing?

A: It's a simple method for peering into the black box.
Probing for contextualized word embeddings

- Answer: mostly syntactic information. Contextualized embeddings provide gains in probe performance on syntactical tasks. (POS tagging)
- Probing for syntax.
- Uses MLP probes.
<table>
<thead>
<tr>
<th>Task</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS</td>
<td>The important thing about Disney is that it is a global [brand]₁. → NN (Noun)</td>
</tr>
<tr>
<td>Constit.</td>
<td>The important thing about Disney is that it [is a global brand]₁. → VP (Verb Phrase)</td>
</tr>
<tr>
<td>Depend.</td>
<td>[Atmosphere]₁ is always [fun]₂ → nsubj (nominal subject)</td>
</tr>
<tr>
<td>Entities</td>
<td>The important thing about [Disney]₁ is that it is a global brand. → Organization</td>
</tr>
<tr>
<td>SRL</td>
<td>[The important thing about Disney]₂ [is]₁ that it is a global brand. → Arg₁ (Agent)</td>
</tr>
<tr>
<td>SPR</td>
<td>[It]₁ [endorsed]₂ the White House strategy... → {awareness, existed after, ... }</td>
</tr>
<tr>
<td>Coref.₀</td>
<td>The important thing about [Disney]₁ is that [it]₂ is a global brand. → True</td>
</tr>
<tr>
<td>Coref.₇</td>
<td>[Characters]₂ entertain audiences because [they]₁ want people to be happy. → True</td>
</tr>
<tr>
<td>Characters entertain [audiences]₂ because [they]₁ want people to be happy. → False</td>
<td></td>
</tr>
<tr>
<td>Rel.</td>
<td>The [burst]₁ has been caused by water hammer [pressure]₂. → Cause-Effect(e₂, e₁)</td>
</tr>
</tbody>
</table>

Table 1: Example sentence, spans, and target label for each task. O = OntoNotes, W = Winograd.
Probing for sentence embeddings

- Ray Mooney, ACL 2014, opening talk: "You can't cram the meaning of a whole f**king sentence into a single f**king vector!"


- Probe sentence encoders for surface-level, syntactic, and semantic information.

- Result: BiLSTM encoder beats simple bag-of-vectors baseline.

- Uses MLP probes.
Issues with probing: attribution

Do we attribute good probe performance to:
1. The upstream model's representation?
2. Training?

Figure credit: John Hewitt
Issues with probing: capacity

Representations are lossless.

If your probe is too complex, all you're doing is feeding embeddings to a model!

Ex: BERT would be a terrible probe.
Issues with probing

To say anything conclusive about probing, we need metrics.

We need to distinguish between:
1. Probes that work because of emergent linguistic representations.
2. Probes that treat deep representations as word embeddings.

→ control tasks

We need to quantify how reflective a probe is of its input representation.

→ selectivity
Designing and Interpreting Probes with Control Tasks

John Hewitt and Percy Liang
Control tasks

1. Generate a label for each word in a vocabulary independently at random (randomness).
2. Assign this label to that word for the rest of the experiment (structure).

<table>
<thead>
<tr>
<th>Control Task Vocab</th>
<th>after</th>
<th>!</th>
<th>ran</th>
<th>The</th>
<th>cat</th>
<th>quickly</th>
<th>dog</th>
</tr>
</thead>
<tbody>
<tr>
<td>! after</td>
<td>42</td>
<td>3</td>
<td>10</td>
<td>15</td>
<td>10</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Control tasks are defined per task.
Control tasks

A probe dependent on linguistic representation should perform badly on control tasks.

Why: control task labels do not correspond with real linguistic knowledge.

So: a probe can only perform well if it uses word embeddings from representation to memorize control task labels.

Control tasks trap probes that are too smart.
Selectivity: a balancing act

We want a probe that:

1. Has enough capacity to draw out info from representation
2. Does not have enough capacity to memorize the task

$selectivity = linguistic\ accuracy\ (1) - control\ accuracy\ (2)$
Selectivity: how reflective a probe is of its input representation.
Experiments
Setting: Tasks

1. Part of Speech tagging
   ○ 45 possible tags (NN, VB, etc.)
   ○ Assign 45 randomized control tags.

2. Dependency edge prediction
   ○ A lot of possibilities.
   ○ 3 control tags: attach to self (i), attach to first (1), or attach to last (T).

---

**Dependency Edge Prediction and Control Task Examples**

**Dependency:**

The Ways and Means Committee will hold a hearing on the bill next Tuesday.

**Control:**

The Ways and Means Committee will hold a hearing on the bill next Tuesday.
Setting: Probe families

\[ f_\theta(h_{1:T}) = \hat{y}_{1:T} \]

1. POS tagging: 3 probes.
   - linear probe: \( y_i \sim \text{softmax}(A h_i) \)
   - 2-layer MLP (MLP-1): \( y_i \sim \text{softmax}(W_2 g(W_1 h_i)) \)
   - 3-layer MLP (MLP-2): \( y_i \sim \text{softmax}(W_3 g(W_2 g(W_1 h_i))) \)

2. Dependency edge prediction: 3 probes, replace linear with bilinear.
   - bilinear probe: \( y_i \sim \text{softmax}(h_{1:T}^T A h_i) \)
Setting: Complexity Control (= Regularization)

Probes can't be too complex.

\[ y_i \sim \text{softmax}(A h_i) \]

1. Rank/hidden dimensionality constraint.
   - Factorize \( A = LR \), force \( L \) to have dimension \( l \)
   - Force MLPs to have hidden state size \( l \)
2. Dropout (temporarily zero out nodes).
3. Constraining the number of training examples.
4. L2 regularization (weight decay).
5. Early stopping.
Dataset

Penn Treebank: dataset of sentences from Wall Street Journal

- Sentences labeled with parts of speech and dependency trees.
Results for different probe families, under various hyperparameter settings.

<table>
<thead>
<tr>
<th>Probe</th>
<th>PoS</th>
<th>Ctl</th>
<th>Select.</th>
<th>Dep</th>
<th>Ctl</th>
<th>Select.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Probes with Default Hyperparameters</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear</td>
<td>97.2</td>
<td>71.2</td>
<td>26.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Bilinear</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>89.0</td>
<td>82.4</td>
<td>6.6</td>
</tr>
<tr>
<td>MLP-1</td>
<td>97.3</td>
<td>92.8</td>
<td>4.5</td>
<td>92.3</td>
<td>93.0</td>
<td>-0.7</td>
</tr>
<tr>
<td>MLP-2</td>
<td>97.3</td>
<td>93.2</td>
<td>4.2</td>
<td>93.9</td>
<td>92.0</td>
<td>1.9</td>
</tr>
<tr>
<td><strong>Probes with 0.4 Dropout</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear</td>
<td>97.1</td>
<td>67.3</td>
<td>29.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Bilinear</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>90.4</td>
<td>73.7</td>
<td>16.7</td>
</tr>
<tr>
<td>MLP-1</td>
<td>97.5</td>
<td>93.4</td>
<td>4.1</td>
<td>93.8</td>
<td>93.1</td>
<td>0.7</td>
</tr>
<tr>
<td>MLP-2</td>
<td>97.4</td>
<td>94.1</td>
<td>3.4</td>
<td>94.7</td>
<td>93.5</td>
<td>1.3</td>
</tr>
<tr>
<td><strong>Probes Designed with Control Tasks</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear</td>
<td>97.0</td>
<td>64.0</td>
<td>33.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Bilinear</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>91.0</td>
<td>83.1</td>
<td>7.9</td>
</tr>
<tr>
<td>MLP-1</td>
<td>97.2</td>
<td>80.6</td>
<td>16.6</td>
<td>90.5</td>
<td>84.3</td>
<td>6.2</td>
</tr>
<tr>
<td>MLP-2</td>
<td>97.2</td>
<td>81.7</td>
<td>15.4</td>
<td>92.8</td>
<td>89.8</td>
<td>3.0</td>
</tr>
</tbody>
</table>

Probes with hyperparameters tuned for selectivity.
<table>
<thead>
<tr>
<th></th>
<th>Parts-of-speech</th>
<th></th>
<th>Dependencies</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>97.2</td>
<td>25.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Bilinear</td>
<td>-</td>
<td>-</td>
<td>89.4</td>
<td>16.6</td>
</tr>
<tr>
<td>MLP-1-layer</td>
<td>97.3</td>
<td>4.65</td>
<td>92.5</td>
<td>3.35</td>
</tr>
<tr>
<td>MLP-2-2-layer</td>
<td>97.3</td>
<td>5.25</td>
<td>93.4</td>
<td>4.17</td>
</tr>
</tbody>
</table>

Figure credit: John Hewitt
Effectiveness of various regularization strategies. What we want: big increase in accuracy, small decrease in selectivity.

- Dropout doesn't really help.
- Constraining rank seems to help
How hard is it to find selective probes?

Results:

- Dropout and early stopping don't help selectivity
- Constraining hidden state dimensionality is effective!
  - Used MLP hidden state size of 10 for POS and 50 for dependency head prediction

Author conclusions:

- Current probes are needlessly overparameterized! They have too much capacity.
- The most selective probes are linear or bilinear models.
- MLPs have the best accuracy on dependency edge prediction.
  - → some syntactical info can't be extracted by a bilinear probe.
POS error analysis

- Linear models tend to classify adjective-noun pairs as noun-noun pairs:
  
  Kan.-based/JJ National/NNP Pizza/NNP
  rental/JJ equipment/NN

- MLPs tend to pluralize singular nouns:

  Environmental/NNP Systems/NNP Co./NNP
  Cara/NNP Operations/NNP Co./NNP
  7.8/CD %/NN stake/NN in/IN Dataprod/NPS/NNP

- Hypothesis: MLPs have enough capacity to get confused by the 's'.
Selectivity and layer differences

Claim: the first layer of ELMo (ELMo1) is better for POS tagging than ELMo2.

Hewitt & Liang: not so fast! ELMo1 is closer to a straight word representation.

Notes for the next slide:

- Recall that ELMo runs a character CNN over the words before feeding into biLSTMs.
- As a baseline, Hewitt & Liang run an untrained biLSTM and call this representation Proj0.
Probe performance on different layers of ELMo

**Part-of-speech Tagging**

| Model | Linear |  | MLP-1 |  |
|-------|--------|--------|--------|
| Proj0 | Accuracy | 96.3 | Selectivity | 20.6 | Accuracy | 97.1 | Selectivity | 1.6 |
| ELMo1 | Accuracy | 97.2 | Selectivity | 26.0 | Accuracy | 97.3 | Selectivity | 4.5 |
| ELMo2 | Accuracy | 96.6 | Selectivity | 31.4 | Accuracy | 97.0 | Selectivity | 8.8 |

**Implication:** Because ELMo1 is closer to a word representation, probes on ELMo1 are leveraging the word identity and not the encoded linguistic knowledge.

**Dependency Edge Prediction**

| Model | Bilinear |  | MLP-1 |  |
|-------|----------|--------|--------|
| Proj0 | Accuracy | 79.9 | Selectivity | -4.3 | Accuracy | 86.5 | Selectivity | -9.0 |
| ELMo1 | Accuracy | 89.7 | Selectivity | 6.7 | Accuracy | 92.5 | Selectivity | -1.0 |
| ELMo2 | Accuracy | 84.5 | Selectivity | 6.2 | Accuracy | 89.5 | Selectivity | 1.4 |

**POS tagging:** We see an increase in selectivity, for a comparably smaller decrease in accuracy.
<table>
<thead>
<tr>
<th>Method</th>
<th>Acc.</th>
<th>Select.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projo</td>
<td>96.3</td>
<td>20.6</td>
</tr>
<tr>
<td>ELMo1</td>
<td><strong>97.2</strong></td>
<td>26.0</td>
</tr>
<tr>
<td>ELMo2</td>
<td>96.6</td>
<td><strong>31.4</strong></td>
</tr>
<tr>
<td>ELMo1-ELMo2</td>
<td>+0.6</td>
<td>-5.4</td>
</tr>
<tr>
<td>ELMo2-Projo</td>
<td>+0.3</td>
<td>+10.8</td>
</tr>
</tbody>
</table>

Figure credit: John Hewitt
"Without considering selectivity, [we might think] that ELMo2 encodes nothing about part-of-speech, since it doesn't beat the Proj0 baseline.

"Taking selectivity into account, we see that probes on ELMo2 are unable to rely on word identity features like those on Proj0. To achieve high accuracy, they must rely on emergent properties of the representation."

-Hewitt & Liang
Summary

- Use **control tasks** to identify models using representations as word embeddings.
- Probes should be **selective**. They should perform poorly on the control task.
- Linear and bilinear probes are the most selective.
- Many probes nowadays are too powerful.
Q1: In Hewitt and Liang et al 2019, why do they claim that linear and bilinear classifiers work better as probes than multi-layer perceptrons?

Linear and bilinear classifiers work better because they are more selective. Selective probes better reflect linguistic properties of the representation. MLPs tend to be too overparameterized, allowing them to memorize control-task mappings.
Where probing is going

- Designing and Interpreting Probes with Control Tasks:
  - appeared on arXiv September 2019
  - published November 2019, EMNLP best paper runner-up

- Probing is a nascent technique: there's no consensus on best practice.

- Thoughts? Is probing a good technique?
**References**


