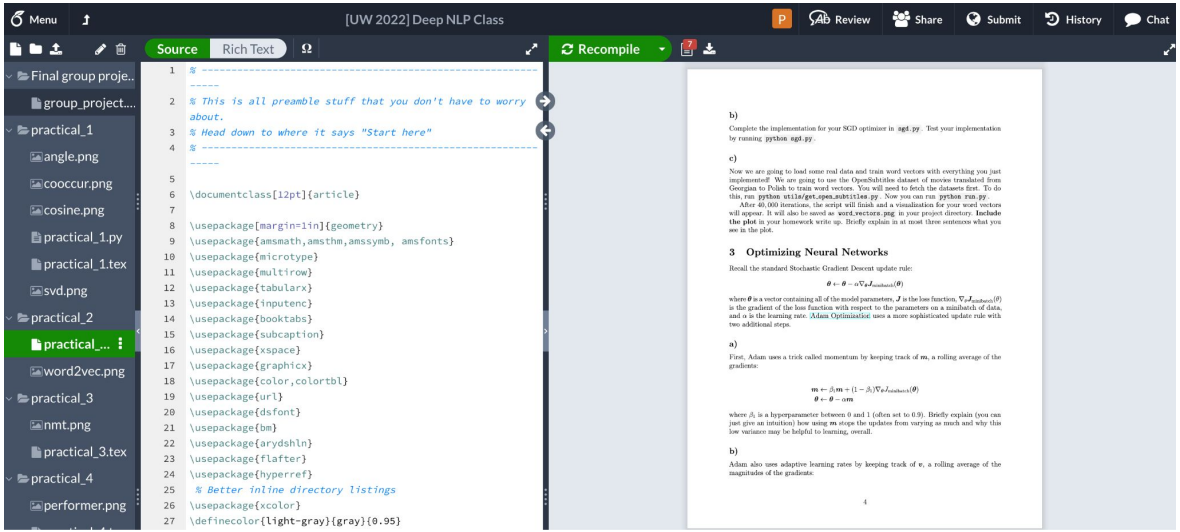
More on Word Vectors

Deep Natural Language Processing, 2022
Paweł Budzianowski

Attendance policy

Submitting reports

- Use template provided in the Moodle/Pegaz
- Overleaf [Disclaimer: Open Advertisement]



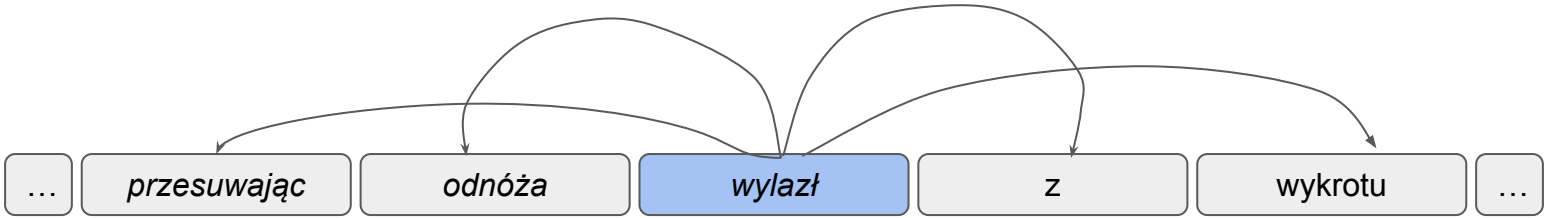
Recordings

GPU Usage

Main idea of the word2vec

Iterate through each word of the whole corpus.

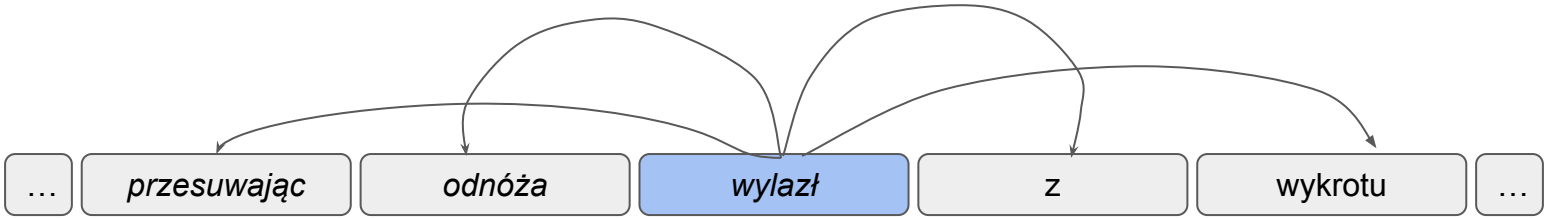
Predict the words around the center word.



Main idea of the word2vec

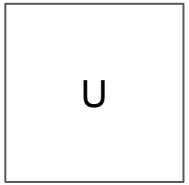
Iterate through each word of the whole corpus.

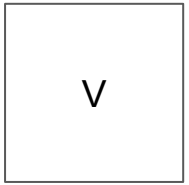
Predict the words around the center word.



$$P(O = o|C = c) = \frac{\exp(\mathbf{u}_o^T \mathbf{v}_c)}{\sum_{w \in Vocab} \exp(\mathbf{u}_w^T \mathbf{v}_c)}.$$

Main idea of the word2vec

outside 

center 

dot product $U \cdot v_c$

softmax probabilities $\text{softmax}(U \cdot v_c)$

- Same predictions at each position!
- We want a model that gives a reasonably high probability to all words that occur in the context.

Stop words issue

- Stop words: AND, OR, THE and more

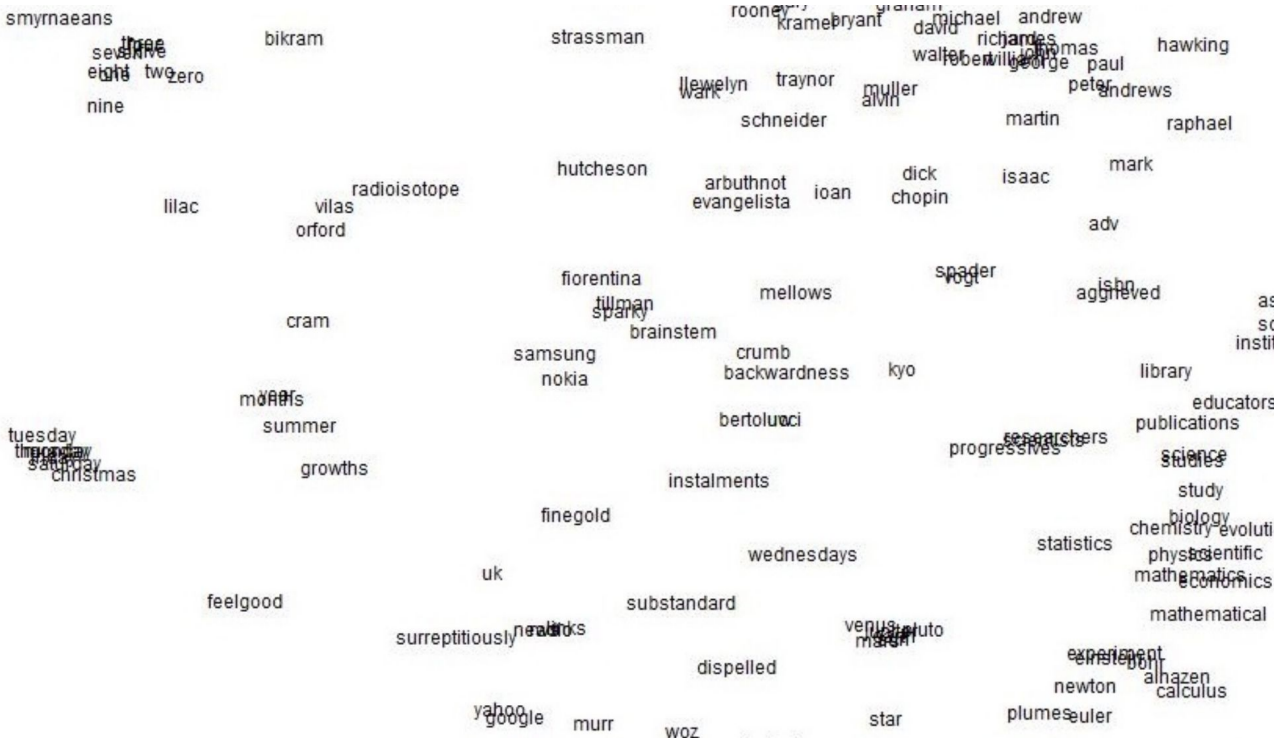
- All word vectors have a high probability component related with high frequency word effect

- The crude way to get rid of it is often the case that the first component is related to it.

Visualising Word Vectors

- 2 dimensional pictures are exceedingly misleading.

Visualising Word Vectors [Manning 2019]



Visualising Word Vectors

- 2 dimensional pictures are exceedingly misleading.

- For example:
  Nokia is close to Samsung and should be also to Yahoo and Google and Finland?

- High dimensional space is not intuitive and some crazy stuff can happen out there that it’s hard to reason about.

Training Skip-gram

Optimization Basics

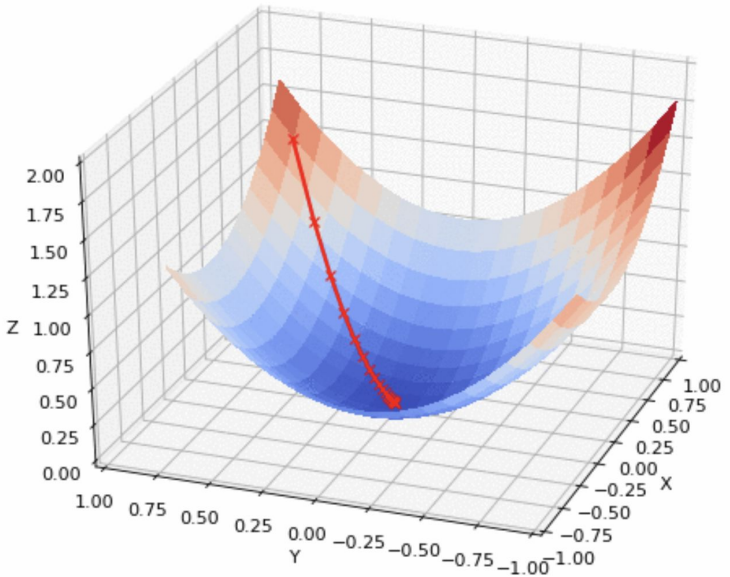
- We have a cost function $J(\theta)$ we want to minimize
- Gradient Descent is an algorithm to minimize $J(\theta)$

- Idea: for current value of θ, calculate gradient of $J(\theta)$ then take small step in the direction of negative gradient and repeat.

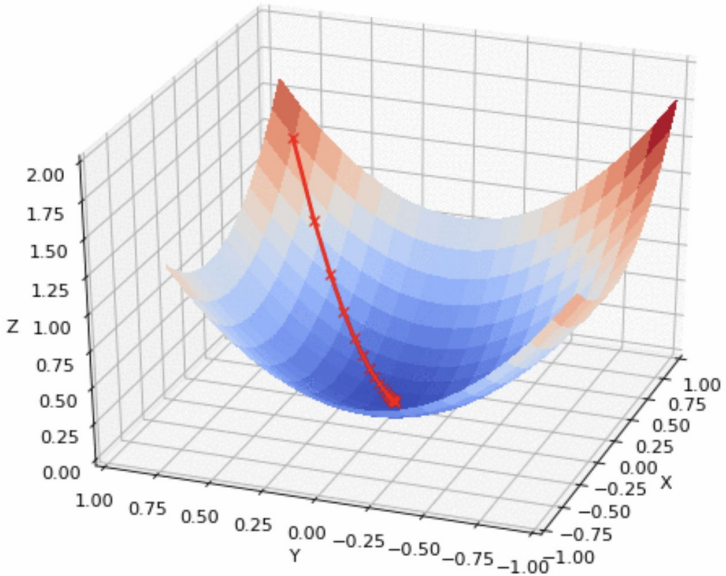
Update rule

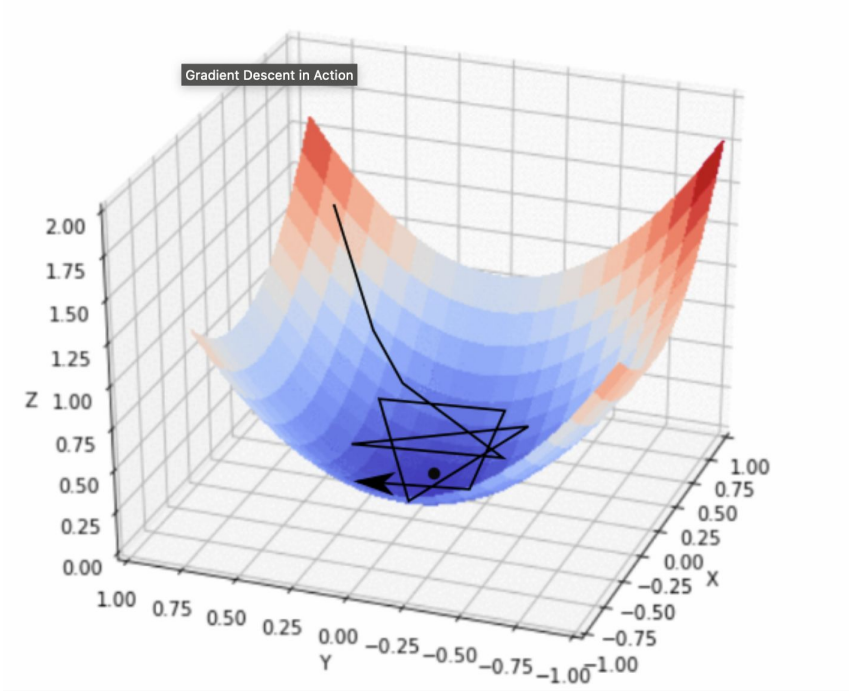
$$\theta_{new} = \theta_{old} - \alpha \nabla J(\theta)$$

Going down the slope

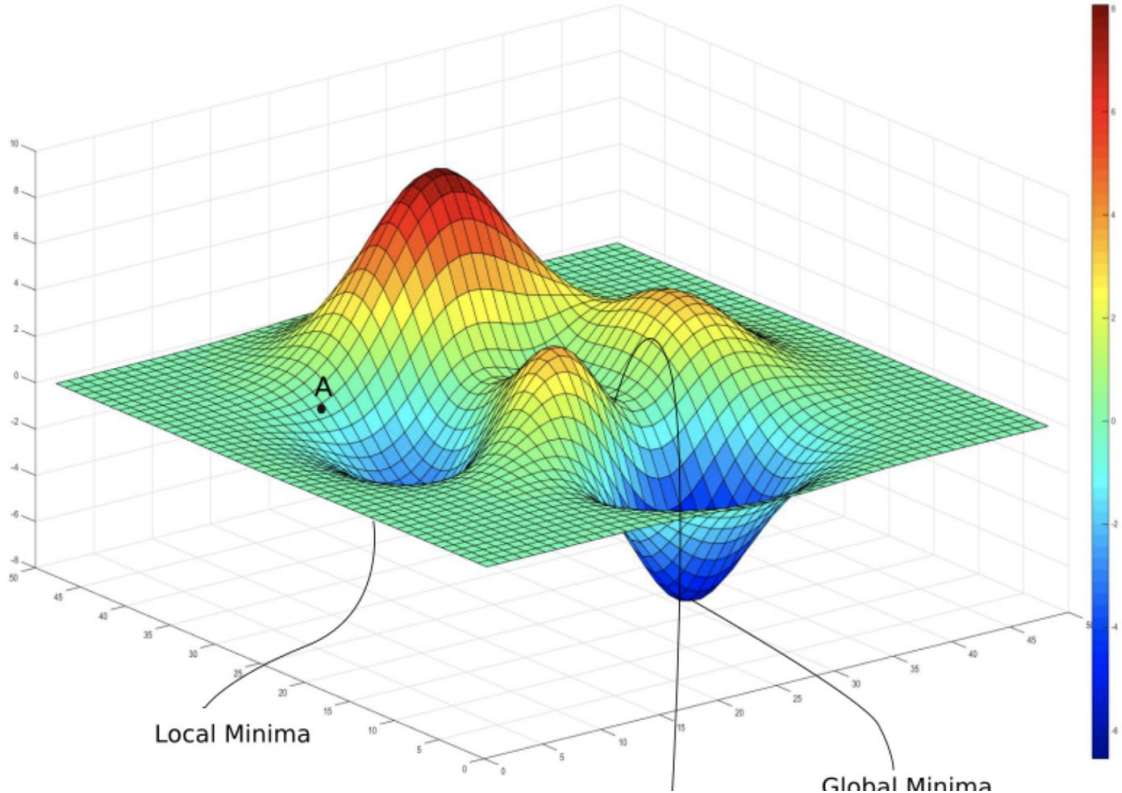


Going down the slope

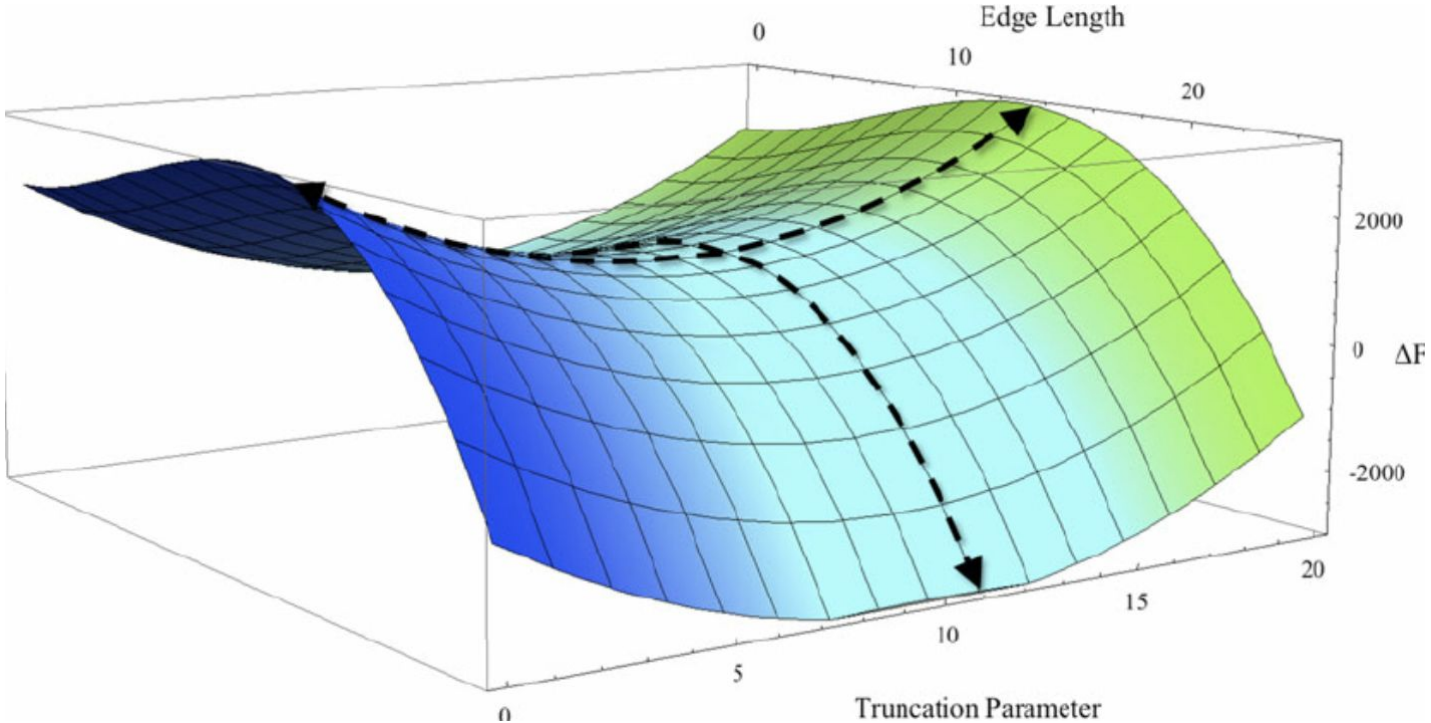




Local vs Global Minima



Saddle points



What is going on

How SGD Selects the Global Minima in Over-parameterized Learning: A Dynamical Stability Perspective

Lei Wu
School of Mathematical Sciences
Peking University
Beijing, 100081, P.R. China
leiwu@pku.edu.cn

Chao Ma
Program in Applied and Computational Mathematics
Princeton University
Princeton, NJ 08544, USA
chaom@princeton.edu

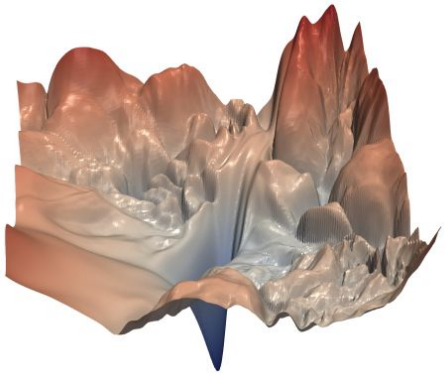
Weinan E
Department of Mathematics and Program in Applied and Computational Mathematics
Princeton University, Princeton, NJ 08544, USA and
Beijing Institute of Big Data Research, Beijing, 100081, P.R. China
weinan@math.princeton.edu

Bad Global Minima Exist and SGD Can Reach Them

Shengchao Liu
Quebec Artificial Intelligence Institute (Mila)
Université de Montréal
liusheng@mila.quebec

Dimitris Papailiopoulos
University of Wisconsin-Madison
dimitris@papail.io

Dimitris Achlioptas
University of Athens
optas@di.uoa.gr



Naive optimisation

- $J(\theta)$ is a function of all windows in the corpus
- So computing the gradient can be potentially very expensive

Naive optimisation

- $J(\theta)$ is a function of all windows in the corpus
- So computing the gradient can be potentially very expensive

- That is why we rely on stochastic gradient update
- This of course introduces the noise that can destroy the learning at some point
- Batch vs Online setting

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \boldsymbol{J}_{\text{minibatch}}(\boldsymbol{\theta})$$

Convergence rates

- For convex *f*, gradient descent with diminishing step sizes satisfies

$$f(x^{(k)}) - f^\star = O(1/\sqrt{k})$$

- When *f* is differentiable with Lipschitz gradient, we get for suitable fixed step sizes

$$f(x^{(k)}) - f^\star = O(1/k)$$

- For convex *f*, SGD with diminishing step sizes satisfies:

$$\mathbb{E}[f(x^{(k)})] - f^\star = O(1/\sqrt{k})$$

Adam Optimized [Kingma and Ba, 2013]

- Fixed learning step size can create problems.
- Adaptive Moment Estimation

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \boldsymbol{J}_{\text{minibatch}}(\boldsymbol{\theta})$$

- Momentum ‘trick’

$$\begin{aligned} \boldsymbol{m} &\leftarrow \beta_1 \boldsymbol{m} + (1 - \beta_1) \nabla_{\boldsymbol{\theta}} J_{\text{minibatch}}(\boldsymbol{\theta}) \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} - \alpha \boldsymbol{m} \end{aligned}$$

- Keeping track of the gradient magnitude

$$\begin{aligned} \boldsymbol{v} &\leftarrow \beta_2 \boldsymbol{v} + (1 - \beta_2) \left(\nabla_{\boldsymbol{\theta}} \boldsymbol{J}_{\text{minibatch}}(\boldsymbol{\theta}) \odot \nabla_{\boldsymbol{\theta}} \boldsymbol{J}_{\text{minibatch}}(\boldsymbol{\theta}) \right) \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} - \alpha \odot \boldsymbol{m} / \sqrt{\boldsymbol{v}} \end{aligned}$$

AdamW Optimizer [Loshchilov and Hutter, 2017]

- Decoupled weight decay regularization
- A GOTO optimizer a.d. 2022

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \boldsymbol{J}_{\text{minibatch}}(\boldsymbol{\theta})$$

Skip-gram in Practice

Naive softmax

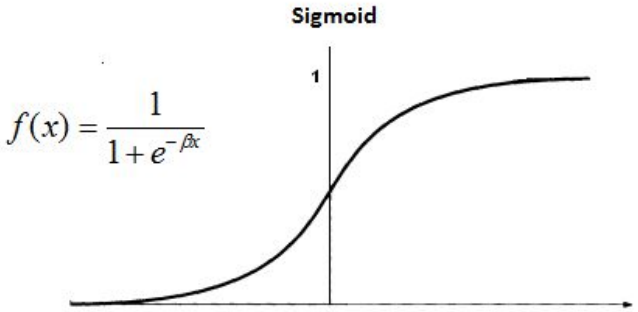
- The normalization factor is way too expensive to compute
- $$P(O = o|C = c) = \frac{\exp(\mathbf{u}_o^T \mathbf{v}_c)}{\sum_{w \in Vocab} \exp(\mathbf{u}_w^T \mathbf{v}_c)}.$$
- In practise, you go for **negative sampling**
- Main idea - train binary logistic classifier for a true pair (center word and the one in the context) against several noise pairs drawn randomly from the vocabulary.

The skip-gram model

Overall objective to maximize:

$$J_t(\theta) = \log \sigma(u_o^T v_c) + \sum_{i=1}^{k} \mathbb{E}_{j \sim P(w)} [\log \sigma(-u_j^T v_c)]$$

The sigmoid function looks like this:



The skip-gram model

- We typically take *k* negative samples
- Maximize probability that real outside word appears or minimize the probability that random words appear around center word

- How do we sample?

$$P(w) = U(w)^{(3/4)} / Z$$

- The power component makes less frequent words sampled more often.

Global knowledge versus local context windows

- Skip-gram may do better on the analogy task but they poorly utilize the statistics of the corpus

- Global models efficiently leverage statistical information but poor performance on analogy tasks

But why not capture co-occurrence counts directly?

- We can use a **co-occurrence matrix X**
- We could do either: windows vs full document
- Window: similarly to word2vec, let's use window around each word. These models captures both syntactic POS and semantic information

- Word-document co-occurrence matrix will give general topics leading to "Latent Semantic Analysis"

Example: Window based co-occurrence matrix

- Window length to choose (similarly to word2vec)
- Symmetric - it doesn't matter whether words are in the left or right context
- Example from the practical:
  Ala ma kota i psa.
  Ala lubi kota.

*	START	Ala	mieć	kot	i	pies	lubić	END
START	0	2	0	0	0	0	0	0
Ala	2	0	1	0	0	0	1	0
mieć	0	1	0	1	0	0	0	0
kot	0	0	1	0	1	0	1	1
i	0	0	0	1	0	1	0	0
pies	0	0	0	0	1	0	0	1
lubić	0	1	0	1	0	0	0	0
END	0	0	0	1	0	1	0	0

Problems?

- Increase in size with vocabulary

- Very high dimensional - requires a lot of storage
- You could go for sparse matrices but it only partially solves the problem

- Subsequent classification models have sparsity issues and hence models are less robust

Solution: low dimensional vectors

- Idea: store most of the important information in a fixed, small number of dimensions: a dense vector

- Usually 25-1000 dimensions, similar to word2vec

- How to reduce dimensionality?

Method 1: Dimensionality Reduction

- Brief look at Algebra 2!

- Singular Value Decomposition of co-occurrence matrix *X*

- Factorizes *X* into $X = U\Sigma V^T$ where *U* and *V* are orthonormal

- Keep only *k* singular values (the largest), in order to generalize new *X* is the best rank *k* approximation to the old *X* in terms of least squares

- Classical linear algebra result - expensive to compute for large matrices

Hacks to *X* (Rohde et al. 2005)

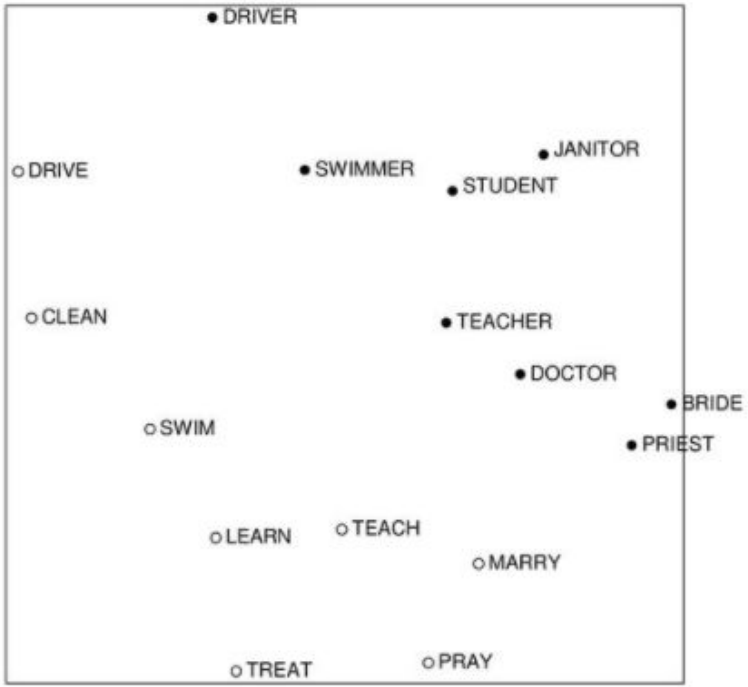
- Scaling (through log) the counts in the cells can help a lot!

- Problem: function words (the, he has) are too frequent -> syntax has **too much impact**.
- Some fixes:
  - Using ceiling function min(X, t), with t = 100.
  - Ignore them all.

- Ramped windows that count closer words more.

- Use Pearson correlations instead of counts, then set negative values to 0.

Interesting semantic patterns emerge in the vectors [Rohde et al. 2005]



Count based vs direct prediction

Two schools:

1) a) Fast training
   b) Efficient usage of statistics,
   c) Primarily used to capture word similarity
   d) High importance to large counts

2) a) Scales with corpus size
   b) Inefficient usage of statistics
   c) Capture complex patterns beyond word similarity

Count based vs direct prediction

- Can we combine both of these schools?

- Let’s look at ratios of co-occurrence probabilities - they should encode meaning components

Count based vs direct prediction

- Can we combine both of these schools?
- Let's look at ratios of co-occurrence probabilities - they should encode meaning components

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Encoding meaning in vector differences

- Q: How can we capture ratios of co-occurrences probabilities as linear meaning components in a word vector space?

- If you could make a dot product a log of co-occurrence probability:

$$w_i \cdot w_j = \log P(i|j)$$

- What we want to arrive at is:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j - \log X_{ij})^2$$

Encoding meaning in vector differences

- A general statement of the desired model

$$F(w_i, w_j, \tilde{w}_k) = \frac{\log P(k|i)}{\log P(k|j)}$$

- Since vector spaces are inherently linear structures, the most natural way to do this is with vector differences

$$F(w_i - w_j, \tilde{w}_k) = \frac{\log P(k|i)}{\log P(k|j)}$$

Encoding meaning in vector differences

- We don’t want to obfuscate the linear structure we are trying to capture

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{\log P(k|i)}{\log P(k|j)}$$

- The distinction between a word and a context word is arbitrary and we are free to exchange the two roles

- The *F* should be a homomorphism between two groups $(\mathbb{R}, +)$ and $(\mathbb{R}_{>0}, \times)$

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_k^T \tilde{w}_k)}$$

Encoding meaning in vector differences

- If *F* is exp the solution takes the form:

$$F(w_i^T \tilde{w}_k) = \log P(k|i) = \log(X_{ik}) - \log(X_i)$$

- We also need the symmetry which is blocked by the $\log(X_i)$ As it is independent of *k* we can push it into the bias. We arrive at:

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik})$$

Encoding meaning in vector differences [Pennington et al., 2014]

- Casting it as a LSP and adding a weighting function gives the final model

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

- Fast training
- Scalable to huge corpora
- Good performance even with small corpus and small vectors

Encoding meaning in vector differences [Pennington et al., 2014]

- Casting it as a LSP and adding a weighting function gives the final model

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

- Fast training
- Scalable to huge corpora
- Good performance even with small corpus and small vectors

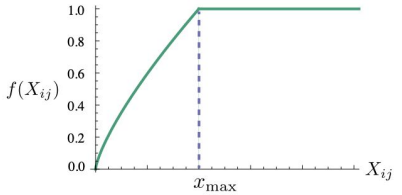


Figure 1: Weighting function f with $\alpha = 3/4$.

Encoding meaning in vector differences [Pennington et al. 2014]

Encoding meaning in vector differences [Pennington et al. 2014]

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

Evaluation issues

How to evaluate word vectors?

Related to general evaluation in NLP: intrinsic vs extrinsic

Intrinsic:

- Evaluation on a specific intermediate subtask
- Fast to compute
- Helps to understand that system
- Not clear if really helpful unless correlation to real taks is established

Extrinsic:

- Evaluation on a real task
- Can take a long time to compute accuracy
- Unclear if the subsystem is the problem or its interaction or other subsystems
- If replacing exactly on subsystems with another improves accuracy -> winning

Evaluation issues

Automatic Evaluation Leaderboard

Rank	Team ID	Best Spec #	Success Rate	Complete Rate	Book Rate	Inform P/R/F1	Turn(succ/all)
1	1	Submission3	93	95.2	94.6	84.1/96.2/88.1	12.5/12.7
2	2	Submission5	91.4	96.9	96.2	80.2/97.3/86.0	15.3/15.7
3	3	Submission1	90.8	94.4	96.7	81.0/95.4/85.9	13.4/13.6
4	4	Submission2	89.8	94.6	96.3	72.4/96.0/80.1	15.1/15.8
5	5	Submission2	83.3	88.5	89.1	81.1/90.3/83.5	13.5/13.8
6	6	Submission1	67.7	88.5	90.8	70.4/85.6/75.2	12.8/14.2
7	7	Submission4	57.8	87.1	85	68.7/81.6/72.6	13.7/16.4

Evaluation issues

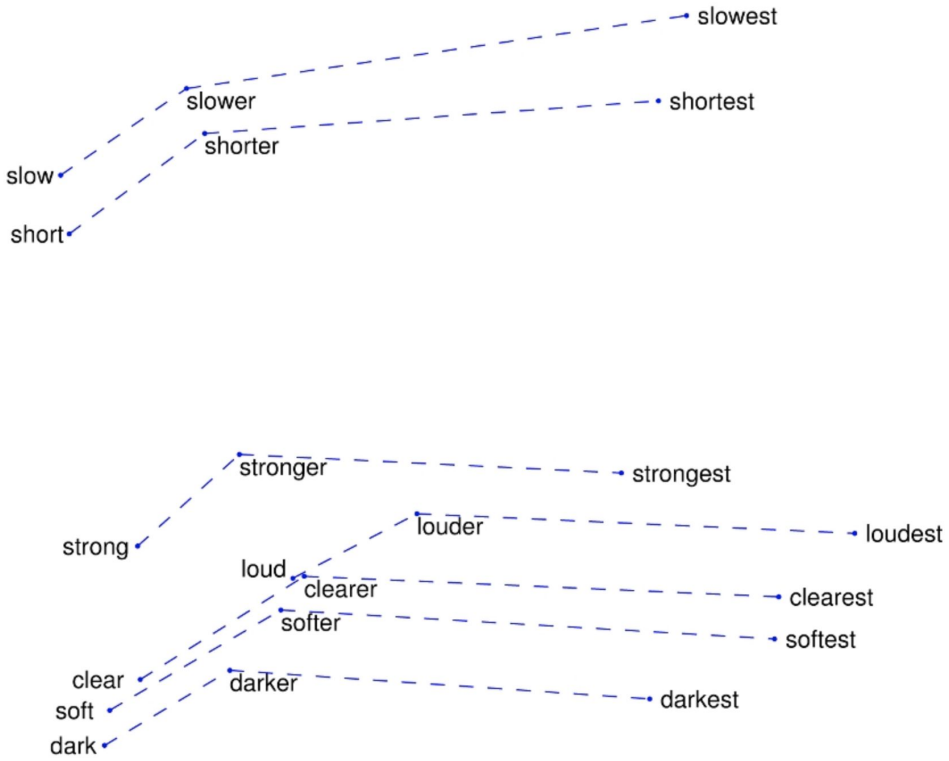
Automatic Evaluation

Rank	Team ID
1	1
2	2
3	3
4	4
5	5
6	6
7	7

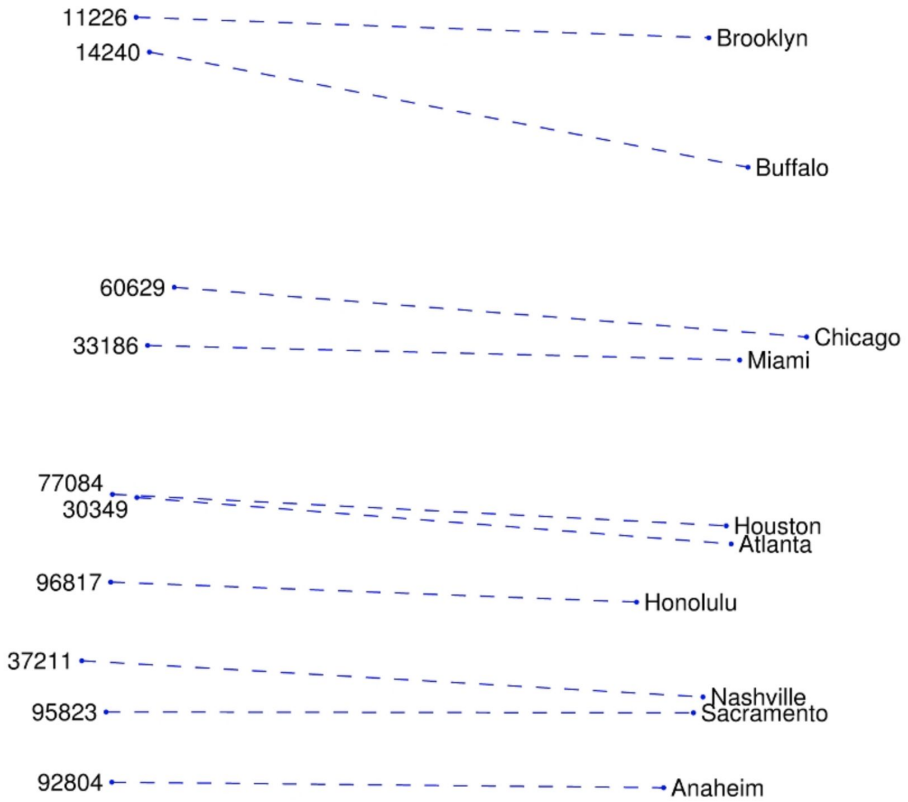
Human Evaluation Leaderboard

Rank	Team ID	Best Spec #	Average Success Rate	Success Rate w/ DB Grounding	Success Rate w/o DB Grounding	Language Understanding Score	Response Appropriateness Score	Turns
1	1	Submission5	74.8	70.2	79.4	4.54	4.47	18.5
1	2	Submission1	74.8	68.8	80.8	4.51	4.45	19.4
3	7	Submission4	72.3	62	82.6	4.53	4.41	17.1
4	6	Submission1	70.6	60.8	80.4	4.41	4.41	20.1
5	3	Submission3	67.8	60	75.6	4.56	4.42	21
6	4	Submission2	60.3	51.4	69.2	4.49	4.22	17.7
7	5	Submission2	58.4	50.4	66.4	4.15	4.06	19.7

Intrinsic word vector evaluation [Pennington et al., 2019]



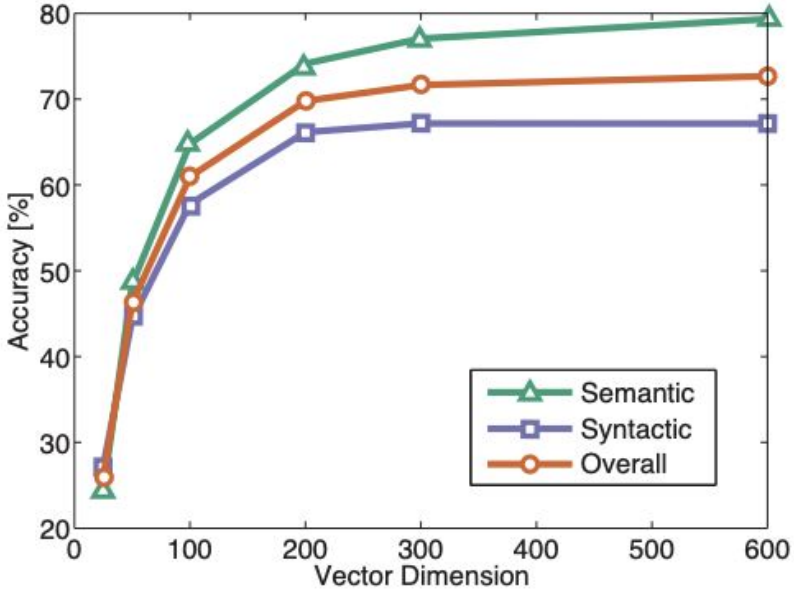
Intrinsic word vector evaluation [Pennington et al., 2019]

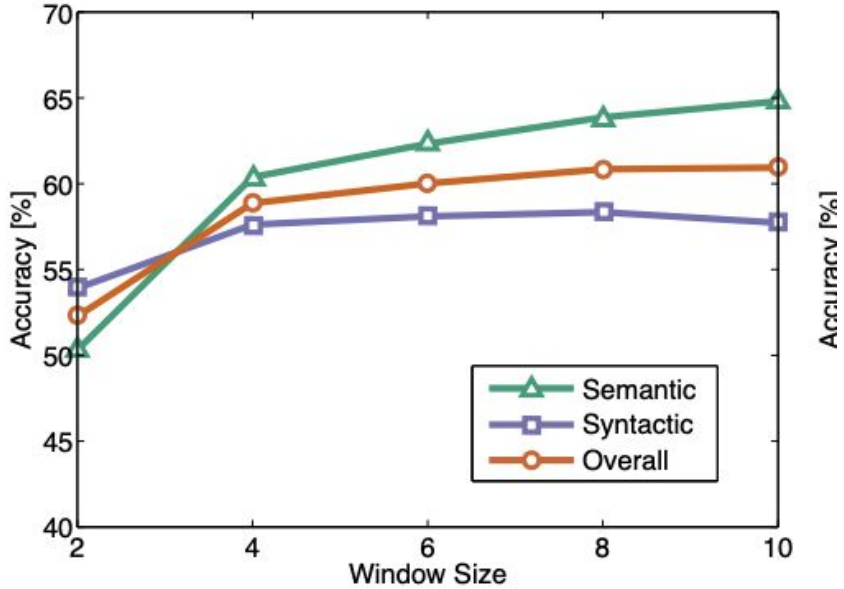


Word Analogy [Mikolov, 2013]

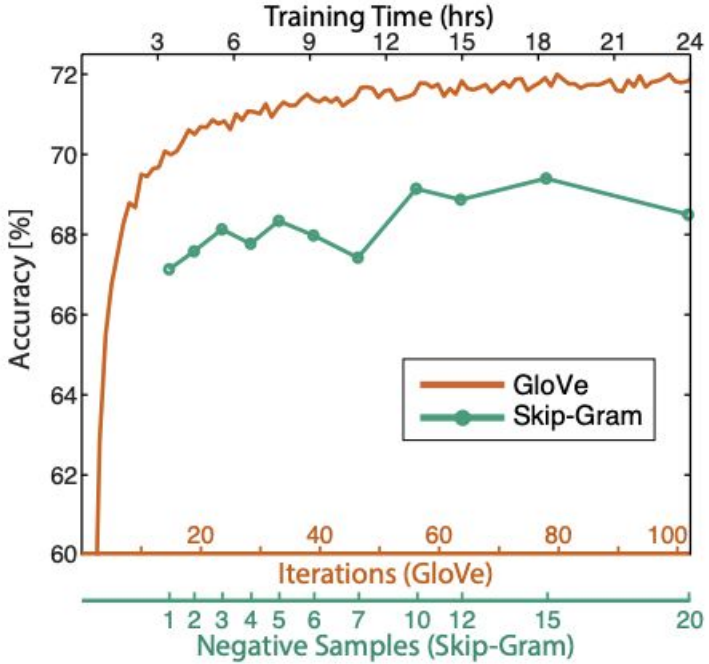
Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
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**

Analogy evaluation and hyperparameters



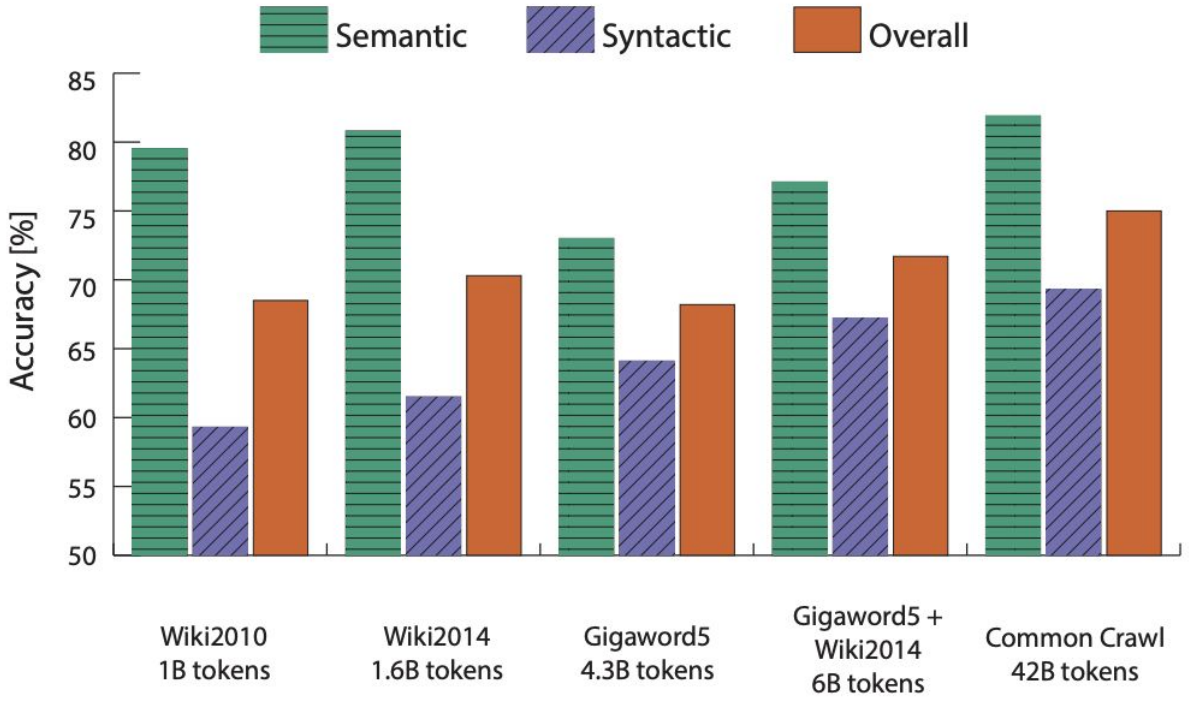


More training time matters



(b) GloVe vs Skip-Gram

Data quality!



Another intrinsic word vector evaluation

- Word vector distances and their correlation with human judgements
- Example dataset: WordSim353

Word 1	Word 2	Human (mean)	1	2	3	4	5	6	7	8	9	10	11	12	13
love	sex	6.77	9	6	8	8	7	8	8	4	7	2	6	7	8
tiger	cat	7.35	9	7	8	7	8	9	8.5	5	6	9	7	5	7
tiger	tiger	10.00	10	10	10	10	10	10	10	10	10	10	10	10	10
book	paper	7.46	8	8	7	7	8	9	7	6	7	8	9	4	9
computer	keyboard	7.62	8	7	9	9	8	8	7	7	6	8	10	3	9
computer	internet	7.58	8	6	9	8	8	8	7.5	7	7	7	9	5	9
plane	car	5.77	6	6	7	5	3	6	7	6	6	6	7	3	7
train	car	6.31	7	7.5	7.5	5	3	6	7	6	6	6	9	4	8

Another intrinsic word vector evaluation

Model	Size	WS353
SVD	6B	35.3
SVD-S	6B	56.5
SVD-L	6B	65.7
CBOW†	6B	57.2
SG†	6B	62.8
GloVe	6B	65.8
SVD-L	42B	74.0
GloVe	42B	**75.9**
CBOW*	100B	68.4

Word senses and word sense ambiguity

- Most words have lots of meanings!

- Especially common words
- And words that have existed for a long time

- Example: dzban

- Does one vector capture all these meanings or do we have a mess?

Linear algebraic structure of word senses with applications to polysemy [Arora et al., 2018]

- An average of different possible vectors for a word
- Sparse coding can recover vectors that approximately capture the senses

spring				
beginning	dampers	flower	creek	humid
until	brakes	flowers	brook	winters
months	suspension	flowering	river	summers
earlier	absorbers	fragrant	fork	ppen
year	wheels	lilies	piney	warm
last	damper	flowered	elk	temperatures

Extrinsic word vector evaluation

Extrinsic evaluation of word vectors - that’s what really matters for us and practitioners

Name entity recognition should help!

Extrinsic word vector evaluation

Extrinsic evaluation of word vectors - that's what really matters for us and practitioners

Name entity recognition should help!

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

Literature:

1. Pennington et al., 2014 - https://aclanthology.org/D14-1162/
2. WordSim353 - https://gabrilovich.com/resources/data/wordsim353/wordsim353.html
3. Arora et al., 2018 - https://arxiv.org/pdf/1601.03764.pdf