Acquiring Semantic Knowledge using Patterns

Roy Schwartz, NLP Lab, The Hebrew University
Learning Club, Dec. 4th, 2014
Overview

• NLP
  – Problems and open questions
  – Main approaches

• Lexico-syntactic Patterns

• Latest Results
  – Interpretable Word Embeddings Using Patterns Features (Schwartz, Reichart and Rappoport, under review)
NLP
NLP

Text Understanding
NLP

Language Representation

Text Understanding
Acquiring Semantic Knowledge using Patterns
@ Roy Schwartz
Language Representation

Text Generation

Text Understanding
Acquiring Semantic Knowledge using Patterns
@ Roy Schwartz
Acquiring Semantic Knowledge using Patterns
@ Roy Schwartz
NLP is Hard

Ambiguity

Noise

Paraphrasing

Complex structures
NLP is Hard

Language is Hard!
NLP Tasks

• High Level (Applications)
  – Search
  – Question Answering
  – Machine Translation
  – Summarization
  – Sentiment Analysis
  – ...

Acquiring Semantic Knowledge using Patterns
@ Roy Schwartz
NLP Tasks

• High Level (Applications)
  – Search
  – Question Answering
  – Machine Translation
  – Summarization
  – Sentiment Analysis
  – ...

• Low Level
  – Syntactic
    • Parsing
    • Part-of-speech Tagging
NLP Tasks

High Level (Applications)
- Search
- Question Answering
- Machine Translation
- Summarization
- Sentiment Analysis
- ...

Low Level
- Syntactic
  - Parsing
  - Part-of-speech Tagging
- Semantic
  - Semantic Role Labeling
  - Textual Entailment
  - Word Clustering
  - Word Representation
NLP Tasks

• High Level (Applications)
  – Search
  – Question Answering
  – Machine Translation
  – Summarization
  – Sentiment Analysis
  – ...

• Low Level
  – Syntactic
    • Parsing
    • Part-of-speech Tagging
  – Semantic
    • Semantic Role Labeling
    • Textual Entailment
    • Word Clustering
    • Word Representation

Acquiring Semantic Knowledge using Patterns
@ Roy Schwartz
NLP Tasks

• **High Level (Applications)**
  – Search
  – Question Answering
  – Machine Translation
  – Summarization
  – Sentiment Analysis
  – ...

• **Low Level**
  – Syntactic
    • Parsing
    • Part-of-speech Tagging
  – Semantic
    • Semantic Role Labeling
    • Textual Entailment
    • Word Clustering
    • **Word Representation**
Language Model

• Compute the probability for every sequence of words
  – Required by virtually every high level task (machine translation, questions answering, summarization, speech recognition, etc.)
Language Model

• Compute the probability for every sequence of words
  – Required by virtually every high level task (machine translation, questions answering, summarization, speech recognition, etc.)

• Impossible to compute (exponentially large number of sequences)
Language Model

• Compute the probability for every sequence of words
  – Required by virtually every high level task (machine translation, questions answering, summarization, speech recognition, etc.)

• Impossible to compute (exponentially large number of sequences)

• A common solution: Markov independence assumption
  – Formally: compute $p(w_i | w_{i-1}, \ldots, w_{i-n})$
  – $n$ usually equals 3 (trigram models)
Neuro-probabilistic Language Models

• Address sparsity by building a (dense) vector word representation (aka *word embeddings*)
  – Replace \( p(w_i | w_{i-1}, ..., w_{i-n}) \) with \( p(w_i | V_{i-1}, ..., V_{i-n}) \)
Neuro-probabilistic Language Models

• Address sparsity by building a (dense) vector word representation (aka *word embeddings*)
  – Replace $p(w_i | w_{i-1}, ..., w_{i-n})$ with $p(w_i | V_{i-1}, ..., V_{i-n})$

• Use deep neural networks to train language models
  – Bengio, 2003; Collobert, 2008 & 2011, *word2vec* (Mikolov 2013{a,b,c})
Neuro-probabilistic Language Models

• Address sparsity by building a (dense) vector word representation (aka \textit{word embeddings})
  – Replace \( p(w_i | w_{i-1}, ..., w_{i-n}) \) with \( p(w_i | V_{i-1}, ..., V_{i-n}) \)

• Use deep neural networks to train language models
  – Bengio, 2003; Collobert, 2008 & 2011, \textit{word2vec} (Mikolov 2013\{a,b,c\})

• Surprisingly, the word representations turned out to be quite successful on their own
Bag of Words Models

• Main type of feature
  – Used in various NLP tasks
  – The idea: use words as features, ignoring words order
  – General principle in computing word embeddings
Main type of feature

- Used in various NLP tasks
- The idea: use words as features, ignoring words order
- General principle in computing **word embeddings**

  ... tokens to date, **friend** lists and recent ... 
  ... by my dear **friend** and companion, Fritz von ... 
  ... even have a **friend** who never fails ... 
  ... by my worthy **friend** Doctor Haygarth of ... 
  ... and as a **friend** pointed out to ... 
  ... partner, in-laws, relatives or **friends** speak a different ... 
  ... petition to a **friend** Go to the ... 
  ... otherwise, to a **friend** or family member ... 
  ...images from my **friend** Rory though - ... 
  ... great, and a **friend** as well as a colleague, who, ...
Bag of Words Models

• Main type of feature
  – Used in various NLP tasks
  – The idea: use words as features, ignoring words order
  – General principle in computing word embeddings

... tokens to date, friend lists and recent ...
... by my dear friend and companion, Fritz von ...
... even have a friend who never fails ...
... by my worthy friend Doctor Haygarth of ...
... and as a friend pointed out to ...
... partner, in-laws, relatives or friends speak a different ...
... petition to a friend Go to the ...
... otherwise, to a friend or family member ...
... images from my friend Rory though - ...
... great, and a friend as well as a colleague, who, ...
...
word2vec’s Skip-Gram Model
Mikolov et al., 2013

• A deep learning method designed to learn an NLM
word2vec’s Skip-Gram Model
Mikolov et al., 2013

- A deep learning method designed to learn an NLM

- For each word \( w \) in the vocabulary \( V \), learn both a “target-embedding” \( v_w \) and a “context-embedding” \( v_c \)
  - \( p(c|w) \) is computed using soft-max:

\[
p(c \mid w) = \frac{e^{v_c \cdot v_w}}{\sum_{w' \in V} e^{v_c \cdot v_{w'}}}
\]
word2vec’s Skip-Gram Model
Mikolov et al., 2013

- A deep learning method designed to learn an NLM

- For each word w in the vocabulary V, learn both a “target-embedding” $v_w$ and a “context-embedding” $v_c$
  - $p(c|w)$ is computed using soft-max:
    $$p(c|w) = \frac{e^{v_c v_w}}{\sum_{w' \in V} e^{v_c v_{w'}}}$$

- For each training sentence, treat each word in turn as a target word
  - Sample (word, context) pairs from a window of nearby words
**word2vec**’s Skip-Gram Model (2)

Mikolov et al., 2013

**Objective function:**

\[
\max_{\theta} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t)
\]

**Algorithm:**

Stochastic gradient descent
Word Embeddings Applications

- Information Retrieval
- Document Classification
- Question Answering
- Named Entity Recognition
- Parsing
- ...
Word Embeddings (Cool!) Properties

- (accurate) Word similarity
Word Embeddings (Cool!) Properties

- (accurate) Word similarity
- Word analogy

(Mikolov et al., 2013)
Word Embeddings Limitations

• Resulting vectors are highly **uninterpretable**
  – Sequences of several hundred numbers
  – Not clear what each number represents
Word Embeddings Limitations

• Resulting vectors are highly **uninterpretable**
  – Sequences of several hundred numbers
  – Not clear what each number represents

• Restricted to a limited set of relations
  – Similarity/Relatedness, some analogies
  – Other relations are not supported: hyponymy (animal $\rightarrow$ dog), antonymy (big/tall), etc.
Lexico-syntactic Patterns
Hearst, 1992

• Patterns that contain words and wildcards
  – “X is a country”, “X such as Y”, etc.
Lexico-syntactictic Patterns
Hearst, 1992

• Patterns that contain words and wildcards
  – “X is a country”, “X such as Y”, etc.

• Patterns potentially capture the context in which a word participates
Lexico-syntactic Patterns
Hearst, 1992

• Patterns that contain words and wildcards
  – “X is a country”, “X such as Y”, etc.

• Patterns potentially capture the context in which a word participates

• For example:
  – A dog participates in patterns (contexts) such as:
    – “X barks”, “X has a tail”, “X and cats”, ...
Pattern Applications

• Acquiring the semantics of relationships between words
  – Discovering hyponymy (animal $\rightarrow$ cat) (Hearst, 1992)
  – Discovering meronymy (cat $\rightarrow$ tail) (Berland & Charniak, 1999)
  – Discovering antonymy (big / small) (Lin, 2003)

• Word clustering and classification
  – Davidov & Rappoport, 2006; Schwartz, Reichart & Rappoport, 2014

• Sentence Level Applications
  – Sarcasm Detection (Tsur, Davidov & Rappoport, 2010)
  – Sentiment Analysis (Davidov, Tsur, & Rappoport, 2010)
  – Authorship Attribution (Schwartz et al., 2013)
Examples of Patterns

• Extracting antonymy (opposite) relations
  – “either X or Y”
  – John is either tall or short
  – either stay or come with us
Examples of Patterns

• Extracting antonymy (opposite) relations
  – “either X or Y”
  – John is either tall or short
  – either stay or come with us

• Extracting hyponymy (is-a) relations
  – “X such as Y”
  – Cut the stems of boxed flowers such as roses
  – I am responsible for preparing a range of fruits such as apples
Word Similarity via Patterns

• Some patterns are useful for identifying words that are similar*
  – mouse / rat, shirt / sweater, etc.

  ❖ This is something that word embeddings are generally good at
Word Similarity via Patterns

• Some patterns are useful for identifying words that are similar*
  – mouse / rat, shirt / sweater, etc.

• These patterns are symmetric
  – Contain exactly two wildcards \((X, Y)\)
  – Words \(w_i, w_j\) that co-occur in these patterns can come in both forms \((X=w_i, Y=w_j)\) and \((X=w_j, Y=w_i)\)

⚠️ This is something that word embeddings are generally good at
Symmetric Patterns (SPs)

• **X and Y**
  – cats and dogs, dogs and cats
  – France and England, England and France

• **X as well as Y**
  – friends as well as colleagues, colleagues as well as friends
  – apples and oranges, oranges and apples

• ….
Automatically Extracted Symmetric Patterns

The (Davidov and Rappoport, 2006) Algorithm

• A graph-based algorithm
  – Input: a corpus of plain text
  – Output: a set of symmetric patterns
Automatically Extracted Symmetric Patterns
The (Davidov and Rappoport, 2006) Algorithm

• A graph-based algorithm
  – Input: a corpus of plain text
  – Output: a set of symmetric patterns

• The idea: search for patterns with interchangeable word pairs
  – For each pattern candidate, compute symmetry measure ($M$)
  – Select the $N$ patterns with the highest $M$ values
Automatically Extracted Symmetric Patterns (2)

The (Davidov and Rappoport, 2006) Algorithm

• The $M$ measure counts the proportion of pattern instances that appear in both directions (“cat and dog” + “dog and cat”)

• High $M$ value $\rightarrow$ A symmetric pattern
DR06 Example

X and Y
DR06 Example

X and Y

Symmetric edges
Acquiring Semantic Knowledge using Patterns
@ Roy Schwartz
Acquiring Semantic Knowledge using Patterns

DR06 Example

Symmetric edges

Asymmetric edges

X and Y

Dog

Cat

Computer

Rat

Camera

M = \#symmetric\_edges

\#all\_edges

M = \#symmetric\_edges

\#all\_edges
So Far

• Word embeddings are great
  – Useful for down stream applications
  – Have cool properties (similarity, word analogies)
So Far

• **Word embeddings are great**
  – Useful for downstream applications
  – Have cool properties (similarity, word analogies)

• **But not perfect**
  – Uninterpretable
  – Unable to recognize various relations
Interpretable Word Embeddings Using Pattern Features

Roy Schwartz, Roi Reichart and Ari Rappoport
(Under Revision)
Outline

• Learn **interpretable** and **high quality** word embeddings
  – Substantially outperform state-of-the-art word2vec embeddings

• Show the benefits of interpretability
  – Use our embeddings to assign dissimilar vectors to antonym pairs (big/small)

• Combine our embeddings with state-of-the-art embeddings to get improved expressive power
Symmetric Patterns to Word Similarity

• Input: a large corpus C
Symmetric Patterns to Word Similarity

• Input: a large corpus C

• Extract a set of SPs $P$ using the DR06 algorithm
  – “X and Y”, “X or Y”, “X and the Y”, “from X to Y”, “X or the Y”, “X as well as Y”, “X or a Y”, “X rather than Y”, “X nor Y”, “X and one Y”, “either X or Y”
Symmetric Patterns to Word Similarity

• Input: a large corpus C

• Extract a set of SPs $P$ using the DR06 algorithm
  – “$X$ and $Y$”, “$X$ or $Y$”, “$X$ and the $Y$”, “from $X$ to $Y$”, “$X$ or the $Y$”, “$X$ as well as $Y$”, “$X$ or a $Y$”, “$X$ rather than $Y$”, “$X$ nor $Y$”, “$X$ and one $Y$”, “either $X$ or $Y$”

• Traverse $C$, extract all instances of all $p$ in $P$
  – cats and dogs
  – House and the rooms
  – from France to England
  – ...

Acquiring Semantic Knowledge using Patterns
@ Roy Schwartz
Symmetric Patterns to Word Similarity (2)

• For each word $w$ in the lexicon, build a count vector ($V_w$) of all other words that co-occur with $w$ in SPs.
Symmetric Patterns to Word Similarity (2)

- For each word $w$ in the lexicon, build a count vector ($V_w$) of all other words that co-occur with $w$ in SPs

- **orange**
  1. ... apples and oranges ...
  2. ... oranges as well as grapes ...
  K. ... neither banana nor orange

- **China**
  1. ... Japan or China ...
  2. ... China rather than Korea ...
  M. ... Vietnam and China ...
Symmetric Patterns to Word Similarity (3)

- Compute the Positive Pointwise Mutual Information (PPMI) between each pair of words

\[
PMI(w_i, w_j) = \log \left( \frac{p(w_i, w_j)}{p(w_i) p(w_j)} \right)
\]

\[
PPMI(w_i, w_j) = \begin{cases} 
PMI(w_i, w_j) < 0 : 0 \\
otherwise : PMI(w_i, w_j)
\end{cases}
\]
The Result: **Interpretable** Word Embeddings based on Symmetric Patterns

\[
V_{\text{dog}}^{sp} = \{ \text{PPMI(\text{dog,house})}, \text{PPMI(\text{dog,mouse})}, \text{PPMI(\text{dog,zebra})}, \text{PPMI(\text{dog,wine})}, \text{PPMI(\text{dog,cat})}, \text{PPMI(\text{dog,dolphin})}, \text{PPMI(\text{dog,bottle})}, \text{PPMI(\text{dog,pen})} \}
\]
The Result: **Interpretable** Word Embeddings based on Symmetric Patterns

\[ V_{\text{dog}}^{\text{sp}} = \{ \text{PPMI(} \text{dog,house}) , \text{PPMI(} \text{dog,mouse} \} \text{,PPMI(} \text{dog,zebra}) , \text{PPMI(} \text{dog,wine}) , \text{PPMI(} \text{dog,cat}) , \text{PPMI(} \text{dog,dolphin}) , \text{PPMI(} \text{dog,bottle}) , \text{PPMI(} \text{dog,pen}) \} \]

\[ |V_{w}^{SP}| = \sim 500K \]

\[ E_{w}(|\text{nonzero}(V_{w}^{SP})|) = \sim 50 \]
Smoothing

- For each word $w$, $V^N_w$ denotes the top $N$ vectors with the smallest cosine distance to $V^{SP}_w$

$$V^{SP'}_w = V^{SP}_w + \alpha \sum_{v \in V^N_w} v$$

- Using $N=50$: $E_w(|\text{nonzero}(V^{SP'}_w)|) = \sim 8K$

- $\alpha$ and $N$ are tuned using a development set
Antonyms

big / small
Antonyms

big / small

• Antonyms appear in similar contexts
  • Here is a X car
  • I live in a X house
Antonyms

**big / small**

- Antonyms appear in similar contexts
  - Here is a \(X\) car
  - I live in a \(X\) house

⇒ In typical word embeddings, \(\cos(V_{\text{big}}, V_{\text{small}})\) is high
Antonyms

big / small

• Some symmetric patterns are indicative of antonymy*
  – “either X or Y” (either big or small), “from X to Y” (from poverty to richness)

* Lin et al. (2003)
Antonyms

• A variant of our model that assigns dissimilar vectors to antonym pairs
Antonyms

• A variant of our model that assigns dissimilar vectors to antonym pairs

• For each word \( w \), compute \( V_w^{AP'} \) similarly to \( V_w^{SP} \), but using the set of antonym patterns

\[
V_w^{AP'} = V_w^{SP} - \beta \cdot V_w^{AP}
\]

❖ \( \beta \) is tuned using a development set
Experiments

- **Word similarity task**
  - Experiments with the SimLex999 dataset (Hill et al., 2014)
  - 999 word pairs, each assigned a similarity score by human annotators
  - $f_{\text{model}}(w_i, w_j) = \cos(V_{\text{model}}^w_i, V_{\text{model}}^w_j)$
  - Evaluation results is the Spearman’s $\rho$ score between model and human judgments
  - Numbers are average scores of 10 folds of 25% (dev) / 75 (test) partitions
  - Baselines: 2 interpretable baselines, 3 state-of-the-art, non-interpretable baselines

<table>
<thead>
<tr>
<th>Interpretable?</th>
<th>Model</th>
<th>Spearman’s $\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-interpretable</td>
<td>GloVe</td>
<td>0.426</td>
</tr>
<tr>
<td></td>
<td>CBOW</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>skip-gram</td>
<td>0.462</td>
</tr>
<tr>
<td>Interpretable</td>
<td>BOW</td>
<td>0.423</td>
</tr>
<tr>
<td></td>
<td>NNSE</td>
<td>0.455</td>
</tr>
<tr>
<td></td>
<td>SP(+)</td>
<td>0.502</td>
</tr>
</tbody>
</table>
Word similarity task

- Experiments with the SimLex999 dataset (Hill et al., 2014)
- 999 word pairs, each assigned a similarity score by human annotators
- $f_{model}(w_i, w_j) = \cos(V_{model}^w_i, V_{model}^w_j)$
- Evaluation results is the Spearman’s $\rho$ score between model and human judgments
- Numbers are average scores of 10 folds of 25% (dev) / 75 (test) partitions
- Baselines: 2 interpretable baselines, 3 state-of-the-art, non-interpretable baselines

<table>
<thead>
<tr>
<th>Interpretable?</th>
<th>Model</th>
<th>Spearman’s $\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-interpretable</td>
<td>GloVe</td>
<td>0.426</td>
</tr>
<tr>
<td></td>
<td>CBOW</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>skip-gram</td>
<td>0.462</td>
</tr>
<tr>
<td>Interpretable</td>
<td>BOW</td>
<td>0.423</td>
</tr>
<tr>
<td></td>
<td>NNSE</td>
<td>0.455</td>
</tr>
<tr>
<td></td>
<td>SP(+)</td>
<td>0.502</td>
</tr>
</tbody>
</table>
Experiments

• Word similarity task
  – Experiments with the SimLex999 dataset (Hill et al., 2014)
  – 999 word pairs, each assigned a similarity score by human annotators
  – \( f_{<\text{model}>}(w_i, w_j) = \cos(V_{<\text{model}>}_{w_i}, V_{<\text{model}>}_{w_j}) \)
  – Evaluation results is the Spearman’s \( \rho \) score between model and human judgments
  – Numbers are average scores of 10 folds of 25% (dev) / 75 (test) partitions
  – Baselines: 2 interpretable baselines, 3 state-of-the-art, non-interpretable baselines

<table>
<thead>
<tr>
<th>Interpretable?</th>
<th>Model</th>
<th>Spearman’s ( \rho )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-interpretable</td>
<td>GloVe</td>
<td>0.426</td>
</tr>
<tr>
<td></td>
<td>CBOW</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>skip-gram</td>
<td>0.462</td>
</tr>
<tr>
<td>Interpretable</td>
<td>BOW</td>
<td>0.423</td>
</tr>
<tr>
<td></td>
<td>NNSE</td>
<td>0.455</td>
</tr>
<tr>
<td></td>
<td>SP(^{(+)})</td>
<td>0.502</td>
</tr>
</tbody>
</table>
# Antonyms

<table>
<thead>
<tr>
<th>Word Pair</th>
<th>( SP )</th>
<th>( SG )</th>
</tr>
</thead>
<tbody>
<tr>
<td>new - old</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>narrow - wide</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>necessary - unnecessary</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>bottom - top</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>absence - presence</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>receive - send</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>fail - succeed</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>
Joint Model

\[ f_{\text{joint}}(w_i, w_j) = \gamma \cdot f_{SP}(w_i, w_j) + (1 - \gamma) \cdot f_{\text{skip-gram}}(w_i, w_j) \]

<table>
<thead>
<tr>
<th>Interpretable?</th>
<th>Model</th>
<th>Spearman’s $\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-interpretable</td>
<td>GloVe</td>
<td>0.426</td>
</tr>
<tr>
<td></td>
<td>CBOW</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>skip-gram</td>
<td>0.462</td>
</tr>
<tr>
<td>Interpretable</td>
<td>BOW</td>
<td>0.423</td>
</tr>
<tr>
<td></td>
<td>NNSE</td>
<td>0.455</td>
</tr>
<tr>
<td></td>
<td>SP(+)</td>
<td>0.502</td>
</tr>
<tr>
<td>Joint</td>
<td></td>
<td>0.528</td>
</tr>
<tr>
<td>Average Human Score</td>
<td></td>
<td>0.651</td>
</tr>
</tbody>
</table>

\(\gamma\) determined using a development set
Joint Model

\[ f_{joint}(w_i, w_j) = \gamma \cdot f_{SP}(w_i, w_j) + (1 - \gamma) \cdot f_{skip-gram}(w_i, w_j) \]

<table>
<thead>
<tr>
<th>Interpretable?</th>
<th>Model</th>
<th>Spearman’s ( \rho )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-interpretable</td>
<td>GloVe</td>
<td>0.426</td>
</tr>
<tr>
<td></td>
<td>CBOW</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td>skip-gram</td>
<td>0.462</td>
</tr>
<tr>
<td>Interpretable</td>
<td>BOW</td>
<td>0.423</td>
</tr>
<tr>
<td></td>
<td>NNSE</td>
<td>0.455</td>
</tr>
<tr>
<td></td>
<td>SP(+)</td>
<td>0.502</td>
</tr>
<tr>
<td>Joint</td>
<td></td>
<td>0.528</td>
</tr>
<tr>
<td>Average Human Score</td>
<td></td>
<td>0.651</td>
</tr>
</tbody>
</table>

- \( \gamma \) determined using a development set
Reminder: word2vec’s Skip-Gram Model
(Mikolov et al., 2013)

Objective function:

$$\max \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j}|w_t)$$
Integrating Symmetric Patterns into Existing NN Architectures?

A Deep Network Model + Symmetric Patterns
Integrating Symmetric Patterns into Existing NN Architectures?

A Deep Network Model + Symmetric Patterns

Pre-processing?
Enhance plain text with symmetric pattern information?

Acquiring Semantic Knowledge using Patterns
@ Roy Schwartz
Integrating Symmetric Patterns into Existing NN Architectures?

A Deep Network Model + Symmetric Patterns

Pre-processing?

Smarter Objective Function?

\[ \max \sum p(c/w) + \text{constraint} \]
Integrating Symmetric Patterns into Existing NN Architectures?

A Deep Network Model + Symmetric Patterns

Pre-processing? Smarter Objective Function?
Summary

• Patterns are useful for extracting semantic information

• Symmetric patterns are as useful (actually more useful) as state-of-the-art word embeddings in modeling word similarity
  – 4–7.9 points gap

• Patterns can capture relations that word embeddings cannot
  – Antonymy

• SPs can be combined along with state-of-the-art embeddings to create an even more accurate representation
  – 6.6 points higher than state-of-the-art
roys02@cs.huji.ac.il
http://www.cs.huji.ac.il/~roys02/