The role of word embeddings in Word Sense Disambiguation

Majid F. Sadi
M.Sc. Student, Zanjan university, Computer Department
Agenda

• Introduction
• Basic WSD concepts
• Deep Learning in WSD
• First Class
• Second Class
• Results
• Conclusion
Word Sense Disambiguation (WSD)

Human language is inherently ambiguous!

<table>
<thead>
<tr>
<th>English</th>
<th>Persian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank (n) :</td>
<td>آن یکی شیر است اندر بادیه آن دگر شیر است</td>
</tr>
<tr>
<td>1. Financial Institution</td>
<td>اندر بادیه آن یکی شیر است کآدم می خورد و آن دگر شیر است کآدم میخورد</td>
</tr>
<tr>
<td>2. Riverside</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Kind of Definition

Word Sense: One of the meanings a word may have depending on the context

- The man cashed a check at the bank.
- He sat on the bank of the river and watched the currents.

WSD: The process of automatically finding the correct sense of the polysemous words in a given text
Properties

WSD is difficult. It’s an AI-Complete task.

Ambiguity is everywhere, in different types and complexities:

<table>
<thead>
<tr>
<th>Term</th>
<th>Meaning</th>
<th>Spelling</th>
<th>Pronunciation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synophone</td>
<td>Different</td>
<td>Different</td>
<td>Similar but not identical</td>
</tr>
<tr>
<td>Synonym</td>
<td>Same</td>
<td>Different</td>
<td>Different</td>
</tr>
<tr>
<td>Polyseme</td>
<td>Different but related</td>
<td>Same</td>
<td>Same or different</td>
</tr>
<tr>
<td>Homophone</td>
<td>Different</td>
<td>Same or different</td>
<td>Same</td>
</tr>
<tr>
<td>Homonym</td>
<td>Different</td>
<td>Same</td>
<td>Same</td>
</tr>
<tr>
<td>Homograph</td>
<td>Different</td>
<td>Same</td>
<td>Same or different</td>
</tr>
<tr>
<td>Heteronym</td>
<td>Different</td>
<td>Same</td>
<td>Different</td>
</tr>
<tr>
<td>Heterograph</td>
<td>Different</td>
<td>Different</td>
<td>Same</td>
</tr>
<tr>
<td>Capitonym</td>
<td>Different when capitalized</td>
<td>Same except for capitalization</td>
<td>Same or different</td>
</tr>
</tbody>
</table>
Complex ambiguity example

Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo.
Applications

- Machine Translation
- Information Retrieval
- Word processing
- Information extraction and text mining
- Content and sentiment analysis
Approaches

WSD Approaches

Supervised: SVM, Naïve Bayes, etc.

Unsupervised: Clustering → WSI

Knowledge Based: LESK
Deep Learning application in Language domain

A neural language Probabilistic model [3]

The first use of word embedding

Goal: Predict the next word in a sequence.

Layers:
1. Embedding layer
2. One or more intermediate layer
3. Softmax Layer

Objective Function: \[ J_\theta = \frac{1}{T} \sum_{t=1}^{T} \log f (w_t, w_{t-1}, ..., w_{t-n+1}). \]
Deep Learning application in Language domain

The first use of word embedding: Predict the next word in a sequence.

Layers:
1. Embedding layer
2. One or more intermediate layer
3. Softmax Layer

Objective Function $J(f, w, \theta) = \sum - \log(P(y_t | \text{context}))$
Word representation:

1. Traditional models
   - Bag of words: Each word in vocabulary is represented with one bit in a huge vector.
     Ex: Hello is [00000010000000] in a vocabulary of size 15.
   - No contexts information

2. Word embeddings
   - Each word is represented as a point in a space with fixed number of dimensions
     Ex: Hello can be like [0.4, -0.11, 0.55, 1, .....]
\text{vector[Queen]} = \text{vector[King]} - \text{vector[Man]} + \text{vector[Woman]}
Word2vec [4]

1. Skip-Gram
   Goal: Predict surrounding words using given word
Word2vec

2. **CBOW**

   Goal: Predict current word given the context

   - **sparse representation**
   - **weights = distributed representation**
     - NB! Shared!
   - **softmax**
How to use ...

Word Embedding ≡ Word Vector ≡ Distributed Representation

• First way:
  Using word embeddings as a feature to a supervised WSD system
Word embedding as a feature
Word embedding as a feature

IMS_Embed [6]

• Using Embeddings as a feature to IMS system
• Trying different word embeddings from different tools
• Changing system parameters like window size and vector dimension
• Running experiment with and without default features
• Awesome results! They will be discussed at the end.
How to use ...

Big problem with word embeddings: They do not capture polysemy.

• Second way:

Train sense embeddings.
How to use ...

• Clustering-based Methods
  • Reisinger and Mooney (2010).

• Non-parametric Sense Embedding methods
  • Neelakantan et al. (2014)

• Ontology-based Methods
  • Sense2Vec, A. Trask, 2015
Evaluation

WSD generic tasks:

- All words
- Lexical sample

Baseline: Most frequent sense (first in WordNet), Lesk algorithm

**Exact Match** = \[100 \times \frac{\text{# exactly matched sense tags}}{\text{# assigned sense tags}}\]

**Precision** (\(p\)) = \[\frac{\text{# correct answers provided}}{\text{# answers to provide}} = \frac{|\{i \in \{1,..,n\} : A'(i) \in A(i)\}|}{|\{i \in \{1,..,n\} : A'(i) \neq \varepsilon\}|}\]

**Recall** (\(R\)) = \[\frac{\text{# correct answers provided}}{\text{# total answers to provide}} = \frac{|\{i \in \{1,..,n\} : A'(i) \in A(i)\}|}{|\{i \in \{1,..,n\} : A'(i) \neq \varepsilon\}|}\]

**Balanced F (\(F_1\))** = \[\frac{2PR}{P + R}\]
## Results

<table>
<thead>
<tr>
<th>System</th>
<th>SE2</th>
<th>SE3</th>
<th>SE7</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS (2010)</td>
<td>65.3</td>
<td>72.9</td>
<td>87.9</td>
</tr>
<tr>
<td>Taghipour and NG (2015)</td>
<td>66.2</td>
<td>73.4</td>
<td>-</td>
</tr>
<tr>
<td>AutoExtend (2015)</td>
<td>66.5</td>
<td>73.6</td>
<td>-</td>
</tr>
<tr>
<td>IMS + C &amp; W</td>
<td>64.3</td>
<td>70.1</td>
<td>88.0</td>
</tr>
<tr>
<td>IMS + Word2vec</td>
<td><strong>69.9</strong></td>
<td><strong>75.2</strong></td>
<td><strong>89.4</strong></td>
</tr>
<tr>
<td>IMS + Retrofitting</td>
<td>65.9</td>
<td>72.8</td>
<td>88.3</td>
</tr>
<tr>
<td>C &amp; W features only</td>
<td>55.0</td>
<td>61.6</td>
<td>83.4</td>
</tr>
<tr>
<td>Word2vec features only</td>
<td>65.6</td>
<td>69.4</td>
<td>87.0</td>
</tr>
<tr>
<td>Retrofitting features only</td>
<td>67.2</td>
<td>72.7</td>
<td>88.0</td>
</tr>
</tbody>
</table>

F1 performance on the three English lexical sample datasets [6]
### Results

<table>
<thead>
<tr>
<th>System</th>
<th>SE2</th>
<th>SE3</th>
<th>SE7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFS Baseline</td>
<td>71.6</td>
<td>70.3</td>
<td>65.8</td>
</tr>
<tr>
<td>Bubelfly</td>
<td>-</td>
<td>68.3</td>
<td>62.7</td>
</tr>
<tr>
<td>Muffin</td>
<td>-</td>
<td>-</td>
<td>66.0</td>
</tr>
<tr>
<td>Muffin + IMS</td>
<td>-</td>
<td>-</td>
<td>68.5</td>
</tr>
<tr>
<td>UBK w2w</td>
<td>-</td>
<td>65.3</td>
<td>56.0</td>
</tr>
<tr>
<td>IMS (pre-trained models)</td>
<td>77.5</td>
<td>74.0</td>
<td>66.5</td>
</tr>
<tr>
<td>IMS (SemCore)</td>
<td>73.0</td>
<td>70.8</td>
<td>64.2</td>
</tr>
<tr>
<td>IMS (OMSTI)</td>
<td>76.6</td>
<td>73.3</td>
<td>67.7</td>
</tr>
<tr>
<td>IMS + Word2vec (SemCor)</td>
<td>74.2</td>
<td>70.1</td>
<td>68.6</td>
</tr>
<tr>
<td>IMS + Word2vec (SemCor)</td>
<td><strong>77.7</strong></td>
<td><strong>74.1</strong></td>
<td><strong>71.5</strong></td>
</tr>
</tbody>
</table>

F1 performance in the nouns subsets of different all-words WSD datasets.[6]
Conclusion

- Word embeddings are not directly applicable to WSD tasks
- Word vectors are a robust feature to supervised WSD
- Sense vectors can be induced to help on Word Sense Induction
References


THANK YOU

Any Questions ?