CSCI 3907/6907

Fall 2019

Lecture 8

Vector Semantics

Slides based on: Jurafsy & Martin's "Speech and Language Processing"

Word Representations

- How should we represent word meaning for computational models?
  - String of letters?
    - N-gram language models simply use surface word forms as word representations
    - Individual words are uncorrelated
  - Index in a vector?
    - Bag-of-words representation
    - Words are uncorrelated

Word Representations

- Above representations treat each word as an independent unit
  - Unless we do complete morphological analysis, each of the following words are independent:
    - work, works, worked, working, workaholic …
  - Even with morphological analysis, the following words will be independent
    - work, job, labor, office, interview …

Models of Meaning

- Ideally representations should reflect word meaning and relationship to other words.
  - Lexical semantics

Lexical Semantics

```
mouse (N)
1.  any of numerous small rodents...
2.  a hand-operated device that controls a cursor...
```

- A **lemma** is the base form of a word.
  - mouse and mice both share the lemma “mouse”
- A **sense** or “concept” is the meaning component of a word.
  - **Homonymous** lemmas are lemmas that have multiple senses.
  - **Word sense disambiguation** is the task of identifying the correct sense of a word in context.

Relations Between Senses

- **Synonyms**: identical or nearly-identical sense of different words.
  - Couch/Sofa, big/large, car/automobile …
- Note that there are probably no examples of perfect synonymy.
  - Even if many aspects of meaning are identical
  - Still may not preserve acceptability based on notions of politeness, slang, register, genre, etc.

Relations Between Senses

- **Antonyms**: senses that are opposite with respect to one feature of meaning; otherwise very similar!
  - Dark/light, short/long, fast/slow, hot/cold, …
- Antonyms may define
  - a binary opposition or are at opposite ends of a scale
    - Long/short (length scale), fast/slow (speed scale)…
  - Or **reversives**, a change or movement in opposite direction
    - Rise/fall, up/down.

Taxonomic Relations

- **Hyponym (subordinate)**: a more specific word, a subclass
  - dog is a hyponym of animal

- **Hypernym (superordinate)**: a more general word, a superclass
  - Animal is a hypernym of dog.

- Entailment relation: hyponyms entail hypernyms.
  - Being a dog entails being an animal.

WordNet

- A large lexical database for English
  - https://wordnet.princeton.edu

WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: bass Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations
Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **bass** (the lowest part of the musical range)
- S: (n) **bass**, bass part (the lowest part in polyphonic music)
- S: (n) **bass**, basso (an adult male singer with the lowest voice)
- S: (n) sea bass, **bass** (the lean flesh of a saltwater fish of the family Serranidae)
- S: (n) freshwater bass, **bass** (any of various North American freshwater fish with lean flesh (especially of the genus Micropterus))
- S: (n) **bass**, bass voice, basso (the lowest adult male singing voice)
- S: (n) **bass** (the member with the lowest range of a family of musical instruments)
- S: (n) **bass** (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

Adjective

- S: (adj) **bass**, deep (having or denoting a low vocal or instrumental range) *"a deep voice"; "a bass voice is lower than a baritone voice"; "a bass clarinet"*

WordNet – Cont.

- Open class words are groups into sets of cognitive synonyms, called **synsets**
  - Each synset corresponds to a distinct concept
  - Synsets are linked using a small set of conceptual relations
    - Super/subordinate relations
    - Meronymy: part-whole relations (e.g. backrest-seat)
    - Antonymy (for adjectives).

Multilingual WordNet

- Variants in other languages also exist, collected in the Open Multilingual WordNet project
  - Accessible through Python's Natural Language Toolkit (NTLK) http://www.nltk.org/howto/wordnet.html

Relations Between Words

- Word **similarity**: not synonyms, but share some elements of meaning
  - Car/bicycle, cow/horse, cat/dog, …
  - Identifying similar words is useful to generalize models to unseen words

Ask humans how similar 2 words are

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

SimLex-999 dataset (Hill et al., 2015)

Relations Between Words

- Word **relatedness**: or word association
  - coffee/cup, car/gasoline : the words are not similar, they share practically no features, but are associated with a shared event.
  - Related words belong to the same **semantic field**.
    - A set of words which cover a particular semantic domain and bear structured relations with each other.

Semantic fields

hospitals

surgeon, *scalpel*, *nurse*, *anaesthetic*, *hospital*

restaurants

waiter, *menu*, *plate*, *food*, *menu, chef*

houses

door, *roof*, *kitchen*, *family*, *bed*

Semantic Frames and Roles

- **Semantic Frame**: a set of words that denote perspectives or participants in a particular type of event.
  - e.g. buy, sell
    - “Sam bought the book from Ling” vs. “Ling sold the book to Sam”

- **Semantic roles**: buyer, seller, goods, money

- Important to identify for some tasks like question answering

FrameNet

- More than 200,000 manually annotated sentences using over 1200 semantic frames.
  https://framenet.icsi.berkeley.edu

  – Use to train semantic role labeling (SRL) systems.
- Examples:

... [$_{\text{Cook}}$ the boys] ... GRILL [$_{\text{Food}}$ their catches] [$_{\text{Heating_instrument}}$ on an open fire].
[$_{\text{Avenger}}$ I] 'll GET EVEN [$_{\text{Offender}}$ with you] [$_{\text{Injury}}$ for this]!

PropBank And Other Resources

- https://propbank.github.io
- Predicate argument relations between words in a sentence
  - No specific semantic frames

- PropBank, FrameNet, OntoNotes, and VerbNet are combined in a unified resource called **Unified Verb Index**: http://verbs.colorado.edu/verb-index/

Connotation

- Words have **affective** meanings, related to emotions, sentiment, opinions, or evaluations
  - positive connotations (*happy*)
  - negative connotations (*sad*)
  - positive evaluation (*great*, *love*)
  - negative evaluation (*terrible*, *hate*).

Affect

- Early work on affective meaning (Osgood et al. 1957) described variations along three dimensions:
  - **Valence**: pleasantness (happy vs. sad)
  - **Arousal**: intensity of emotion (excited vs. calm)
  - **Dominance**: degree of control (courageous vs. heartbroken)

Affect

- Each word is a vector of a 3-dimensional space:

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24
life	6.68	5.59	5.89

  - A convenient presentation of words as a vector of numbers that also reflect meaning! (but how to calculate these values?)

Distributional Semantics

- Directly modeling all aspects of lexical semantics seems to be difficult …
- **Distributional Hypothesis**: Words that occur in similar contexts tend to have similar meanings.
  - Oculist and eye-doctor occur in the same context (around words like eye or examined)
  - The distribution of words around other words can serve as meaning representations.

What does ongchoi mean?

Suppose you see these sentences:

- Ongchoi is delicious **sautéed with garlic**.
- Ongchoi is superb **over rice**
- Ongchoi **leaves** with salty sauces

And you've also seen these:

- …spinach **sautéed with garlic over rice**
- Chard stems and **leaves** are **delicious**
- Collard greens and other **salty** leafy greens

Conclusion:

- Ongchoi is a leafy green like spinach, chard, or collard greens

Distributional Semantics

- Define a word by its distribution in language use
  - A word's distribution is the set of contexts in which it occurs, the neighboring words or grammatical environments
  - We can calculate counts of these contexts and store them in a vector
    - Each word is then a point in an N-dimensional space (N contexts)

Similarity Based Models

- Each word is a vector (or point) in a multi-dimensional space. Similar words are clustered together.

Vector Semantics

- Word vectors are called **embeddings**
- Word embeddings can be learned automatically from raw text without labeling or supervision
- Word embeddings are currently the standard way of representing words in most NLP applications

Term-document matrix

- Each document is represented by a vector of words (bag of words)
- Also called **co-occurrence** matrix
- In this example, each document is a 4-dimensional vector

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Visualizing document vectors

Document Vectors

- Document vectors were originally defined for the task of **information retrieval** to identify most related documents to a query

- A query is also represented as a vector of the same length (and the same words as dimensions)

  – Vectors are compared using a vector similarity metric such as Euclidean distance or cosine.

Words as Vectors

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

battle is "the kind of word that occurs in Julius Caesar and Henry V"

fool is "the kind of word that occurs in comedies, especially Twelfth Night"

More common: word-word matrix

- Two **words** are similar in meaning if their context vectors are similar. Context is defined using other words in the vocabulary.

sugar, a sliced lemon, a tablespoonful of	**apricot**	jam, a pinch each of,
their enjoyment. Cautiously she sampled her first	**pineapple**	and another fruit whose taste she likened
well suited to programming on the digital	**computer**.	In finding the optimal R-stage policy from
for the purpose of gathering data and	**information**	necessary for the study authorized in the

	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

Visualizing Word Vectors

	aardvark	...	computer	data	pinch	result	sugar	...
apricot	0	...	0	0	1	0	1	
pineapple	0	...	0	0	1	0	1	
digital	0	...	2	1	0	1	0	
information	0	...	1	6	0	4	0	

Comparing Word Vectors

- Vector length is rather arbitrary
  - Depending on dataset, two words may have high co-occurrence counts compared to other similar words

- Cosine similarity is typically used to compare vectors instead of Euclidean distance

Visualizing cosines (well, angles)

- "apricot" is orthogonal to "data"

	large	data
apricot	2	0
digital	0	1
information	1	6

Cosine as a similarity metric

- -1: vectors point in opposite directions
- +1: vectors point in same directions
- 0: vectors are orthogonal
- In word co-occurrence matrix, counts are non-negative, so cosine ranges between 0-1

$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

Which Words Are Most Similar?

	large	data	computer
apricot	2	0	0
digital	0	1	2
information	1	6	1

$$\cos(\vec{v},\vec{w}) = \frac{\vec{v}\bullet\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\vec{v}}{|\vec{v}|}\bullet\frac{\vec{w}}{|\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

$$\cos(\text{apricot},\text{information}) = \frac{2+0+0}{\sqrt{4+0+0}\sqrt{1+36+1}} = \frac{2}{2\sqrt{38}} = .16$$

$$\cos(\text{digital},\text{information}) = \frac{0+6+2}{\sqrt{0+1+4}\sqrt{1+36+1}} = \frac{8}{\sqrt{38}\sqrt{5}} = .58$$

Frequency Alone is Not Great

- Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information
- But overly frequent words like *the*, *it,* or *they* are not very informative about the context
- Need a function that resolves this frequency paradox!

tf-idf: combine two factors

- **tf: term frequency**. frequency count (usually log-transformed):

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

- **idf: inverse document frequency:**

Frequent words like "the" or "good" have very low idf

$$\mathrm{idf}_i = \log\left(\frac{N}{\mathrm{df}_i}\right)$$

Total # of docs in collection

# of docs that have word i

tf-idf value for word t in document d:

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

Idf - example

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

Example

- The word “good” appears in all Shakespearean plays, so it’s not helpful as feature

raw frequencies

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

tf-idf

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Summary: tf-idf

- Compare two words using tf-idf cosine to see if they are similar

- Compare two documents
  - Take the centroid of vectors of all the words in the document
  - Centroid document vector is:

$$d = \frac{w_1 + w_2 + \ldots + w_k}{k}$$

PPMI

- The positive point-wise mutual information (PPMI)
  - An alternative weighting function, often used for weighting term-term co-occurrence matrix

- tf-idf relies on the concept of documents; PPMI does not

- PMI measures how often two events occur more compared to what is expected by chance (that is, if the events were independent)

$$\mathrm{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)}$$

PPMI

$$\mathrm{PMI}(w, c) = \log_2 \frac{P(w, c)}{P(w)P(c)}$$

- PMI values range from negative to positive infinity. Negative PMI values are unreliable
  - Accurate estimates for rare words require huge amounts of data!

- Positive PMI (PPMI) replaces all negative values with zeros

Count(w,context)

	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \quad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \quad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$\text{PPMI}_{ij} = \max(\log_2 \frac{p_{ij}}{p_{i*}p_{*j}}, 0)$$

$$\begin{aligned}
P(\text{w=information,c=data}) &= \frac{6}{19} = .316 \\
P(\text{w=information}) &= \frac{11}{19} = .579 \\
P(\text{c=data}) &= \frac{7}{19} = .368 \\
\text{ppmi(information,data)} &= \log 2(.316/(.368 * .579)) = .568
\end{aligned}$$

	p(w,context)					p(w)
	computer	data	pinch	result	sugar	
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

Weighting PMI

- PMI is biased toward infrequent events
  - Very rare words have very high PMI values

- Two solutions:
  - Give rare words slightly higher probabilities
  - Use Laplace smoothing (which has a similar effect)

Add-2 Smoothed Count(w,context)

	computer	data	pinch	result	sugar
apricot	2	2	3	2	3
pineapple	2	2	3	2	3
digital	4	3	2	3	2
information	3	8	2	6	2

	p(w,context) [add-2]					**p(w)**
	computer	data	pinch	result	sugar	
apricot	0.03	0.03	0.05	0.03	0.05	0.20
pineapple	0.03	0.03	0.05	0.03	0.05	0.20
digital	0.07	0.05	0.03	0.05	0.03	0.24
information	0.05	0.14	0.03	0.10	0.03	0.36
p(context)	0.19	0.25	0.17	0.22	0.17	

PPMI versus add-2 smoothed PPMI

PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

PPMI(w,context) [add-2]

	computer	data	pinch	result	sugar
apricot	0.00	0.00	0.56	0.00	0.56
pineapple	0.00	0.00	0.56	0.00	0.56
digital	0.62	0.00	0.00	0.00	0.00
information	0.00	0.58	0.00	0.37	0.00

LSA

- A term-document matrix is sparse
  - Most elements are zero

- **Latest Semantic Analysis** (LSA) finds a low-rank approximation of tf-idf weighted term-document matrix.
  - Using Singular value decomposition (SVD)

LSA

- LSA results in low-dimensional dense vectors for the rows (words) and columns (documents).

https://liqiangguo.files.wordpress.com/2011/06/svd1.jpg

Sparse versus dense vectors

- Why dense vectors?
  - [-0.10,-0.47,-0.49,-0.06, 0.02, 0.43, 0.43, 0.40] **vs.** [0,0.47,0,0, 0.02, 0.43, 0, 0.40]
  - Short vectors may be easier to use as **features** in machine learning (less weights to tune)
  - Dense vectors may **generalize** better than storing explicit counts
  - They may do better at capturing synonymy:
    - *car* and *automobile* are synonyms; but are distinct dimensions
      - a word with *car* as a neighbor and a word with *automobile* as a neighbor should be similar, but aren't
  - **In practice, they work better**

Word2vec

- Instead of **counting** how often each word *w* occurs near "*apricot*"
- Train a classifier on a binary **prediction** task:
  - Is *w* likely to show up near "*apricot*"?

- We don't actually care about this task
  - But we'll take the learned classifier weights as the word embeddings

Unsupervised Training

- A word *s* near *apricot*
  - Acts as gold ‘correct answer’ to the question
  - “Is word *w* likely to show up near *apricot*?”

- No need for hand-labeled supervision

- The idea comes from **neural language modeling**
  - Bengio et al. (2003)
  - Collobert et al. (2011)

Word2Vec: **Skip-Gram** Task

- Word2vec provides a variety of options. Let's do
  - "skip-gram with negative sampling" (SGNS)

Skip-gram algorithm

1. Treat the target word and a neighboring context word as positive examples

2. Randomly sample other words in the lexicon to get negative samples

3. Use logistic regression to train a classifier to distinguish those two cases
   - Binary Classification

4. Use the weights as the embeddings

Skip-Gram Training Data

- Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

c1 c2 target c3 c4

Asssume context words are those in +/- 2 word window

Skip-Gram Goal

- Given a tuple (t,c) = target, context
  - (*apricot*, *jam*)
  - (*apricot*, *aardvark*)

Return probability that c is a real context word:

$$P(+|t,c)$$
$$P(-|t,c) = 1-P(+|t,c)$$

How to compute p(+|t,c)?

- Intuition:
  - Words are likely to appear near similar words
  - Model similarity with dot-product!
  - Similarity(t,c) $\propto t \cdot c$
- *Problem:*
  - *Dot product is not a probability!*
    - *(Neither is cosine)*

Turning dot product into a probability

- The sigmoid (logit) function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Turning dot product into a probability

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

$$\begin{aligned} P(-|t,c) &= 1 - P(+|t,c) \\ &= \frac{e^{-t\cdot c}}{1+e^{-t\cdot c}} \end{aligned}$$

For all the context words

- Assume all context words are independent, then simply multiply the probabilities.

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

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

Skip-Gram Training Data

- Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

c1 c2 t c3 c4

- Training data: input/output pairs centering on *apricot*
- Asssume a +/- 2 word window

Skip-Gram Training

- Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

c1 c2 t c3 c4

positive examples +	
t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

- For each positive example, we'll create *k* negative examples.
- Using *noise* words
- Any random word that isn't *t*

Skip-Gram Training

- Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

c1 c2 t c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

negative examples -

t	c	t	c
apricot	aardvark	apricot	twelve
apricot	puddle	apricot	hello
apricot	where	apricot	dear
apricot	coaxial	apricot	forever

Choosing noise words

- Could pick w according to their unigram frequency P(w)
- More common to chosen then according to $p_\alpha(w)$

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_w count(w)^\alpha}$$

- α= ¾ works well because it gives rare noise words slightly higher probability. To show this, imagine two events p(a)=.99 and p(b) = .01:

$$P_\alpha(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97$$

$$P_\alpha(b) = \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = .03$$

Setup

- Let's represent words as vectors of some length (say 300), randomly initialized.
- So we start with 300 * V random parameters
- Over the entire training set, we'd like to adjust those word vectors such that we
  - Maximize the similarity of the target word, context word pairs (t,c) drawn from the positive data
  - Minimize the similarity of the (t,c) pairs drawn from the negative data.

Objective Criteria

- We want to maximize…

$$\sum_{(t,c)\in+} logP(+|t,c) + \sum_{(t,c)\in-} logP(-|t,c)$$

- Maximize the + label for the pairs from the positive training data, and the − label for the pairs sample from the negative data.

The Skip-Gram Model

Summary: Skip-Gram Training

- Start with V random 300-dimensional vectors as initial embeddings
- Use logistic regression, the second most basic classifier used in machine learning after naïve bayes
  - Take a corpus and take pairs of words that co-occur as positive examples
  - Take pairs of words that don't co-occur as negative examples
  - Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
  - Throw away the classifier code and keep the embeddings.

Context Embeddings

- We can choose to throw away the context embeddings, or use them in combination with the word embeddings:
  - Sum up the word embedding and context embedding of each word, or
  - Concatenate word and context embeddings.

Skip-Gram vs. PPMI co-occurrence

- An analysis of Skip-gram with negative sampling shows that it is implicitly equivalent to factorizing a word-context matrix with PMI weighting, shifted by a global constant
  - https://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf
  - Working directly with the equivalent shifted PPMI objective is better at optimizing the SGNS objective.

Visualizing Word Embeddings

- Cluster words by similarity
  - e.g. k-means clustering or hierarchical clustering
- Find nearest neighbors for each word

Two-dimensional projection

Project high-dimensional vectors to 2-dimensions in a way that preserves distances to some extent:

- PCA
- T-SNE

Evaluating embeddings

- Compare to human scores on word similarity-type tasks:
  - WordSim-353 (Finkelstein et al., 2002)
  - SimLex-999 (Hill et al., 2015)
  - Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
  - TOEFL dataset: *Levied is closest in meaning to: imposed, believed, requested, correlated*

Properties of Embeddings

Similarity depends on window size C

- C = ±2 The nearest words to *Hogwarts:*
  - *Sunnydale*
  - *Evernight*
- C = ±5 The nearest words to *Hogwarts:*
  - *Dumbledore*
  - *Malfoy*
  - *halfblood*

Analogy: relational meaning

vector(*'king'*) - vector(*'man'*) + vector(*'woman'*) vector('queen')

vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) vector('Rome')

heiress
niece
aunt
sister
countess
duchess
empress
madam
nephew
heir
woman
uncle
earl
queen
duke
brother
emperor
sir
man
king
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.5
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5

slowest
slower
shortest
shorter
slow
short
stronger
strongest
louder
loudest
strong
loud
clearer
clearest
softer
softest
clear
darker
darkest
soft
dark
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5
0.6

Semantic Properties of Word Embeddings

- Embeddings encode the distributional properties in the training dataset:
  - Word usage changes over time. Comparing word embeddings from different time periods help study evolution of word meaning.
  - Embeddings also reflect any cultural bias or stereotypes present in the text used in training.

Diachronic word embeddings for studying language change

Visualizing Changes

Project 300 dimensions down into 2

~30 million books, 1850-1990, Google Books data

The Evolution of Sentiment Words

Negative words change faster than positive words

Summary

- **Concepts** or word senses
  - Have a complex many-to-many association with **words** (homonymy, multiple senses)
  - Have relations with each other (Synonymy, Antonymy, Superordinate)
  - But are hard to define formally

- **Embeddings** = vector models of meaning
  - More fine-grained than just a string or index
  - Especially good at modeling similarity/analogy
  - Can use sparse models (tf-idf) or dense models (word2vec, GLoVE)
  - Useful in practice but know they encode cultural stereotypes.

Extensions

- Word embeddings are great, but have many limitations:
  - Each word has a single embedding
    - But we know many words have multiple senses!
  - What about unseen words?
    - Totally novel words might be hard to deal with, but what about inflections or derived words?

Extensions

- From word to sense embeddings:
  - https://arxiv.org/pdf/1805.04032.pdf

- Embeddings with subword information
  - FastText http://www.fasttext.cc/

Dense embeddings you can download

- **Word2vec** (Mikolov et al.)
https://code.google.com/archive/p/word2vec/

- **Fasttext** http://www.fasttext.cc/

- **Glove** (Pennington, Socher, Manning)
http://nlp.stanford.edu/projects/glove/

Supplementary Tutorial

- A jupyter notebook that walks you through the steps of training a word2vec model using the Gensim library