Some thoughts on
Prior Knowledge, Deep Architectures and
NLP

Jason Weston, Ronan Collobert, Pavel Kuksa
NEC Labs America, Princeton, USA
Prior Knowledge and Learning

successful learning = data + prior knowledge

If you have limited data, you need more prior knowledge

If you have more data, you need less prior knowledge

Asymptotic case: you only need labeled data, no other knowledge
Realistic case: extra knowledge usually helps

(Note: you can think of labeling data as a kind of knowledge)
Ways of Encoding Prior Knowledge

\[ x = \phi(\text{input}) \quad \text{human-chosen features for task} \]

\[ \text{choose } F \quad \text{choose family of models e.g. kernel / NN architecture} \]

\[ \text{choose } f \in F \quad \text{choose objective, e.g. loss + regularizer} \]

\[ y = \text{argmax}_{\hat{y}} f(x, \hat{y}) \quad \text{choice of decoding algorithm (constraints?)} \]

\[ \text{output} \quad \text{choose labeling task, potentially use in a cascade} \]

+ Multi-tasking: share architecture over tasks

Differences in ways: robust/brittle, cheap/expensive knowledge, train/test speed
Differences in Ways of Encoding Prior Knowledge

- **Robust/Brittle** -
  Brittle: system must use chosen representation / coding
  Robust: priors injected into system, but it can ignore them / adapt

  - e.g. bag-of-words: if you encode this prior and it’s **wrong** will your classifier suffer a lot?

- **Cheap/Expensive knowledge** - Expert linguist or ML researcher needs to work for months / years to encode prior?

  - e.g. labeling more data, defining complicated rules or features

- **Train/Test speed** - How will it effect speed?

  - e.g. add features derived from a parse tree → have to compute parse tree
Shallow System Cascades Vs Deep Architectures

NNs good for learning features rather than humans working them out
Easier to define priors in NNs via multi-tasking & choice of architecture

very high level representation:

\[
\begin{array}{c}
\text{MAN} \\
\text{SITTING}
\end{array}
\]

... etc ...

slightly higher level representation

raw input vector representation:

\[ \boldsymbol{v} = [23, 19, 20, 18] \]

POISON HELP! 1-800-222-V222
Ways of Encoding Prior Knowledge

\[
\begin{align*}
\text{input} \quad & \quad \downarrow \\
x &= \phi(\text{input}) \\
& \quad \downarrow \\
\text{choose } \mathcal{F} \quad & \quad \downarrow \\
& \quad \text{choose family of models} \quad \text{e.g. kernel / NN architecture} \\
& \quad + \\
\text{choose } f \in \mathcal{F} \quad & \quad \downarrow \\
& \quad \text{choose objective, e.g. loss + regularizer} \\
& \quad + \\
y &= \text{arg}\max_{\hat{y}} f(x, \hat{y}) \\
& \quad \downarrow \\
\text{output} \quad & \quad \downarrow \\
& \quad \text{choose labeling task, potentially use in a cascade} \\
& \quad + \\
\text{Multi-tasking: share architecture over tasks}
\end{align*}
\]
Images: use prior about invariances (rotations/translations)

Text: hand-built generative model that creates labeled data? [Allows the model to absorb a lot of expert knowledge which is not required at test time.]
Labeling more data!

An easy way for humans to encode their knowledge.

Parsing error rates have reduced a “small amount” in the last 10 years.

*If parsing researchers were labeling data instead of improving algorithms, would we have better error rates?*
Ways of Encoding Prior Knowledge

- Input $x = \phi(\text{input})$
- Choose family of models $\mathcal{F}$, e.g., kernel/NN architecture.
- Choose $f \in \mathcal{F}$.
- Choose objective, e.g., loss + regularizer.
- Choose decoding algorithm, e.g., $y = \arg\max_{\hat{y}} f(x, \hat{y})$.
- Choose labeling task, potentially use in a cascade.

- Multi-tasking: share architecture over tasks.
SRL: example of features in an NLP classifier

Assume segments to label are nodes of predicted parse tree.

Extract hand-made features e.g. from the parse tree

Feed these features to a shallow classifier like SVM
**SRL: many hand built features** e.g. [Pradhan et al.]

**Issues:**
1) Expensive knowledge, computation time of features  
2) Features rely on other solutions (parsing, named entity)  
2) Technology task-transfer is difficult

<table>
<thead>
<tr>
<th>Predicate and POS tag of predicate</th>
<th>Voice: active or passive (hand-built rules)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phrase type: adverbal phrase, prepositional phrase, ...</td>
<td>Governing category: Parent node's phrase type(s)</td>
</tr>
<tr>
<td>Head word and POS tag of the head word</td>
<td>Position: left or right of verb</td>
</tr>
<tr>
<td>Path: traversal from predicate to constituent</td>
<td>Predicted named entity class</td>
</tr>
<tr>
<td>Word-sense disambiguation of the verb</td>
<td>Verb clustering</td>
</tr>
<tr>
<td>Length of the target constituent (number of words)</td>
<td>NEG feature: whether the verb chunk has a &quot;not&quot;</td>
</tr>
<tr>
<td>Partial Path: lowest common ancestor in path</td>
<td>Head word replacement in prepositional phrases</td>
</tr>
<tr>
<td>First and last words and POS in constituents</td>
<td>Ordinal position from predicate + constituent type</td>
</tr>
<tr>
<td>Constituent tree distance</td>
<td>Temporal cue words (hand-built rules)</td>
</tr>
<tr>
<td>Dynamic class context: previous node labels</td>
<td>Constituent relative features: phrase type</td>
</tr>
<tr>
<td>Constituent relative features: head word</td>
<td>Constituent relative features: head word POS</td>
</tr>
<tr>
<td>Constituent relative features: siblings</td>
<td>Number of pirates existing in the world...</td>
</tr>
</tbody>
</table>
SRL: Cascade of features
Ways of Encoding Prior Knowledge

input
→
\[ x = \phi(\text{input}) \]
→
choose \( \mathcal{F} \)
→
choose \( f \in \mathcal{F} \)
→
\[ y = \arg\max_{\hat{y}} f(x, \hat{y}) \]
→
output

- choice of training examples (e.g. add invariances?)
- human-chosen features for task
- choose family of models e.g. kernel / NN architecture
- choose objective, e.g. loss + regularizer
- choice of decoding algorithm (constraints?)
- choose labeling task, potentially use in a cascade

+ Multi-tasking: share architecture over tasks
The company operates stores mostly in Iowa and Nebraska.

→ Learn a linear classifier $f(x) = w \cdot x + b$

Simple, fast.
Problem: no account of word order!

Q: Is the meaning lost?  
A: On easy problems ok, on harder problems - yes! (e.g. in information extraction)
sliding windows: somewhat brittle, cheap, fast

Classic Window approach makes assumption distances $> m$ are useless

Could regularize instead: larger window, further words have less weight.
cons: slow, extra parameters.
Convolutions: quite robust, cheap, fast

Extract local features – share weights through time/space

Used with success in image (Le Cun, 1989) and speech (Bottou & Haffner, 1989)
Convolutional NN: architecture vs. invariances [Image: Y. Bengio, 2007]

**MNIST error rates:**

- SVM, Gaussian Kernel: 1.40
- 2-layer NN, 800 HU, Cross-Entropy Loss: 1.60
- Convolutional net LeNet-5, [no distortions]: 0.95
- Virtual SVM, deg-9 poly, 2-pixel jittered: 0.56
- 2-layer NN, 800 HU, cross-entropy [elastic distortions]: 0.70
- Convolutional net, cross-entropy [elastic distortions]: 0.40
End-to-end learning. Input/output to each layer chosen by network.
Shallow System Cascades Vs Deep Architectures

Cascade of “layers” trained separately. (Disjoint = convex ?)

Input fixed. Output fixed. → suboptimal deep network?
The “Brain Way”

We propose a radically different, machine learning, approach:

- **Avoid** building a parse tree. Humans don’t need this to talk.
- **We try to avoid** all hand-built features → monolithic systems.
- Humans **implicitly** learn these features. Neural networks can too.

End-to-end system

+ Fast predictions (∼0.01 secs per sentence)
The Big Picture (1/2)

- Blah Blah Blah
  - Embedding
  - Local features
  - Global features
  - Tags

- Deep architecture
- Unification of NLP tasks
The Deep Learning Way (1/2)

Input Sentence

- Text: the cat sat on the
- Indices: s(1) s(2) s(3) s(4) s(5)

Lookup Table

- Embedding
- Global Features
- Linear
- HardTanh
- Softmax

Tags

Word of interest
The Deep Learning Way (2/3)

Input Text

Input Sentence

*word of interest*

**the cat sat on the mat**

*indices*

s(1) s(2) s(3) s(4) s(5) s(6)

*pos w.r.t. word*

-1  0  1  2  3  4

Lookup Tables

$LT_w$

$LT_{pw}$

TDNN Layer

Max Over Time

Local features

HardTanh

Linear

Global features

HardTanh

Linear

Softmax

Tags
The Deep Learning Way

(2/3)

Input Sentence

the cat sat on the mat

word of interest

verb of interest

s(1) s(2) s(3) s(4) s(5) s(6)

-1 0 1 2 3 4

Lookup Tables

$LT_w$

$LT_{pw}$

$LT_{pv}$

TDNN Layer

Max Over Time

HardTanh

Linear

Softmax

Tags

Local features

Global features

Embedding

Blah Blah Blah

Text

indices

pos w.r.t. word

-2 -1 0 1 2 3

pos w.r.t. verb

-1 0 1 2 3 4

24
The Lookup Tables

the cat eats the fish

hash table

feature vector size

feature vectors for each word!

trained by backpropagation

Y. Bengio & R. Ducharme, A Neural Probabilistic Language Model, NIPS 2001
The Lookup Tables (again)

the cat eats the fish

indices in a dictionary

binary vectors

fed to some linear model
with weights shared through time

feature vectors for each word!
yesterday, after Microsoft bought Google, the dollar went down under half a euro and the fish market exploded.
yesterday, after Microsoft bought Google, the dollar went down under half a euro and the fish market exploded.
Speed comparisons: Kernel Machines can be slow...

SVM

Linear SVM training is fast. Bag-of-words is fast.

...but nonlinear SVMs are slow to compute:

\[ f(x) = \sum \alpha_i K(x_i, x) \]

Can encode prior in the kernel. But often too slow.

E.g. String kernels are slow: have to apply to every support vector.

NNs

Fast: Stochastic gradient descent. Online.

One update roughly twice the cost of computing \( f(x) = \) fast
Ways of Encoding Prior Knowledge

\[
\text{input} \quad \downarrow \\
x = \phi(\text{input}) \quad \text{choice of training examples (e.g. add invariances?)}
\]

\[
\downarrow \\
\text{choose } \mathcal{F} \quad \text{human-chosen features for task}
\]

\[
\downarrow \\
\text{choose } f \in \mathcal{F} \quad \text{choose family of models e.g. kernel / NN architecture}
\]

\[
\downarrow \\
y = \arg\max_{\hat{y}} f(x, \hat{y}) \quad \text{choose objective, e.g. loss + regularizer}
\]

\[
\downarrow \\
\text{output} \quad \text{choice of decoding algorithm (constraints?)}
\]

\[
\downarrow \\
+ \quad \text{choose labeling task, potentially use in a cascade}
\]

\[
+ \quad \text{Multi-tasking: share architecture over tasks}
\]
Ways of Encoding Prior Knowledge

input → choice of training examples (e.g. add invariances?)

\[ x = \phi(\text{input}) \]

→ human-chosen features for task

choose \( \mathcal{F} \) → choose family of models e.g kernel / NN architecture

choose \( f \in \mathcal{F} \) → choose objective, e.g. loss + regularizer

\[ y = \arg\max_{\hat{y}} f(x, \hat{y}) \]

→ choice of decoding algorithm (constraints?)

output → choose labeling task, potentially use in a cascade

Multi-tasking: share architecture over tasks
Ways of Encoding Prior Knowledge

\[
\begin{align*}
\text{input} & \quad \text{choice of training examples (e.g. add invariances?)} \\
\downarrow & \\
\phi(\text{input}) & \\
\downarrow & \\
\text{choose } F & \quad \text{human-chosen features for task}
\end{align*}
\]

\[
\begin{align*}
\downarrow & \\
\text{choose } f \in F & \quad \text{choose family of models e.g kernel / NN architecture}
\end{align*}
\]

\[
\text{choose objective, e.g. loss + regularizer}
\]

\[
\begin{align*}
\downarrow & \\
y = \arg\max_{\hat{y}} f(x, \hat{y}) & \quad \text{choice of decoding algorithm (constraints?)}
\end{align*}
\]

\[
\downarrow \\
\text{output} & \quad \text{choose labeling task, potentially use in a cascade}
\]

\[
\begin{align*}
\downarrow & \\
\text{Multi-tasking: share architecture over tasks}
\end{align*}
\]
Output labels

POS, Parsing, SRL: encodings of prior knowledge
Brittle: system must use chosen representation / coding in cascade
Ways of Encoding Prior Knowledge

input
↓
\( x = \phi(\text{input}) \)

human-chosen features for task

choose \( \mathcal{F} \)

choose family of models
e.g. kernel / NN architecture

choose \( f \in \mathcal{F} \)

choose objective, e.g. loss + regularizer

\( y = \arg \max_{\hat{y}} f(x, \hat{y}) \)

choice of decoding algorithm (constraints?)

output

choose labeling task, potentially use in a cascade

Multi-tasking: share architecture over tasks
Multi-tasking NLP Tasks

- Part-Of-Speech Tagging (POS): syntactic roles (noun, adverb...)
- Chunking: syntactic constituents (noun phrase, verb phrase...)
- Name Entity Recognition (NER): person/company/location...
- Semantic Role Labeling (SRL): semantic role

[John]_{ARG0} [ate]_{REL} [the apple]_{ARG1} [in the garden]_{ARGM-LOC}

Labeled data: Wall Street Journal (≈ 1M words)
Unlabeled data: Wikipieda (≈ 630M words)
Good overview in Caruana (1997)
Improving Word Embedding: Multi-Task with Wordnet

Rare words are not trained properly

Sentences with similar words should be tagged in the same way:

★ The cat sat on the mat
★ The feline sat on the mat

Wordnet

★ pull together linked words
★ push apart other pair of words
Semi-Supervised: Multi-task with Unsupervised Task

Language Model: “is a sentence actually english or not?”

Implicitly captures
★ syntax
★ semantics

Trained over Wikipedia (~ 631M words)

Random windows from Wikipedia → +ve examples
“The black cat sat on the mat” “Champion Federer wins again”

Random distorted windows from Wikipedia → -ve examples
“The black car sat on the mat” “Champion began wins again”
Language Model: Embedding

<table>
<thead>
<tr>
<th>france</th>
<th>jesus</th>
<th>xbox</th>
<th>reddish</th>
<th>scratched</th>
</tr>
</thead>
<tbody>
<tr>
<td>454</td>
<td>1973</td>
<td>6909</td>
<td>11724</td>
<td>29869</td>
</tr>
<tr>
<td>spain</td>
<td>christ</td>
<td>playstation</td>
<td>yellowish</td>
<td>smashed</td>
</tr>
<tr>
<td>italy</td>
<td>god</td>
<td>dreamcast</td>
<td>greenish</td>
<td>ripped</td>
</tr>
<tr>
<td>russia</td>
<td>resurrection</td>
<td>psNUMBER</td>
<td>brownish</td>
<td>brushed</td>
</tr>
<tr>
<td>poland</td>
<td>prayer</td>
<td>snes</td>
<td>bluish</td>
<td>hurled</td>
</tr>
<tr>
<td>england</td>
<td>yahweh</td>
<td>wii</td>
<td>creamy</td>
<td>grabbed</td>
</tr>
<tr>
<td>denmark</td>
<td>josephus</td>
<td>nes</td>
<td>whitish</td>
<td>tossed</td>
</tr>
<tr>
<td>germany</td>
<td>moses</td>
<td>nintendo</td>
<td>blackish</td>
<td>squeezed</td>
</tr>
<tr>
<td>portugal</td>
<td>sin</td>
<td>gamecube</td>
<td>silvery</td>
<td>blasted</td>
</tr>
<tr>
<td>sweden</td>
<td>heaven</td>
<td>psp</td>
<td>greyish</td>
<td>tangled</td>
</tr>
<tr>
<td>austria</td>
<td>salvation</td>
<td>amiga</td>
<td>paler</td>
<td>slashed</td>
</tr>
</tbody>
</table>

Dictionary size: 30,000 words. Even rare words are well embedded!
MTL: Semantic Role Labeling

Wsz=100

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Test Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.5</td>
</tr>
<tr>
<td>6</td>
<td>8.5</td>
</tr>
<tr>
<td>11</td>
<td>13.5</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
</tr>
</tbody>
</table>

SRL
SRL+POS
SRL+CHUNK
SRL+POS+CHUNK
SRL+POS+CHUNK+NER
SRL+SYNONYMS
SRL+POS+CHUNK+NER+SYNONYMS
SRL+LANG.MODEL
SRL+POS+CHUNK+NER+LANG.MODEL
MTL: Semantic Role Labeling

![Graphs showing test error vs epoch for different models with Wsz=15 and Wsz=100. Each graph compares error rates for models like SRL, SRL+POS, SRL+CHUNK, etc., across epochs.]
**MTL: Unified Brain for NLP**

Improved results with Multi-Task Learning (MTL)

<table>
<thead>
<tr>
<th>Task</th>
<th>Alone</th>
<th>MTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRL</td>
<td>18.40%</td>
<td>14.30%</td>
</tr>
<tr>
<td>POS</td>
<td>2.95%</td>
<td>2.91%</td>
</tr>
<tr>
<td>Chunking – error rate</td>
<td>4.5%</td>
<td>3.8%</td>
</tr>
<tr>
<td>Chunking – F-measure</td>
<td>91.1</td>
<td>92.7</td>
</tr>
</tbody>
</table>

**SRL:** state-of-the-art: 16.54% — Pradhan et al. (2004) Note: F1 ongoing...

**POS:** state-of-the-art ~ 3% or less

**Chunking:** Best system had 93.48% F1-score at CoNLL-2000 challenge [http://www.cnts.ua.ac.be/conll2000/chunking](http://www.cnts.ua.ac.be/conll2000/chunking). State-of-the-art is 94.1%. We get 94.9% by adding POS features.
Conclusions: NNs, NLP and prior knowledge

- **Generic end-to-end deep learning system** for NLP tasks

- **Common belief in NLP**: explicit syntactic features necessary for semantic tasks
  We showed it is **not necessarily true**

- **NNs can learn** good features and can **multi-task**

- **Other forms of prior knowledge** can be added similarly to other systems.
  But we prefer to avoid hand-made features and tags if we can... *as they might lead to brittle systems that don’t scale to harder tasks...*