Distributional Hypothesis

- Zellig Harris: words that occur in the same contexts tend to have similar meanings
- Firth: a word is known (characterized) by the company it keeps
- Statistical basis for lexical semantics
- How can we learn computational representations of words
  - Representational learning: unsupervised
  - Contrast with feature engineering

Lemmas and Senses

- Lemma or citation form: general form of a word (e.g., mouse)
  - May have multiple senses
  - May come in multiple parts of speech
  - May cover variants (*word forms*) such as for plurals, gender, ...
- Homonymous lemmas
  - With multiple senses
  - Challenges in word sense disambiguation
- Principle of contrast: difference in form indicates difference in meaning

Synonyms and Antonyms

- Synonyms: Words with identical meanings
  - Interchangeable without affecting *propositional* meaning
  - Are there any true synonyms?
- Antonyms: Words with opposite meanings
  - Opposite ends of a scale
  - Antonyms would be more similar than different
- Reversives: subclass of antonyms
  - Movement in opposite directions, e.g., rise versus fall

Word Similarity

Crucial for solving many important NL tasks

- Similarity: Ask people
- Relatedness ≈ association in psychology, e.g., coffee and cup
  - Semantic field: domain, e.g., surgery
  - Indicates relatedness, e.g., surgeon and scalpel

Vector Space Model

Foundation of information retrieval since early 1960s

- Term-document matrix
  - A row for each word (term)
  - A column for each document
  - Each cell being the number of occurrences
  - Dimensions
    - Number of possible words in the corpus, e.g., $\approx [10^4, 10^5]$
    - Size of corpus, i.e., number of documents: highly variable (small, if you talk only of Shakespeare; medium, if New York Times; large, if Wikipedia or Yelp reviews)
- The vectors (distributions of words) provide some insight into the content even though they lose word order and grammatical structure

Document Vectors and Word Vectors

- Document vector: each column vector represents a document
  - The document vectors are sparse
  - Each vector is a point in the 10^5 dimensional space (one dimension per word)
- Word vector: each row vector represents a word
  - Better extracted from another matrix

Word-Word Matrix

- $|V| \times |V|$ matrix
  - Each row and column: a word
  - Each cell: number of times the row word appears in the *context* of the column word
  - The context could be
    - Entire document $\Rightarrow$ co-occurrence in a document
    - Sliding window (e.g., ± 4 words) $\Rightarrow$ co-occurrence in the window

Measuring Similarity

- Inner product $\equiv$ dot product: Addition of element-wise products

$$\vec{v} \cdot \vec{w} = \sum_i v_i w_i$$

  - Highest for similar vectors
  - Zero for orthogonal (dissimilar) vectors
- Inner product is biased by vector length

$$|\vec{v}| = \sqrt{\sum_i v_i^2}$$

- Cosine of the vectors: Inner product divided by length of each

$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}||\vec{w}|}$$

- Normalize to unit length vectors if length doesn't matter
  - Cosine $=$ inner product (when normalized for length)
  - Not suitable for applications based on clustering, for example

TF-IDF: Term Frequency–Inverse Document Frequency

Basis of relevance; used in information retrieval

- TF: higher frequency indicates higher relevance

$$\mathrm{tf}_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{count}(t, d) & \text{if count(t, d) is positive} \\ 0 & \text{otherwise} \end{cases}$$

- IDF: terms that occur selectively are more valuable when they do occur

$$\mathrm{idf}_t = \log_{10} \frac{N}{\mathrm{df}_t}$$

  - N is the total number of documents in the corpus
  - df_t is the number of occurrences in which t occurs
- TF-IDF weight

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

  - These weights become the vector elements

Applying TF-IDF Vectors

- Word similarity as cosine of their vectors
- Define a document vector as the mean (centroid)

$$d_D = \frac{\sum_{t \in D} \vec{w}_t}{|D|}$$

  - D: document
  - w_t: TF-IDF vector for term t

Pointwise Mutual Information (PMI)

How often two words co-occur relative to if they were independent

- For a target word w and a context word c

$$\mathrm{PMI}(w,c) = \lg \frac{P(w,c)}{P(w)P(c)}$$

  - Negative: less often than naïvely expected by chance
  - Zero: exactly as naïvely expected by chance
  - Positive: more often than naïvely expected by chance
- Not feasible to estimate for low values
  - If $P(w) = P(c) = 10^{-6}$, is $P(w,c) \geq 10^{-12}$?
- PPMI: Positive PMI

$$\mathrm{PPMI}(w_i, c_j) = \max(\lg \frac{P(w,c)}{P(w)P(c)}, 0)$$

Estimating PPMI: Positive Pointwise Mutual Information

- Given co-occurrence matrix $F = W \times C$, estimate cells

$$p_{ij} = \frac{f_{ij}}{\sum_i^W \sum_j^C f_{ij}}$$

- Sum across columns to get a word's frequency

$$p_{i*} = \sum_j^C p_{ij}$$

- Sum across rows to get a context's frequency

$$p_{*j} = \sum_i^W p_{ij}$$

- Plug in these estimates into the PPMI definition

$$\text{PPMI}(w, c) = \max(\lg \frac{p_{ij}}{p_{i*} \times p_{*j}}, 0)$$

Correcting PPMI's Bias

- PPMI is biased: gives high values to rare words
  - Replace $P(c)$ by $P_\alpha(c)$

$$P_\alpha(c) = \frac{\text{count}(c)^\alpha}{\sum_d \text{count}(d)^\alpha}$$

- Heuristically suggested $\alpha = 0.75$
- Improved definition for PPMI

$$\text{PPMI}(w, c) = \max(\lg \frac{P(w, c)}{P(w)P_\alpha(c)}, 0)$$

Word2Vec

- TF-IDF vectors are long and sparse
- How can we achieve short and dense vectors?
  - 50–500 dimensions
  - Dimensions of 100 and 300 are common
- Easier to learn on: fewer parameters
- Superior generalization and avoidance of overfitting
  - Better for synonymy since the words aren't themselves the dimensions

Skip Gram with Negative Sampling

Representation learning

- Instead of counting co-occurrence
- Train a classifier on a binary task: whether a word w will co-occur with another word v ($\approx$ context)
- Implicit supervision—gold standard for free!
  - If we observe that v and w co-occur, then that's a positive label for the above classifier
  - A target word and a context word are positive examples
  - Other words, which don't occur in the target's context, are negative examples
- With a context window of ± 2 ($c_{1:4}$), consider this snippet

. . . lemon,	a	tablespoon	of	apricot	jam,	a	pinch of . . .
		c_1	c_2	t	c_3	c_4	

  - Estimate probability $P(\text{yes}|t, c)$

Skip Gram Probability Estimation

- Intuition: $P(\text{yes}|t, c) \propto \text{similarity}(t, c)$
  - That is, the embeddings of co-occurring words are similar vectors
- Similarity is given by inner product, which is not a probability distribution
- Transform via sigmoid

$$P(\text{yes}|t, c) = \frac{1}{1 + e^{-t \cdot c}}$$

$$P(\text{no}|t, c) = \frac{e^{-t \cdot c}}{1 + e^{-t \cdot c}}$$

- Naïve (but effective) assumption that context words are mutually independent

$$P(\text{yes}|t, c_{1:k}) = \prod_{i=1}^{k} \frac{1}{1 + e^{-t \cdot c_i}}$$

Learning Skip Gram Embeddings

- Positive examples from the window
- Negative examples couple the target word with a random word ($\neq$ target)
- Number of negative samples controlled by a parameter
- Probability of selecting a random word from the lexicon
  - Uniform
  - Proportional to frequency: won't hit rarer words a lot
  - Discounted as in the PPMI calculations, with $\alpha = 0.75$

$$P_\alpha(w) = \frac{\text{count}(w)^\alpha}{\sum_v \text{count}(v)^\alpha}$$

- Maximize similarity with positive examples
- Minimize similarity with negative examples
  - Maximize and minimize inner products, respectively

Learning Skip Gram Embeddings by Gradient Descent

- Two concurrent representations for each word
  - As target
  - As context
- Randomly initialize W (each column is a target) and C (each row is a context) matrices
- Iteratively, update W and C to increase similarity for target-context pairs and reduce similarity for target-noise pairs
- At the end, do any of these
  - Discard C
  - Sum or average W^T and C
  - Concatenate vectors for each word from W and C
- Complexity increases with size of context and number of noise words considered

CBOW: Continuous Bag of Words

Alternative formulation and architecture to skip gram

- Skip gram: maximize classification of words given nearby words
  - Predict the context
- CBOW
  - Classify the middle word given the context
- CBOW versus skip gram
  - CBOW is faster to train
  - CBOW is better on frequent words
  - CBOW requires more data

Semantic Properties of Embeddings

Semantics $\approx$ meaning

- Context window size
  - Shorter: immediate context $\Rightarrow$ more syntactic
    - ± 2 Hogwarts $\rightarrow$ Sunnydale (school in a fantasy series)
  - Longer: richer context $\Rightarrow$ more semantic
    - Topically related even if not similar
    - ± 5 Hogwarts $\rightarrow$ Dumbledore, half-blood
- Syntagmatic association: first-order co-occurrence
  - When two words often occur near each other
  - Wrote vis à vis book, poem
- Paradigmatic association: second-order co-occurrence
  - When two words often occur near the same other words
  - Wrote vis à vis said, remarked

Analogy

A remarkable illustration of the magic of word embeddings

- Common to visualize embeddings by reducing the dimensions to two
  - t-SNE (T-distributed Stochastic Neighbor Embedding), which produces a small dimension representation that respects similarity (Euclidean distance) between vectors
- *Offsets* (differences) between vectors reflect analogical relations
  - $\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}} \approx \overrightarrow{\text{queen}}$
  - $\overrightarrow{\text{Paris}} - \overrightarrow{\text{France}} + \overrightarrow{\text{Italy}} \approx \overrightarrow{\text{Rome}}$
  - Similar ones for
    - Brother:Sister::Nephew:Niece
    - Brother:Sister::Uncle:Aunt

Language Evolution

- Changes in meanings over time
- Consider corpora divided over time (decades)
  -
  -
- Framing changes, e.g., in news media
  - Obesity: lack of self-discipline in individuals $\Rightarrow$ poor choices of ingredients by the food industry
- Likewise, changing biases with respect to ethnic names or female names

Bias

- Word embeddings discover biases in language and highlight them
  - (From news text) $\overrightarrow{\text{man}} - \overrightarrow{\text{programmer}} + \overrightarrow{\text{woman}} \approx \overrightarrow{\text{homemaker}}$
  - $\overrightarrow{\text{doctor}} - \overrightarrow{\text{father}} + \overrightarrow{\text{mother}} \approx \overrightarrow{\text{nurse}}$
- GloVE (an embedding approach) discovers implicit association biases
  - Against African Americans
  - Against old people
- Sometimes these biases would be hidden and simply misdirect the applications of embeddings, e.g., as features for machine learning
- These biases could also be read explicitly as "justification" by a computer of someone's bias

Evaluation

- Use manually labeled data, e.g., on conceptual similarity or analogies
- Use existing language tests, e.g., TOEFL (Test of English as a Foreign Language)

FasText

- Deals with unknown words
- Uses character-level, i.e., *subword*, n-grams
  - $\langle$ word start
  - $\rangle$ word end
  - Where $\Rightarrow$ where, $\langle$wh, whe, her, ere, re$\rangle$ (original plus five trigrams)
- Learn the skipgram embedding for each n-gram
- Obtain word embedding as sum of the embeddings of its n-grams