Distributional Semantics

Carlos Ramisch

Aix*Marseille université

AIM-WEST workshop, October 20-21, 2015

How can one represent the meaning of a word?

- Tags
- Synsets
- Ontologies
- Logic predicates
- ...

Princeton's WordNet

- Lexical units grouped by **synsets**
- Relations between words and synsets: antonymy, synonymy, hypernymy, etc
- Contains POS, glosses, examples, frequency, etc

WordNet online

`http://wordnet.princeton.edu/`

Exercise: WordNet lookup

1. Go to `http://wordnet.princeton.edu/`
2. Click on "Use WordNet Online"
3. Look for the lexical unit **mouse**
4. Click on "S:" on the first sense and explore the relations
5. Look for the lexical unit **pick up**
6. Click on "S:" on the first sense and explore the relations
7. Select "Display options" - "Show all" to see more information

Good to know

WordNet is free, and has a simple interface in Ubuntu package:

```
sudo apt-get install wordnet
wn mouse -synsn
```

Problems

- Wordnet has limited coverage
- Wordnets are expensive
- Granularity
- Similarity vs Relatedness

Outline

Distributional Hypothesis

"You shall know a word by the company it keeps" - Firth (1957)

- Words which are similar in meaning occur in similar contexts
- Similar words have similar distributions
- Context provides meaning

Example

What is a *winyby*?

- *They had the best winyby in town*
- *I bought some winyby for our party*
- *One bottle of red wine and one bottle of winyby, please*
- *We are hangover, we drank too much winyby last night*
- *The winyby distillery was founded in the 14th century*

Example

What is a *winyby*?

- *They had the best winyby in town*
- *I bought some winyby for our party*
- *One bottle of red wine and one bottle of winyby, please*
- *We are hangover, we drank too much winyby last night*
- *The winyby distillery was founded in the 14th century*

Example

What is a *winyby*?

- *They had the best winyby in town*
- *I bought some winyby for our party*
- *One bottle of red wine and one bottle of winyby, please*
- *We are hangover, we drank too much winyby last night*
- *The winyby distillery was founded in the 14th century*

Example

What is a *winyby*?

- *They had the best winyby in town*
- *I bought some winyby for our party*
- *One bottle of red wine and one bottle of winyby, please*
- *We are hangover, we drank too much winyby last night*
- *The winyby distillery was founded in the 14th century*

Example

What is a *winyby*?

- *They had the best winyby in town*
- *I bought some winyby for our party*
- *One bottle of red wine and one bottle of winyby, please*
- *We are hangover, we drank too much winyby last night*
- *The winyby distillery was founded in the 14th century*

Example

What is a *winyby*?

- *They had the best winyby in town*
- *I bought some winyby for our party*
- *One bottle of red wine and one bottle of winyby, please*
- *We are hangover, we drank too much winyby last night*
- *The winyby distillery was founded in the 14th century*

Distributional Semantic Models (DSMs)

- Contexts can be represented as sparse vectors
- Cells are co-occurrence counts
- Similarity is the cosine between vectors
- Similar to IR boolean model (TF/IDF)
- Also called Vector-Space Models

Example

	get	see	use	hear	eat	kill
knife	51	20	84	0	3	0
cat	52	58	4	4	6	26
dog	115	83	10	42	33	17
boat	59	39	23	4	0	0
cup	98	14	6	2	1	0
pig	12	17	3	2	9	27
banana	11	2	2	0	18	0

Source: Stefan Evert's 2010 NAACL tutorial on DSMs

DSM parameters

- Corpus preprocessing
- Context definition
- Dimensionality reduction
- Association scores
- Similarity scores
- Evaluation

Corpus preprocessing

- Sentence splitting
- Tokenisation
- POS tagging
- MWE annotation
- Syntactic parsing

Context definition

- Document, paragraph, sentence
- Sliding window vs. syntactic dependencies
- Content words or function words
- Number of words to the left/right of target
- Context weighting (proportional to distance)

Dimensionality reduction

- Cooccurrence matrix is sparse
- Principal Component Analysis
- Singular Value Decomposition
- Context filtering
  - Keep most frequent p contexts
  - Keep contexts occurring more than th times

Association scores

How to distinguish significant from accidental cooccurrence?

- Many association scores exist: LL, dice, t-score, pmi, Fisher, etc.

Most famous one: PMI

$$PMI(w_1, w_2) = \log \frac{Nc(w_1, w_2)}{c(w_1)c(w_2)}$$

Similarity scores

How do you compare context vectors?

- Geometric interpretation
- Distance vs similarity
- Normalization
- Cosine similarity

$$cos(\overrightarrow{w_1}, \overrightarrow{w_2}) = \frac{\sum_{i=1}^{n} w_1^i \times w_2^i}{||\overrightarrow{w_1}|| \times ||\overrightarrow{w_2}||}$$

Evaluation

- Correlation with human scores
  Pearson, Spearman, Kendall-τ-b
- Compare with traditional thesauri
- Precision/recall/f1

Applications

- Similarity
- Word clustering
- Lexical substitution
- Textual entailment
- Paraphrasing
- Lexicon enrichment

Word embeddings

- Another way of creating vectors for words
- Based in neural networks that "predict" the target based on context (or vice-versa)
- Layer of network becomes a word's "embedding"
- Famous implementation: word2vec

Outline

Minimantics

Minimal distributional semantics
`https://github.com/ceramisch/minimantics`

```
minimantics/src
make
cd ../..
```

Step 1: Select target words

1. Select the words in the WS353 dataset

```
cut -f1 wordsim353.tsv |
tail -n +2 |       # remove header
sed 's/,/\n/g' |   # 1 word per line
sed 's/./\L&/g' | # lowercase
sort |             # remove duplicates
uniq > targets.txt
```

2. Verify that the result contains 437 words.

Step 2: Extract context-target pairs

1. Extract all pairs of words co-occurring in a window of 4 words

```
zcat corpus/01.gz corpus/02.gz |
./extract-pairs.py targets.txt 4 | # takes a while...
gzip > targets-contexts.gz
```

Steps 3: filter contexts

- Keep top-500 contexts sorted by descending frequency

```
zcat targets-contexts.gz |
grep -v " 1$" |                    # remove hapax
sort -t " " -k 1,1 -k 3,3nr |  # TAB separator
awk -f thresh-fk.awk THRESHOLD=500 |
cat > targets-contexts-f.txt
```

Steps 4: measure association

- Calculate association measures between targets and relevant contexts

```
minimantics/src/build_profiles targets-contexts-f.txt |
cat > profiles-filter500.txt
```

Step 5: Compute similarity

1. Compute all target-target similarities

```
minimantics/src/calculate_similarity -T 8 -a pmi \
          -s cosine -t targets.txt -n targets.txt \
          profiles-filter500.txt > targets-sim.txt
```

2. Merge WS353 dataset and target-target matrix

```
./eval-ws.py wordsim353.tsv targets-sim.txt |
cat > wordsim353-system.tsv
```

Step 6: Evaluate

- Compare ranks of human similarity scores and system scores

```
./minimantics/scripts/csv_evalrank.py \
            --gold-threshold=5 wordsim353.tsv \
            wordsim353-system.tsv
```

Questions?