Deep Learning for Text Mining

Yiming Yang

- The slides contain significant contributions from my students: Ruochen Xu, Wei-Cheng Chang, Yuexin Wu and Zihang Dai
- The slides combine multiple open-source tutorials/blogs/papers in the literature as listed in the references.

1

Topics

- Part 1: Word Embedding
- Part 2. Recurrent Neural Networks (RNN)
- Part 3. Convolutional Neural Networks (CNN)
- Part 4. Contextualized Text Representations

2

Part 1. Word Embedding

- Motivation
- Methods
  - SVD (Singular Value Decomposition) based
  - GloVe (Global Vectors for Word Representation)
  - CBOW (Continuous Bag-Of-Words)

Motivation

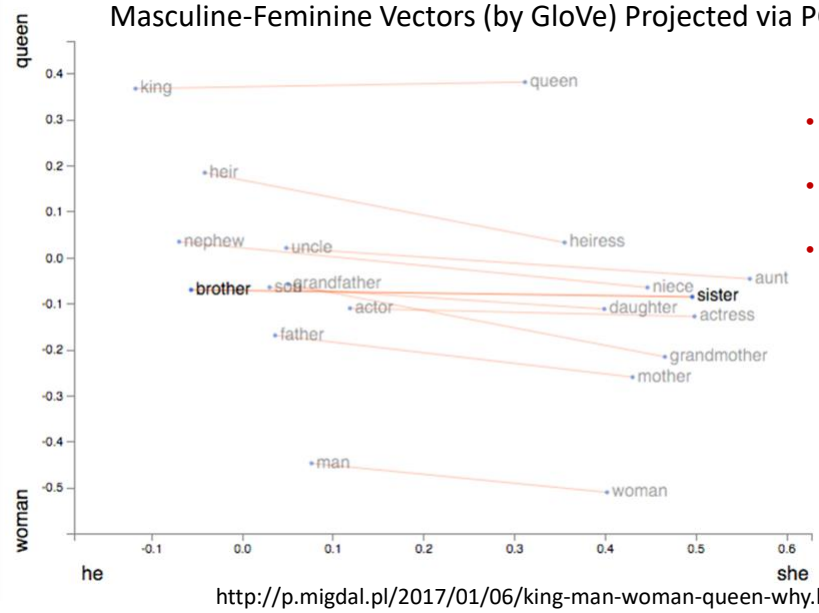
- Word2vec mapping has shown to be useful in information retrieval (IR), natural language processing (NLP**)**, machine translation (MT), text reading comprehension, etc.
- The key idea is to use a k-dimensional vector (e.g., k=300) to capture the *meaning* of each word based on *distributional semantics*

Distributional Semantics

- J.R. Firth's hypothesis (1957): *"You shall know a word by the company it keeps."*
- Key Idea: represent each word based on word usage patterns in a training corpus of documents
- Two families of algorithms
  - Global matrix factorization (SVD-based methods, GloVe?, etc.)
  - Local-window based learning (CBOW, Skip-gram, etc.)

5

Masculine-Feminine Vectors (by GloVe) Projected via PCA



- Each word is a vector (spot).
- Each relation is a line.
- Parallel lines show the "similarity" or "analogy" among entity-relation triplets.

http://p.migdal.pl/2017/01/06/king-man-woman-queen-why.html

6

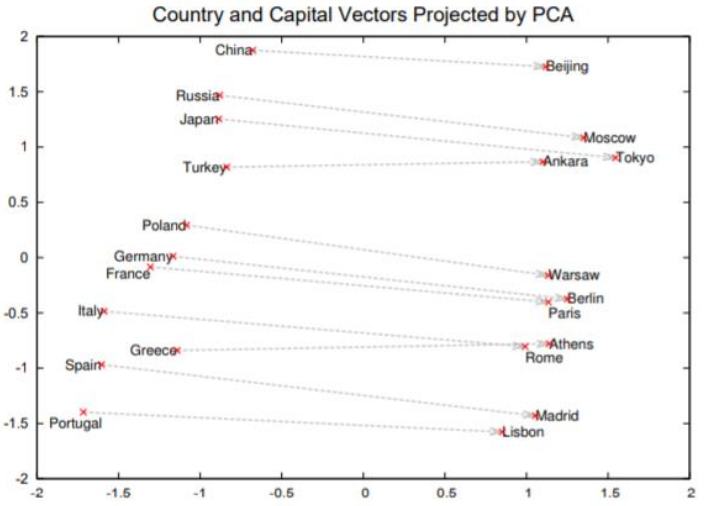


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

The Word Analogy Task

- The problem (illustrated by an example)

  $\mathcal{R}$(man, woman) ≈ $\mathcal{R}$ (king, ?)

- Selecting the word based on the embeddings $v_w \in R^k$ of all words

  $$w^* = argmin_w \left\| (v_w - v_{king}) - (v_{woman} - v_{man}) \right\|$$

- Geometrically, we want to find the line going through vertex v_{king} and parallel to line of ($v_{woman} - v_{man}$) as possible.

Part 1. Word Embedding

- Motivation
- Methods
  - SVD (Singular Value Decomposition) based
  - GloVe (Global Vectors for Word Representation)
  - CBOW (Continuous Bag-Of-Words)

SVD-based Methods

- **Representative Example**
  - Latent Semantic Analysis (LSA) [Deerwester et al., 1990]
- **Input**
  - Word-Doc Matrix $X = R^{|V| \times N}$ $where$ X_{ij} is the weight of word i in doc j
- **Low-rank SVD gives**

$$X \approx W_k \times \Sigma_k \times {D_k}^T$$

where $W_k \in R^{|V| \times k}$ contains the embedded words as the row vectors.

Variants of SVD-based Methods

- **As Baselines** in comparison with GloVe (Pennington et al., EMNLP 2014)
  - SVD where $X_{ij} \coloneqq TF_{ij}$ (the TF value of word i in doc j)
  - SVD-S where $X \rightarrow X'$ and $X'_{ij} \coloneqq \sqrt{X_{ij}}$
  - SVD-L where$X \rightarrow X'$ and $X'_{ij} \coloneqq \log(1 + X_{ij})$
- **Key property of the above methods**
  - Based on *global* patterns of word *occurrences* instead of local patterns in word *co-occurrences*
- **Questions regarding the design choices**
  - Why not try TF-IDF weight for X_{ij} in X?
  - Why not try SVD of $X \in R^{|V|\times|V|}$ with X_{ij} set to the number of times word j occurs within the local context windows of word i (the same as in GloVe)?

Part 1. Word Embedding

- Motivation
- Methods
  - SVD (Singular Value Decomposition) based
  - GloVe (Global Vectors for Word Representation)
  - CBOW (Continuous Bag-Of-Words)

GloVe [Jeffrey Pennington et. al., EMNLP 2014]

- **Input matrix** $X \in R^{|V|\times|V|}$
  - A word-by-word matrix with X_{ij} as the number of times of word j occurring within the context window of word i, multiplied to $\frac{1}{d}$ when word j is d $(\in \{1,\cdots,10)$ words apart from word i.
- **Output matrix** $W \in R^{|V|\times k}$
  - The matrix of word embeddings where each row is a k-dimensional vector ($k \ll |V|$) for a word, denoted as w_i for $i = 1,2,\cdots,|V|$, with a bias term (automatically learned) per word, namely b_i .
- **Objective of GloVe** (being solved via stochastic gradient descent)

$$\min_{W,b} \sum_{i,j=1}^{m} f(X_{ij})\left(w_i^T w_j + b_i + b_j - logX_{ij}\right)^2$$

where the weighting function is defined as:

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

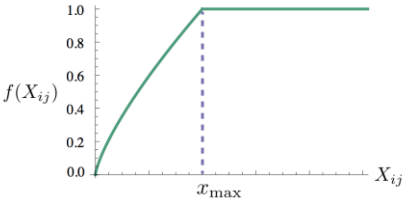


Figure 1: Weighting function f with $\alpha = 3/4$.

13

GloVe as Matrix Factorization

- **Objective:** $\min_{W,b} L(W,b|X)$

$$L(W,b|X) = \sum_{i,j=1}^{m} f(X_{ij})(\underbrace{w_i^T w_j + b_i + b_j}_{log\hat{X}_{ij}} - logX_{ij})^2$$

- **Equivalently, we can rewrite $w_i^T w_j$ above as $(W^T W)_{ij}$**

1) Thus, optimizing W w.r.t. $L(W,b|X)$ is indeed a matrix factorization problem.

2) Despite the name, GloVe focuses on window-based local co-occurrence information about word pairs, not global word-doc distributions.

14

Comparing GloVe with SVD (of a word-doc matrix)

- GloVe made the following differences
  - Using *local* co-occurrence counts (discounted based on proximity) in a log scale
  - Adding the bias term for each word
  - Weighting pairwise losses by function $f(X_{ij})$
- GloVe outperformed SVD (??) in a downstream task-oriented evaluation
  - It would be interesting to run SVD on GloVe's X.

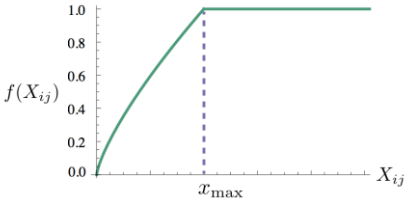


Figure 1: Weighting function f with $\alpha = 3/4$.

Experiment and Evaluation Results

- Word analogy task
- Word similarity task
- Other downstream tasks (not included here)
  - Name entity recognition (NER)
  - Sentimental classification
  - Language modeling
  - Neural machine translation

Word Analogy Task

- Ask questions like "a is to b as c is to __?"
- Dataset contains 19,544 questions, divided into two subsets.
  - Semantic: "Athens is to Greece as Berlin is to __?"
  - Syntactic: "dance is to dancing as fly is to __?"
- Find v_d that is closest to $v_b - v_a + v_c$ according to cosine similarity

Word Analogy Results [Jeffrey Pennington et. al., EMNLP 2014]

- Baseline Methods
  - HPCA: PMI version of LSA (PCA) [10]
  - vLBL, ivLBL: log-bilinear model [9]
  - SG: skip gram (another variant of w2v)
  - CBOW: continuous bag-of-words
  - SVD-S: take SVD of $\sqrt{X_{trunc}}$
  - SVD-L: take SVD of $\log(1 + X_{trunc})$
- Metric: Accuracy
- Size: number of tokens in training corpus

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**

Part 1. Word Embedding

- Motivation
- Methods
  - SVD (Singular Value Decomposition) based
  - GloVe (Global Vectors for Word Representation)
  - CBOW (Continuous Bag-Of-Words)

Continuous Bag of Words (CBOL) Model

-- Tomáš Mikolov's Word2vec algorithm (arXiv:1301.3781)

- Another approach to leverage local context for word embedding
- Assuming the probability of word w depending on the words in the context window (of size C),

$$\Pr[w|w_1, w_2, \ldots, w_C] \propto \exp(v_w \cdot (\frac{1}{C}\sum_i v_{w_i}))$$

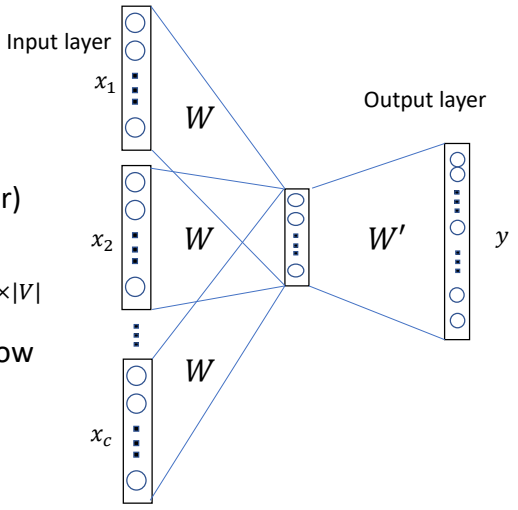
where v_w is the embedding vector of word w.

Continuous bag-of-words (CBOW) model

- Toy example
  - Word sequence "The cat chills on a mat"

x_1 x_2 x_3 x_4 x_5 x_6

  - Input: x_1, x_2, x_4, x_5 (each is a one-hot vector)
  - Output: x_3 (denote y in the figure)
  - Parameter matrices: $W \in R^{|V| \times k}, W' \in R^{k \times |V|}$
  - The the embedding of word w_j is the j-th row of matrix W, denoted by $v_{w_j} = W^T x_j$.



Forward propagation of CBOW [Xin Rong, arXiv 2016]

- In the hidden layer

$$h = \frac{1}{C} W^T (x_1 + x_2 + \cdots + x_C) = \frac{1}{C} (v_{w1} + v_{w2} + \cdots + v_{wc})^T,$$

- In the output layer

$$u_j = v'^T_{w_j} h \quad \text{where } v'_{w_j} \text{ is the } j\text{-th column of matrix } W'.$$

- The output softmax function

$$\hat{y}_j = Pr\left[w_{y_j} \middle| w_1, \ldots, w_C\right] = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})}$$

Loss function of CBOW

- The objective function is the negative log-likelihood

$$\begin{aligned} \mathrm{E} &= -\log p(w_O|w_{I1},\dots,w_{IC}) \\ &= -u_{j^*} + \log(\textstyle\sum_{j'=1}^{|V|} \exp(u_{j'})) \\ &= -\mathrm{v}'^{T}_{w_O} h + \log(\textstyle\sum_{j'=1}^{|V|} \exp(v'^{T}_{w_{j'}} h)) \end{aligned}$$

where j^* is the index of the actual output word in the output layer.

- In practice, the vocabulary may be too large, so people often use the Noise Contrastive Estimation technique, which randomly sampled words instead of going through all words in the log-sum-exp term.

Parameter Learning with Backpropagation

- Output-layer partial derivative

$$\begin{aligned} \frac{\partial E}{\partial u_j} &= \frac{\partial}{\partial u_j}\left(-u_{j^*} + \log(\textstyle\sum_{j'=1}^{|V|} \exp(u_{j'}))\right) \\ &= -t_j + \hat{y}_j \end{aligned}$$

(the prediction error of the output layer)

where $t_j = 1\ if\ j = j^*\ otherwise\ t_j = 0.$

- Partial derivative with respect to elements of W'

$$\frac{\partial E}{\partial W'_{ij}} = \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial W'_{ij}} = (\hat{y}_j - t_j) h_i \qquad u_j = v'^{T}_{w_j} h = \textstyle\sum_{i=1}^{k} W'_{ij} h_i$$

j-th column of W'

- The update rule for matrix W'

$$W'^{(new)}_{ij} := W'^{(old)}_{ij} - \eta(\hat{y}_j - t_j) h_i$$

Parameter Learning with Backpropagation (cont'd)

- Partial derivative w.r.t. the hidden layer

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j}\frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{|V|}(\hat{y}_j - t_j)\cdot W'_{ij}$$

- Partial derivative w.r.t. the elements of W

$$\frac{\partial E}{\partial W_{li}} = \frac{\partial E}{\partial h_i}\cdot\frac{\partial h_i}{\partial W_{li}} = \sum_{j=1}^{|V|}(\hat{y}_j - t_j)\cdot W'_{ij}\bar{x}_l$$

- The update rule for matrix W

$$W_{li}^{(new)} := W_{li}^{(old)} - \eta\frac{\partial E}{\partial W_{li}}$$

$$\bar{x} \equiv \frac{1}{C}(x_1 + x_2 + \cdots + x_C) \in R^{|V|}$$

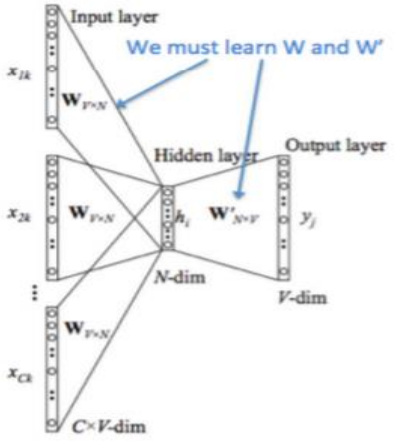
$$h = W^T\bar{x} \in R^k$$

$$h_i = W_{1i}\bar{x}_1 + W_{2i}\bar{x}_2 + \cdots + W_{|V|i}\bar{x}_{|V|}) \in R$$

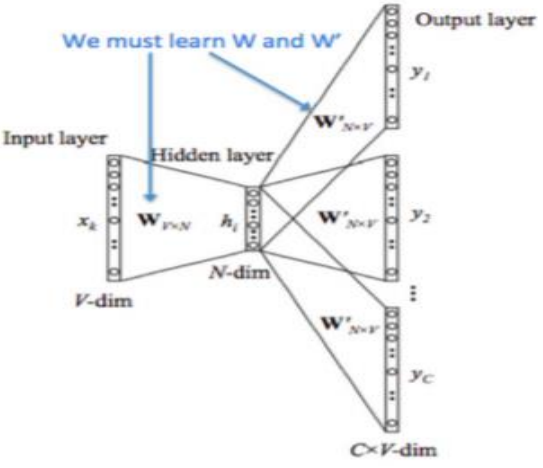
$$\frac{\partial h_i}{\partial W_{li}} = \bar{x}_l$$

Skip-Gram Model (Tomáš Mikolov, NIPS 2013)

CBOL



Skip-Gram



Word Analogy Results [Jeffrey Pennington et. al., EMNLP 2014]

- Baseline Methods
  - HPCA: PMI version of LSA using PCA [10]
  - vLBL, ivLBL: log-bilinear model [9]
  - SG: skip gram (another variant of w2v)
  - CBOW: continuous bag-of-words
  - SVD-S: take SVD of $\sqrt{X_{trunc}}$
  - SVD-L: take SVD of $\log(1 + X_{trunc})$
    - X_{trunc}: keep the top 10,000 most frequent words only in the input matrix X
- Metric: Accuracy
- Size: number of tokens in training corpus

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**

Summary

- Representative Word Embedding Methods
  - SVD based: a matrix factorization algorithm, focusing (but not necessarily) on global word distribution over doc's
  - GloVe: a matrix factorization algorithm, focusing on pairwise word-word co-occurrence information in local proximity windows
  - CBOW (and Skip-Gram): a window-based algorithm, representing the local context of each word by aggregating the representations of its nearest neighbors
- Widely used for downstream tasks (IR, NLP, MT, etc.)
  - Typically as a preprocessing step in RNN, CNN, etc. (later lectures)

Reference

1. Sanjeev Arora Blog: Off the convex path [link1, link2, link3]
2. Indexing by latent semantic analysis. Scott Deerwester et. al., Journal of the American Society for Information Science 1990
3. GloVe: Global Vectors for Word Representation. Jeffrey Pennington et. al., EMNLP 2014
4. Xin Rong: **word2vec Parameter Learning Explained.** CoRR abs/1411.2738 (2016)
5. A Latent Variable Model Approach to PMI-based Word Embeddings. Sanjeev Arora et. Al., TACL 2016
6. RANK-WALK: A latent variable model approach to word embedding. Sanjeev Arora et. al., arXiv:1502.03520
7. Distributed representations of words and phrases and their compositionality. Tomas Mikolov et. al., NIPS 2013
8. Efficient Estimation of Word Representations in Vector Space. Tomas Mikolov et. al., arXiv 2013
9. Word2Vec tutorial [link]
10. Learning word embeddings efficiently with noise-contrastive estimation. A. Mnih and K. Kavukcuoglu, NIPS 2013
11. Word embeddings through Hellinger PCA. R. Lebret and R. Collobert. EACL, 2014