Text Analysis: Word Representation Methods

Cam Tu Nguyen

阮锦绣

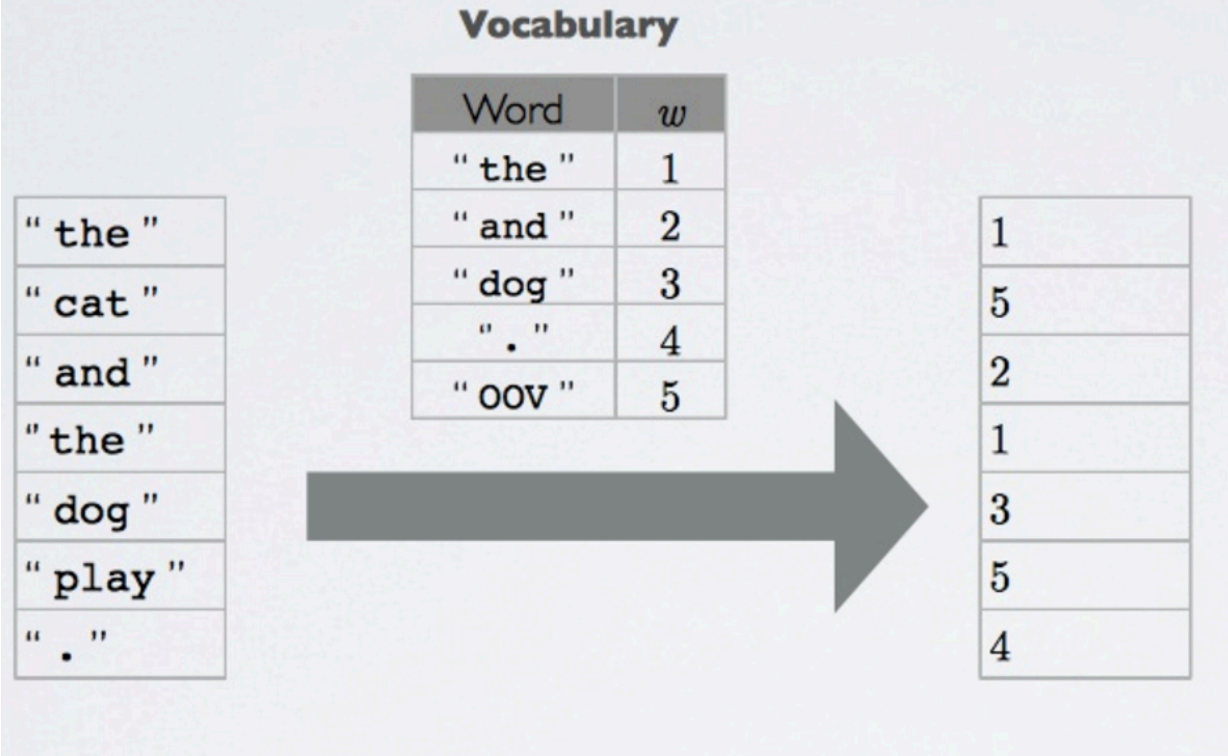
Software Institute, Nanjing University
nguyenct@lamda.nju.edu.cn
ncamtu@gmail.com

Outline

- Motivation
- SVD-based method
- Language Model
- CBOW
- Skip-gram Model
- Efficient Training for Word Embedding

One-hot encoding

- From vocabulary of words that maps lemmatized words to a unique ID (position of word in vocabulary).
- Typical vocabulary sizes will vary between 10 000 and 250 000



One-hot encoding

- From its word ID, we get a basic representation of a word through the one-hot encoding of the ID.
- The one-hot vector of an ID is a vector filled with 0s, except for a 1 at the position associated with the ID.
  - Ex: for vocabulary size V=10, the one-hot vector of word ID w=4 is: e(w) = [0 0 0 1 0 0 0 0 0 0]
- A one-hot encoding makes no assumption about word similarity
  - All words are equally different from each other
- This is a natural representation to start with, though a poor one.

Better Word Representation: Motivations

Harvard

Similar words

Word	Cosine distance
yale	0.638970
cambridge	0.612665
university	0.597709
faculty	0.588422
harvey_mudd	0.578338
johns_hopkins	0.575645
graduate	0.570294
undergraduate	0.565881
professor	0.563657
mcgill	0.562168
ph_d	0.558665
california_berkeley	0.555539
yale_university	0.550480

Better Word Representation: Motivations

Can be used as features for other natural language processing tasks and machine learning algorithms.

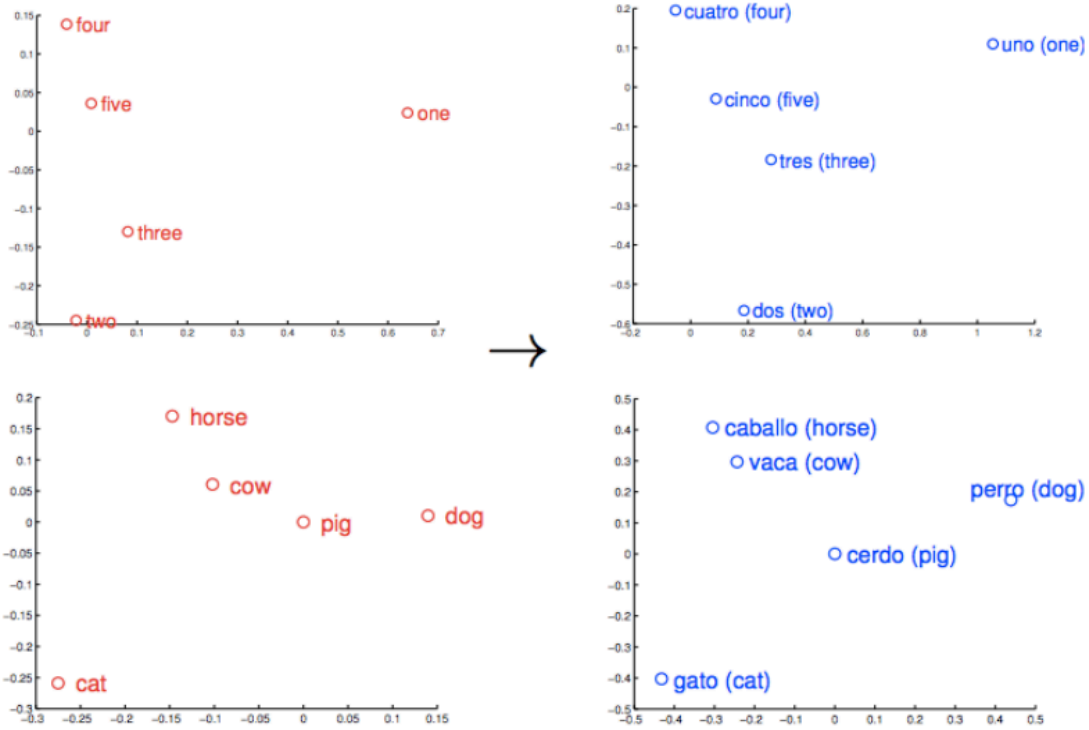
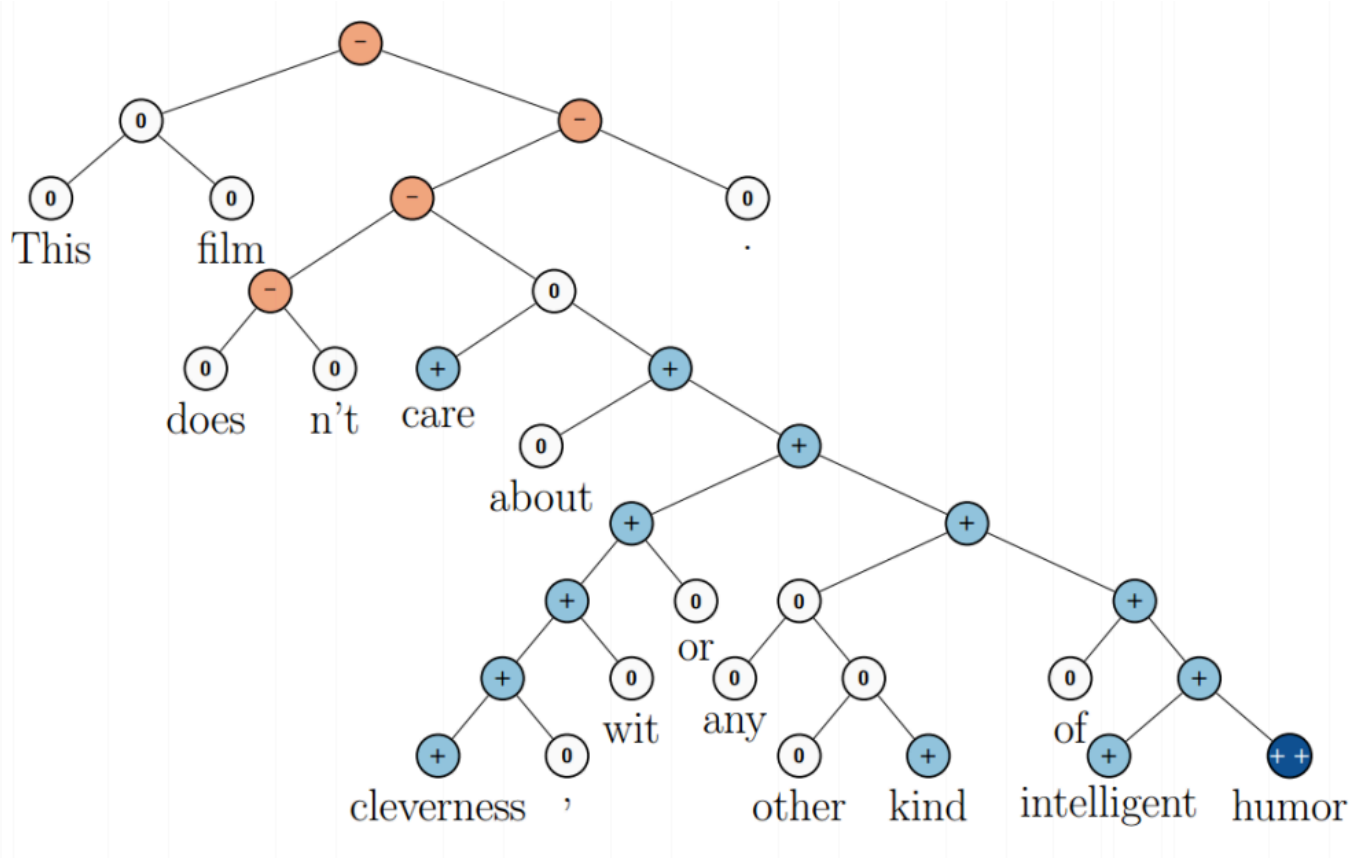


Figure 1: Distributed word vector representations of numbers and animals in English (left) and Spanish (right). The five vectors in each language were projected down to two dimensions using PCA, and then manually rotated to accentuate their similarity. It can be seen that these concepts have similar geometric arrangements in both spaces, suggesting that it is possible to learn an accurate linear mapping from one space to another. This is the key idea behind our method of translation.

Quoted after Mikolov

Better Word Representation: Motivations

Can be used as features for other natural language processing tasks and machine learning algorithms.



Recursive Neural Tensor Network

Quoted after Socher

Outline

- Motivation
- SVD-based method
- Language Model
- CBOW
- Skip-gram Model
- Efficient Training for Word Embedding

SVD-based Method

- Loop over a massive dataset and accumulate word co-occurrence counts

- Perform Singular Value Decomposition on X to get USV^T decomposition.

  - We use the row of U as the word embeddings for all words in our dictionary.

Word-Word Co-occurrence Matrix

- Conjecture: words that are related will often appear in the same documents.
- Let our data set contains just three (short) documents.
  - I enjoy flying
  - I like NLP
  - I like deep learning

Co-occurrence Matrix

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Singular Value Decomposion

- Applying SVD to $X=USV^T$

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

- Reducing dimensionality by selecting first k singular vectors

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

The obtained vector representation for word encode semantic and syntactic information.

SVD-based Method

- Problems
  - The dimensions of the matrix change very often (new words are added very frequently and corpus changes in size).
  - The matrix is extremely sparse since most words do not co-occur.
  - The matrix is very high dimensional in general
  - Quadratic cost to train (i.e. to perform SVD)
  - Requires the incorporation of some hacks on X to account for the drastic imbalance in word frequency.
  - Difficult to interpret the underlying intuition

Outline

- Motivation
- SVD-based method
- Language Model
- CBOW
- Skip-gram Model
- Efficient Training for Word Embedding

Language Models

- A language model assigns a probability to a sequence of tokens
  - Example:
    - "The cat jumped over the puddle" → high probability
    - "stock boil fish is toy" → low probability (it makes no sense!)
  - Probability on any given sequence of n words: $P(w_1, w_2, ..., w_n)$
- Unigram language model: assumes the word occurrences are completely independent.

$$P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} P(w_i)$$

- Bigram Language model:

$$P(w_1, w_2, ..., w_n) = \prod_{i=2}^{n} P(w_i|w_{i-1})$$

Language Models

- General: n-gram models
  - Conditional probability tables for $P(w_n | w_{1:n-1})$

- Problems of classical n-gram models
  - Need (complicated) smoothing for the raw counts since the number of n-tuples is exponential in n.
  - N-gram models are unable to take advantage of large contexts since the data sparsity becomes extreme.
    - Different entries are estimated independently of each other.
    - No concept of similarity between words.

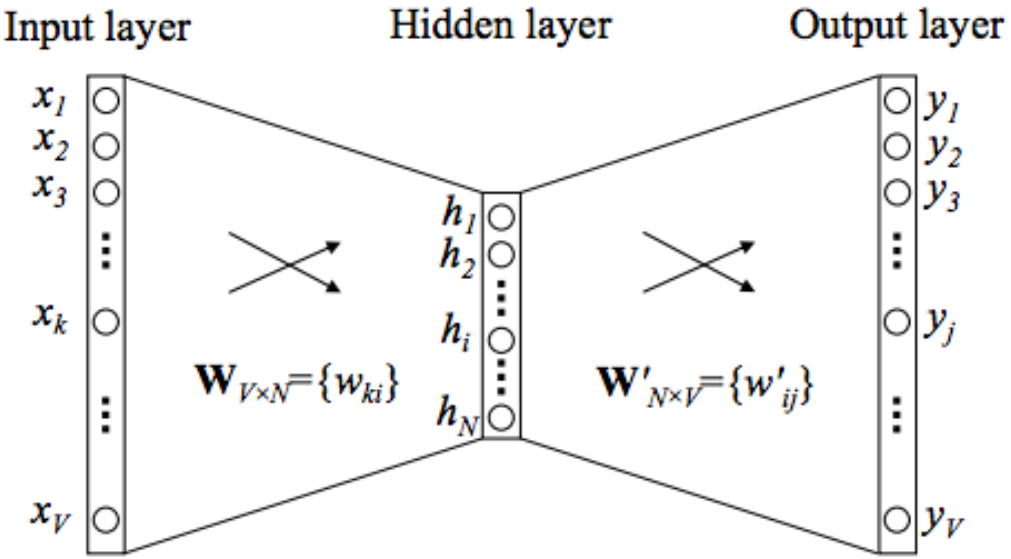
Improvement: Neural Probabilistic Language Model (NPLM, proposed by Yoshua Bengio), Continuous Bag-of-words model, Skip-gram model (proposed by Mikolov et al.).

Outline

- Motivation
- SVD-based method
- Language Model
- CBOW
- Skip-gram Model
- Efficient Training for Word Embedding

CBOW: One-word context

- One-word context: predicts one target word given one context word (like bigram)



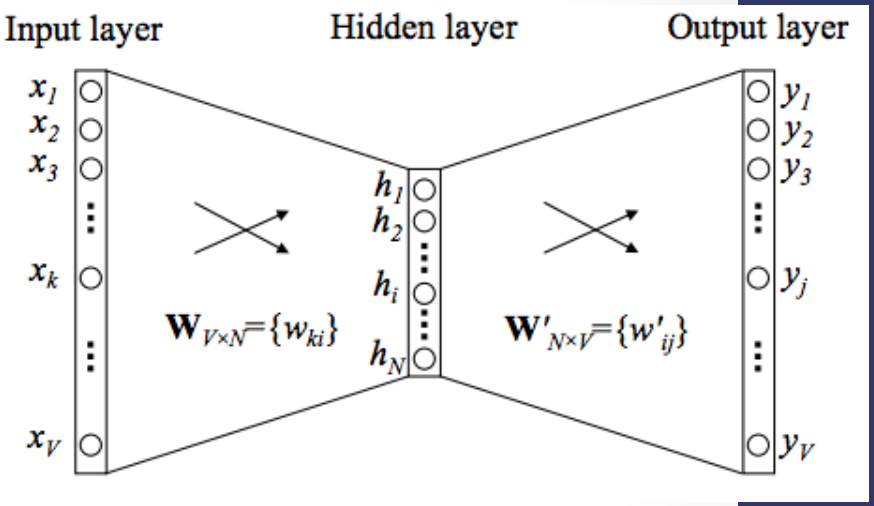
Input: one-hot encoded vector
Parameters: to be estimated W, and W'
V: the vocabulary size.

CBOW: One-word context

- Given a context word (id=k), i.e. $x_k = 1 \text{ and } x_{k'} = 0 \forall k' \neq k$, we have:

$$\mathbf{h} = \mathbf{W}^T \mathbf{x} = \mathbf{W}^T_{(k,\cdot)} := \mathbf{v}^T_{w_I}$$

  - The activation function of the hidden layer units is simply linearly (directly passing)



- The net input for each output layer neuron: $u_j = \mathbf{v}'^T_{w_j} \mathbf{h}$,

  where $\mathbf{v}'_{w_i}$ is the j-th column of the matrix $\mathbf{W}'$

- The output of output layer neuron:

$$p(w_j|w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})},$$

CBOW: One-word context

- Training objective (for one training sample)

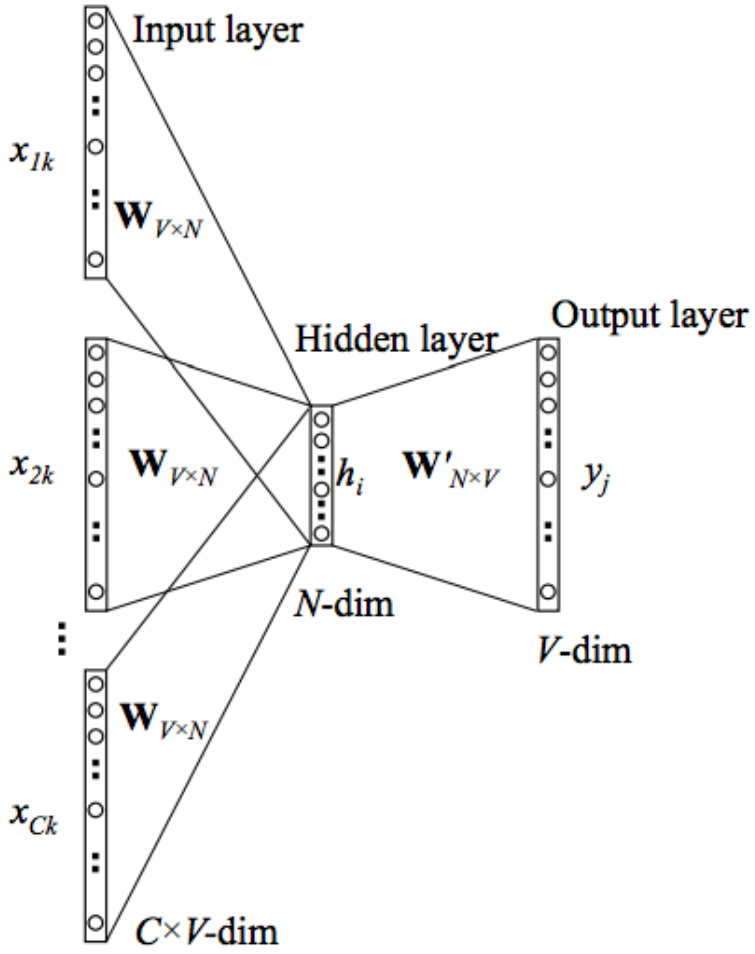
$$\begin{aligned} \max p(w_O|w_I) &= \max y_{j^*} \\ &= \max \log y_{j^*} \\ &= u_{j^*} - \log \sum_{j'=1}^{V} \exp(u_{j'}) := -E, \end{aligned}$$

- Update equations (on board)
  - Notice: to update parameters for one example, we need to loop through all the words in the vocabulary (expensive if the vocabulary size is large).
  - Efficient method will be explained later.

See more: Word2vec parameter learning explained, https://arxiv.org/pdf/1411.2738v4.pdf

CBOW: Multi-word context

- CBOW model with a multi-word context setting
  - When computing hidden layer output, instead of directly copying the input vector of the input context word, CBOW model takes the **average of the vectors of the input context words.**



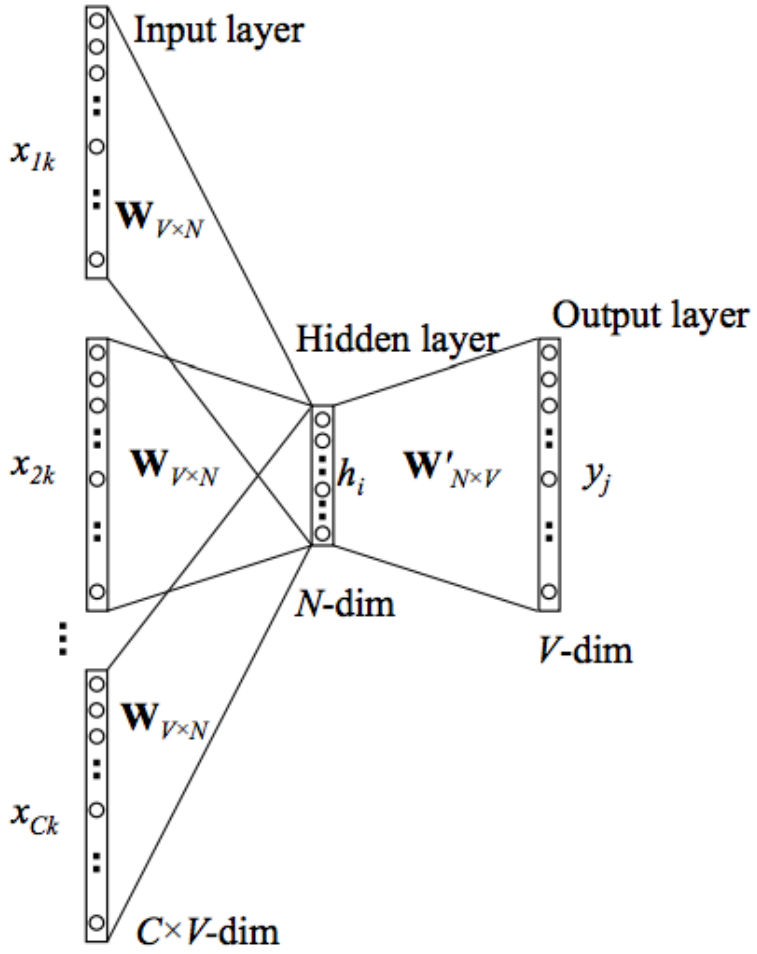
CBOW: Multi-word context

- We have:

$$\begin{aligned}\mathbf{h} &= \frac{1}{C}\mathbf{W}^T(\mathbf{x}_1+\mathbf{x}_2+\cdots+\mathbf{x}_C)\\ &= \frac{1}{C}(\mathbf{v}_{w_1}+\mathbf{v}_{w_2}+\cdots+\mathbf{v}_{w_C})^T\end{aligned}$$

- The loss function

$$\begin{aligned}E &= -\log p(w_O|w_{I,1},\cdots,w_{I,C})\\ &= -u_{j^*}+\log\sum_{j'=1}^{V}\exp(u_{j'})\\ &= -\mathbf{v}'_{w_O}{}^T\cdot\mathbf{h}+\log\sum_{j'=1}^{V}\exp(\mathbf{v}'_{w_j}{}^T\cdot\mathbf{h})\end{aligned}$$



CBOW: Multi-word context

- The update equation for the hidden-output weights stay the same as one-word-context model, which is:

$$\mathbf{v}'^{(\text{new})}_{w_j} = \mathbf{v}'^{(\text{old})}_{w_j} - \eta \cdot e_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V.$$

- The update equation for input-hidden weights is similar to the one-word-context model:

$$\mathbf{v}^{(\text{new})}_{w_{I,c}} = \mathbf{v}^{(\text{old})}_{w_{I,c}} - \frac{1}{C} \cdot \eta \cdot \mathrm{EH}^T \qquad \text{for } c = 1, 2, \cdots, C.$$

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot w'_{ij} := \mathrm{EH}_i$$

Outline

- Motivation
- SVD-based method
- Language Model
- CBOW
- Skip-gram Model
- Efficient Training for Word Embedding

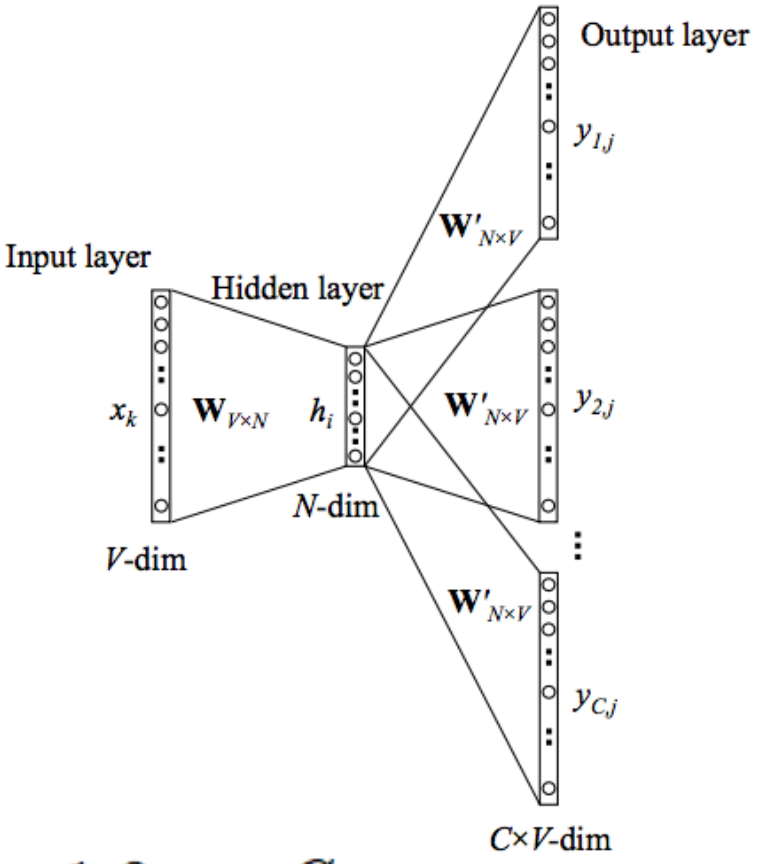
Skip-Gram Model

- The skip-gram model (Mikolov et al., 2013)
  - The opposite of the CBOW model
  - The target word is now at the input layer, and the context words are on the output layer.
  - Hidden layer output:

$$\mathbf{h} = \mathbf{W}_{(k,\cdot)}^T := \mathbf{v}_{w_I}^T,$$

  - Net input of the j-th unit on the c-panel of the output layer

$$u_{c,j} = u_j = \mathbf{v}'_{w_j}{}^T \cdot \mathbf{h}, \text{ for } c = 1, 2, \cdots, C$$



Skip-Gram Model

- The skip-gram model (Mikolov et al., 2013)
  - Hidden layer output:

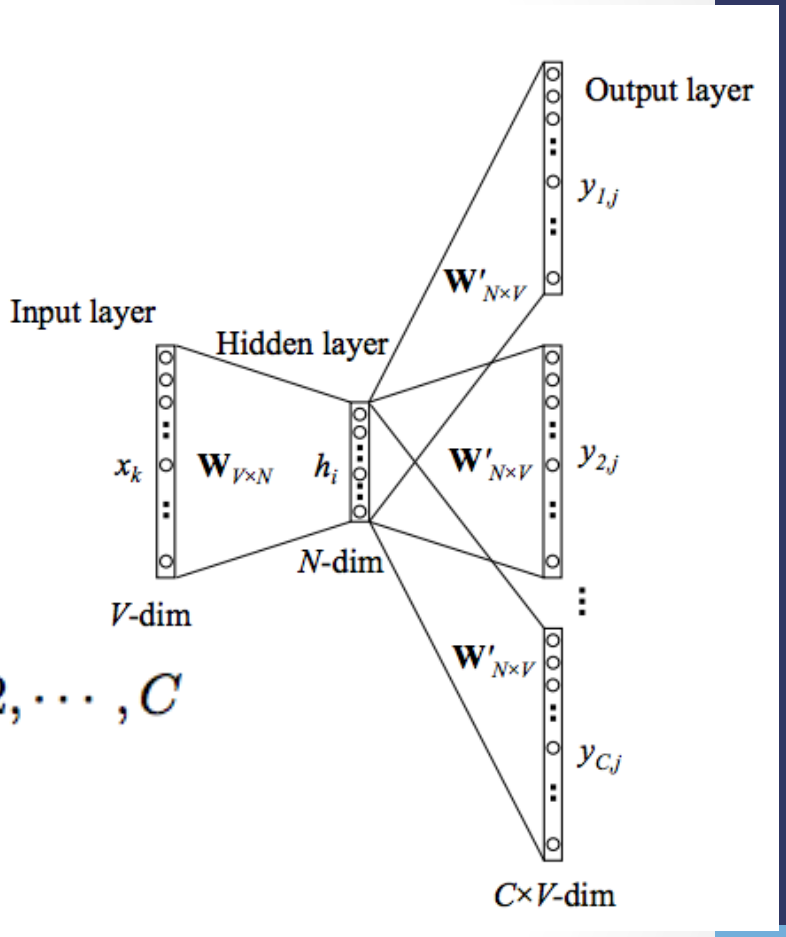
$$\mathbf{h} = \mathbf{W}_{(k,\cdot)}^T := \mathbf{v}_{w_I}^T,$$

  - Net input of the j-th unit on the c-panel of the output layer

$$u_{c,j} = u_j = {\mathbf{v}'_{w_j}}^T \cdot \mathbf{h}, \text{ for } c = 1, 2, \cdots, C$$

  - Output C multinomial distribution on the output layer:

$$p(w_{c,j} = w_{O,c} | w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V} \exp(u_{j'})}$$



Skip-Gram Model

- The loss function

$$
\begin{aligned}
E &= -\log p(w_{O,1}, w_{O,2}, \cdots, w_{O,C} | w_I) \\
&= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j_c^*})}{\sum_{j'=1}^{V} \exp(u_{j'})} \\
&= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp(u_{j'})
\end{aligned}
$$

Skip-gram Model

- Training:
  - Take the derivative of E w..r.t the net input of every unit on every panel of the output layer

$$\frac{\partial E}{\partial u_{c,j}} = y_{c,j} - t_{c,j} := e_{c,j}$$

$$\mathrm{EI}_j = \sum_{c=1}^{C} e_{c,j}$$

$$V\text{-dimensional vector } \mathrm{EI} = \{\mathrm{EI}_1, \cdots, \mathrm{EI}_V\}$$

  - Take the derivative of E w.r.t W'

$$\frac{\partial E}{\partial w'_{ij}} = \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} \cdot \frac{\partial u_{c,j}}{\partial w'_{ij}} = \mathrm{EI}_j \cdot h_i$$

Skip-gram Model

- Training:

  Update equation for hidden-output matrix W’

$$w_{ij}'^{(\text{new})} = w_{ij}'^{(\text{old})} - \eta \cdot \text{EI}_j \cdot h_i$$

  or

$$\mathbf{v}_{w_j}'^{(\text{new})} = \mathbf{v}_{w_j}'^{(\text{old})} - \eta \cdot \text{EI}_j \cdot \mathbf{h} \qquad \text{for } j = 1, 2, \cdots, V.$$

  Update for input-hidden matrix:

$$\mathbf{v}_{w_I}^{(\text{new})} = \mathbf{v}_{w_I}^{(\text{old})} - \eta \cdot \text{EH}^T$$

$$\text{EH}_i = \sum_{j=1}^{V} \text{EI}_j \cdot w_{ij}'.$$

Outline

- Motivation
- SVD-based method
- Language Model
- CBOW
- Skip-gram Model
- Efficient Training for Word Embedding

Computational Difficulties of CBOW and Skip-gram

- Skip-gram:

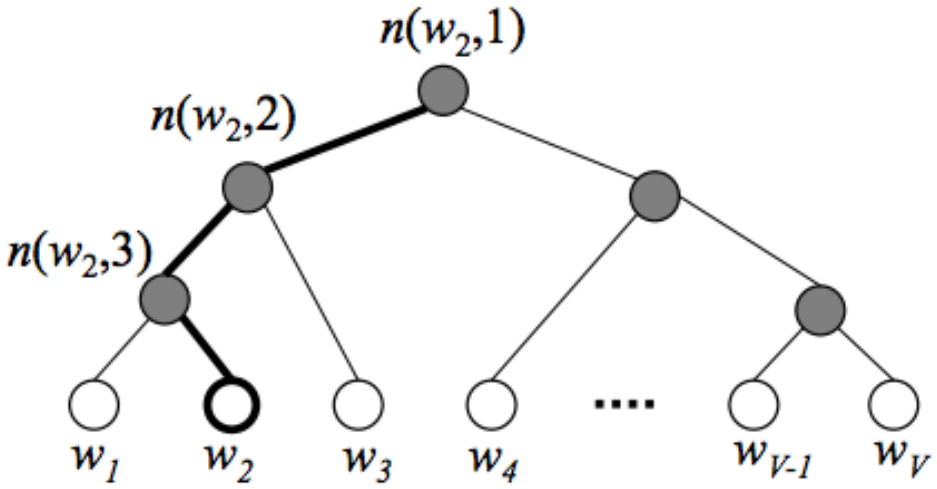
$$
\begin{aligned}
E &= -\log p(w_{O,1}, w_{O,2}, \cdots, w_{O,C} | w_I) \\
&= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j_c^*})}{\sum_{j'=1}^{V} \exp(u_{j'})} \\
&= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp(u_{j'})
\end{aligned}
$$

- CBOW:

$$
\begin{aligned}
E &= = -\log p(w_O | w_{I,1}, \cdots, w_{I,C}) \\
&= -u_{j^*} + \log \sum_{j'=1}^{V} \exp(u_{j'}) \\
&= -{\mathbf{v}'_{w_O}}^T \cdot \mathbf{h} + \log \sum_{j'=1}^{V} \exp({\mathbf{v}'_{w_j}}^T \cdot \mathbf{h})
\end{aligned}
$$

Hierarchical Softmax

- Words are organized into a binary tree



- V words must be leaf units of the tree
- There are V-1 inner units
- Each inner unit is associated with an output vector $\mathbf{v}'_{n(w,j)}$

Hierarchical Softmax

- The error function

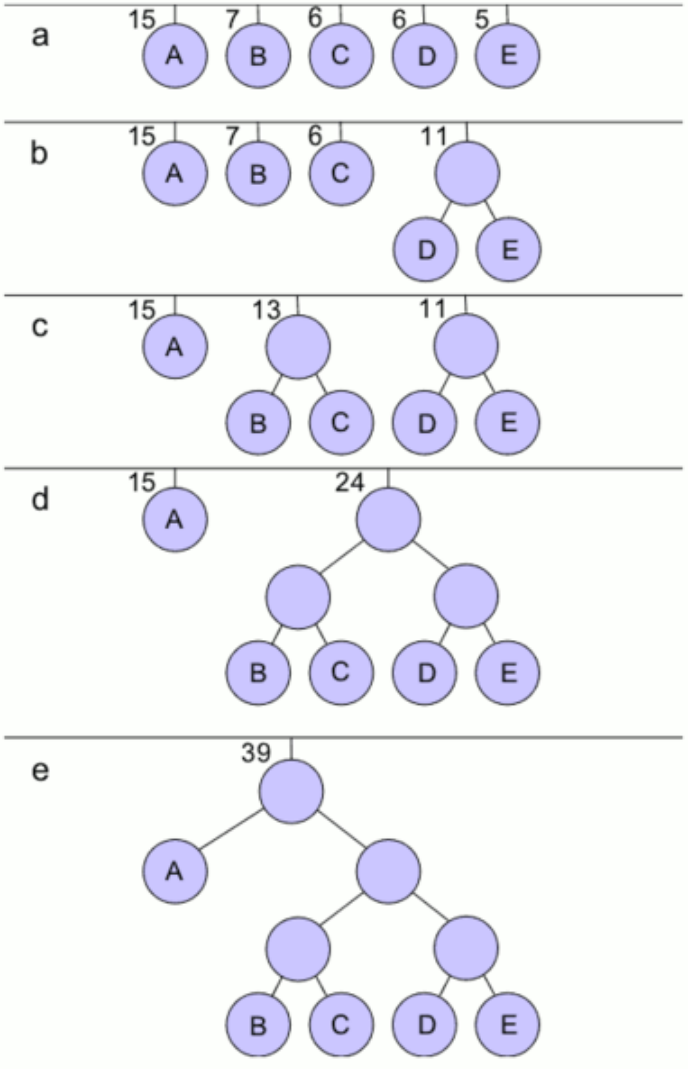
$$E = -\log p(w = w_O|w_I) = -\sum_{j=1}^{L(w)-1} \log \sigma([\![\cdot]\!]\mathbf{v}'^{T}_{j}\mathbf{h})$$

  - Where L(w) is the path from root to the word (w); the length is around log(V) only.

How to obtain the hierarchical structure for words?

- There are several approaches:
  - Knowledge-based: Wordnet (is-a relationship)
  - Automatic:
    - Huffman Tree: words with high frequent are represented with shorter code.

Huffman Coding Tree



Negative Sampling

- Sample a few words as negative sample:
  - The probabilistic distribution, which is needed for the sampling process can be arbitrarily chosen.
    - Mikolov et al. uses a unigram distribution raised to 3/4th power.
  - The training objective

$$E = -\log \sigma({\mathbf{v}'_{w_O}}^T \mathbf{h}) - \sum_{w_j \in \mathcal{W}_{\text{neg}}} \log \sigma(-{\mathbf{v}'_{w_j}}^T \mathbf{h})$$

Word2vec Tool: gensim

Word2vec Tool: Gensim

```
model = word2vec.Word2Vec.load_word2vec_format( word2vec_vectors_filename, binary=True )

for word in words:
    if word in model:
        print model[ word ]
```

Training a model

```
>>> # import modules & set up logging
>>> import gensim, logging
>>> logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=log
>>>
>>> sentences = [['first', 'sentence'], ['second', 'sentence']]
>>> # train word2vec on the two sentences
>>> model = gensim.models.Word2Vec(sentences, min_count=1)
```

Training a model with iterator

```
>>> class MySentences(object):
...     def __init__(self, dirname):
...         self.dirname = dirname
...
...     def __iter__(self):
...         for fname in os.listdir(self.dirname):
...             for line in open(os.path.join(self.dirname, fname)):
...                 yield line.split()
>>>
>>> sentences = MySentences('/some/directory') # a memory-friendly iterator
>>> model = gensim.models.Word2Vec(sentences)
```

Further Reading

1. Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv: 1301.3781 (2013).
2. Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.
3. Rong, Xin. "word2vec parameter learning explained." arXiv preprint arXiv:1411.2738 (2014).
4. TensorFlow Lib: https://www.tensorflow.org/versions/r0.11/tutorials/word2vec/index.html
5. Deep Learning Glossary: http://www.wildml.com/deep-learning-glossary/
6. Deep Learning Textbook: http://www.deeplearningbook.org/