Polysemy Resolution in Word Embedding

Sanchit Agrawal   Ishu Dharmendra Garg
Ujjawal Soni   Shreyas Harish

Department of Computer Science and Engineering
Indian Institute of Technology Madras, India

NLP July-Nov 2016

Outline

Word Embeddings

- Word Embeddings are mapping that map a word w from a vocabulary to a real-valued vector $\vec{w}$ in an embedding space of dimensionality relatively less than vocabulary size.
- The two categories of methods to generate Word Embeddings are:
  - Count based
    Example: Latent Sematic Analysis
  - Prediction based
    Example: Neural Network Language Model (NNLM), word2vec
- These embeddings boost the performance in NLP tasks when they are used as the underlying input representation.

Problem of Polysemous Words

- Most words are polysemous in language like English, i.e. having multiple meanings. Example:
  - light - electromagnetic waves or to set on fire or the opposite of heavy
  - bank - a riverside or a financial institution
- In most methods, like word2vec, each word is represented by a single vector.
- Quality of the embeddings will be detrimentally affected as different meanings of a word are conflated into a single representation.
- We cannot reasonably hope that a single vector can effectively capture the correct meaning in all the contexts.

Proposed Solution

- We propose to learn vectors for senses (i.e. meanings) instead of words.
- Various senses of the words can be enumerated using the synsets defined in knowledge graphs like WordNet.
- Then these sense vectors can be used in other tasks as follows:
  - If the appropriate sense of the word is known, or can be found using WSD, use the sense vector.
  - If an appropriate relation can be found that maps words to appropriate senses, then those vectors can be used.
  - Create a word vector as the weighted average of its sense vectors if the sense cannot be ascertained.

Proposed Methods

Learning Sense Embedding

We propose two methods to learn Sense Embeddings:

- Method 1: WSD on text corpus
  - Obtain a large sense-annotated text corpus by running WSD
  - Run word2vec on the corpus to obtain sense vectors
- Method 2: DeepWalk on knowledge graph
  - Perform several random walks on a knowledge graph like WordNet
  - For each random walk, store the vertices visited in the knowledge graph
  - Run word2vec on the corpus to obtain sense vectors

Considering advantages and disadvantages, and experimenting with them, we decided to finally pursue the DeepWalk approach.

Advantages and Disadvantages of Method 1

Learning Sense Embedding

Advantages:

- Exploits the knowledge hidden in very large corpora.
- Can be applied well to limited technical domains where WSD is easy to perform (on manual and books)
- Can easily generalize to new languages for which corpora and WSD are available

Disadvantages:

- WSD is hard. Having it a sub-task to our goal makes it considerably harder and likelier to affect the quality of embeddings
- Even the best WSD techniques are unable to disambiguate a large fraction of words for a given context
- WSD techniques frequently assign an incorrect sense to a word

Details of Method 2: DeepWalk on knowledge graph

Learning Sense Embedding

Generate a corpora using the following random walk algorithm on knowledge graph (WordNet 3.0 in our experiments):

- Randomly choosing a vertex from the graph
- Move to a random neighbour with a probability α ($= 0.15$ for our experiments) or else terminate the walk
- We create an artificial corpus of large number ($10,000,000$ for our experiments) such contexts

. Each context in this artificial generated corpus is a list of sense ids. We run word2vec on the generated corpora to obtain the sense vectors.

Advantages and Disadvantages of Method 2

Learning Sense Embedding

Advantages:

- Helps bypassing the WSD step
- Exploits the information of expertly curated knowledge graphs
- Can potentially cover all human knowledge (use Wikipedia)
- Captures better semantics since similar synsets appear together

Disadvantages:

- Doesn't distinguish fine senses of similar words, lexical derivatives etc
- Cannot generalize easily to new languages
- May not be appropriate for technical domains, since terms have very specific connotations
- Fails to capture the lexical & grammatical relations of the language

Results
Word Similarity

	MEN	RG-65	WS-353-Sim
word2vec	14.1588	1.2306	2.8249
sense embeddings	10.6590	0.9055	1.8451
hybrid model	10.6453	0.8031	1.8409

Table: RMS error on various datasets (lower is better)

	MEN	RG-65	WS-353-Sim
word2vec	0.3907	0.4545	0.4946
sense embeddings	0.6423	0.8142	0.7144
hybrid model	0.6489	0.8310	0.7220

Table: Pearson correlation on various datasets (higher is better)

Results

Word Relatedness

	WS-353-Rel
word2vec	3.2309
sense embeddings	2.3684
hybrid model	2.3346

Table: RMS error on various datasets (lower is better)

	WS-353-Rel
word2vec	0.2254
sense embeddings	0.4584
hybrid model	0.4647

Table: Pearson correlation on various datasets (higher is better)

Visualizing word2vec Embeddings

Polysemous word - 'bank'

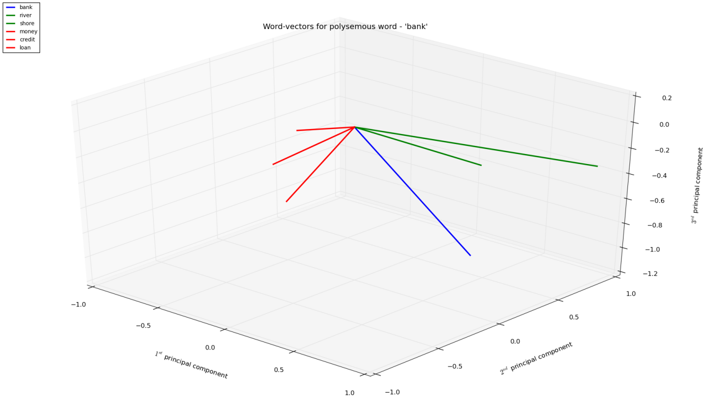

Figure: "bank" vector plotted with its senses

Visualizing word2vec Embeddings

Polysemous word - 'wood'

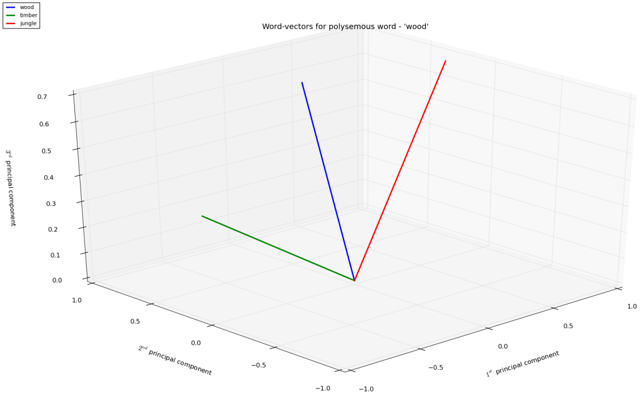

Figure: "wood" vector plotted with its 2 senses

Visualizing Sense Embeddings

Polysemous word - 'bank'

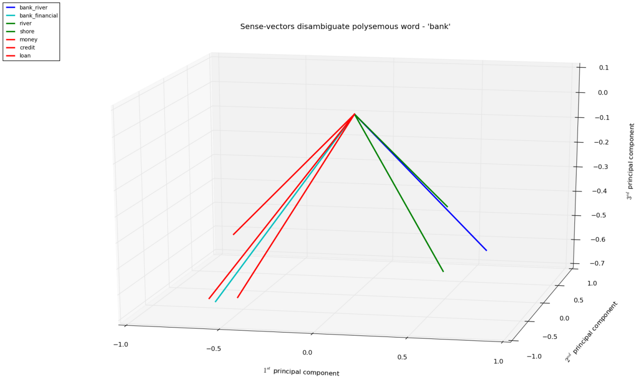

Figure: Disambiguating the senses of "bank"

Visualizing Sense Embeddings

Polysemous word - 'wood'

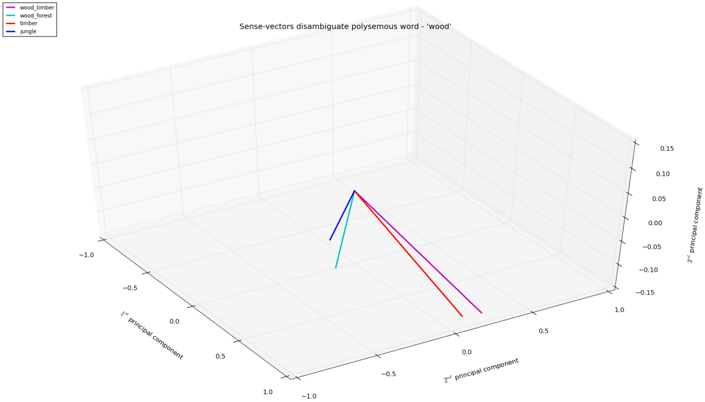

Figure: Disambiguating the senses of "wood"

Visualizing Sense Embeddings

Visualizing word relatedness

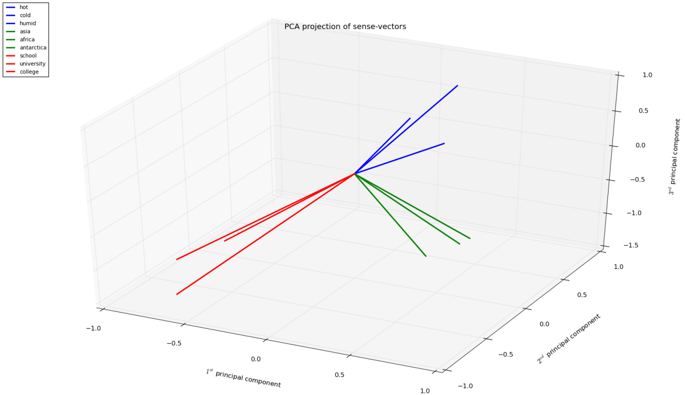

Figure: Relatedness between weathers, continents, and educational institutes

Visualizing Sense Embeddings
Visualizing word relatedness

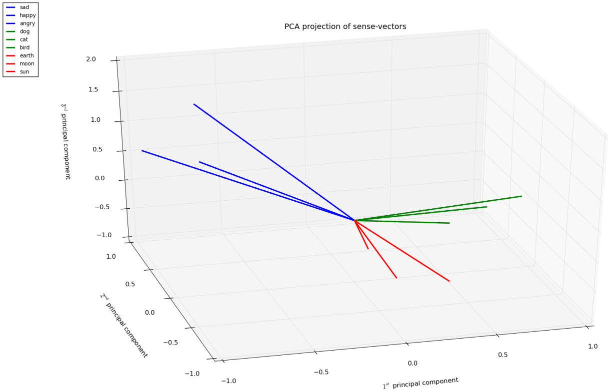

Figure: Relatedness between moods, animals, and celestial bodies

Visualizing Sense Embeddings

Visualizing word relatedness

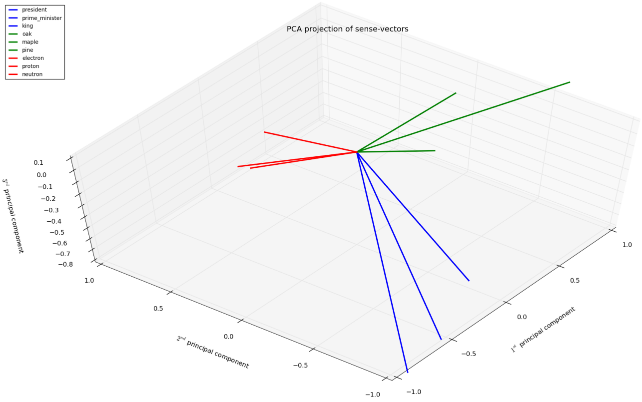

Figure: Relatedness between positions of power, trees, and atomic particles