CS 4248
Natural Language Processing

Professor NG Hwee Tou

Department of Computer Science
School of Computing
National University of Singapore
nght@comp.nus.edu.sg

Materials

- NNM4NLP Chapter 10, 11

How to Obtain Word Embeddings

- Random initialization
- Supervised task-specific pre-training
- Unsupervised pre-training

Supervised Task-Specific Pre-Training

- Setup:
  - Task A has limited labeled training data
  - Task B is another auxiliary task which has much more labeled training data
  - E.g., task A: syntactic parsing; task B: POS tagging

Supervised Task-Specific Pre-Training

- Goal: Exploit the large labeled training data for task B to help task A
- Pre-train word vectors for task B, so that we get word vectors that are good predictors for task B
- Use these pre-trained word vectors for training task A

Unsupervised Pre-Training

- "Unsupervised": Using raw texts without human manual annotation
- Based on word prediction in raw texts
  - Predict a word based on its context (e.g., neural language modeling uses context of k previous words)
  - Predict the context based on a word
- Actually a supervised learning task (self-supervised)

Word Embeddings

- When using pre-trained word vectors:
  - Pre-trained vectors in embedding matrix $E \in \mathbb{R}^{|V| \times d}$ can be:
    - Further tuned to the specific task with neural network training
    - Fixed during neural network training
    - Trade-off: Adapt the word vectors to the specific task versus losing generalization properties learned for words in the raw texts but not in the training data of the specific task

Word2Vec

- A software package implementing 2 different context representations
  - CBOW (continuous bag of words)
  - Skip-gram
- Fast, efficient to train, highly scalable to billions of words of text

Word2Vec

- Two vectors for each word (one vector when the word is the center word and another vector when the word is a context word)
- The vectors are the parameters to be learned
- Idea: A center word is similar to the words in its context
- Make the vector $\boldsymbol{w}$ of the center word close to the vectors $\boldsymbol{c}$ of the words in its context ($\boldsymbol{w} \cdot \boldsymbol{c}$ is large)

Word2Vec

- Training distinguishes a set D of correct word-context pairs from a set $\overline{D}$ of incorrect word-context pairs
- Positive examples $(\mathrm{w}, \mathrm{c}) \in D$ from a raw text corpus
- Negative sampling: For each $(\mathrm{w}, \mathrm{c}) \in D$, sample k words and add each of these words (w_i, c) as a negative example to $\overline{D}$

Word2Vec

- Negative words are sampled according to their corpus frequency: $\frac{\#(w)^{0.75}}{\sum_{w'} \#(w')^{0.75}}$ (gives higher relative weight to less frequent words)
- Example:
  - $\mathrm{Count}(w_1) = 98 \ \ (p(w_1) = 0.98)$
  - $\mathrm{Count}(w_2) = 2 \ \ (p(w_2) = 0.02)$
  - $\tilde{p}(w_1) = \frac{98^{0.75}}{98^{0.75}+2^{0.75}} = 0.95$
  - $\tilde{p}(w_2) = \frac{2^{0.75}}{98^{0.75}+2^{0.75}} = 0.05$

Word2Vec

- $s(w, c) = \boldsymbol{w} \cdot \boldsymbol{c}$
- $P(D = 1|w, c) = \frac{1}{1+e^{-s(w,c)}} = \frac{1}{1+e^{-\boldsymbol{w}\cdot\boldsymbol{c}}}$
- $P(D = 0|w, c) = 1 - P(D = 1|w, c)$

Word2Vec

- Training objective: Minimize the negative log-likelihood of the data $D \cup \overline{D}$:

$$\mathcal{L}(\Theta; D, \overline{D}) = -\sum_{(w,c)\in D} \log P(D = 1|w, c)$$

$$-\sum_{(w,c)\in \overline{D}} \log P(D = 0|w, c)$$

Word2Vec: CBOW

- c: $c_{1:k}$ (context is the window of surrounding words centered at w)
- Predict the target word w from the context $c_{1:k}$ of surrounding words
- $\boldsymbol{c} = \sum_{i=1}^{k} \boldsymbol{c_i}$
- $\log P(D = 1 | w, c_{1:k}) = \log \frac{1}{1+e^{-(\boldsymbol{w} \cdot \boldsymbol{c_1} + \boldsymbol{w} \cdot \boldsymbol{c_2} + \cdots \boldsymbol{w} \cdot \boldsymbol{c_k})}}$
- Ignore order of surrounding words

Word2Vec: Skip-gram

- Predict the context $c_{1:k}$ of surrounding words from the target word w
- Assume each word in $c_{1:k}$ is independent of each other
- $P(D = 1|w, c_{1:k}) = \prod_{i=1}^{k} P(D = 1|w, c_i) = \prod_{i=1}^{k} \frac{1}{1+e^{-\boldsymbol{w}\cdot\boldsymbol{c_i}}}$
- $\log P(D = 1|w, c_{1:k}) = \sum_{i=1}^{k} \log \frac{1}{1+e^{-\boldsymbol{w}\cdot\boldsymbol{c_i}}}$

Word2Vec

- After training, two embedding matrices are computed:
  - $\boldsymbol{E}^{\boldsymbol{W}} \in \mathbb{R}^{|V_W| \times d_{\mathrm{emb}}}$ (kept)
  - $\boldsymbol{E}^{\boldsymbol{C}} \in \mathbb{R}^{|V_C| \times d_{\mathrm{emb}}}$ (discarded)

Using Word Embeddings

- Word similarity
- Cosine similarity

$$\mathrm{sim}_{\cos}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\|_2 \|\boldsymbol{v}\|_2} = \frac{\sum_i \boldsymbol{u}_{[i]} \cdot \boldsymbol{v}_{[i]}}{\sqrt{\sum_i (\boldsymbol{u}_{[i]})^2} \sqrt{\sum_i (\boldsymbol{v}_{[i]})^2}}$$

- When $\boldsymbol{u}$ and $\boldsymbol{v}$ are of unit length ($\|\boldsymbol{u}\|_2 = \|\boldsymbol{v}\|_2 = 1$), cosine similarity reduces to a dot-product

$$\mathrm{sim}_{\cos}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u} \cdot \boldsymbol{v} = \sum_i \boldsymbol{u}_{[i]} \cdot \boldsymbol{v}_{[i]}$$

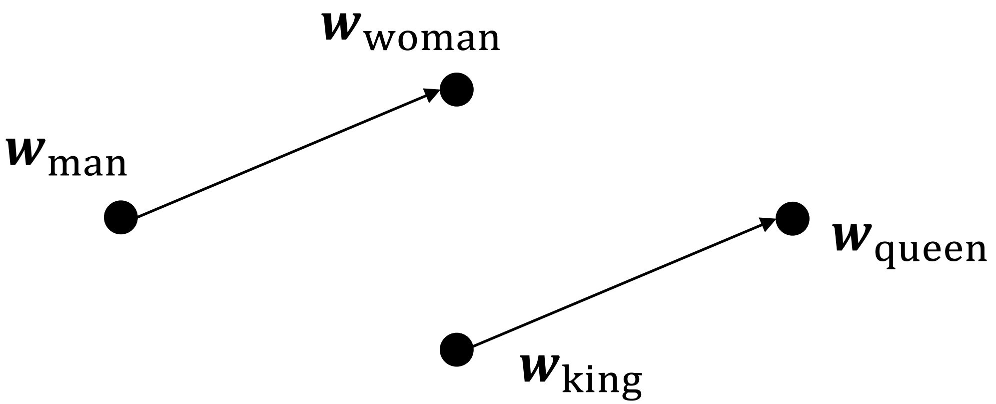
Using Word Embeddings

- Word analogy task via word vector algebra
- For word embeddings trained using Word2vec:

$$\boldsymbol{w}_{\text{king}} - \boldsymbol{w}_{\text{man}} + \boldsymbol{w}_{\text{woman}} \approx \boldsymbol{w}_{\text{queen}}$$

$$\boldsymbol{w}_{\text{France}} - \boldsymbol{w}_{\text{Paris}} + \boldsymbol{w}_{\text{London}} \approx \boldsymbol{w}_{\text{England}}$$

$$\text{analogy}(m\!: w \rightarrow k\!:?) = \underset{v \in V \setminus \{m,w,k\}}{\text{argmax}} \cos(\boldsymbol{v}, \boldsymbol{k} - \boldsymbol{m} + \boldsymbol{w})$$



Using Word Embeddings

- Finding the k most similar words to w
- $\boldsymbol{E}$: row-normalized embedding matrix
- $\mathrm{sim}_{\cos}(w_1, w_2) = \boldsymbol{E}_{[w_1]} \cdot \boldsymbol{E}_{[w_2]}$

Using Word Embeddings

- $\boldsymbol{s} = \boldsymbol{w}\boldsymbol{E}^{\mathrm{T}}$
- $\boldsymbol{s}$ is a vector of similarities, where $\boldsymbol{s}_{[i]}$ is the similarity of w to the ith word in the vocabulary
- k most similar words: find the indices corresponding to the k highest values in $\boldsymbol{s}$
- Vector-matrix multiplication can be computed rapidly

Using Word Embeddings

- Odd-one out
  - Given a list of words, find the word that does not belong
  - Compute the similarity between each word to the average word vector of the group, and return the least similar word

Using Word Embeddings

- Similarity of short documents (e.g., web queries, newspaper headlines, tweets)
- Treat each document as a bag of words
- $D_1 = w_1^1, w_2^1, \dots, w_m^1 \qquad D_2 = w_1^2, w_2^2, \dots, w_n^2$
- $\mathrm{sim}_{\mathrm{doc}}(D_1, D_2) = \sum_{i=1}^{m} \sum_{j=1}^{n} \mathrm{sim}_{\cos}(\boldsymbol{w_i^1}, \boldsymbol{w_j^2})$