Lexical Semantics 1: Word Senses and Word Similarity: WordNet and Vector Semantics

Adam Meyers
New York University
Outline

• Basics of Lexical Semantics
• Word Senses and WordNet
• Word Similarity using Word Embeddings
• Deep Learning
Lemmas and Wordforms

- **Lemma**: basic word form (paired with POS) used in lexicon
  - base form representing a set of inflected forms
    - Singular nouns: *book* → *book, books*
    - Bare infinitive verbs: *be* → *be, being, been, am, is, are, were, was*
    - Base adjective: *angry* → *angry, angrier, angriest*

- **Word form**: word how it actually occurs
  - a single wordform can be related to multiple lemmas:
    - *bases* is the plural of *basis* and *base*
    - *leaves* is the plural of *leaf* and *leave* (as in *leaves of absense*)
      - also present-tense 3rd Pers Sing form (VBZ) of the verb *leave*
    - A word form can be defined phonologically (different homophones)
      - /tu/ has 3 lemmas corresponding to *two, to and too*
    - A word form can be defined orthographically (different homographs)
      - Thus *does* corresponds to 2 lemmas with different pronunciations: (1) present 3rd person singular of the verb *do* and (2) plural of the noun *doe*
Senses

• Conventional Dictionaries and Thesauri map lemmas to sets of different meanings called senses

• Granularity of senses: a standard problem in lexicography
  – **merge** 2 senses together or **split** one sense into 2?

• Lets lookup *bank* in WordNet (Version 3), a thesaurus/dictionary that we will be featuring
  – Compare definitions 1 and 2
  – Compare definitions 2 and 9
    • Is it possible that these definitions should be merged together?
    • All organization names can stand in for buildings that house them
      – Thus 9 is predictable from 2 (by metonymy)
    • A single instance can have both “senses” at once:
      – *The bank on the corner hired 3 security guards*
WordNet Senses

• Conventional Dictionaries
  – Senses are defined informally as text
  – The relations between senses is not represented

• WordNet
  – a sense is defined by a Synset, the set of words that share that meaning. Intuition = define properties by extensions
  – Formal semantic definitions are often extensional
    • Both Frege and Russel's set-theoretic definitions of natural numbers
      – a number $N$ is defined as the set of sets with cardinality $N$
    • Many other definitions of meaning used in (computational) linguistics
  – Senses are hierarchical (connected by graph, edges represent IS-A)
    • Senses have hyponyms (sub-senses) and hypernyms (super-senses)
      – furniture $\supset$ seat $\supset$ chair $\supset$ recliner
    • The full graph allows multiple inheritance
      – and even cycles such as: \textit{restrain} $\supset$ \textit{inhibit} $\supset$ \textit{restrain} $\supset$ \textit{inhibit} ...
The Super Classes of shrimp

Entity
  ↓
Physical Entity
  ↘
Physical Object
  ↘
Whole Unit
  ↘
Living Thing
  ↘
Organism
  ↘
Animal
  ↘
Invertebrate
  ↘
Anthropod
  ↘
Crustacean
  ↘
Seafood
  ↘
Shrimp
Some CL uses for WordNet

• Sense disambiguation (multi-sense words only):
  – Given some text tagged with WN senses
  – Train a classifier that can automatically tag raw data
  – Issue: fine-grained senses are difficult to tag (manually or automatically)
    • Many approaches collapse several WordNet senses together

• Calculating Semantic Similarity
  – Some models of similarity of meaning are based on distance in the sense hierarchy
  – Issue: same length paths do not reflect same similarity
    • distinctions are based on available information, not carefully calculated similarity distances
    • cruder relative path distances are used or path distances are combined with other information, e.g., similarity in text (in terms of n-grams)
Distributional Methods for Sense Disambiguation and Word Similarity

• ML with features based on N-grams (sequences of neighboring tokens), dependencies (relations with words in parse trees), etc.
• Word Embeddings
  – Represent each word by a large vector indicating the words that occur in any of the same sentences in some corpus.
  – Similarity calculated using similarity between vectors
    • e.g., cosine similarity
    • similarity between other words measures word similarity
    • similarity between instance and automatically or manually annotated synset instance can indicate sense
Word Word Matrix Using Pointwise Mutual Information

- Word Word Matrix (aka *word embedding*)
  - Rows represent word \( R \)
  - Columns (aka *dimensions*) represent words co-occurring with word \( C \)
  - Can be generalized to multi-words (n-grams, phrases, \( \ldots \))
    - word to multi-word
    - multi-word to multi-word
  - Context can be defined other ways, e.g., proximity in syntactic tree

- Approximation of meaning: “A word is defined by the company it keeps” (Firth)

- Scores in Matrix
  - How related is word \( R \) to word \( C \) represented by column \( C \)
  - Pointwise Mutual Information

\[
PMI = \log \left( \frac{\text{prob}(\text{word}_R, \text{word}_C)}{\text{prob}(\text{word}_R) \times \text{prob}(\text{word}_C)} \right)
\]
Modifications to PMI

• Negative values should be treated as 0
• PMI is high for low frequency words
  – *banana* occurs once in the corpus of 1K words
  – *face* occurs twice in that corpus
  – *Banana face* occurs once in that corpus
  – \( \text{PMI}(\text{banana}, \text{face}) = \log_2\left(\frac{.5}{.001 \times .002}\right) = 12.42 \)
  – Smoothing – different methods that raise the denominator slightly which offset this effect
  – Example: La Place – add a small constant to all e.g., add 1 (banana = 2, banana face = 2, face = 3)
    – \( \text{PMI}(\text{banana}, \text{face}) = \log_2\left(\frac{.667}{.002 \times .003}\right) = 11.6 \)
Sample Word Embedding 1

• Assume a “bag of words” approach
  – Order of words don't matter
  – Other types of embeddings (e.g., skip grams) model word order as well
  – Assume that words are stemmed
• Use words in a window of $K$ words before and $K$ words after word $R$
• Let's assume $K = 5$ (for this example)
• Eliminate stop words and high frequency (low IDF) words
• Use integers in vectors (scores usually between 0 and 1)
Sample Word Embedding 2
From Hypothetical Recipe Corpus

- Rows = words being classified
- Columns = words in context
- Higher number indicates more likely to be in window of +/- 5 from word labeling row

<table>
<thead>
<tr>
<th></th>
<th>cup</th>
<th>ounce</th>
<th>taste</th>
<th>chicken</th>
<th>stir</th>
<th>bake</th>
<th>chocolate</th>
</tr>
</thead>
<tbody>
<tr>
<td>beef</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>cabbage</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>lemon</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>parsley</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>pepper</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>salt</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>sugar</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>
### Cosine similarity for Word Vectors from Previous Slide

<table>
<thead>
<tr>
<th></th>
<th>beef</th>
<th>cabbage</th>
<th>lemon</th>
<th>parsley</th>
<th>pepper</th>
<th>salt</th>
<th>sugar</th>
</tr>
</thead>
<tbody>
<tr>
<td>beef</td>
<td>1</td>
<td>.63</td>
<td>.54</td>
<td>.57</td>
<td>.72</td>
<td>.66</td>
<td>.41</td>
</tr>
<tr>
<td>cabbage</td>
<td>.63</td>
<td>1</td>
<td>.25</td>
<td>.51</td>
<td>.53</td>
<td>.58</td>
<td>.51</td>
</tr>
<tr>
<td>lemon</td>
<td>.54</td>
<td>.25</td>
<td>1</td>
<td>.86</td>
<td>.64</td>
<td>.68</td>
<td>.74</td>
</tr>
<tr>
<td>parsley</td>
<td>.57</td>
<td>.51</td>
<td>.86</td>
<td>1</td>
<td>.81</td>
<td>.86</td>
<td>.69</td>
</tr>
<tr>
<td>pepper</td>
<td>.72</td>
<td>.53</td>
<td>.64</td>
<td>.81</td>
<td>1</td>
<td>.97</td>
<td>.44</td>
</tr>
<tr>
<td>salt</td>
<td>.66</td>
<td>.58</td>
<td>.68</td>
<td>.86</td>
<td>.97</td>
<td>1</td>
<td>.56</td>
</tr>
<tr>
<td>sugar</td>
<td>.41</td>
<td>.51</td>
<td>.74</td>
<td>.69</td>
<td>.44</td>
<td>.56</td>
<td>1</td>
</tr>
</tbody>
</table>
Demo for find similar words

Word Sense Disambiguation

• Demo of A word sense diambiguator demo
  – [http://www.ling.gu.se/~lager/Home/pwe_ui.html](http://www.ling.gu.se/~lager/Home/pwe_ui.html)

• Shared tasks include Semcor
  – [http://web.eecs.umich.edu/~mihalcea/downloads.html#semcor](http://web.eecs.umich.edu/~mihalcea/downloads.html#semcor)

• Using Word Vectors for Word Sense Disambiguation
  – Vectors represent word senses rather than words
    • Need sense annotated corpus
  – Create vectors for words in new text
  – Compute similarity of words in new text with sense vectors and choose most similar sense
Paraphrase and Entailment

- SemEval Text Similarity Task:
  - 2016 (Task 1)
  - [http://alt.qcri.org/semeval2014/task1/](http://alt.qcri.org/semeval2014/task1/) (webpage)

- Input pairs of text “snippets”
  - English/English (like previous year tasks)
  - Spanish/English pairs (innovation for 2016)

  - previous snippets, with one member of pair translated

- System produces score from 0 to 5 indicating similarity
- Manually tagged data (test, dev, training sets)
- Data collection of snippets based on heuristics and manually annotate

  - One heuristic is based on word embedding similarity – embedding of sentence = sum of the embeddings of words
Human Judge similarity 0 to 5 (from Agirre et al 2016)

- 5 mean exactly the same thing
  - *The bird is in the sink* ↔ *Birdie is washing itself in the water basin*
- 4 mostly the same, differences unimportant
  - *In May 2010, the troops attempted to invade Kabul* ≈ *The US army invaded Kabul on May 7th last year, 2010*
- 3 roughly same with important differences/omissions
  - *John said he is considered a witness but not a suspect* → *“He is not a suspect anymore.” John said*
- 2 same topic, share some details
  - *They flew out of the nest in groups* ≠ *They flew out of the nest together*
- 1 same topic
  - *The woman is playing the violin* ≠ *The young lady enjoys listening to guitar*
- 0 disimilar
  - *John went horse back riding with a whole group of friends* ≠ *Sunrise at dawn is a magnificent view to take in if you wake up early enough for it*
Evaluation

• Systems scored by the Pearson correlation between their scores and the Manual Annotation
• Samsung's system got the highest score: .7781
• I looked at papers about the top 3 systems
  – All used word embeddings in one form or another
Top System (Samsung) used Word Embeddings

• Vectors contained words & multi-word phrases
  – Methods for combining embeddings of words into embeddings of sentences
• Used other features, e.g., from WordNet
• Used dependency parses of snippets
• Machine Learning Algorithms (e.g., SVM)
  – To predict 0 to 5 Textual Similarity Score
  – Features include cosine similarity of roots of parses
    • Similarity derived by combining children similarities according to an algorithm
• Most top systems used Word Embeddings
Real Vectors have Many Dimensions

• Preceding “toy” examples use few dimensions
• Vectors often have tens of thousands of dimensions, e.g., there can be thousands of possible co-occurring words
• More dimensions
  – Better output (higher recall and precision)
  – Slower speed (e.g., takes longer to compute similarity)
• Large Vectors are sparse (lots of zeros)
• Context: window of 3 to 17 (or the whole sentence)
• Reducing dimensions to make smaller, less sparse vectors
  – Capture Generalizations, more efficient processing, etc.
  – One such method is called Latent Semantic Analysis
  – Many other methods for refining vector-based analyses
Latent Semantic Analysis: Reducing Dimensions

- Original 2-D Vector
- Rotate/Move so points are closer to the X and Y axes
- Eliminate one dimension
Other factors

• Softmax functions: functions that normalize a range of values from 0 to 1, so they can be used as probabilities
• Eliminating dimensions that do not discriminate between vectors, high/low frequency words, words with low IDF, etc.
• Bag of Words Feature (so far)
• Skip Gram Features (include proximity of words)
• Adding more features
Word Embeddings using Deep Learning, aka, Neural Networks

- **Warning:** Summarizing complex process in 3 slides
- **Goal:** derive predictive function from input vector output vector
  - represents goal for learning process
  - input vector encodes features, e.g., word, base form, previous word, following word, POS, previous POS, etc.
  - output vector = estimated probabilities of classifications. e.g., \( P(\text{instance of } \text{bank}) \) is **bank sense 1**
- **Intermediate vectors** (hidden layers)
  - fixed number of dimensions (typically 50–1000)
  - initially random parameters for formulas linking all dimensions in input to all dimensions in layer 1 to all dimensions in layer 2, ..., to all dimensions in output
  - eventually represent interdependencies between features
  - weights updated during training on randomly selected pieces of the data and the predictive function gradually improves at predicting the output correctly
Deep Learning Network

- Red = Input, Green = Output, Edges labeled with changing parameters
Training in Deep Learning

- Randomly sample data in batches
- Keep updating parameters used by the predictive function
- For each batch, change the parameters that set values in the hidden vectors
  - change parameter to make it closer to the optimal values
    - If incorrect prediction, big change
- Stop at some point:
  - convergence: weights stop changing much
  - fixed number of iterations
- Resulting function to predicts output class for new data
  - Map end result into probabilities (using function like softmax)
- Similarity calculated same way as with PMI or TFIDF vectors, e.g., cosign similarity
Summary

- **WordNet**
  - Resource that details senses of words and relations between word senses (an is-a ontology)
  - Some annotated data available based on WordNet
- **Vector characterization of words (word embeddings)**
  - Dimensions represent words in context within a window
  - Related words/word-senses/translations/etc. have similar embeddings
- Dimensions are weighted using TF-IDF, PMI and other metrics
- Similarity is calculated with Cosine Similarity, Jaccard similarity, …
- Real systems use large sparse vectors which are converted into smaller dense vectors, using various “deep learning” methods
- These methods use “hidden layers” to randomly induce dependencies between features. These methods tend to get higher scores than previous ML methods, although it may be difficult to figure out the dependencies that are used (or why they are predictive).
Readings on WordNet and Word Similarity

• J & M: Chapter 19.1, 19.2 and 19.3
• WordNet
  – Read the first 2 papers found here:
    • http://wordnetcode.princeton.edu/5papers.pdf
  – Read NLTK section 2.5 and try the NLTK WordNet module
• Optional Readings Throughout Slides
Deep Learning at NYU

• Machine Translation
  – Prof. Kyunghyun Cho (http://www.kyunghyuncho.me/)

• Natural Language Semantics
  – Prof. Sam Bowman (https://www.nyu.edu/projects/bowman/)

• ACE Event Detection
  – Thien Nguyen (http://www.cs.nyu.edu/~thien/)

• And Others
Deep Learning Documentation (Reading Optional) and Code

- Jurafsky and Martin 3rd Edition (Chapters 15 and 16)
- Word2Vec
  - https://www.tensorflow.org/versions/r0.12/tutorials/word2vec/index.html
  - https://deeplearning4j.org/word2vec
  - https://github.com/dav/word2vec