AdvNLP/E Lecture 4

Lexical and Distributional Semantics 2

Dr Julie Weeds, Spring 2026

Previously

- Lexical semantics (week 1)
  - word senses
  - semantic relationships
  - semantic similarity
- Distributional semantics (week 1)
  - bootstrapping semantics from context
  - cosine similarity
  - (positive) pointwise mutual information
  - Challenges
- N-gram Language modelling (week 2 and 3)
  - n-gram modelling
  - generalization and smoothing
  - Neural networks and neural language models
  - Feedforward neural networks
  - RNNs

Challenges for measures of distributional similarity

- Mixture of senses (lecture 1)
- Mixture of relationships (lecture 1)
- Sparsity (this time!)

Lecture 3 overview

- Part 1
  - Sparsity and Zipf's Law
  - Smoothing
  - Dimensionality Reduction
- Part 2
  - Word Embeddings
  - Word2Vec
  - GloVe
  - Composition

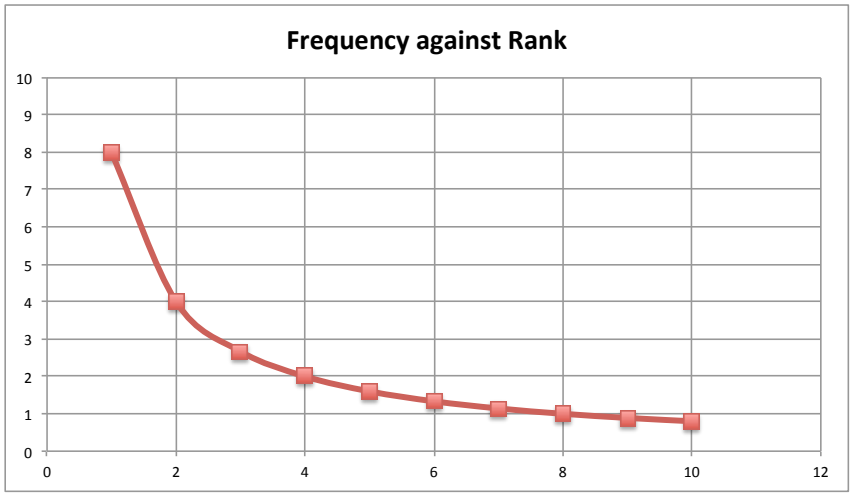
Sparsity

- Google News corpus contains **320M words (tokens)** with a vocabulary of **82K (types).**
- What does that suggest about the distribution of types?
- Mean frequency = 3900
- Is that enough?
- Maybe if it was a Normal distribution with a small standard deviation ….

Zipf's Law

- **"The product of the frequency of a word and its rank is approximately constant."**

Rank	8/Rank	Freq
1	8	8
2	4	4
3	2.667	3
4	2	2
5	1.6	2
6	1.333	1
7	1.143	1
8	1	1
9	0.889	1
10	0.8	1
	23.43	24



Hapax Legomena : words which only occur once. However large the corpus, these make up approximate half the vocabulary.

Consequences of Zipf's Law

- 82k dimensional co-occurrence vectors will be very sparse (lots of zeros)
- difficult to compare vectors because of all of this unseen stuff
- What can we do?

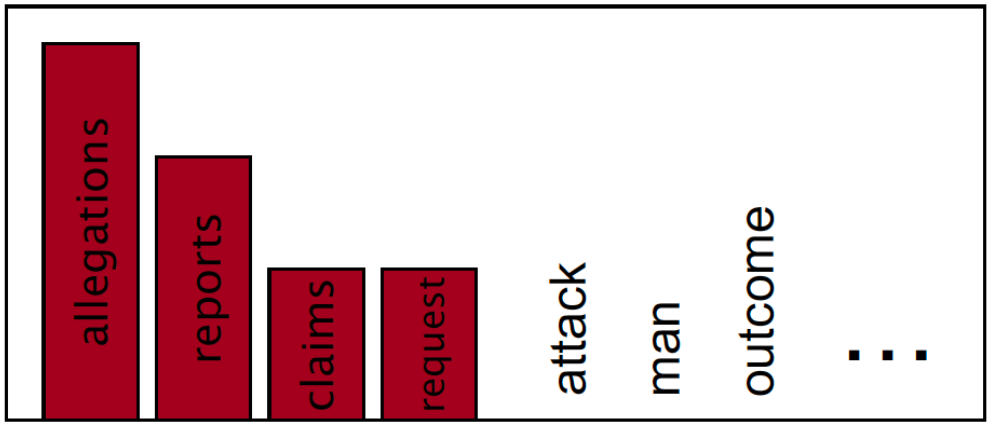
Possible solutions

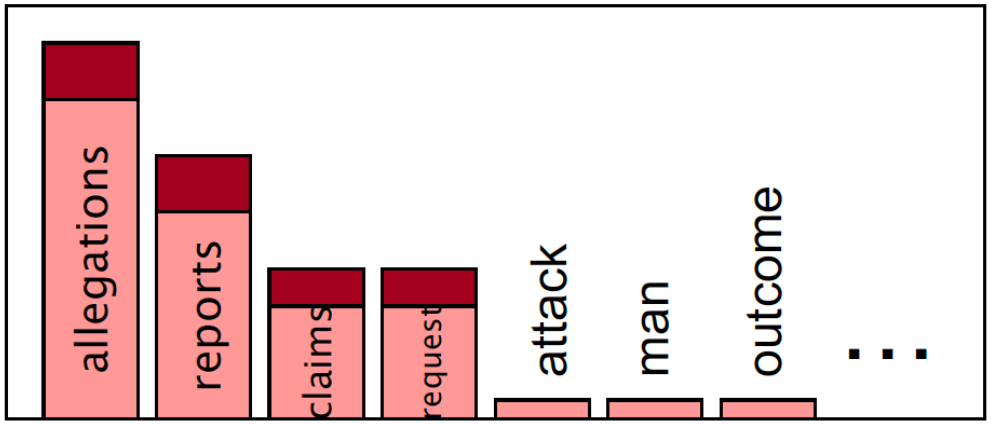
1. Smoothing
2. Dimensionality reduction
3. Word embeddings / fixed dimensionality representations from language models

Smoothing intuition

- Anything is possible – even previously unobserved events
- When we have sparse statistics, steal probability mass from observed events generalise to unobserved events

count (w| "denied")	smoothed count
allegations 3	allegations 2.5
reports 2	reports 1.5
claims 1	claims 0.5
request 1	request 0.5
	OTHER 2
total 7	**total 7**





Add-one smoothing

- Also called **Laplace smoothing**
- Pretend we saw each word one more time than we did (in each possible scenario)
- Just add on one to each count in the frequency matrix

feature	banana	meat	credit	Total
yellow	10	2	3	15
red	2	14	19	35
eat	20	9	1	30
spend	1	2	27	30
card	3	2	50	55
the	25	25	50	100
is	20	20	40	80
tiger	3	17	0	20
man	6	9	10	25
monkey	10	0	0	10
Total	**100**	**100**	**200**	**400**

feature	banana	meat	credit	Total
yellow	11	3	4	18
red	3	15	20	38
eat	21	10	2	33
spend	2	3	28	33
card	4	3	51	58
the	26	26	51	103
is	21	21	41	83
tiger	4	18	1	23
man	7	10	11	28
monkey	11	1	1	13
Total	**110**	**110**	**210**	**430**

Applying Add-1 Smoothing

- Can be very effective for some problems
  - If you did **AppliedNLP**, you might remember Add-1 Smoothing being used in the *Naïve Bayes text classifier*
  - Prevents a zero probability being assigned to a word occurring in a particular class just because it has never been seen with that class before
  - But here, the number of zeros isn't so huge
  - number of classes << size of English vocabulary
- Rarely used for co-occurrence data / language models
  - assigns too much mass to unseen co-occurrences (even add-k)
  - leads to massive, unwieldy models
  - and it wouldn't help us with the sparsity problem for distributional semantics anyway…..

Think about it

- Why is Add-1 smoothing unlikely to be very effective in helping us with the sparsity problem for distributional semantics?

Distributional smoothing

Dagan, Pereira and Lee (1994), define S(w) as the k-most similar words to w according to some distributional similarity measure. The co-occurrence probability of two words is then:

$$P_{SIM}(w_2|w_1) = \sum_{w' \epsilon S(w_1)} P(w_2|w') \frac{sim(w', w_1)}{\sum_{w' \epsilon S(w_1)} sim(w', w_1)}$$

Smoothing Example

- *snort* and *aardvark* are both rare words
- observed co-occurrence frequency is zero
- estimate probability of co-occurrence based on
  - co-occurrence of snort with neighbours of aardvark

$$P_{SIM}(snort|aardvark) = \sum_{w' \epsilon \{pig,\, animal, \ldots\}} P(snort|w') \frac{sim(w', aardvark)}{\sum sim(w', aardvark)}$$

  - and/or co-occurrence of aardvark with neighbours of snort

$$P_{SIM}(aardvark|snort) = \sum_{w' \epsilon \{grunt,\, sniff, \ldots\}} P(aardvark|w') \frac{sim(w', snort)}{\sum sim(w', snort)}$$

Distributional smoothing in practice

- Build sparse distributional representations of all words
- Automatically generate thesaurus (find k nearest neighbours of every word)
- Re-estimate co-occurrence probabilities of every pair of words using distributional smoothing formula
- Regenerate thesaurus

Think about it

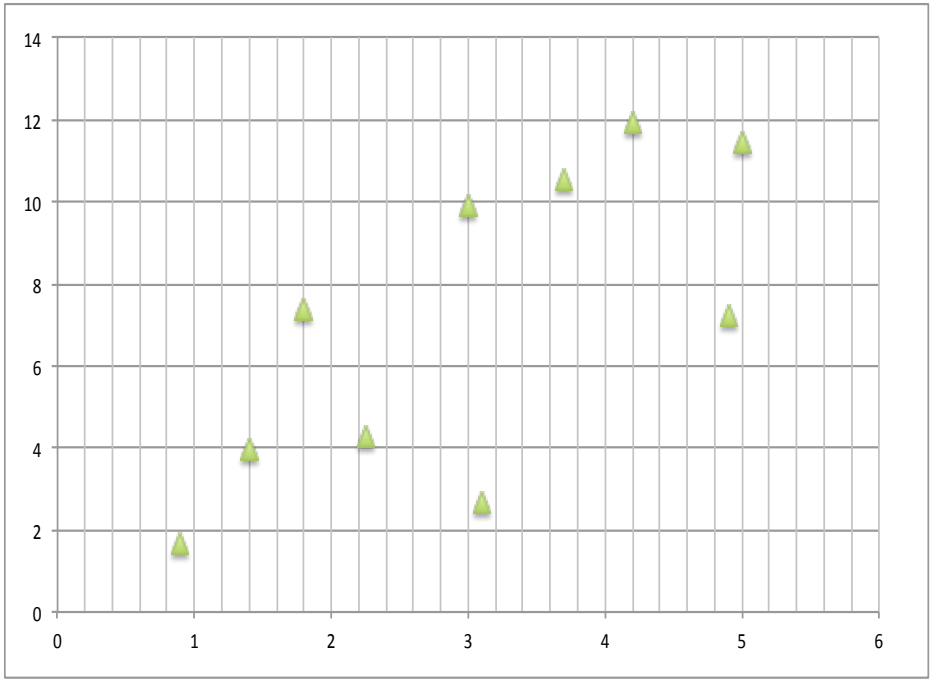
- What disadvantages can you see to this potential solution to the sparsity problem?

Dimensionality Reduction

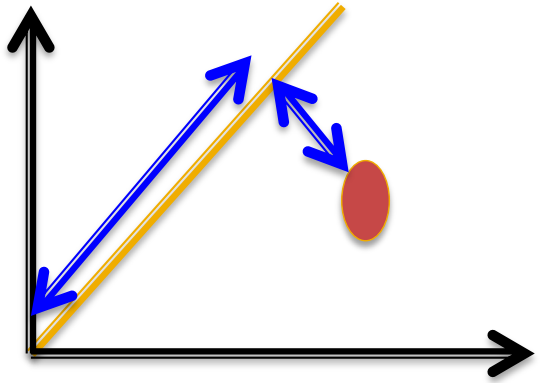
- Principal Component Analysis (PCA) – statistical technique
- Singular Value Decomposition (SVD) – matrix factorisation technique
- Non-negative Matrix Factorisation (NNMF)

Intuition: Find the k axes/bases on which there is most variation, transform the data so these are the bases, ignore other variations.

Principal Component Analysis (PCA) (2d->1d)



- Find the regression line (least squares linear regression)
- Each point can be represented by how far along the line it is and how far away from the line.



For a good line of best fit, most of the variation is contained within the principal component (distance along the line). Ignoring non-principal components leads to good approximation and good compression

PCA (2)

- In general, we want to reduce *n* dimensions (*n* very large) to *k* dimensions (e.g., *k* = 300)
- PCA can be applied iteratively to find all of the *k* dimensions (which are further required to be orthogonal)
- Variance in the data (not captured by principal components) can be computed as a measure of the loss of information

Latent Semantic Analysis (LSA)

- Deerwester et al. 1990, Landauer et al. 1998
- Represent text as a matrix X where each row is a unique word and each column is a document (or other context).
- Values are transformed frequencies (typically tf-idf or PPMI)
- Apply SVD where X is decomposed into a product of 3 matrices

SVD (Singular Value Decomposition)

$$X = W.S.P$$

$$\begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \\ 4 & 2 & 2 \\ 0 & 0 & 5 \end{pmatrix} = \begin{pmatrix} 2 & ? \\ ? & ? \\ ? & ? \\ ? & ? \\ ? & ? \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1.5 \end{pmatrix} \begin{pmatrix} 1 & ? & ? \\ 0 & ? & ? \end{pmatrix}$$

$$[5 \times 3] = [5 \times 2][2 \times 2][2 \times 3]$$

- S is a diagonal matrix, the dimensions of which define the reduced space
- W is the reduced dimensionality set of word vectors
- P is the reduced dimensionality set of document/co-occurrence vectors

- W, S and P found using standard matrix techniques e.g., by finding eigenvalues and eigenvectors
- Also known as eigendecomposition

NNMF

- Non-negative matrix factorization
- Many possible solutions to the factorization formulae
- NNMF has further constraint that all of the values in factorized space are non-negative
- Potentially leads to more interpretable spaces
  - why?

Using PCA, SVD, LSA or NNMF

ADVANTAGES

- can massively reduce dimensionality
- theoretically should capture similarities between words in the dimensions (co-occurrences with words describing dogs *could* all be on some "dogginess" dimension

DISADVANTAGES

- computationally expensive
- generally impossible to interpret/interrogate dimensions

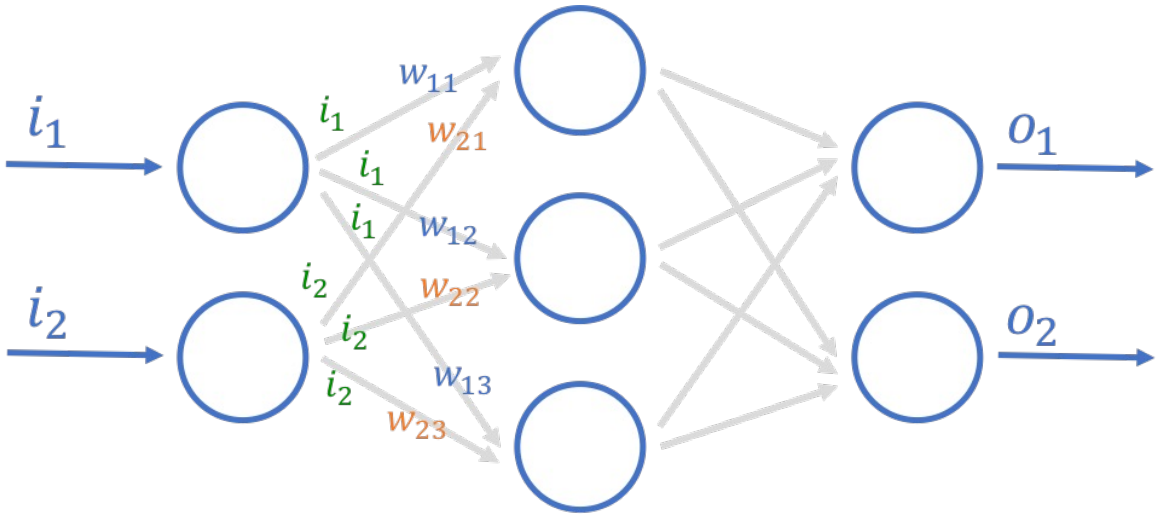
Week 4 Lecture: part 2

Word embeddings

Word Embeddings

- Start with the assumption that each word can be represented by a fixed set of dimensions
- Learn the best values to optimise performance on some task e.g., the log-likelihood of some training corpus
  - What's the probability of seeing this sequence of words?
  - What values would make it more likely that we would see this sequence of words (which has been observed and therefore we know it should be highly likely)?

A Simple NN



$$\begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \cdot \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} (w_{11} \times i_1) + (w_{21} \times i_2) \\ (w_{12} \times i_1) + (w_{22} \times i_2) \\ (w_{13} \times i_1) + (w_{23} \times i_2) \end{bmatrix}$$

One hot encoding

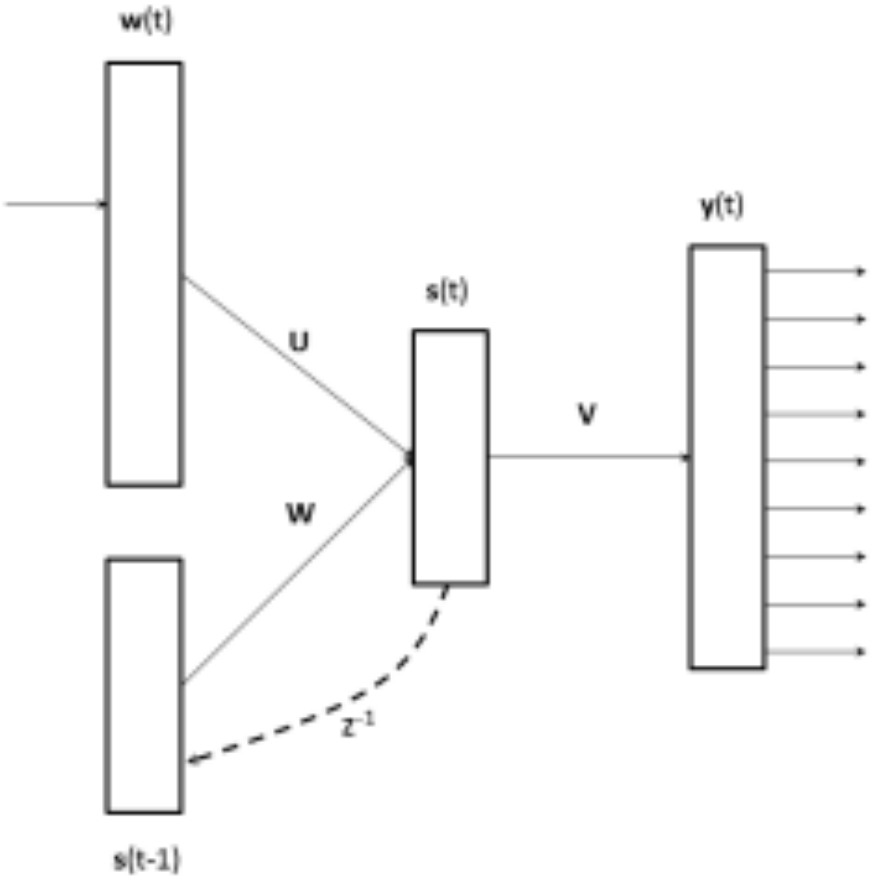
- Input space with V dimensions where V is size of vocabulary
- Exactly one of the input features is one, the others are zero
- This selects a column from the weight matrix

$$\begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \end{bmatrix} \qquad \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} w_{21} \\ w_{22} \\ w_{23} \end{bmatrix}$$

Embedding for word 1 (1,0)

Embedding for word 2 (0,1)

Recurrent Neural Network Language Model (Mikolov et al. NAACL 2013)

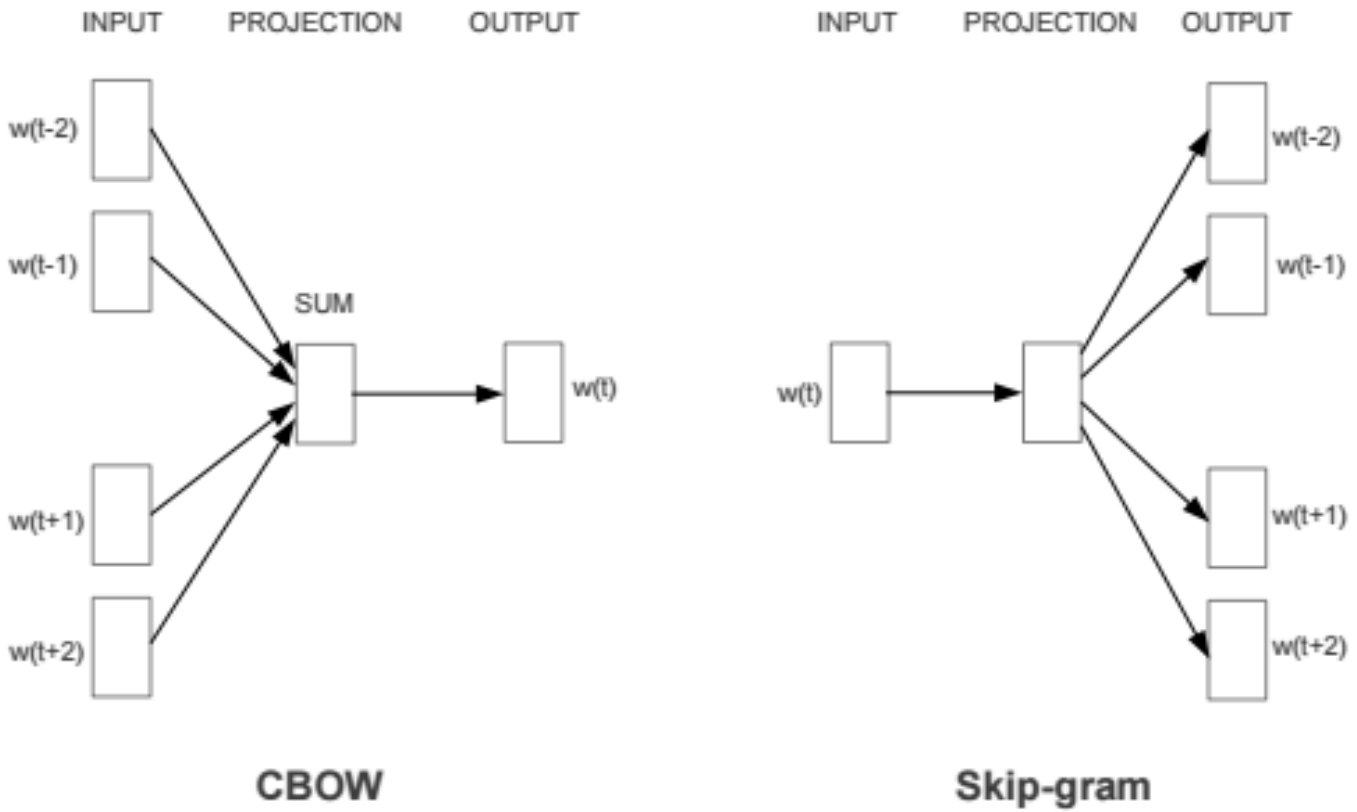


- Input vector, w(t) is the one-hot encoding of the current word
- Input word projected into embedding space via weight matrix U
- RNN so hidden layer also takes as input weighted representation of the hidden layer from the last time step
- Output layer is the same size as Vocab and is a probability distribution over next word

Word2Vec Intuition

- The current word in the corpus is referred to as the **target word**
- Use a **fixed length context window** (e.g., +/- 2 words either side of the target word
- Each word has 2 different embeddings
  - Remember – an embedding is a representation / vector / set of weights
  - One embedding used when it is a target word
  - The other embedding when it is a context word
- In CBOW, the context embeddings are used to predict the target embedding
- In Skip-gram, the target embedding is used to predict the context embeddings

Word2Vec (Mikolov et al. arXiv 2013)



Continuous Bag of Words (CBOW) : predict the current word given the context
Skip-gram : predict the context given the current word

Word2Vec training

- Embeddings are randomly initialised
- Iterate over text:
  - compute objective function for target word using positive and negative samples
- Stochastic gradient descent / backpropagation is used to 'improve' the weights
- Training continues .....

Gradient descent

- Randomly initialise parameter settings (W)
- Compute predictions
- Compute cost function (J)
- Compute (partial) derivatives of cost function with respect to parameters
- Back-propagate i.e. update parameters:

$$W = W - \alpha \frac{dJ}{dW}$$

learning rate

- Repeat until convergence / cost is acceptably small

Skip-gram Intuition

- Is word w_2 likely to show up near word w_1?
- Can we train a classifier on running text?
- Treat the target word and a neighbouring context word as positive examples
- Randomly sample other words in lexicon to get negative examples
- Train a classifier to distinguish those two cases

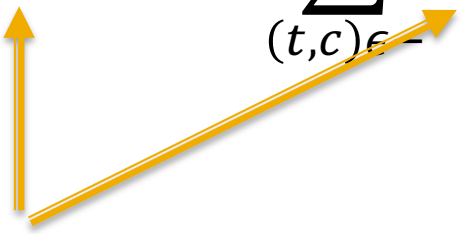
Skip-gram training

- Randomly initialise two sets of embeddings – each word in the vocabulary has a target embedding, t_w and a context embedding, c_w
- Iteratively shift target and context embeddings so that:
  - target embedding of word w is more like the context embeddings of words that occur nearby and less like the embeddings of words that don't occur nearby

Skip-gram Objective

$$J = \sum_{(t,c)\epsilon+} \log P(+|t,c) + \sum_{(t,c)\epsilon-} \log P(-|t,c)$$

$$J = \sum_{(t,c)\epsilon+} \sigma(c.t) + \sum_{(t,c)\epsilon-} \sigma(-c.t)$$



sigmoid function

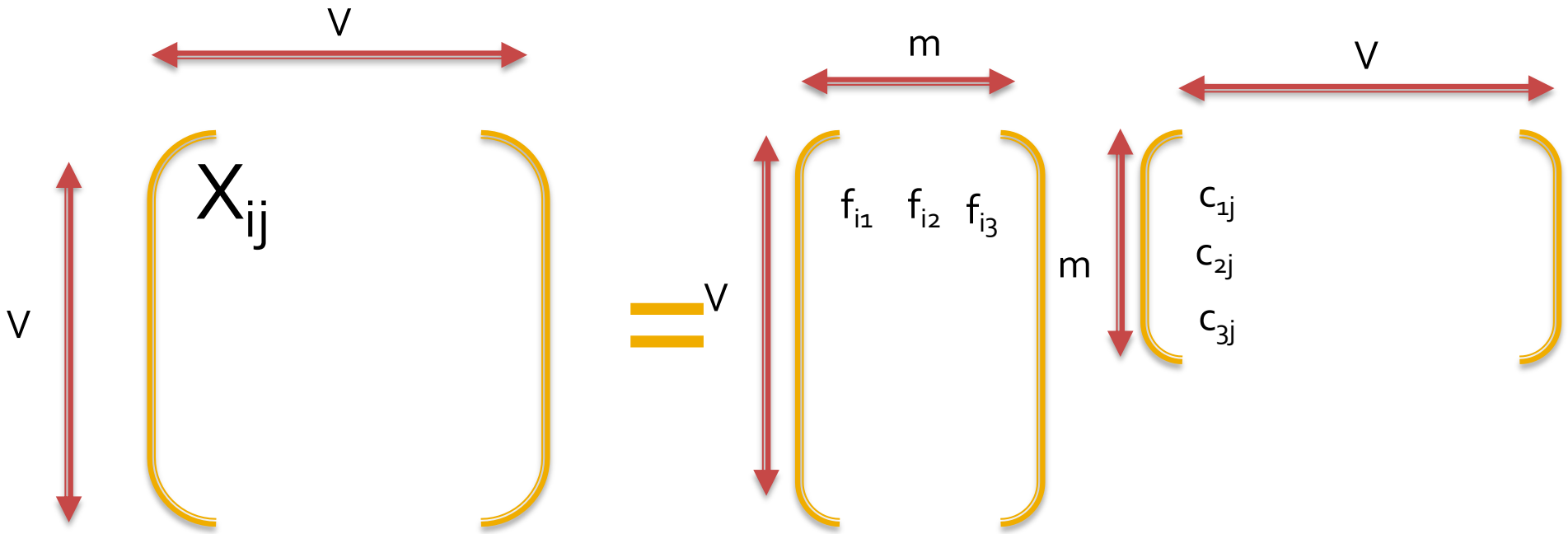
Word2Vec parameters

- Number of epochs
  - how many times is the corpus iterated over?
- Learning rate
  - how much are the weights updated each time?
- Batch size
  - how many predictions are considered before updates made?
- Window size
  - how many words of context are considered?

GloVe (Pennington et al. 2014)

- alternative approach to building low-dimensional dense word representations
- combines
  - mathematical simplicity of matrix factorization; with
  - speed and efficiency of neural techniques

GloVe Factorization

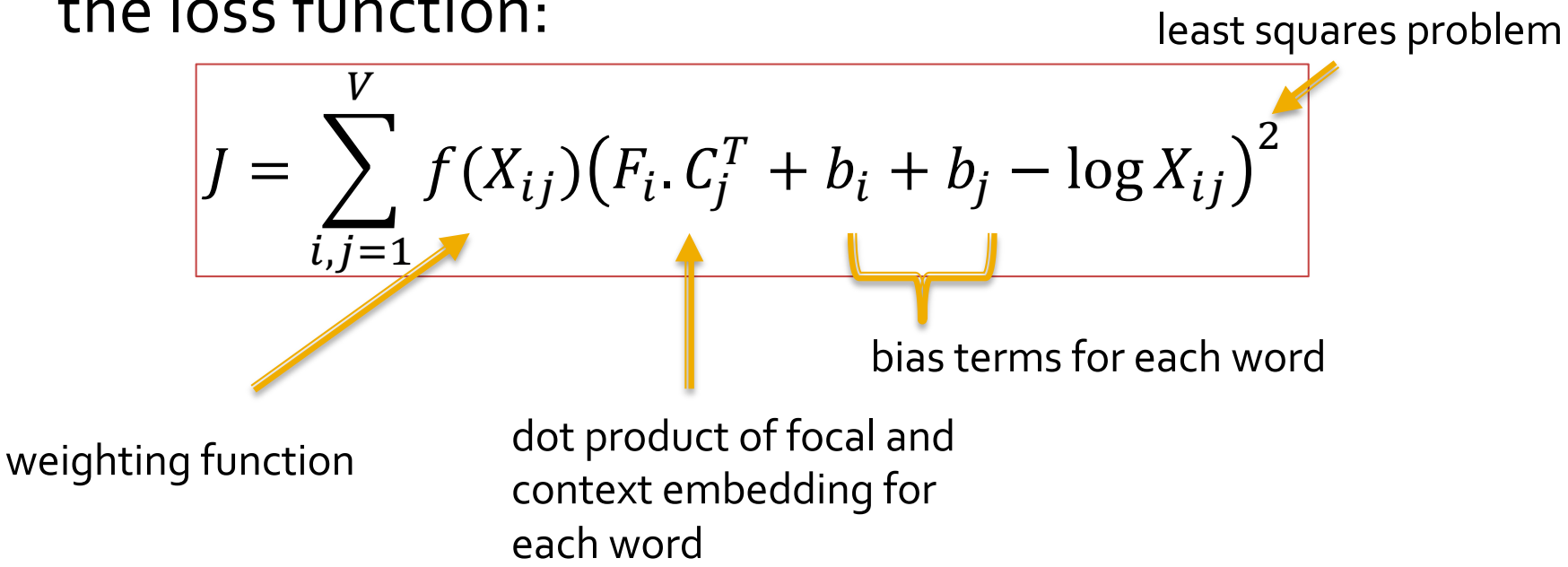


X_{ij} = co-occurrence frequency of word i and word j

Aim: find a set of m dimensional focal embeddings (F) and m-dimensional context embeddings (C) such that $X=FC^T$

GloVe Training

- Build standard co-occurrence matrix (based on context windows)
- Randomly initialise focal and context embeddings
- Batch sample co-occurrence matrix and minimise the loss function:



Weighting Function

$$f(x) = \begin{cases} (x/x_{max})^{\alpha} & \text{if } x < x_{max} \\ 1 \quad . & \text{otherwise} \end{cases}$$

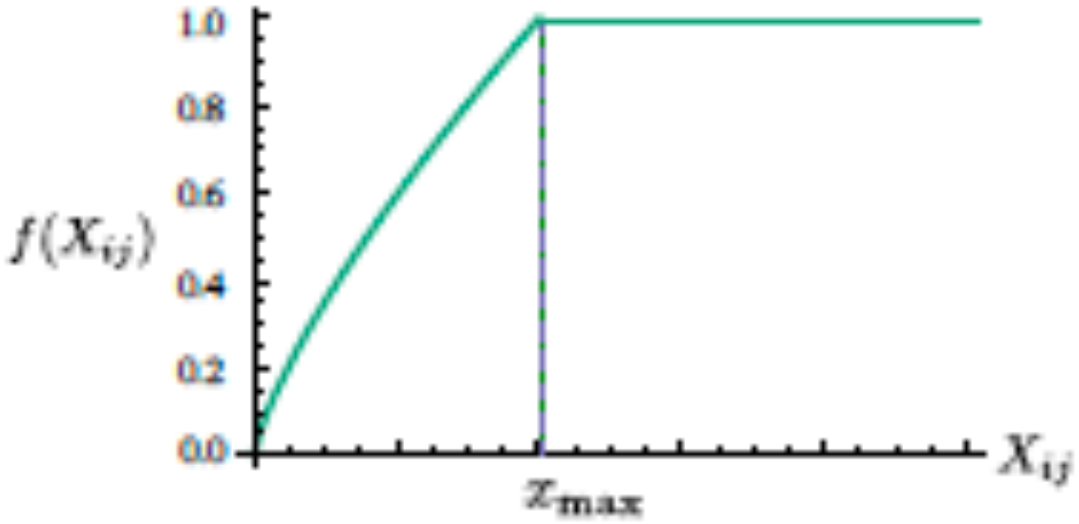


Figure 1: Weighting function f with $\alpha = 3/4$.

- Aim: reduce the impact of less reliable, lower frequency events
- x_{max} is typically around 100

Correlation with human judgements

Baroni et al. (2014)'s comparison of word2vec embeddings against best-performing count-based distributional representations on word similarity tasks

Dataset	Count-based (PPMI + NNMF)	Word2Vec
Rubinstein & Goodenough	0.74	0.84
WordSim353 (similarity)	0.70	0.80
WordSim353 (relatedness)	0.59	0.70
MEN dataset	0.72	0.80

"Don't Count, predict!"

What is so special about word2vec or GloVe?

- Omer & Levy 2014 showed that word2vec embeddings are an implicit factorisation of standard co-occurrence vectors with weights as PPMI
- BUT there are a lot more hyper-parameters which are tuneable (Omer & Levy 2015)

Hyper-parameters exploited by Word2Vec et al. (1)

1. dynamic context window: weight words by their distance from the focus word
2. subsampling: dilute/remove very frequent words
3. deleting rare words: ignore these when computing context windows

Hyper-parameters exploited by Word2Vec et al. (2)

4. context distribution smoothing: when computing PMI, smooth the unigram distribution
5. negative sampling/shifted PMI: shift all of the PMI values by k so only really important contexts are seen as features
6. Combining word and context vectors
7. Eigenvalue weighting in SVD

Disadvantages of low-density representations

- Dimensions are un-interpretable
  - there may be a 'dogginess' dimension
  - but no way of knowing which one
- Non-determinism
  - random initialisation
  - different solution found each time
  - even if globally optimal solution exists, space could be rotated on different runs
  - cannot compare embeddings produced on different runs / for different corpora
  - but it does provide a way of testing for statistical significance of results

Where next with distributional semantics … what about phrases?

- What is the similarity of "*nurse*" to "*man*"?
- What is the similarity of "*male nurse*" to "*man*"?
- How do we know the meaning of the word "*male nurse*"?
- Is this true for all (noun) phrases?

Composition: intersective

$$F(X\ Y) = F(X) \cap F(Y)$$

- Captures the intuition that features of the phrase must have occurred with ALL of the constituents of the phrase
- E.g., a feature is associated with "male nurse" if it is associated with "male" AND it is associated with "nurse"
- Commonly implemented with pointwise multiplication of vectors (or minimum)
- As more words are composed, will tend to zero vector (particularly if the vectors are sparse)

Composition: additive

$$F(X\ Y) = F(X) \cup F(Y)$$

- Captures the intuition that features of the phrase must have occurred with ONE of the constituents of the phrase
- E.g., a feature is associated with "male nurse" if it is associated with "male" OR it is associated with "nurse"
- Commonly implemented with pointwise addition of vectors (or maximum)
- Re-normalising makes this the centroid (or average) of the constituents
- As more words are composed, will tend to the centre of the semantic space

Evaluation of compositional models

- Very difficult
- Correlation with human similarity judgements for phrases
- Semantic similarity in context tasks
- Paraphrase recognition / textual entailment tasks
- Application-based evaluation

Challenges for compositional models

- Not all phrases are compositional
- Word order is important (surely?) but often totally ignored!
- Negation!!!!!
- Other functional words (e.g., determiners)
- What are the distributional contexts of a sentence anyway?

Reading

- Mikolov, Yih and Zweig (NAACL, 2013: Linguistic regularities in continuous space word representations

References

- Baroni, Dinu and Kruszewski (ACL, 2014): Don't count, predict! A systematic comparison of context-counting vs context-predicting semantic vectors
- Dagan, Pereira and Lee (ACL, 1994): Similarity-based estimation of word co-occurrence probabilities
- Levy and Goldberg (NIPS, 2014): Neural Word Embeddings as Implicit Matrix Factorization
- Levy, Goldberg and Dagan (TACL, 2015): Improving Distributional Similarity with Lessons Learned from Word Embeddings
- Landauer, Foltz and Laham (Discourse Processes, 1998): An Introduction to Latent Semantic Analysis
- Mikolov, Chen, Corrado and Dean (arXiv pre-print, 2013): Efficient Estimation of Word Representations in Vector Space
- Mikolov, Yih and Zweig (NAACL, 20130: Linguistic regularities in continuous space word representations
- Zweig and Burges (Microsoft Research Technical Report, 2011): The Microsoft Research sentence completion challenge