COMP9444: Neural Networks and Deep Learning

Week 7a. Word Vectors (Word2Vec and GloVe)

Sonit Singh
School of Computer Science and Engineering

July 09, 2024

Outline

- Statistical Language Processing
- *n*-gram models
- co-occurence matrix
- word representation
- Word2Vec
- GloVe
- word relationships

Word Embeddings

"Words that are used and occur in the same contexts tend to purport similar meanings."

Z. Harris (1954)

"You shall know a word by the company it keeps."

J.R. Firth (1957)

Aim of Word Embeddings:

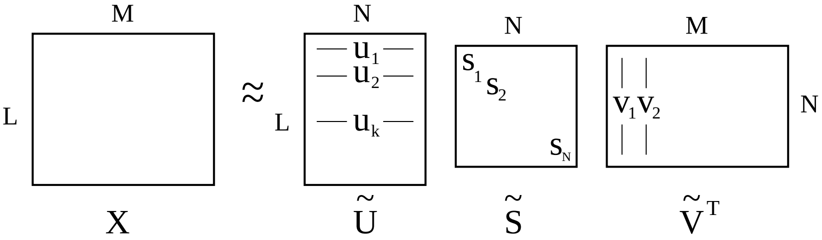
Find a vector representation of each word, such that words with nearby representations are likely to occur in similar contexts.

History of Word Embeddings

- Structuralist Linguistics (Firth, 1957)
- Recurrent Networks (Rumelhart, Hinton & Williams, 1986)
- Latent Semantic Analysis (Deerwester et al., 1990)
- Hyperspace Analogue to Language (Lund, Burgess & Atchley, 1995)
- Neural Probabilistic Language Models (Bengio, 2000)
- NLP (almost) from Scratch (Collobert et al., 2008)
- word2vec (Mikolov et al., 2013)
- GloVe (Pennington, Socher & Manning, 2014)

Singular Value Decomposition (Recap from Week 5b)

Co-occurrence matrix $X_{(L\times M)}$ can be decomposed as $\mathrm{X} = \mathrm{U\,S\,V}^{\mathrm{T}}$ where $U_{(L\times L)}$, $V_{(M\times M)}$ are unitary (all columns have unit length) and $S_{(L\times M)}$ is diagonal, with diagonal entries $s_1 \geq s_2 \geq \ldots \geq s_M \geq 0$



We can obtain an approximation for X of rank $N < M$ by truncating U to $\tilde{\mathrm{U}}_{(L\times N)}$, S to $\tilde{\mathrm{S}}_{(N\times N)}$ and V to $\tilde{\mathrm{V}}_{(N\times M)}$. The kth row of $\tilde{\mathrm{U}}$ then provides an N-dimensional vector representing the k^{th} word in the vocabulary.

Word2Vec and GloVe

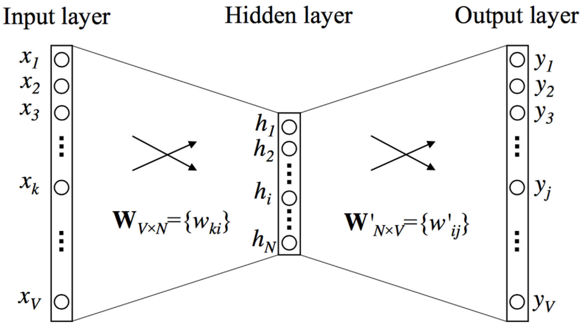
For language processing tasks, typically, L is the number of words in the vocabulary (about $60,000$) and M is either equal to L or, in the case of document classification, the number of documents in the collection. SVD is computationally expensive, proportional to $L \times M^2$ if $L \geq M$. Can we generate word vectors in a similar way but with less computation, and incrementally?

- Word2Vec
  - predictive model
  - maximize the probability of a word based on surrounding words
- GloVe
  - count-based model
  - reconstruct a close approximation to the co-occurrence matrix X

word2vec (Mikolov et al., 2013)

- **Idea**: predict rather than count
- Instead of counting how often each word *w* occurs near ”university” train a classifier on a **binary prediction task**:
  - Is *w* likely to show up near ”university”?
- We don’t actually care about this task
  - But we’ll take the learned classifier weights as the word embeddings
- Use running text as implicitly supervised training data
- No need for hand-labeled supervision

Word2Vec 1-word Context Model



The k^{th} row $\mathbf{v}_k$ of $\mathbf{W}$ is a representation of word k.
The j^{th} column $\mathbf{v}'_j$ of $\mathbf{W}'$ is an (alternative) representation of word j.
If the (1-hot) input is k, the linear sum at each output will be $u_j = {\mathbf{v}'_j}^{\mathrm{T}}\mathbf{v}_k$

Cost Function

Softmax can be used to turn these linear sums u_j into a probability distribution estimating the probability of word j occurring in the context of word k

$$\mathrm{prob}(j|k) = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})} = \frac{\exp(\, \mathbf{v}_j'^{\mathrm{T}} \mathbf{v}_k)}{\sum_{j'=1}^{V} \exp(\mathbf{v}_{j'}'^{\mathrm{T}} \mathbf{v}_k)}$$

We can treat the text as a sequence of numbers $w_1, w_2, \ldots, w_T$ where $w_i = j$ means that the i^{th} word in the text is the j^{th} word in the vocabulary.

We then seek to maximize the log probability

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \le r \le c, r \ne 0} \log\ \mathrm{prob}(w_{t+r}|w_t)$$

where c is the size of training context (which may depend on w_t)

Word2Vec issues

- Word2Vec is a linear model in the sense that there is no activation function at the hidden nodes
- this 1-word prediction model can be extended to multi-word prediction in two different ways:
  - Continuous Bag of Words
  - Skip-Gram
- need a computationally efficient alternative to Softmax (Why?)
  - Hierarchical Softmax
  - Negative Sampling
- need to sample frequent words less often

Word2Vec Weight Updates

If we assume the full softmax, and the correct output is j^*, then the cost function is

$$E = -u_{j^*} + \log \sum_{j'=1}^{V} \exp(u_{j'})$$

the output differentials are

$$e_j = \frac{\partial E}{\partial u_j} = -\delta_{jj^*} + \frac{\partial}{\partial u_j} \log \sum_{j'=1}^{V} \exp(u_{j'})$$

where

$$\delta_{jj^*} = \begin{cases} 1, & \text{if} \quad j = j^*, \\ 0, & \text{otherwise.} \end{cases}$$

Word2Vec Weight Updates

hidden-to-output differentials

$$\frac{\partial E}{\partial w'_{ij}} = \frac{\partial E}{\partial u_j}\frac{\partial u_j}{\partial w'_{ij}} = e_j\, h_i$$

hidden unit differentials

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j}\frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j\, w'_{ij}$$

input-to-hidden differentials

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i}\frac{\partial h_i}{\partial w_{ki}} = \sum_{j=1}^{V} e_j\, w'_{ij}\, x_k$$

CBOW vs Skip-Gram

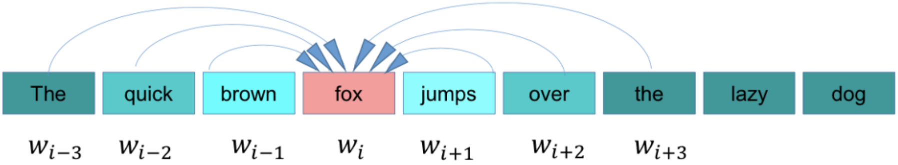


Figure: Continous Bag of Words (CBOW)

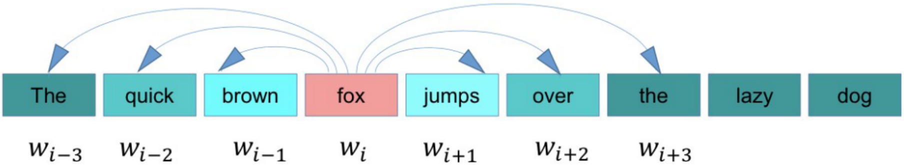
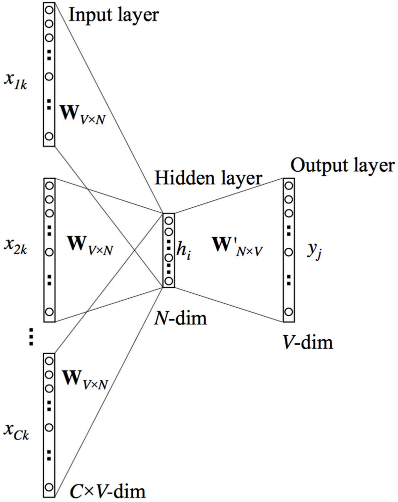


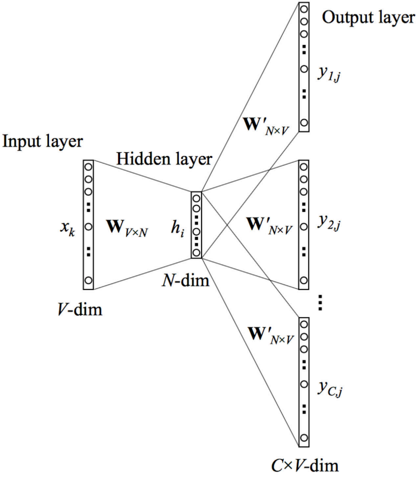
Figure: Skip-Gram model

Continuous Bag of Words (CBOW)



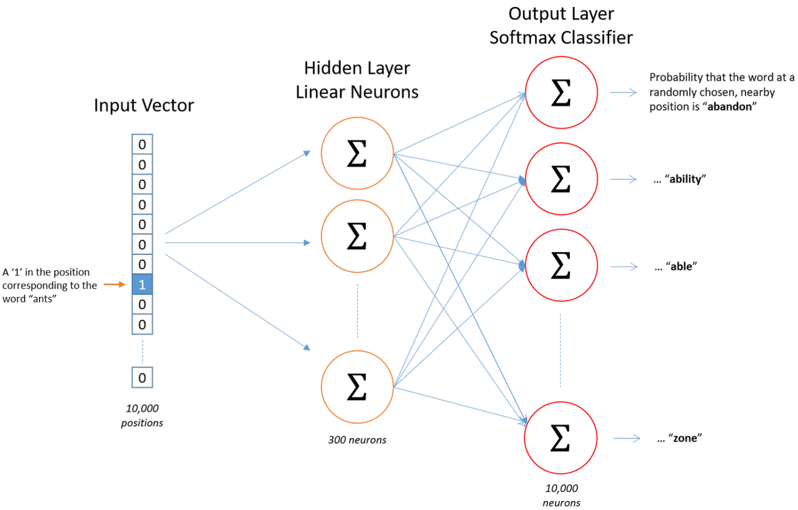
- If several context words are each used independently to predict the center word, the hidden activation becomes a sum (or average) over all the context words
- Note the difference between this and NetTalk – in word2vec (CBOW) all context words share the same input-to-hidden weights

Word2Vec Skip-Gram Model



- try to predict the context words, given the center word
- this skip-gram model is similar to CBOW, except that in this case a single input word is used to predict multiple context words
- all context words share the same hidden-to-output weights

Computational complexity in computing Softmax



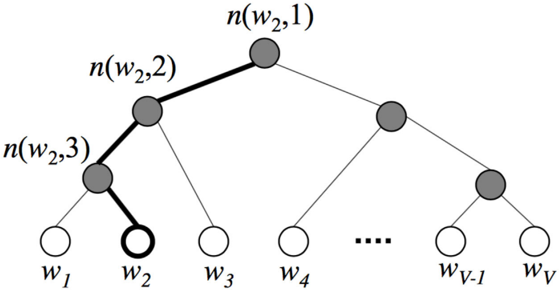
Hierarchical Softmax

- target words are organized in a Huffman-coded Binary Tree
- each output of the network corresponds to one branch point in the tree
- only those nodes that are visited along the path to the target word are evaluated (which is $\log_2(V)$ nodes on average)

word	count
fat	3
fridge	2
zebra	1
potato	3
and	14
in	7
today	4
kangaroo	2

$\longrightarrow$

Hierarchical Softmax



$$[n' = \mathrm{child}(n)] = \begin{cases} +1, & \text{if } n' \text{ is left child of node } n, \\ -1, & \text{otherwise.} \end{cases}$$

$$\sigma(u) = 1/(1 - \exp(-u))$$

$$\mathrm{prob}(w = w_t) = \prod_{j=1}^{L(w)-1} \sigma([n(w, j+1) = \mathrm{child}(n(w,j))]\mathbf{v}'_{n(w,j)}{}^{\mathrm{T}}\,\mathbf{h})$$

Negative Sampling

- Consider for each word w binary classifier: if given word C is good context for w, or not
- The idea of negative sampling is that we train the network to increase its estimation of the target word j^* and reduce its estimate not of all the words in the vocabulary but just a subset of them $\mathcal{W}_{\mathrm{neg}}$, drawn from an appropriate distribution.

$$E = -\log \sigma(\mathbf{v}'_{j^*}{}^{\mathrm{T}}\mathbf{h}) - \sum_{j \in \mathcal{W}_{\mathrm{neg}}} \log \sigma(-\mathbf{v}'_j{}^{\mathrm{T}}\mathbf{h})$$

- Given a sentence “**I love neural networks and deep learning**”
- Let’s assume **networks** as the centre word, the *positive* words (that are in the correct context words for the given centre word).
  Positive words: neural, love, deep learning
- **Negative words**: bus, auto, food, Japan, etc. ...

Negative Sampling

- The number of samples is 5-20 for small datasets, 2-5 for large datasets.
- Empirically, a good choice of the distribution from which to draw the negative samples is $P(w) = U(w)^{3/4}/Z$ where $U(w)$ is the unigram distribution determined by the previous word, and Z is a normalizing constant.
- Negative Sampling is a simplified version of Noise Constrastive Estimation (NCE).
  It is not guaranteed to produce a well-defined probability distribution,
  but in practice it does produce high-quality word embeddings.

Sub-sampling of Frequent Words

In order to diminish the influence of more frequent words, each word in the corpus is discarded with probability

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

where $f(w_i)$ is the frequency of word w_i and $t \sim 10^{-5}$ is an empirically determined threshold.

Word2Vec Summary)

- Go through each word in the whole corpus and
  - predict surrounding words for eac hword (skip-gram)
  - predict centre word, based on surrounding words (CBOW)
- This captures co-occurrence of words one at a time.

Why not capture co-occurrence counts directly?

Global Vectors (GloVe)

GloVe leverages the **co-occurrence statistics** of words in a corpus to learn the word embeddings.
Co-occurrence probabilities: Given two words *i* and *j* that occur in text, their co-occurrence probability is defined as the probability of seeing *i* in the context of *j*

$$P(j/i) = \frac{\text{count}(j \text{ in context of } i)}{\sum_k(\text{count}(k \text{ in context if } i))}$$

Claim: If we want to distinguish between two words, it is not enough to look at their co-occurrences, we need to look at the ratio of their co-occurrences with other words.

– Formalizing this intuition gives us an optimization problem

GloVe Objective

Notation:

- i: word, j: a context word
- w_i: The word embedding for i
- c_j: The context embedding for j
- b_i^w, b_j^c: Two bias terms: word and context specific
- X_{ij}: The number of times word i occurs in the context of j

Intuition:

1. Construct a word-content matrix whose $(i, j)^{th}$ entry is $log(X_{ij})$
2. Find vectors w_i, c_j and the biases b_i, c_j such that the dot product of the vectors added to the biases approximates the matrix entries

GloVe Objective

Notation:

- i: word, j: a context word
- w_i: The word embedding for i
- c_j: The context embedding for j
- b_i^w, b_j^c: Two bias terms: word and context specific
- X_{ij}: The number of times word i occurs in the context of j

Objective:

$$J(\theta) = \sum_{i,j=1}^{|V|} (w_i^T c_j + b_i + b_j - log\ X_{ij})^2$$

Problem: Pairs that frequently co-occur tend to dominate the objective

Solution: Correct for this by adding an extra term that prevents this

GloVe Objective

Notation:

- i: word, j: a context word
- w_i: The word embedding for i
- c_j: The context embedding for j
- b_i^w, b_j^c: Two bias terms: word and context specific
- X_{ij}: The number of times word i occurs in the context of j

Objective:

$$J(\theta) = \sum_{i,j=1}^{|V|} f(X_{ij})(w_i^T c_j + b_i + b_j - log\ X_{ij})^2$$

f: A weighting function that assigns lower relative importance to frequent co-occurrences

GloVe Objective

Objective:

$$J(\theta) = \sum_{i,j=1}^{|V|} f(X_{ij})(w_i^T c_j + b_i + b_j - log\ X_{ij})^2$$

Learn word vectors such that their dot product equals the logarithm of the words' probabilty of co-occurrence,
i.e., For each pair of words i, j, minimise distance between **dot product** and overall **log count** in corpus

GloVe vs Word2Vec

- GloVe (Count-based)
  - Fast training (if corpus is not too large)
  - Efficient use of statistics
  - Primarily used to capture word similarity
  - Good performance with less dimensionality and even less corpora

- Word2Vec (Prediction-based)
  - Scales with corpus size
  - Inefficient use of stats
  - Can capture complex patterns beyond word similarity
  - Improved performance on extrinsic tasks

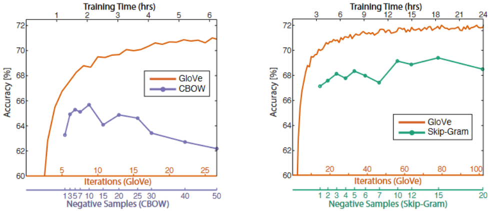
GloVe

Table 2: Results on the word analogy task, given as percent accuracy. Underlined scores are best within groups of similarly-sized models; bold scores are best overall. HPCA vectors are publicly available[2]; (i)vLBL results are from (Mnih et al., 2013); skip-gram (SG) and CBOW results are from (Mikolov et al., 2013a,b); we trained SG† and CBOW† using the `word2vec` tool[3]. See text for details and a description of the SVD models.

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**

Table 3: Spearman rank correlation on word similarity tasks. All vectors are 300-dimensional. The CBOW* vectors are from the `word2vec` website and differ in that they contain phrase vectors.

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



(a) GloVe vs CBOW (b) GloVe vs Skip-Gram

Figure 4: Overall accuracy on the word analogy task as a function of training time, which is governed by the number of iterations for GloVe and by the number of negative samples for CBOW (a) and skip-gram (b). In all cases, we train 300-dimensional vectors on the same 6B token corpus (Wikipedia 2014 + Gigaword 5) with the same 400,000 word vocabulary, and use a symmetric context window of size 10.

Sentence Completion Task

Q1. Seeing the pictures of our old home made me feel .... and nostalgic.

A. fastidious
B. indignant
C. wistful
D. conciliatory

Q2. Because the House had the votes to override a presidential veto, the President has no choice but to ....

A. object
B. abdicate
C. abstain
D. compromise

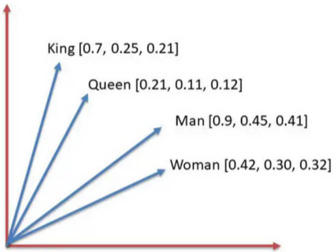
(use model to choose which word is most likely to occur in this context)

Linguistic Regularities

King + Woman - Man $\simeq$ Queen

More generally,
A is to B as C is to ??

$$d = \operatorname{argmax}_x \frac{(v_c + v_b - v_a)^{\mathrm{T}} v_x}{||v_c + v_b - v_a||}$$



King – Man + Woman = [0.7, 0.25, 0.21] - [0.9, 0.45, 0.41] + [0.42, 0.30, 0.32]

= [0.22, 0.1, 0.12] ≈ Queen

Word Analogy Task

Q1. **evening** is to **morning** as **dinner** is to ....

A. breakfast

B. soup

C. coffee

D. time

Q2. **bow** is to **arrow** as .... is to **bullet**

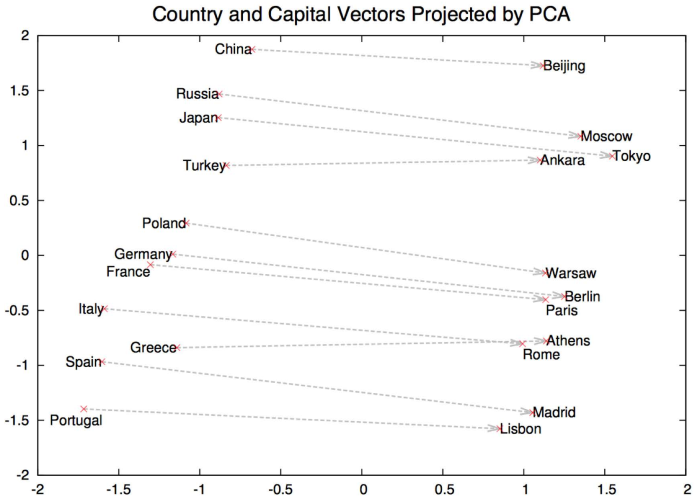
A. defend

B. lead

C. shoot

D. gun

Capital Cities



Word Analogies

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Word Relationship

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Summary

- *Word vectors*, also sometimes called *word embeddings* or *word representations* are distributed representations of words.
- Two kinds of embeddings
  - Sparse vectors: Words are represented by a simple function of the counts of nearby words.
  - Dense vectors: Representation is created by training a classifier to distinguish nearby and far-away words
- The ***contexts*** in which a word appears tells us a lot about what it means. Distributional similarities use the set of contexts in which words appear to measure their similarity
- Word2Vec and GloVe are two important dense representations of words.
- Various choice:
  - dimensionality of embeddings (50, 100, 200, 300, 500)
  - scale, quality and type of text to get word embeddings
  - size of the context window

References

- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'13). Curran Associates Inc., Red Hook, NY, USA, 3111–3119.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Xin Rong. 2014. word2vec Parameter Learning Explained, arXiv link.
- GloVe https://nlp.stanford.edu/projects/glove/
- On Word Embeddings
- Word Embedding Demo: Tutorial

References

- Bandyopadhyay, S., Xu, J., Pawar, N. and Touretzky, D. 2022. Interactive Visualizations of Word Embeddings for K-12 Students. Proceedings of the AAAI Conference on Artificial Intelligence. 36, 11 (Jun. 2022), 12713-12720. DOI:https://doi.org/10.1609/aaai.v36i11.21548.
- TensorFlow Word Embeddings
- Word Embeddings in PyTorch
- Word2Vec Demo
- Word Embedding Demo

Important Blogs and Demos

- Word2Vec
- The Illustrated Word2Vec by Jay Alammar
- Word2Vec Negative Sampling Explained
- Hierarchical Softmax
- GloVe Vectors
- Image captioning
- Text Classification with Torchtext library
- Seq2Seq model for Machine Translation