Deep Learning for Natural Language Processing – A Rabbit’s Perspective

Joakim Nivre
Uppsala University
Department of Linguistics and Philology
Chris Manning
ACL Presidential Address
Beijing, 2015
At DL 2015, Neil Lawrence said ...

“NLP is kind of like a rabbit in the headlights of the deep learning machine, waiting to be flattened.”
At DL 2015, Neil Lawrence said ...

“NLP is kind of like a rabbit in the headlights of the deep learning machine, waiting to be flattened.”
At DL 2015, Neil Lawrence said ...

“NLP is kind of like a rabbit in the headlights of the deep learning machine, waiting to be flattened.”
Features of Statistical Parsers

Confessions of a bottom-feeder:
Dredging in the Statistical Muck

Mark Johnson
Brown Laboratory for Linguistic Information Processing

CoNLL 2005

With much help from Eugene Charniak, Michael Collins and Matt Lease
Features of Statistical Parsers

Confessions of a bottom-feeder: Dredging in the Statistical Muck

Mark Johnson
Brown Laboratory for Linguistic Information Processing

CoNLL 2005

With much help from Eugene Charniak, Michael Collins and Matt Lease
Natural Language Processing
Natural Language Processing

Making computers perform useful and intelligent tasks involving natural language
Natural Language Processing

Making computers perform useful and intelligent tasks involving natural language

End-to-end applications:

• Machine translation
• Virtual assistants
Natural Language Processing

Making computers perform useful and intelligent tasks involving natural language

End-to-end applications:
- Machine translation
- Virtual assistants

Component technologies:
- Word sense disambiguation
- Co-reference resolution

“Carter told Mubarak he shouldn’t run again.”
Syntactic Parsing

the dog chased the cat from the room
Syntactic Parsing

Figure 1: Simplified UD annotation for equivalent sentences from English (top) and Finnish (bottom).
Syntactic Parsing

- Component technology – not end-to-end application
- Structured prediction task – exponential output space
PROCESSING ENGLISH WITH A
GENERALIZED PHRASE STRUCTURE GRAMMAR

Jean Mark Gawron, Jonathan King, John Lamping, Egon Loebner,
E. Anne Paulson, Geoffrey K. Pullum, Ivan A. Sag, and Thomas Wasow

Computer Research Center
Hewlett Packard Company
1501 Page Mill Road
Palo Alto, CA 94304

ABSTRACT

This paper describes a natural language processing system implemented at Hewlett-Packard's Computer Research Center. The system's main components are: a Generalized Phrase Structure Grammar (GPSG); a top-down parser; a logic transducer that outputs a first-order logical representation; and a "disambiguator" that uses sortal information to convert "normal-form" first-order logical expressions into the query language for HIRE, a relational database hosted in the SPHERE system. We argue that theoretical developments in GPSG syntax and in Montague semantics have specific advantages to bring to this domain of computational linguistics. The syntax and semantics of the system are totally domain-independent, and thus, in principle, highly portable. We discuss the prospects for extending domain-independence to the lexical semantics as well, and thus to the logical semantic representations.

can be achieved without detailed syntactic analysis. There is, of course, a massive pragmatic component to human linguistic interaction. But we hold that pragmatic inference makes use of a logically prior grammatical and semantic analysis. This can be fruitfully modeled and exploited even in the complete absence of any modeling of pragmatic inferencing capability. However, this does not entail an incompatibility between our work and research on modeling discourse organization and conversational interaction directly. Ultimately, a successful language understanding system will require both kinds of research, combining the advantages of precise, grammar-driven analysis of utterance structure and pragmatic inferencing based on discourse structures and knowledge of the world. We stress, however, that our concerns at this stage do not extend beyond the specification of a system that can efficiently extract literal meaning from isolated sentences of arbitrarily complex grammatical structure. Future systems will exploit the literal meaning thus extracted in more
NLP in the 1980s
NLP in the 1980s

Languages described by formal systems

• Inventory of elementary units (lexicon)
• Rules for combining units (grammar)
NLP in the 1980s

Languages described by formal systems

- Inventory of elementary units (lexicon)
- Rules for combining units (grammar)

Created by linguists in a theoretical framework

- Linguistic levels: morphology, syntax, semantics
- Generate all and only well-formed expressions
NLP in the 1980s

Languages described by formal systems
  • Inventory of elementary units (lexicon)
  • Rules for combining units (grammar)

Created by linguists in a theoretical framework
  • Linguistic levels: morphology, syntax, semantics
  • Generate all and only well-formed expressions

Combined with algorithms for analysis/synthesis
Issues
Issues

Coverage

• Hard to build a complete description of a language
• Languages are constantly changing
Issues

Coverage
• Hard to build a complete description of a language
• Languages are constantly changing

Robustness
• Language use is not always well-formed
• Made worse by lack of coverage
Issues

Coverage

• Hard to build a complete description of a language
• Languages are constantly changing

Robustness

• Language use is not always well-formed
• Made worse by lack of coverage

Ambiguity

• Natural language grammars inherently ambiguous
• Combinatorial explosion from interacting rules and levels
• Practical applications require disambiguation
Three New Probabilistic Models for Dependency Parsing: An Exploration*

Jason M. Eisner
CIS Department, University of Pennsylvania
200 S. 33rd St., Philadelphia, PA 19104-6389, USA
jeisner@linc.cis.upenn.edu

Abstract

After presenting a novel $O(n^3)$ parsing algorithm for dependency grammar, we develop three contrasting ways to stochasticize it. We propose (a) a lexical affinity model where words struggle to modify each other, (b) a sense tagging model where words fluctuate randomly in their selectional preferences, and (c) a generative model where the speaker fleshes out each word's syntactic and conceptual structure without regard to the implications for the hearer. We also give preliminary empirical results from evaluating the three models' parsing performance on annotated Wall Street Journal training text (derived from the Penn Treebank). In these results, the generative model performs significantly better than the others, and

Figure 1: (a) A bare-bones dependency parse. Each word points to a single parent, the word it modifies; the head of the sentence points to the EOS (end-of-sentence) mark. Crossing links and cycles are not allowed. (b) Constituent structure and subcategorization may be highlighted by displaying the same dependencies as a lexical tree.
NLP in the 1990s
NLP in the 1990s

Probabilistic models of language
  • Generative models of $P(X,Y)$
NLP in the 1990s

Probabilistic models of language
  • Generative models of $P(X, Y)$

Parameters estimated from (annotated) data
  • Maximum-likelihood estimation
  • Smoothing to cope with sparse data
Probabilistic models of language

- Generative models of $P(X,Y)$

Parameters estimated from (annotated) data

- Maximum-likelihood estimation
- Smoothing to cope with sparse data

Inference algorithms for analysis:

- Exact argmax search using dynamic programming
How does this help?
How does this help?

Ambiguity

• Disambiguation through probability ranking
• Learning from data more effective than heuristics
How does this help?

Ambiguity

• Disambiguation through probability ranking
• Learning from data more effective than heuristics

Robustness

• Probability ranking allows constraint relaxation
• No sharp line between well-formed and deviant
A New Paradigm
A New Paradigm

Emphasis on robust large-scale processing
A New Paradigm

Emphasis on robust large-scale processing

Quantitative evaluation
  • Naturally occurring test data
  • Exact numerical metrics
A New Paradigm

Emphasis on robust large-scale processing

Quantitative evaluation
  • Naturally occurring test data
  • Exact numerical metrics

Data-driven development
  • Naturally occurring training data
  • Models induced using statistical inference
Limitations
Limitations

Statistical models of the (early) 1990s:

• Generative models of $P(X,Y)$
• Maximum likelihood estimation (with smoothing)
• No advanced learning algorithms – just counting
Limitations

Statistical models of the (early) 1990s:
- Generative models of $P(X,Y)$
- Maximum likelihood estimation (with smoothing)
- No advanced learning algorithms – just counting

Main problem:
- Rigid independence assumptions (local context)
- Required for effective learning and efficient inference
Online Large-Margin Training of Dependency Parsers

Ryan McDonald  Koby Crammer  Fernando Pereira
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA
{ryantm, crammer, pereira}@cis.upenn.edu

Abstract

We present an effective training algorithm for linearly-scored dependency parsers that implements online large-margin multi-class training (Crammer and Singer, 2003; Crammer et al., 2003) on top of efficient parsing techniques for dependency trees (Eisner, 1996). The trained parsers achieve a competitive dependency models of the same vintage even though it scores parsing decisions in isolation and thus may suffer from the label bias problem (Lafferty et al., 2001).

Discriminatively trained parsers that score entire trees for a given sentence have only recently been investigated (Riezler et al., 2002; Clark and Curran, 2004; Collins and Roark, 2004; Taskar et al., 2004). The most likely reason for this is that discriminative training requires repeatedly re-parsing the training corpus with the current model to determine the
NLP in the 2000s
NLP in the 2000s

Conditional or discriminative models

- Conditional models of $P(Y|X)$
- Linear models for prediction $X \rightarrow Y$
NLP in the 2000s

Conditional or discriminative models

- Conditional models of $P(Y|X)$
- Linear models for prediction $X \rightarrow Y$

Parameters estimated from (annotated) data

- Learning as numerical optimization
- Regularisation to prevent overfitting
NLP in the 2000s

Conditional or discriminative models
  • Conditional models of $P(Y|X)$
  • Linear models for prediction $X \rightarrow Y$

Parameters estimated from (annotated) data
  • Learning as numerical optimization
  • Regularisation to prevent overfitting

Inference algorithms for analysis:
  • Exact argmax search not always possible
  • Heuristic methods like beam search and reranking
How does this help?
How does this help?

Independence assumptions can be relaxed

• No need to estimate joint distribution \( P(X,Y) \)
• Features over input \( X \) come for free
How does this help?

Independence assumptions can be relaxed

• No need to estimate joint distribution $P(X,Y)$
• Features over input $X$ come for free

Prediction accuracy improves with rich features

• Arbitrary combinations of input and output features
• Fall back on heuristic inference for efficiency if needed
Independence assumptions can be relaxed

- No need to estimate joint distribution $P(X, Y)$
- Features over input $X$ come for free

Prediction accuracy improves with rich features

- Arbitrary combinations of input and output features
- Fall back on heuristic inference for efficiency if needed
Limitations
Limitations

Sparse discrete features

- Most features are binarized symbolic features (1-hot)
- Feature vectors get extremely high-dimensional but sparse
Limitations

Sparse discrete features

- Most features are binarized symbolic features (1-hot)
- Feature vectors get extremely high-dimensional but sparse

Feature engineering

- Linear models require complex hand-crafted features
- Features have to be selected in trial-and-error experiments
Limitations

Sparse discrete features
- Most features are binarized symbolic features (1-hot)
- Feature vectors get extremely high-dimensional but sparse

Feature engineering
- Linear models require complex hand-crafted features
- Features have to be selected in trial-and-error experiments

Rigid locality
- Features are mostly local and restricted to fixed windows
we saw her duck
we saw her duck

\[\text{parse}(s) = \arg \max_{y \in \mathcal{Y}(s)} \sum_{(h,m) \in y} \text{score}(\phi(s,h,m))\]
we saw her duck

\[ \text{score}(\phi(s, h, m)) = \sum_{i=1}^{k} w_i \times \phi_i(s, h, m) \]
we saw her duck

score(\phi(s, h, m)) = \sum_{i=1}^{k} w_i \times \phi_i(s, h, m)
we saw her duck

\[
\text{score}(\phi(s, h, m)) = \sum_{i=1}^{k} w_i \times \phi_i(s, h, m)
\]
we saw her duck

score(φ(s, h, m)) = \sum_{i=1}^{k} w_i \times φ_i(s, h, m)
we saw her duck

\[ \text{score}(\phi(s, h, m)) = \sum_{i=1}^{k} w_i \times \phi_i(s, h, m) \]
We present a simple and effective scheme for dependency parsing which is based on bidirectional-LSTMs (BiLSTMs). Each sentence token is associated with a BiLSTM vector representing the token in its sentential context, and feature vectors are constructed by concatenating a few BiLSTM vectors. The BiLSTM is trained jointly with the parser objective, resulting in very effective feature extractors for parsing. We demonstrate the effectiveness of the approach by applying it to a greedy transition-based parser as well as to a globally optimized graph-based parser. The resulting parsers have very simple architectures, and match or surpass the state-of-the-art accuracies on English and Chinese.

1 Introduction

The focus of this paper is on feature representation for dependency parsing, using recent techniques from the neural-networks ("deep learning") literature. Modern approaches to dependency parsing can be broadly categorized into graph-based and transition-based parsers (Kübler et al., 2009). Graph-based parsers (McDonald, 2006) treat parsing as a search-based structured prediction problem in which the goal is learning a scoring function over dependency trees such that the correct tree is scored above all other trees. Transition-based parsers (Nivre, 2004; Nivre, 2008) treat parsing as a sequence of actions that produce a parse tree, and a classifier is trained to score the possible actions at each stage of the process and guide the parsing process. Perhaps the simplest graph-based parsers are arc-factored (first order) models (McDonald, 2006), in which the scoring function for a tree decomposes over the individual arcs of the tree. More elaborate models look at larger (overlapping) parts, requiring more sophisticated inference and training algorithms (Martins et al., 2009; Koo and Collins, 2010). The basic transition-based parsers work in a greedy manner, performing a series of locally-optimal decisions, and boast very fast parsing speeds. More advanced transition-based parsers introduce some search into the process using a beam (Zhang and Clark, 2008). Regardless of the details of the parsing framework being used, a crucial step in parser design is choosing the right feature function for the underlying statistical model. Recent work (see Section 2.2 for an overview) attempt to alleviate parts of the feature function design problem by moving from linear to non-linear models, enabling the modeler to focus on a small set of "core" features and leaving it up to the machine-learning machinery to come up with good feature combinations (Chen and Manning, 2014; Pei et al., 2015; Lei et al., 2014; Taub-Tabib et al., 2015). However, the need to carefully define a set of core features remains. For example, the work of Chen and Manning (2014) uses 18 different elements in its feature function, while the work of Pei et al. (2015) uses 21 different elements. Other works, notably Dyer et al. (2015) and Le and Zuidema (2014), propose more sophisticated feature representations, in which the feature engineering is replaced with architecture engineering.

In this work, we suggest an approach which is much simpler in terms of both feature engineering and model architecture.
Illustration from Kiperwasser and Goldberg, Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations Networks, TACL 2016.
\[
\text{parse}(s) = \arg\max_{y \in \mathcal{Y}(s)} \sum_{(h, m) \in y} \text{score}(\phi(s, h, m))
\]
\[
\text{parse}(s) = \arg \max_{y \in \mathcal{Y}(s)} \sum_{(h, m) \in y} \text{score}(\phi(s, h, m))
\]
In contrast, our feature extractor uses merely the sparse feature vector for an arc linking modifier.

Given the scores of the arcs, the highest scoring production is usually computed using a dynamic-programming algorithm.

The architecture is illustrated in Figure 2.

Figure 2: Illustration of the neural model scheme of the graph-based parser when calculating the score of a given parse tree. The parse tree is depicted below the sentence. Each dependency arc in the sentence is scored using an MLP that defines a hinge loss with respect to a gold tree, and finds the best scoring tree using a dynamic-programming algorithm.

Our feature function is then a concatenation of a representation which is suitable for the parsing task.

The resulting feature vectors are then scored using a MLP and the highest scoring incorrect tree is rejected, giving the score for the correct parse.

The training objective is to set the score of a tree to be infinite if it is invalid, and zero otherwise.

The resulting parse is then a concatenation of a representation which is suitable for the parsing task.

The final model is:

\[
\text{parse}(s) = \arg \max_{y \in \mathcal{Y}(s)} \sum_{(h,m) \in y} \text{score}(\phi(s,h,m))
\]

\[
v_i = \text{BIRNN}(x_{1:n}, x_i)
\]

\[
x_i = e(w_i) \circ e(p_i)
\]
\[
\text{parse}(s) = \arg \max_{y \in \mathcal{Y}(s)} \sum_{(h,m) \in y} \text{score}(\phi(s,h,m))
\]

\[
\text{score}(\phi(s,h,m)) = \text{MLP}(v_h \circ v_m)
\]

\[
x_i = e(w_i) \circ e(p_i)
\]

\[
v_i = \text{BiRNN}(x_{1:n}, x_i)
\]
parse(s) = \arg \max_{y \in \mathcal{Y}(s)} \sum_{(h, m) \in y} \text{score}(\phi(s, h, m))

score(\phi(s, h, m)) = \text{MLP}(v_h \circ v_m)

x_i = e(w_i) \circ e(p_i)

v_i = \text{BiRNN}(x_{1:n}, x_i)
Improves the accuracy of English dependency parsing from 90% to 95%

Why?

\[
\text{parse}(s) = \arg \max_{y \in \mathcal{Y}(s)} \sum_{(h,m) \in y} \text{score}(\phi(s, h, m))
\]

\[
\text{score}(\phi(s, h, m)) = \text{MLP}(v_h \circ v_m)
\]
## Embeddings

<table>
<thead>
<tr>
<th>l-Hot</th>
<th>Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 ... 0 0 1 0</td>
<td>0.1 0.9 -0.2 0.3 ... -0.1 -0.5</td>
</tr>
<tr>
<td>binary, sparse</td>
<td>continuous, dense</td>
</tr>
<tr>
<td>dim (\sim 10^5)–(10^6)</td>
<td>dim (\sim 10^2)</td>
</tr>
</tbody>
</table>

\[ x_i = e(w_i) \circ e(p_i) \]
Embeddings

I-Hot

| 0 | 0 | 0 | 0 | ... | 0 | 0 | 1 | 0 |

binary, sparse
dim $\sim 10^5$–$10^6$

Embedding

| 0.1 | 0.9 | -0.2 | 0.3 | ... | -0.1 | -0.5 |

continuous, dense
dim $\sim 10^2$

• Embeddings inherently more expressive

\[ x_i = e(w_i) \circ e(p_i) \]
Embeddings

**1-Hot**

- Binary, sparse
- dim $\sim 10^5$–$10^6$

**Embedding**

- Continuous, dense
- dim $\sim 10^2$

- Embeddings inherently more expressive
- Can capture similarities between items (sparsity)
Embeddings

<table>
<thead>
<tr>
<th>I-Hot</th>
<th>Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 ... 0 0 1 0</td>
<td>0.1 0.9 -0.2 0.3 ... -0.1 -0.5</td>
</tr>
</tbody>
</table>

- binary, sparse
  dim ~ 10^5–10^6
- continuous, dense
  dim ~ 10^2

- Embeddings inherently more expressive
- Can capture similarities between items (sparsity)
- Can be pre-trained on large unlabeled corpora (OOV)

\[ x_i = e(w_i) \circ e(p_i) \]
Embeddings

<table>
<thead>
<tr>
<th>I-Hot</th>
<th>Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 ... 0 0 1 0</td>
<td>0.1 0.9 -0.2 0.3 ... -0.1 -0.5</td>
</tr>
</tbody>
</table>

- Binary, sparse
  - $\text{dim} \sim 10^5 - 10^6$
- Continuous, dense
  - $\text{dim} \sim 10^2$

- Embeddings inherently more expressive
- Can capture similarities between items (sparsity)
- Can be pre-trained on large unlabeled corpora (OOV)
- Can be learned/tuned specifically for the task at hand

$$x_i = e(w_i) \circ e(p_i)$$
Not only words ...
Illustration from Chen and Manning, A Fast and Accurate Parser Using Neural Networks, EMNLP 2014.
Not only words …

Illustration from Chen and Manning, A Fast and Accurate Parser Using Neural Networks, EMNLP 2014.

\[ x_i = e(w_i) \circ e(p_i) \]
Not only words …

Illustration from Chen and Manning, A Fast and Accurate Parser Using Neural Networks, EMNLP 2014.
Feature Learning

\[ \text{score}(\phi(s, h, m)) = \text{MLP}(v_h \circ v_m) \]
Feature Learning

Output layer

Hidden layer

Input layer

- Feature representations are learned in hidden layers

Illustration from Chen and Manning, A Fast and Accurate Parser Using Neural Networks, EMNLP 2014.

\[ \text{score}(\phi(s, h, m)) = \text{MLP}(v_h \circ v_m) \]
Feature Learning

- Feature representations are learned in hidden layers
- All features are learned together – end-to-end training

Illustration from Chen and Manning, A Fast and Accurate Parser Using Neural Networks, EMNLP 2014.
Feature Learning

- Feature representations are learned in hidden layers
- All features are learned together – end-to-end training
- Hyper-parameter tuning considerably more complex

Illustration from Chen and Manning, A Fast and Accurate Parser Using Neural Networks, EMNLP 2014.
Recurrent Networks

Illustration from Kiperwasser and Goldberg, Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations Networks, TACL 2016.

\[ v_i = \text{BiRNN}(x_{1:n}, x_i) \]
Recurrent Networks

- RNNs fit the sequential structure of language
Recurrence Networks

- RNNs fit the sequential structure of language
- Information can be transferred over arbitrary distances

Illustration from Kiperwasser and Goldberg, Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations Networks, TACL 2016.
Recurrent Networks

- RNNs fit the sequential structure of language
- Information can be transferred over arbitrary distances
- Makes global information available locally

\[ v_i = \text{BIRNN}(x_{1:n}, x_i) \]
A New Paradigm?
A New Paradigm?

Empirical results have improved a lot in five years

- But tasks and evaluation criteria remain very similar
A New Paradigm?

Empirical results have improved a lot in five years
  • But tasks and evaluation criteria remain very similar

Deep learning gives more effective features
  • Features are continuous and dense (not discrete and sparse)
  • Features are learned (not hand-crafted)
  • Features can be tuned to (multiple) specific tasks
  • Features can capture unbounded dependencies
A New Paradigm?

Empirical results have improved a lot in five years
  • But tasks and evaluation criteria remain very similar

Deep learning gives more effective features
  • Features are continuous and dense (not discrete and sparse)
  • Features are learned (not hand-crafted)
  • Features can be tuned to (multiple) specific tasks
  • Features can capture unbounded dependencies

Structured prediction models has not changed as much
  • But better features allow us to simplify models
Deep learning in NLP is not a revolution — it is an evolution!
An Afterthought
An Afterthought

Is this a limited rabbit’s perspective?

• Is linguistic structure prediction a thing of the past?
• Is the future in end-to-end training of end-to-end applications?
• Is this the real deep learning revolution?
An Afterthought

Is this a limited rabbit’s perspective?
  • Is linguistic structure prediction a thing of the past?
  • Is the future in end-to-end training of end-to-end applications?
  • Is this the real deep learning revolution?

Will linguistic structure be relevant in the future?
  • Do end-to-end models learn linguistic structure implicitly?
  • If so, do they learn the structure predicted by linguistic theory?
  • Do they benefit from incorporating linguistic structure prediction?
  • Do they benefit from an inductive bias towards linguistic structure?
An Afterthought

Is this a limited rabbit’s perspective?
  • Is linguistic structure prediction a thing of the past?
  • Is the future in end-to-end training of end-to-end applications?
  • Is this the real deep learning revolution?

Will linguistic structure be relevant in the future?
  • Do end-to-end models learn linguistic structure implicitly?
  • If so, do they learn the structure predicted by linguistic theory?
  • Do they benefit from incorporating linguistic structure prediction?
  • Do they benefit from an inductive bias towards linguistic structure?

Will rabbits go extinct?