Approximating Semantics

COSC 6336: Natural Language Processing
Spring 2020

Some content in these slides has been adapted from Jurafsky & Martin 3rd edition, and lecture slides from Rada Mihalcea, Ray Mooney and the deep learning course by Manning and Socher.

UNIVERSITY of HOUSTON

Today's Lecture

- ★ Semantic representation
  - ○ Word level
  - ○ Document level
- ★ Neural word embeddings
  - ○ Word2vec
  - ○ Glove
  - ○ FastText.zip

Semantic Representation of Words

- ★ *Polysemy:* Many words in natural language have more than one meaning:
  - Take one pill daily
  - Take the first right past the stop light
- ★ A computer program has no basis to knowing which sense is appropriate
- ★ Many relevant tasks require an accurate disambiguation:
  - QA (Who is the head of state of X country? vs who's the president?)
  - Information Retrieval (search for Michael Jordan)
  - Machine Translation (I am an NLP researcher, vs I am at the plaza)
- ★ How do humans manage word sense disambiguation (WSD)?

In the early days of WSD

Bar-Hillel (1960) posed the following:

Little John was looking for his toy box. Finally, he found it. The box was in the pen. John was very happy.

Is "**pen**" a writing instrument or an enclosure where children play?

…declared it unsolvable, left the field of MT!

How do we map words to meanings?

How do we map words to meanings?

- ★ Dictionaries
  - ○ Oxford English Dictionary
  - ○ Collins
  - ○ Longman Dictionary of Ordinary Contemporary English (LDOCE)
- ★ Thesauruses – add synonymy information
  - ○ Roget Thesaurus
- ★ Semantic networks – add more semantic relations
  - ○ WordNet
  - ○ EuroWordNet

Example from WordNet

```
>>> for ss in wn.synsets('coati'):
...     print(ss.name(), ss.lemma_names())
...
(u'coati.n.01', [u'coati', u'coati-mondi', u'coati-mundi', u'coon_cat', u'Nasua_narica'])
>>>
```

Early days of WSD

1970s - 1980s

- Rule based systems
- Rely on hand-crafted knowledge sources

1990s

- Corpus based approaches
- Dependence on sense tagged text
- (Ide and Veronis, 1998) overview history from early days to 1998.

2000s

- Hybrid Systems
- Minimizing or eliminating use of sense tagged text
- Taking advantage of the Web

Example WSD Approach with dictionaries

Simplified Lesk (Kilgarriff & Rosensweig, 2000):

1. Retrieve from MRD all sense definitions of the word to be disambiguated
2. Determine the overlap between each sense definition and the current context
3. Choose the sense that leads to highest overlap

Example: disambiguate PINE in

"Pine cones hanging in a tree"

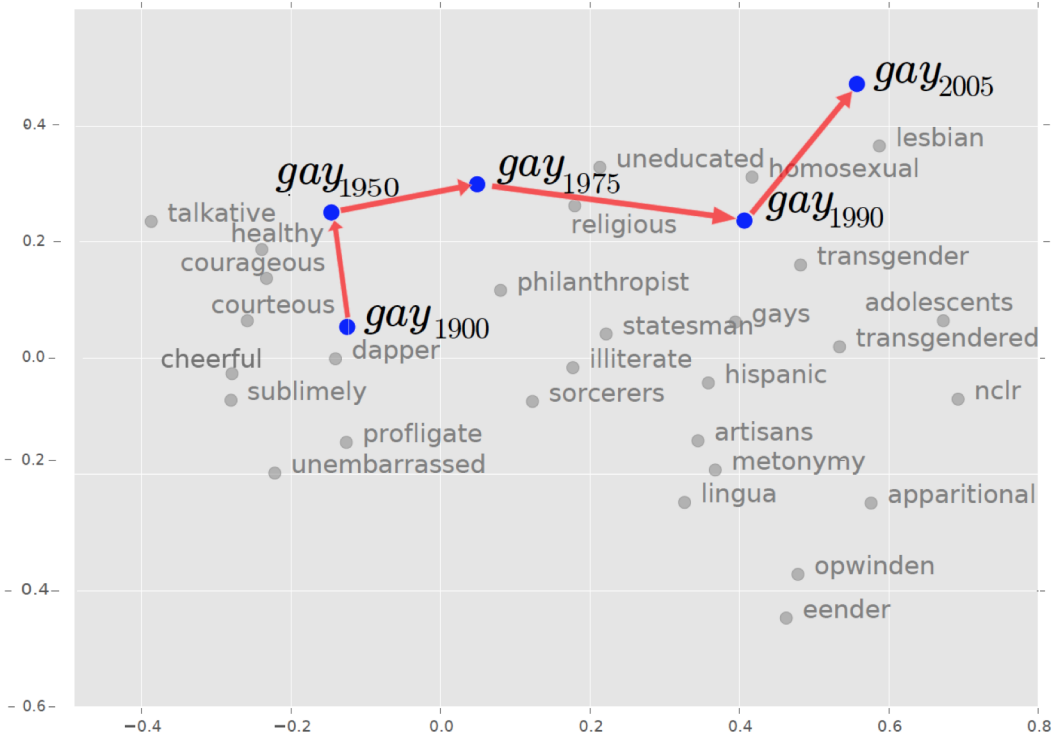
• PINE

1. kinds of evergreen tree with needle-shaped leaves
2. waste away through sorrow or illness

Pine#1 ∩ Sentence = 1
Pine#2 ∩ Sentence = 0

Limitations of Machine Readable Dictionaries

- Brittle
- Fail to capture changes in meaning over time
- Subjective
- Requires human involvement
- Low coverage of languages



UNIVERSITY of HOUSTON

From words to documents

Why vector models of meaning?

"**fast**" is similar to "**rapid**"

"**tall**" is similar to "**height**"

Question answering:

*Q: "How **tall** is Mt. Everest?"*

*Candidate A: "The official **height** of Mount Everest is 29029 feet"*

Vector Semantics

Key Idea: “ You shall know a word by the company it keeps!” (Firth, 1957)

The coati is extremely noisy

Vector Semantics

Key Idea: “ You shall know a word by the company it keeps!” (Firth, 1957)

The coati is extremely noisy. Coatis love fruits, insects and mice.

Vector Semantics

Key Idea: “ You shall know a word by the company it keeps!” (Firth, 1957)

The coati is extremely noisy. Coatis love fruits, insects and mice. They live in North America and are relatives of the racoon.

Vector Semantics: Intuition

- ★ Model the meaning of a word by “embedding” in a vector space.
- ★ The meaning of a word is a vector of numbers
  - Vector models are also called “**embeddings**”.
- ★ Contrast: word meaning is represented in many computational linguistic applications by a vocabulary index (“word number 545”)

Vector Semantics

Sparse vector representations:

- ★ Mutual-information weighted word co-occurrence matrices

Dense vector representations:

- ★ Singular value decomposition (and Latent Semantic Analysis)
- ★ Neural-network-inspired models (skip-grams, CBOW)
- ★ Brown clusters

Co-occurrence matrices

- ★ We represent how often a word occurs in a document: **Term-document matrix**
- ★ Or how often a word occurs with another: **term-term matrix** (or **word-word co-occurrence matrix** or **word-context matrix**)

Term-document matrix

- Each cell: count of word *w* in a document *d*:
  - Each document is a count vector in $\mathbb{N}^{|V|}$: a column below

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Document similarities in term-document matrices

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Word similarities in term-document matrices

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Word-word or word-context matrices

- ★ Context is now a window, not a document
- ★ A word is now defined by a vector over counts of context vectors
- ★ Vectors are now of length |V|
- ★ Shorter windows more syntactically oriented
- ★ Longer windows more semantically oriented

Alternative weighting

- ★ Raw counts are blunt notions of association
- ★ We could use a measure of how informative a given context word is about the target word:
  - Point-wise mutual information (PMI)
  - Or it's close relative: Positive PMI (PPMI)
  - TF-IDF

Motivating dense vector representations

- ★ Term-document and term-term co-occurrence vectors are high dimensional:
  - Anywhere from 20K to 13M
  - Sparse
  - Too many parameters to learn!
  - Dense vectors may be better at representing semantic relatedness

Dense Vectors

Neural Word Embeddings

Word Embeddings

We build dense vectors for words built in such a way so that words that appear in similar contexts have similar vectors.

$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

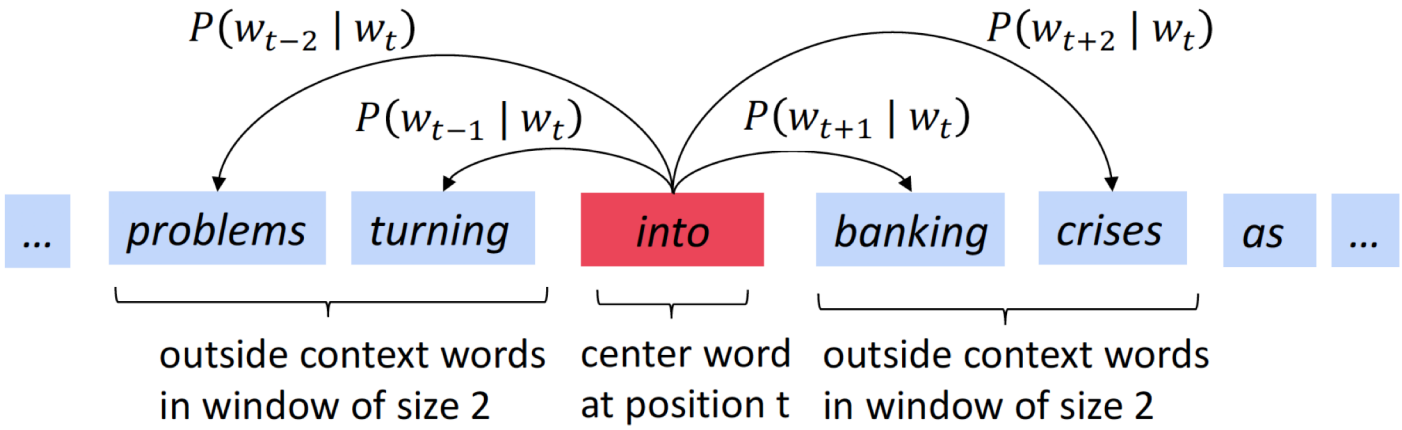
word2vec

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. In Proceedings of Workshop at ICLR, 2013.

- ★ Neural network architecture for efficiently computing continuous vector representations of words from very large data sets.
- ★ Proposes two strategies:
  - Continuous bag-of-words
  - Continuous skip-gram

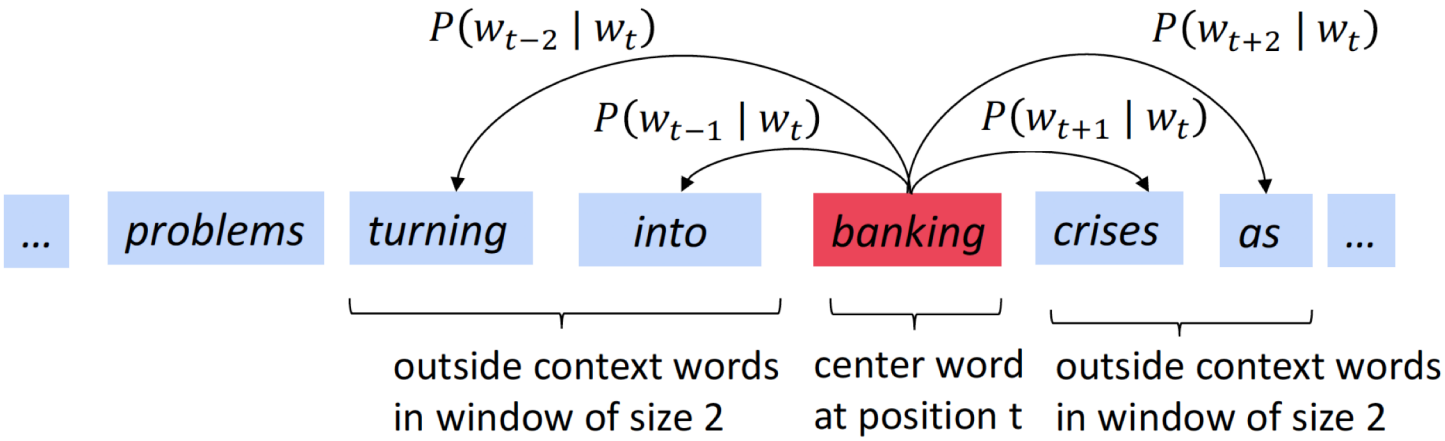
Word2Vec

Example for computing $p(w_{t+j}|w_t)$



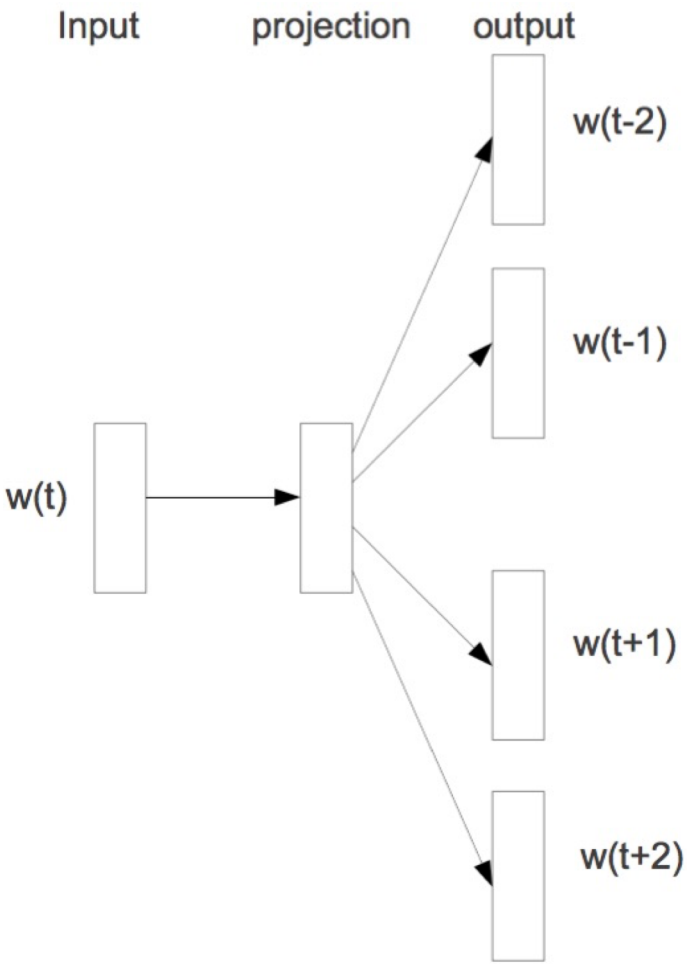
Word2Vec

Example for computing $p(w_{t+j}|w_t)$



Skip-gram

- ★ This an other models: train on a “fake task”: predict the context given a word



Skip-gram model

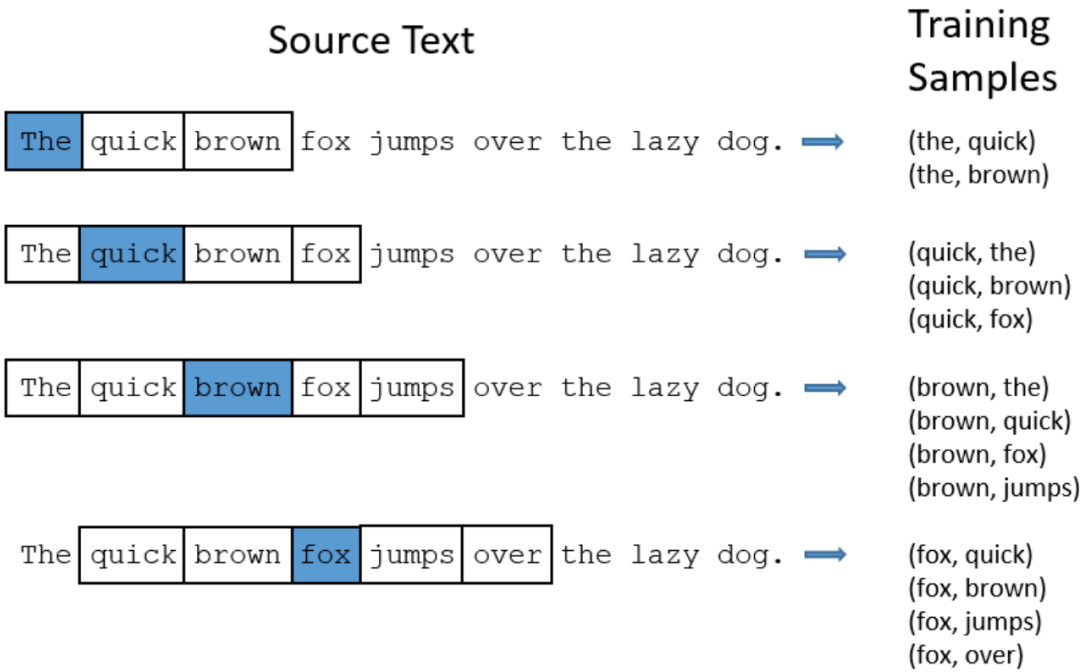
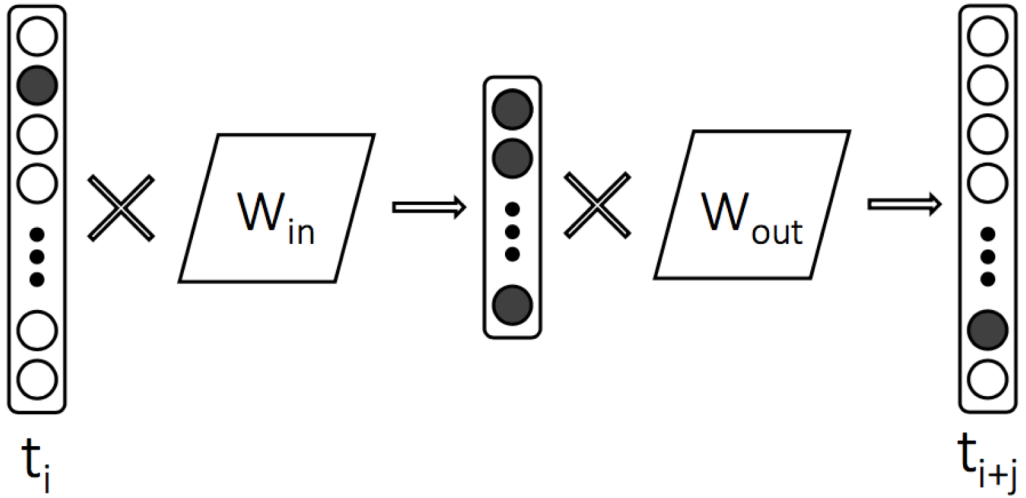


Image borrowed from Chris McCormick http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Skip-gram detail



$$\mathcal{L}_{skip-gram} = -\frac{1}{|S|}\sum_{i=1}^{|S|}\sum_{-c\leq j\leq +c, j\neq 0} log(p(t_{i+j}|t_i))$$

Efficient implementation

- Soft-max output:

$$p(t_{i+j}|t_i) = \frac{\exp\left((W_{out}\vec{v}_{t_{i+j}})^\intercal (W_{in}\vec{v}_{t_i})\right)}{\sum_{k=1}^{|T|} \exp\left((W_{out}\vec{v}_{t_k})^\intercal (W_{in}\vec{v}_{t_i})\right)}$$

- To calculate the denominator you have to sum over the whole vocabulary. Very inefficient!
- Strategies:
  - Hierarchical softmax
  - Negative sampling

Negative sampling

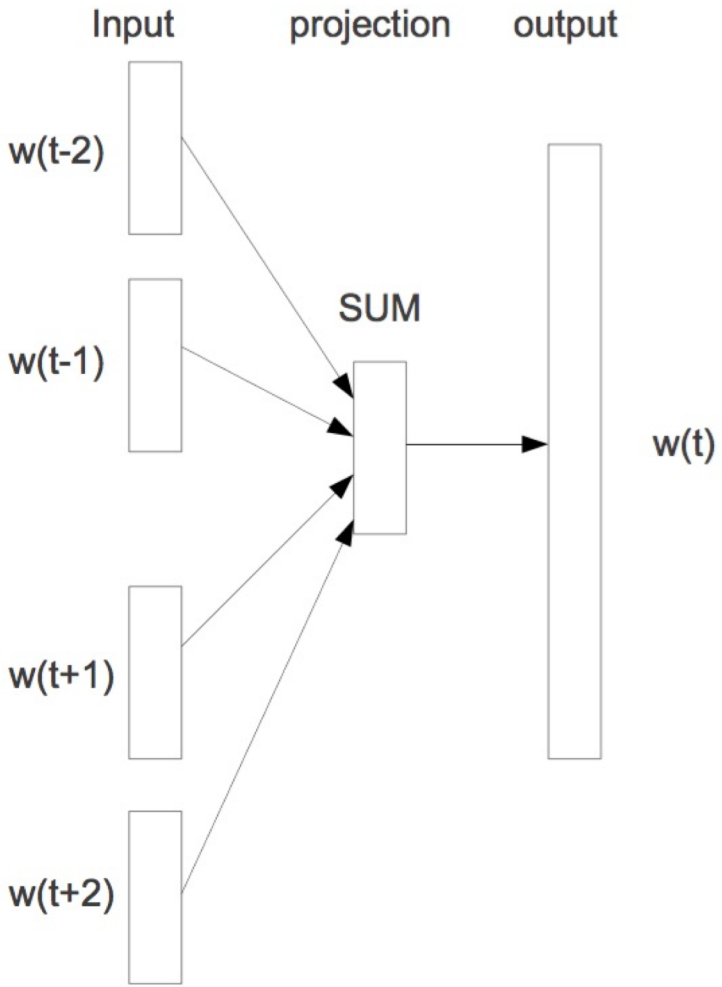
$$\log \sigma({v'_{w_O}}^{\top} v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^{\top} v_{w_I}) \right]$$

positive sample

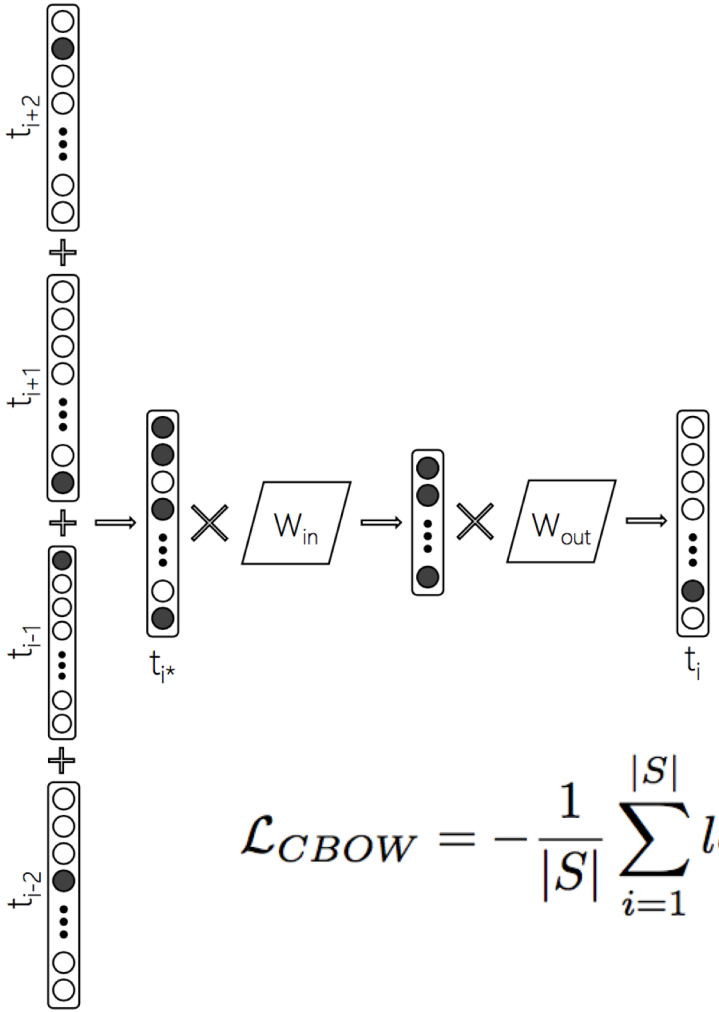
negative samples

Continuous bag-of-words

- ★ Problem: predict a word given its context.
- ★ All the words in the context use the same codification.
- ★ The representation of the words in the context are summed (compositionality).



CBOW detail



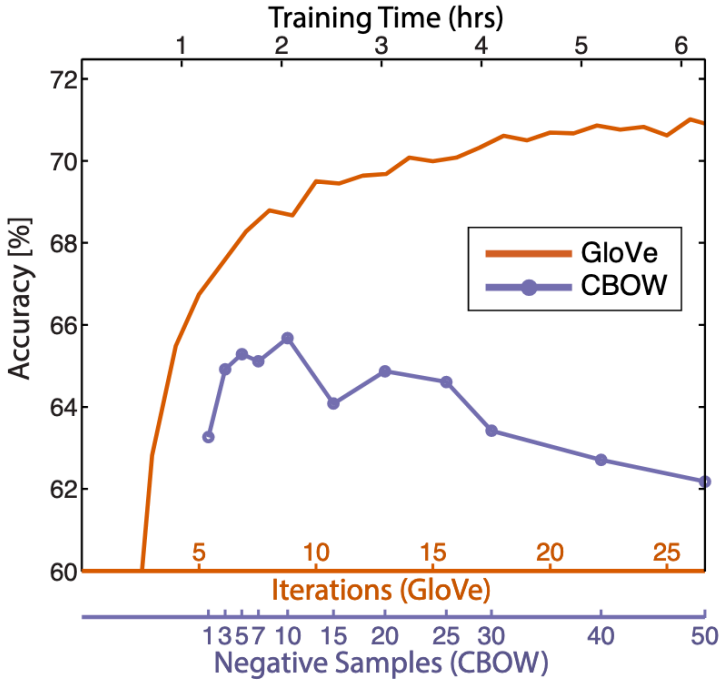
$$\mathcal{L}_{CBOW} = -\frac{1}{|S|}\sum_{i=1}^{|S|} log(p(t_i| \sum_{-c \leq j \leq +c, j \neq 0} t_{i+j}))$$

GloVe

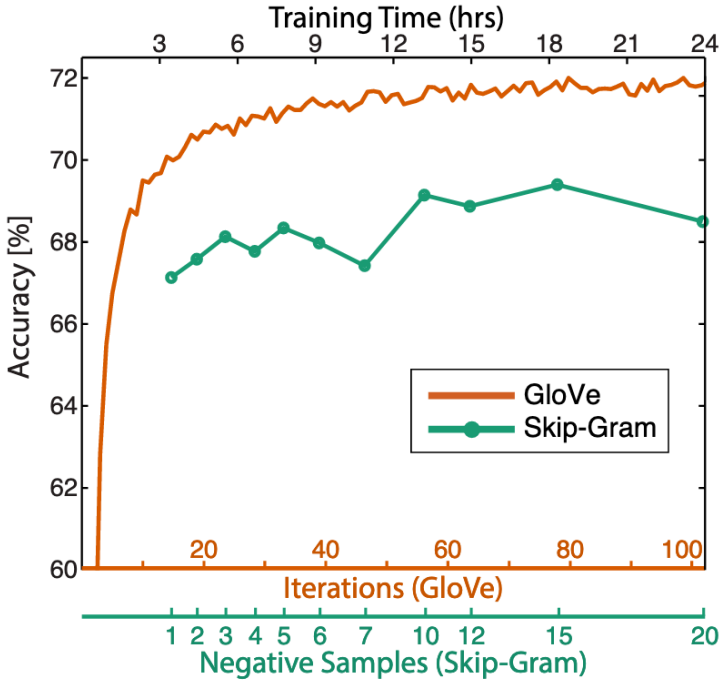
Pennington, J., Socher, R., & Manning, C. (2014). *Glove: Global vectors for word representation*. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

- ★ Learns from word co-occurrence corpus statistics instead than from individual text window samples.
- ★ It can be seen as a generalization of the skip-gram model with an additional saturation function that controls the influence of high-frequency co-occurrences.

GloVe performance



(a) GloVe vs CBOW

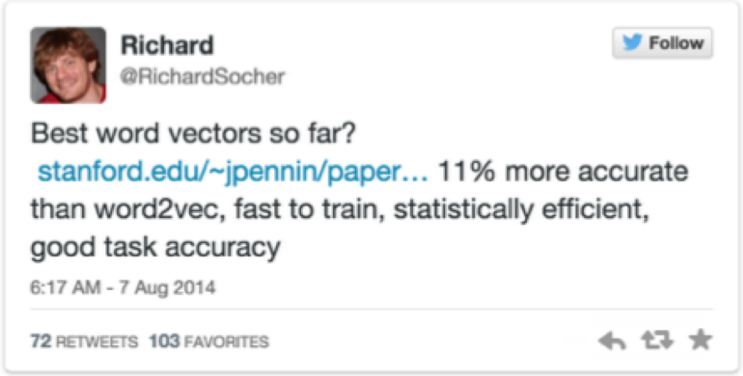


(b) GloVe vs Skip-Gram

GloVe Criticism

Model	Dim.	Size	Sem.	Syn.	Tot.
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

results on the word analogy task



vs

On the importance of comparing apples to apples: a case study using the GloVe model

Yoav Goldberg, 10 August 2014

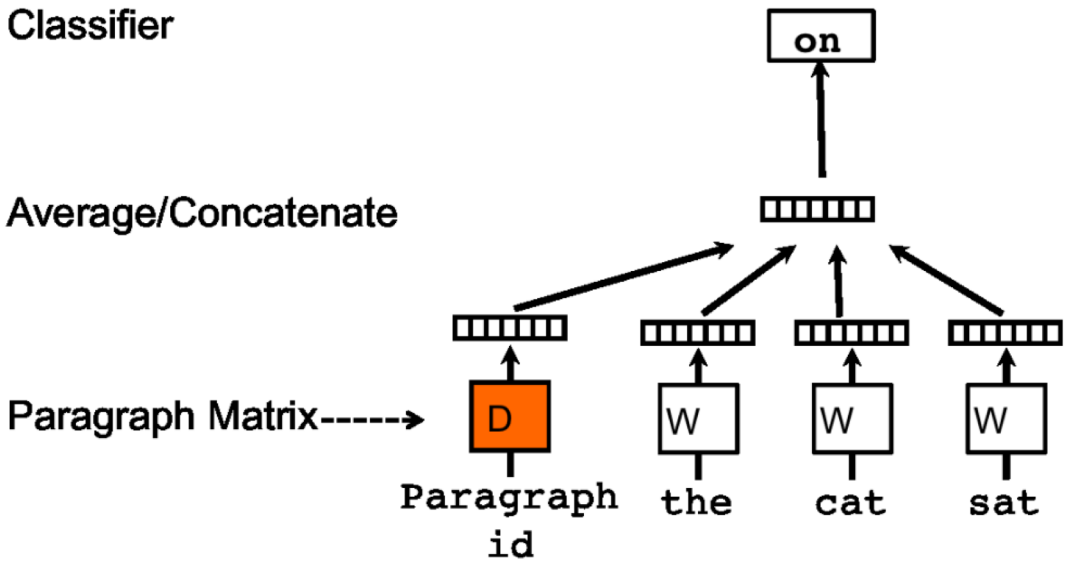
TL;DR: the GloVe model is not better than word2vec on analogy question when properly compared. The models have different strengths, but the overall accuracy is very similar. When evaluating embedding models, it is crucial to compare apples to apples and control for as much of the variation as possible. In particular, the difference in model quality seem to stem from using a different feature set and not from using a different optimization objective.

"similar accuracy"

Taken from a presentation from Roelof Pieters (www.csc.kth.se/~roelof/)

UNIVE
HOL

paragraph2vec



Le, Q., & Mikolov, T. (2014). *Distributed representations of sentences and documents*. In Proceedings of the 31st International Conference on Machine Learning (ICML-14) (pp. 1188-1196).

paragraph2vec performance

Table 1. The performance of our method compared to other approaches on the Stanford Sentiment Treebank dataset. The error rates of other methods are reported in (Socher et al., 2013b).

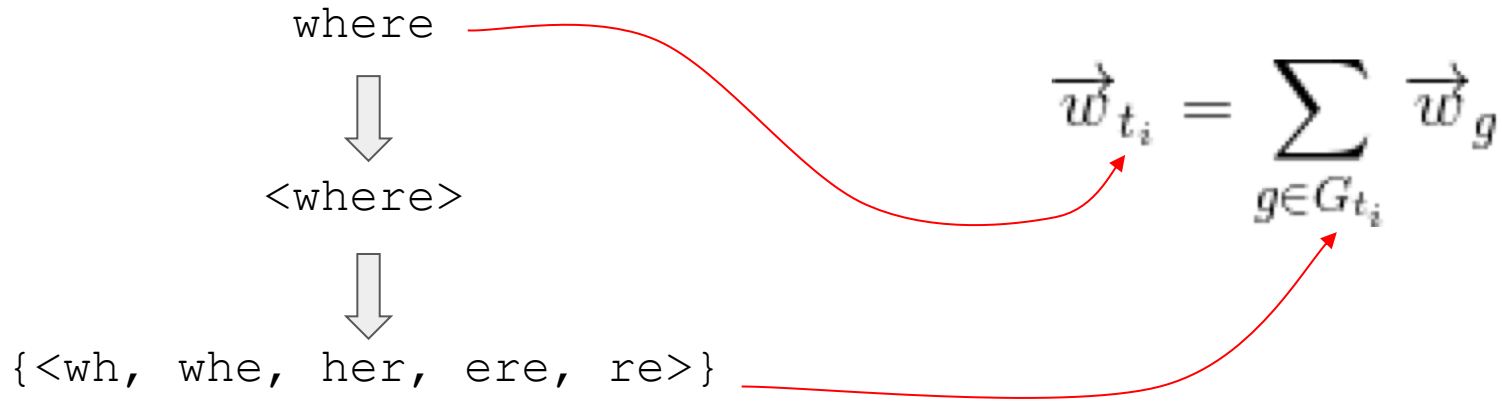
Model	Error rate (Positive/ Negative)	Error rate (Fine-grained)
Naïve Bayes (Socher et al., 2013b)	18.2 %	59.0%
SVMs (Socher et al., 2013b)	20.6%	59.3%
Bigram Naïve Bayes (Socher et al., 2013b)	16.9%	58.1%
Word Vector Averaging (Socher et al., 2013b)	19.9%	67.3%
Recursive Neural Network (Socher et al., 2013b)	17.6%	56.8%
Matrix Vector-RNN (Socher et al., 2013b)	17.1%	55.6%
Recursive Neural Tensor Network (Socher et al., 2013b)	14.6%	54.3%
Paragraph Vector	**12.2%**	**51.3%**

fastText word embeddings

Bojanowski, Piotr, et al. "*Enriching Word Vectors with Subword Information*." Transactions of the Association for Computational Linguistics 5 (2017): 135-146.

- ★ Extends the Skip-gram model to take into account morphological information.
- ★ The model finds representations for n-grams.
- ★ A word representation is built from the representation of its constituent n-grams.
- ★ Uses a fast implementation based on hashing of the n-grams. 1.5x slower than conventional Skip-gram.

FastText word embeddings



Discussion

Advantages of neural embeddings:

- ★ Dense representations that leverage distributional semantics
- ★ Pretrained representations available

Disadvantages:

- ★ Pretrained representations
- ★ Polysemy is not appropriately handled

Current Research in Word Embeddings

- ★ Task specific embeddings
- ★ Optimal dimension
- ★ Context dependent embeddings
- ★ Transformers