Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

October 16, 2019

Natural Language Processing

Anoop Sarkar
anoopsarkar.github.io/nlp-class

Simon Fraser University

Part 1: Word Vectors

One-hot vectors

Singular Value Decomposition

Word2Vec

GloVe

One-hot vectors

- Let $|V|$ be the size of the vocabulary
- Assign each word to a unique index from $1 \ldots |V|$
- e.g. *aarvark* is 1, *a* is 2, etc.
- Represent each word as as a $\mathbb{R}^{|V| \times 1}$
- The vector has one at index i and all other values are 0

One-hot vectors

Figure from [1]

$$w^{aardvark} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{a} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{at} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots \; w^{zebra} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

One-hot vectors

- Problems with similarity over one-hot vectors
- Consider similarity between words as dot product between their word vectors:

$$w_{\mathrm{cat}} \cdot w_{\mathrm{dog}} = w_{\mathrm{joker}} \cdot w_{\mathrm{dog}} = 0$$

- Idea: reduce the size of the large sparse one-hot vector
- Embed large sparse vector into a dense subspace.

One-hot vectors

Singular Value Decomposition

Word2Vec

GloVe

Window based co-occurrence matrix

- Assume a window around each word (window size 2, 5, ...)
- Collect co-occurrence counts for each pair of words in the vocabulary.
- Create a matrix X where each element $X_{i,j} = c(w_i, w_j)$
- $c(w_i, w_j)$ is the number of times we observe word w_i and w_j together
- X is going to be very sparse (lots of zeroes)

Window based co-occurrence matrix

DocID:	Title
doc0	Human machine interface for Lab ABC computer applications
doc1	A survey of user opinion of computer system response time
doc2	The EPS user interface management system
doc3	System and human system engineering testing of EPS
doc4	Relation of user-perceived response time to error measurement
doc5	The generation of random, binary, unordered trees
doc6	The intersection graph of paths in trees
doc7	Graph minors IV: Widths of trees and well-quasi-ordering
doc8	Graph minors: A survey

Window based co-occurrence matrix

	and	minors	generation	testing	engineering	computer	relation	human	measurement
and	0	1	0	1	1	0	0	1	0
minors	1	0	0	0	0	0	0	0	0
generation	0	0	0	0	0	0	0	0	0
testing	1	0	0	0	1	0	0	1	0
engineering	1	0	0	1	0	0	0	1	0
computer	0	0	0	0	0	0	0	1	0
relation	0	0	0	0	0	0	0	0	1
human	1	0	0	1	1	1	0	0	0
measurement	0	0	0	0	0	0	1	0	0
unordered	0	0	1	0	0	0	0	0	0

Singular Value Decomposition

- Collect $X = |V| \times |V|$ word co-occurrence matrix.
- Apply SVD on X to get $X = USV^T$

Transpose

Transpose of V is V^T which switches the row and column of V

- Select first k columns of U to get k-dimensional vectors
- The matrix S is a diagonal matrix with entries $\sigma_1, \ldots, \sigma_i, \ldots, \sigma_{|V|}$

Variance

The amount of variance captured by the first k dimensions is given by

$$\frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{|V|} \sigma_i}$$

Dimensionality reduction with SVD

Figure from [1]

Applying SVD to X:

$$
\begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} & & \\ & X & \\ & & \end{bmatrix} \end{array}
= \begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix} \end{array}
\begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \end{array}
\begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix} \end{array}
$$

Dimensionality reduction with SVD

Figure from [1]

Reducing dimensionality by selecting first k singular vectors:

$$\begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} & & \\ & \hat{X} & \\ & & \end{bmatrix}\end{array} = \begin{array}{cc} & k \\ |V| & \begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix}\end{array} \begin{array}{cc} & k \\ k & \begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}\end{array} \begin{array}{cc} & |V| \\ k & \begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix}\end{array}$$

Why SVD is not the ideal solution

- Computational complexity is high $\mathcal{O}(|V|^3)$
- Cannot be trained as part of a larger model.
- It is not a component that can be part of a larger neural network
- Cannot be trained discriminatively for a particular task

One-hot vectors

Singular Value Decomposition

Word2Vec

GloVe

Word2Vec

- Word2Vec is a family of model + learning algorithm
- The goal is to learn dense word vectors

Continuous bag of words

- Takes the average of the context; predicts the target word
- Trained with gradient descent on cross entropy loss for word prediction

Skip-gram

- Considers each context word independently and constructs (target-word, context-word) pairs
- Trained using negative sampling and loss on predicting good vs. bad pairs

Word2Vec: Continuous Bag of Words

CBOW

the general __________ the troops

Predicting a center word from the surrounding words
(also window-based)

For each word we want to learn two vectors:

- $v_i \in \mathbb{R}^k$ (input vector) when the word w_i is in the context
- $u_i \in \mathbb{R}^k$ (output vector) when the word u_i is in the center

Word2Vec: Continuous Bag of Words

Algorithm

the general __________ the troops

$v_{\text{the}}\, v_{\text{general}}$ $v_{\text{the}}\, v_{\text{troops}}$

- Average the context vectors:
$$\hat{v} = \frac{v_{\text{the}} + v_{\text{general}} + v_{\text{the}} + v_{\text{troops}}}{4}$$
- For each word $i \in V$ we have a word vector $u_i \in \mathbb{R}^k$
- Compute the dot product $z_i = u_i \cdot \hat{v}$
- Convert $z_i \in \mathbb{R}$ into a probability:
$$\hat{y}_i = \frac{exp(z_i)}{\sum_{k=1}^{|V|} exp(z_k)}$$
- If the correct center word is w_i then the max should be $\hat{y}_i$.

Word2Vec: Continuous Bag of Words

the general ___________ the troops

$v_{\text{the}}\, v_{\text{general}}$ $v_{\text{the}}\, v_{\text{troops}}$

- Average the context vectors to get $\hat{v}$
- Let matrix $U = [u_1, \ldots, u_{|V|}] \in \mathbb{R}^{|V| \times k}$ with word vectors $u_i \in \mathbb{R}^k$
- Compute the matrix product $z = U \cdot \hat{v}$ where $z = [z_1, \ldots, z_{|V|}] \in \mathbb{R}^{|V|}$ and each $z_i \in \mathbb{R}$
- Compute vector $\hat{y} \in \mathbb{R}^{|V|}$. Each element $\hat{y}_i = \frac{exp(z_i)}{\sum_{k=1}^{|V|} exp(z_k)}$
- We write this as $\hat{y} = \mathrm{softmax}(z)$
- If the correct center word is w_i then the ideal output y is a one-hot vector with index i as 1 and all other elements are 0.

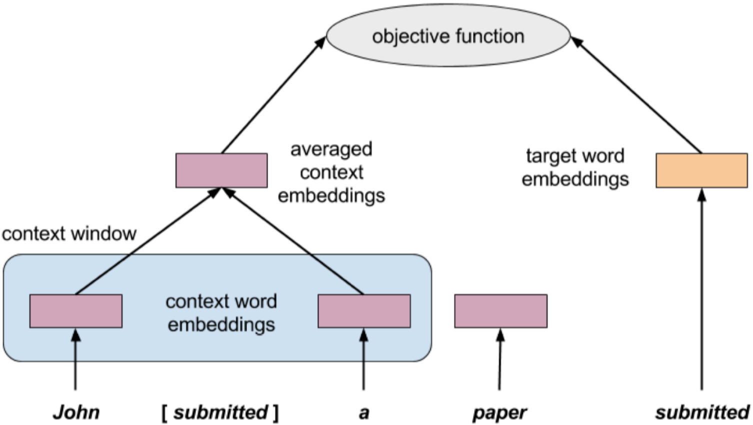
Word2Vec: Continuous Bag of Words

Learning

- Goal: learn k-dimensional word vectors u_i, v_i for each $i = 1, \ldots |V|$
- For each training example the correct center word w_j is represented as a one-hot vector y where $y_j = 1$.
- $\hat{y} = \mathrm{softmax}(U \cdot \hat{v})$ where $\hat{v}$ is the average of the context words
- Loss function is the cross entropy:
$$H(\hat{y}, y) = -\log(\hat{y}_j) \text{ for } j \text{ where } y_j = 1$$
- If c is the index of the correct word, consider case where prediction $\hat{y}_c = 0.99$ then the loss or penalty is low $H(\hat{y}, y) = -1 \cdot \log(0.99) = 0.01$
- If the prediction was bad $\hat{y}_c = 0.01$ then the loss is high $H(\hat{y}, y) = -1 \cdot \log(0.01) = 4.6$

CBOW Loss Function

Figure from [2]



Gradient descent

Objective function

$$
\begin{aligned}
&\text{Minimize } J \\
&\quad = \; -\log P(u_c \mid \hat{v}) \\
&\quad = \; -u_c \cdot \hat{v} + \log \sum_{j=1}^{|V|} exp(u_j \cdot \hat{v})
\end{aligned}
$$

Gradient descent

- Initialize $u^{(0)}$ and $v^{(0)}$
- $J(u, v) = -u_c \cdot \hat{v} + \log \sum_{j=1}^{|V|} exp(u_j \cdot \hat{v})$
- $t \leftarrow 0$
- Iterate to minimize loss $H(\hat{y}, y)$ on each training example:
  - Pick a training example at random
  - Calculate:

$$\begin{aligned} \hat{y} &= \mathrm{softmax}(U \cdot \hat{v}) \\ \Delta_u &= \left.\frac{dJ(u,v)}{du}\right|_{u,v=u^{(t)},v^{(t)}} \\ \Delta_v &= \left.\frac{dJ(u,v)}{dv}\right|_{u,v=u^{(t)},v^{(t)}} \end{aligned}$$

  - Using a learning rate γ find new parameter values:

$$\begin{aligned} \mathbf{u}^{(t+1)} &\leftarrow \mathbf{u}^{(t)} - \gamma\Delta_u \\ \mathbf{v}^{(t+1)} &\leftarrow \mathbf{v}^{(t)} - \gamma\Delta_v \end{aligned}$$

One-hot vectors

Singular Value Decomposition

Word2Vec

GloVe

GloVe

Co-occurrence matrix

Let X denote the word-word co-occurrence matrix.
X_{ij} is number of times word j occurs in the context of word i.
Let $X_i = \sum_k X_{ik}$
And $P_{ij} = P(w_j \mid w_i) = \frac{X_{ij}}{X_i}$

GloVe objective

Probability that word j occurs in context of word i:

$$Q_{ij} = \frac{exp(u_i \cdot v_j)}{\sum_{w=1}^{|V|} exp(u_w \cdot v_i)}$$

Compute global cross-entropy loss:

$$J = -\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} X_{ij} \log Q_{ij}$$

GloVe

Cross Entropy Loss

$$J = -\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} \underbrace{X_{ij}}_{X_i P_{ij}} \log Q_{ij}$$

$$X_{i,j} = X_i P_{ij} \text{ because: } P_{ij} = \frac{X_{ij}}{\sum_k X_{ik}} = \frac{X_{ij}}{X_i}$$

$$J = -\sum_i X_i \underbrace{\sum_j P_{ij} \log Q_{ij}}_{H(P_i, Q_i)}$$

where H is the cross entropy of Q_{ij} which uses the parameters u, v wrt the observed frequencies P_{ij}.

GloVe

Simplify objective function

In the objective $-\sum_{ij} X_i \cdot P_{ij} \log Q_{ij}$ the distribution Q_{ij} requires an expensive normalization over the entire vocabulary.

Simplify J to $\hat{J}$ using the squared error of the logs of $\hat{P}$ and $\hat{Q}$ without normalization:

$$\hat{J} = -\sum_{i,j=1}^{|V|} \underbrace{X_i}_{\text{replace with function } f(X_{ij})} \left(\log \underbrace{\hat{Q}_{ij}}_{exp(u_i \cdot v_j)} - \log \underbrace{\hat{P}_{ij}}_{X_{ij}} \right)^2$$

$$\hat{J} = -\sum_{ij} f(X_{ij})(u_i \cdot v_j - \log X_{ij})^2$$

The GloVe model efficiently leverages global statistical information by training only on the nonzero elements in a word-word co-occurrence matrix.

[1] Christopher Manning, Richard Socher, Francois Chaubard, Michael Fang, Guillaume Genthial, Rohit Mundra.
Natural Language Processing with Deep Learning: Word Vectors I: Introduction, SVD and Word2Vec
Winter 2019.

[2] O. Melamud and J. Goldberger and I. Dagan
context2vec: Learning Generic Context Embedding with Bidirectional LSTM.
CoNLL 2016.

Acknowledgements

Many slides borrowed or inspired from lecture notes by Michael Collins, Chris Dyer, Kevin Knight, Chris Manning, Philipp Koehn, Adam Lopez, Graham Neubig, Richard Socher and Luke Zettlemoyer from their NLP course materials.

All mistakes are my own.

A big thank you to all the students who read through these notes and helped me improve them.