Large Scale Learning Which Is Actually Useful

Ronan Collobert
ronan@collobert.com

Jason Weston, Bing Bai, Pavel Kuksa, Léon Bottou

NEC Laboratories America
The Goal

We want to have a conversation with our computer
The Goal

- We want to have a conversation with our computer
- Opinion, sentiment analysis
- Business profile
- Semantic search
- Question answering, call center
- Machine Translation

It is complex tasks and it is large-scale!
Natural Language Processing

- **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}
How Large-Scale Is It By The Way?

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 of words)
Unlabeled data: Infinite
SVMs with 1M of Labeled Examples

Linear SVMs

★ Stochastic Gradient Descent see e.g. Bottou
★ SVMPerf Joachims, 2005
★ Pegasos Shalev-Shwartz, Singer & Srebro, 2007
★ LibLinear Lin, Weng & Keerthi, 2008

It can handle it, see workshop.

Non-linear SVMs

★ LaSVM, 8M of examples. 8 days. Loosli, Canu & Bottou, 2007
Unfortunately: 150K support vectors! (a non-noisy task…)
★ Non-convex SVMs (Collobert, Weston & Bottou, 2007) can reduce drastically the number of SVs in a noisy situation
★ LaSVM+non-convex?

Even if we could do it, non-linear SVMs are slow at testing time
SVMs with $\infty$ Unlabeled Examples

- Most Transductive SVM algorithms can handle only toys

- Linear Transductive SVMs
  - SVMLin, 5M unlabeled examples, 15 minutes.
    Sindhwani, Keerthi, 2007

- Non-Linear Transductive SVMs
  - CCCP-TSVM, 60K unlabeled examples, 42 hours.
    Collobert, Weston & Bottou, 2007

- Any online Transductive SVMs?
Large Scale = Complex Models

In general, if a lot of data is available a complex system is necessary, because the task is complex...
Large Scale = Complex Models

In general, if a lot of data is available a complex system is necessary.

Not using all available examples because it does not improve generalization (see workshop?) means the system is underfitting.
Large Scale = Complex Models

In general, if a lot of data is available a complex system is necessary.

Not using all available examples because it does not improve generalization (see workshop?) means the system is underfitting.

Two extreme choices to get a complex system:

- **Large Scale Engineering**: design a lot of complex features, use a fast existing linear machine learning algorithm.

- **Large Scale Machine-Learning**: use simple features, design a complex model which will implicitly learn the right features.

Solution in the middle?
Choose some good **hand designed features**

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

Feed them to a **shallow classifier** like SVM
SRL State-of-the-art: the ASSERT system

Cascade features: e.g. extract POS, construct a parse tree

Extract hand-made features from the parse tree

Feed these features to a shallow classifier like SVM
NLP: Large Scale Learning

Blah Blah Blah

Embedding

Local features

Global features

Tags

Deep architecture

Unification of NLP tasks
Embedding

Local features

Global features

Task 1 Tags

Task 2 Tags
Neural Networks have been obsolete 20 years ago, why using them again?

- **Stochastic Gradient Descent** on SVMs = Margin Perceptron with weight decay (see Leon Bottou page)

- **Transductive** Neural Networks are really fast (Karlen, Weston, Erkan, Collobert, 2008) and they lead to as good generalization performance as TSVMs...

- Let see what they can do on 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
- $LT_w$

Embedding

Global features
- HardTanh
- Linear
- HardTanh
- Linear
- Softmax

Tags
The Deep Learning Way (2/2)

Input Sentence

- Input text: the cat sat on the mat
- Word of interest: cat
- Verb of interest: sat
-  

Indices:
- w.r.t. word: -1 0 1 2 3 4
- w.r.t. verb: -2 -1 0 1 2 3

Lookup Tables:
- $LT_w$
- $LT_{pw}$
- $LT_{pv}$

Convolution Layer
- Max Over Time
- Local features
- Global features

Tags
- Softmax
- Linear
- HardTanh
- Tags
Convolutions

Extract local features – share weights through time/space

Used with success in image (Le Cun, 1989) and speech (Bottou & Haffner, 1989)

Lookup-table is a special case: convolution with kernel size of 1 and input $i^{th}$ word

$(0, 0, \ldots, 1, 0, \ldots, 0)$ 1 at position $i$

Bengio et al (2001)
yesterday, after Microsoft bought Google, the dollar went down under half a euro and the fish market exploded.
After Microsoft bought Google, the dollar went down under half a euro and the fish market exploded.
Multi-Task Learning

Task 1

Lookup Tables

Convolution

Max

Classical NN Layer(s)

Softmax

Task 2

Lookup Tables

Convolution

Max

Classical NN Layer(s)

Softmax

Auxiliary task

Good overview in Caruana (1997)
Dictionary size of WSJ: about 36,000 words. Contains 1M of words.

15% of the most frequent words in the dictionary are seen 90% of the time.

Possible improvements:

- **Word clustering** (according to POS for e.g.).
  See Collobert & Weston, 2007

- **Thresholding** the number of words in the dictionary
  - order the words by frequency
  - words above the threshold are mapped to a special word “UNKNOWN”
Improving Word Embedding

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
Language Model: Think Massive

Language Model: “is a sentence actually english or not?”
Implicitly captures: ★ syntax ★ semantics

Bengio & Ducharme (2001) Probability of next word given previous words. Overcomplicated – we do not need probabilities here

English sentence windows: Wikipedia (∼ 631M words)
Non-english sentence windows: middle word randomly replaced

Multi-class margin cost:

\[
\sum_{s \in S} \sum_{w \in D} \max(0, 1 - f(s, w^*_s) + f(s, w))
\]

\(S\): sentence windows \(D\): dictionary
\(w^*_s\): true middle word in \(s\)
\(f(s, w)\): network score for sentence \(s\) and middle word \(w\)
Common Pitfall

Shuffling the data is really important (might be tricky on large datasets)...

![Graph showing valid-error and valid-error-shuffled data points]
Child Learning

Start with a simple problem, and then increase slowly the complexity.

In our case:

- order words by frequency in the dictionary
- start with a small vocabulary
- allow progressively more and more words...
- increase gradually the window size
## 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

We get: 14.30%. State-of-the-art: 16.54% — Pradhan et al. (2004)

250× faster than state-of-the-art. ∼ 0.01s to label a WSJ sentence.
MTL: Unified Network 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>5.4%</td>
<td>4.9%</td>
</tr>
<tr>
<td>Chunking – F1-score</td>
<td>91.5%</td>
<td>93.6%</td>
</tr>
</tbody>
</table>

POS: state-of-the-art ~ 3%

Chunking: Best system had 93.48% F1-score at CoNLL-2000 challenge [http://www.cnts.ua.ac.be/conll2000/chunking]. State-of-the-art is 94.1%. We get 94.9% by using POS features.
\[ SENmatch(Q, S) = \sum w(a) \max_{b \in PAS(S)} PASmatch(a, b), \quad w(a) = \frac{idf_{va}}{\sum_{k \in PAS(Q)} idf_{vk}} \]

\[ PASmatch(a, b) = \frac{V_{stfidf}^T(a) \cdot V_{stfidf}^T(b)}{\|V_{stfidf}^T(a)\|_2 \|V_{stfidf}^T(b)\|_2} \cdot I(a, b) \]

where \[ I(a, b) = \begin{cases} 
1 & v_a \text{ and } v_b \text{ are synonyms} \\
0 & \text{otherwise} 
\end{cases} \]
Noodle Experiments

Precision-Recall graph for simple queries

Precision-Recall graph for complex queries
With Semantic Args

Who\{WP acquired\} VBD Compaq\{NPP \} ?\| .\| .\|
Who\{ARG0 acquired\} \{rel Compaq\{ARG1 \} ?\| .\| .\|

Although HP has since acquired Compaq, the Presario name was not discontinued due to its marketability.

Compaq was in turn acquired by HP in 2002, bringing Tandem back to its original roots.

Hewlett-Packard acquired Compaq in 2002.

Compaq itself was acquired by Hewlett-Packard in 2002.

Pure TFIDF

Who\{WP acquired\} VBD Compaq\{NPP \} ?\| .\| .\|
Who\{ARG0 acquired\} \{rel Compaq\{ARG1 \} ?\| .\| .\|

In 1998, Compaq acquired Digital, and DECUS became a Compaq user group.

Tandem was acquired by Compaq in 1997.

Hewlett-Packard acquired Compaq in 2002.

Compaq itself was acquired by Hewlett-Packard in 2002.

The company reached $13 Billion in annual sales before it was acquired by
Noodle vs Google
Summary

We developed a deep neural network architecture for NLP

Advantages
★ General to any NLP tagging task
★ State-of-the-art performance
★ No hand designed features
★ Joint training
★ Can exploit massive unlabeled data
★ Extremely fast: 0.02s for all tags of a sentence

Inconvenients
★ Neural networks are a powerful tool: hard to handle

Early Impacts
★ Easy to apply to other tasks or languages: extending to Japanese
★ Fast: developed a semantic search system
Conclusion

- SVMs are a very good model for research studies
- Very complex (large-scale) tasks require more ambitious models