Word Embedding

孙林
360 AI研究院
2017.6

OUTLINE

- Background
- Methods
- Evaluation
- Tools
- Summary

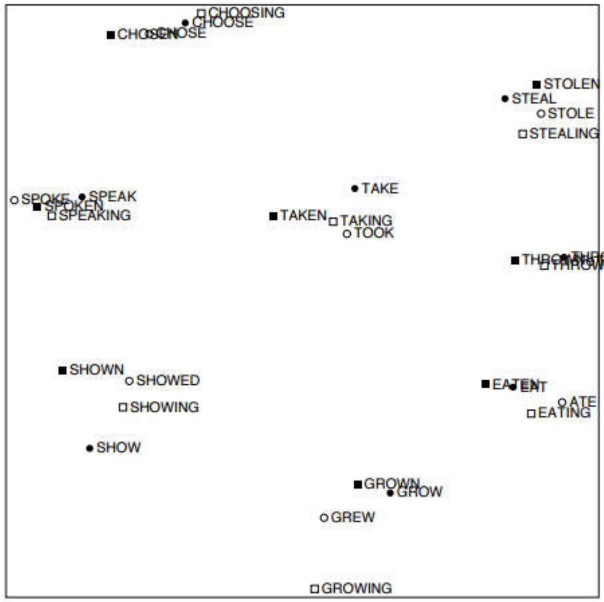
From symbolic to distributional/distributed representations

- sparse high-dimensional -> dense low-dimensional
- one-hot representation -> word embedding

$$[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]$$

$$\text{motel}\ [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$$
$$\text{hotel}\ [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0] = 0$$

$$\textit{shown} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

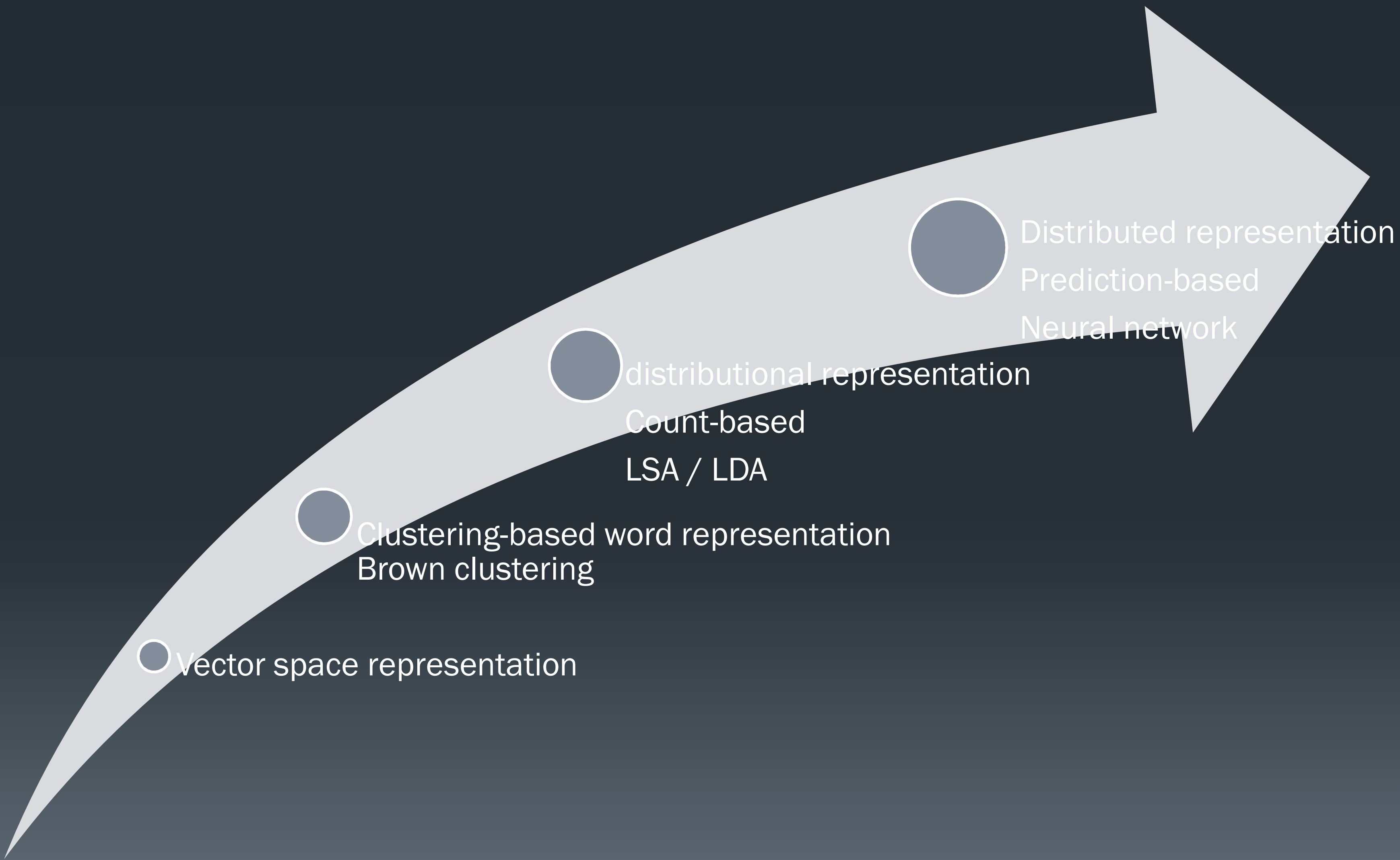


[Rohde et al. 2005. An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence]

distributional hypothesis

- Harris (1954), Firth(1957)
  - You can get a lot of value by representing a word by means of its neighbors
  - "You shall know a word by the company it keeps"
- two respects
  - context representation
  - modelling the relationship between word and context

roadmap


Vector space representation
Clustering-based word representation
Brown clustering
distributional representation
Count-based
LSA / LDA
Distributed representation
Prediction-based
Neural network

Background

- https://www.researchgate.net/publication/301779119_A_Survey_of_Word_Embedding_Literature_Context_Representations_and_the_Challenge_of_Ambiguity
- https://rare-technologies.com/making-sense-of-word2vec/
- http://u.cs.biu.ac.il/~yogo/nnlp.pdf

OUTLINE

- Background
- Methods
  - Clustering-based word representation
  - distributional representation (Count-based)
  - distributed representation
- Evaluation
- Tools
- Summary

distributional clustering

- https://arxiv.org/pdf/cmp-lg/9408011.pdf
- Brown clustering
  - http://blog.csdn.net/u014516670/article/details/50574147
  - http://blog.csdn.net/dark_scope/article/details/8879656

distributional representation(Count-based)

- co-occurrence matrix

	context1	context2	context3	context4
word1	count1	count2	count3	count4
word2	count5	count6	count7	count8
word3	count9	count10	count11	count12

- Similarity: cosine(word1, word2)
- Word1 = {context $_i$: count $_i$ }
- Word2 = {context $_j$: count $_j$ }

Details of co-occurrence matrix

- Content
  - Word   word-word   matrix
  - N-gram   word-ngram   matrix
  - Document   word-doc   matrix
  - …
- Count
  - Tf-idf
  - PMI
  - log(count)
  - …
- Matrix Factorization
  - SVD
  - NMF
  - CCA
  - Hellinger PCA
  - …

LSA(pLSA, LDA) & GloVe

- LSA
  - Word-document
  - tf-idf
  - SVD
- GloVe
  - Word-word
  - Log(dynamic_window(count))
  - Latent Factor Model
  - https://nlp.stanford.edu/pubs/glove.pdf

Distributed representation(prediction-based)

- Use Language model

$$P(w_1, w_2, ..., w_m) = P(w_1)\ P(w_2|w_1)\ P(w_3|w_1, w_2) \ldots P(w_i \mid w_1, w_2, ..., w_{i-1}) \ldots P(w_m \mid w_1, w_2, ..., w_{m-1})$$

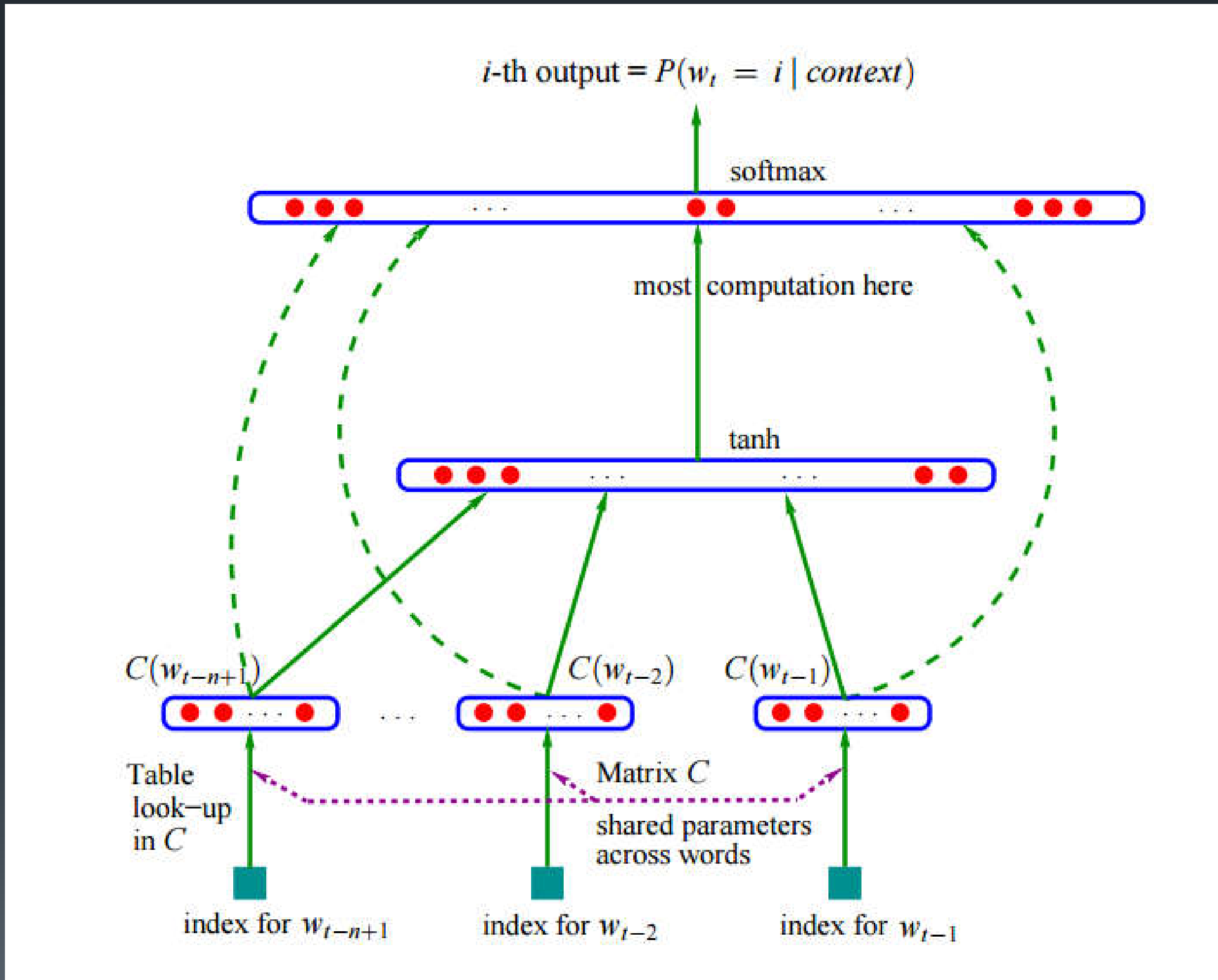
- N-gram language model

$$P(w_i \mid w_1, w_2, ..., w_{i-1}) \approx P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

$$P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1})}$$

- Trigram model(n=3)

NNLM



$$\hat{P}(w_t|w_{t-1},\cdots w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

$$y = b + Wx + U\tanh(d + Hx)$$

$$x = (C(w_{t-1}), C(w_{t-2}), \cdots, C(w_{t-n+1}))$$

Log-Bilinear Language Model (LBL)

$$E(w_i; w_{i-(n-1):i-1}) = \boldsymbol{b}^{(2)} + \boldsymbol{e}(w_i)^{\mathrm{T}}\boldsymbol{b}^{(1)} + \boldsymbol{e}(w_i)^{\mathrm{T}} H \left[\boldsymbol{e}(w_{i-(n-1)}); \ldots; \boldsymbol{e}(w_{i-1})\right]$$

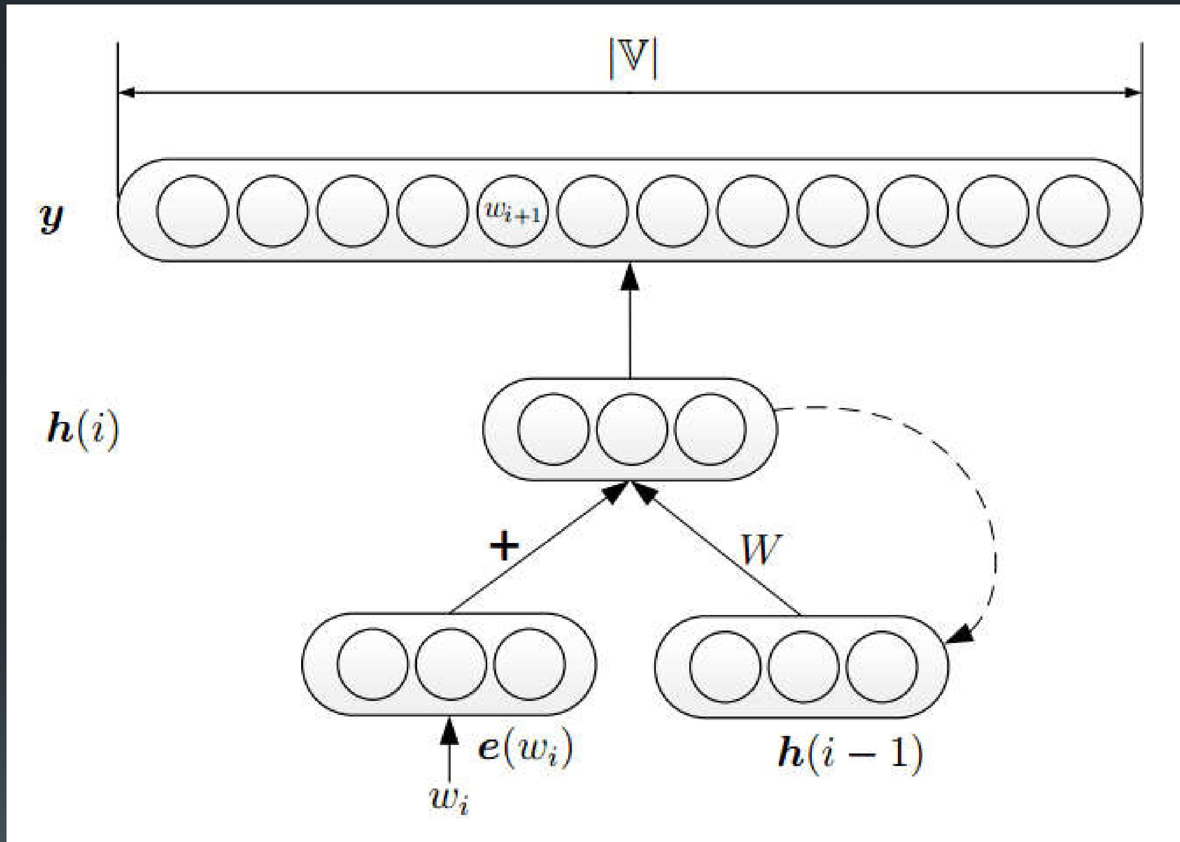
$$y_j = \sum_{i=1}^{n-1} C(w_j)^T H_i C(w_i)$$

$$h = \sum_{i=1}^{t-1} H_i C(w_i)$$

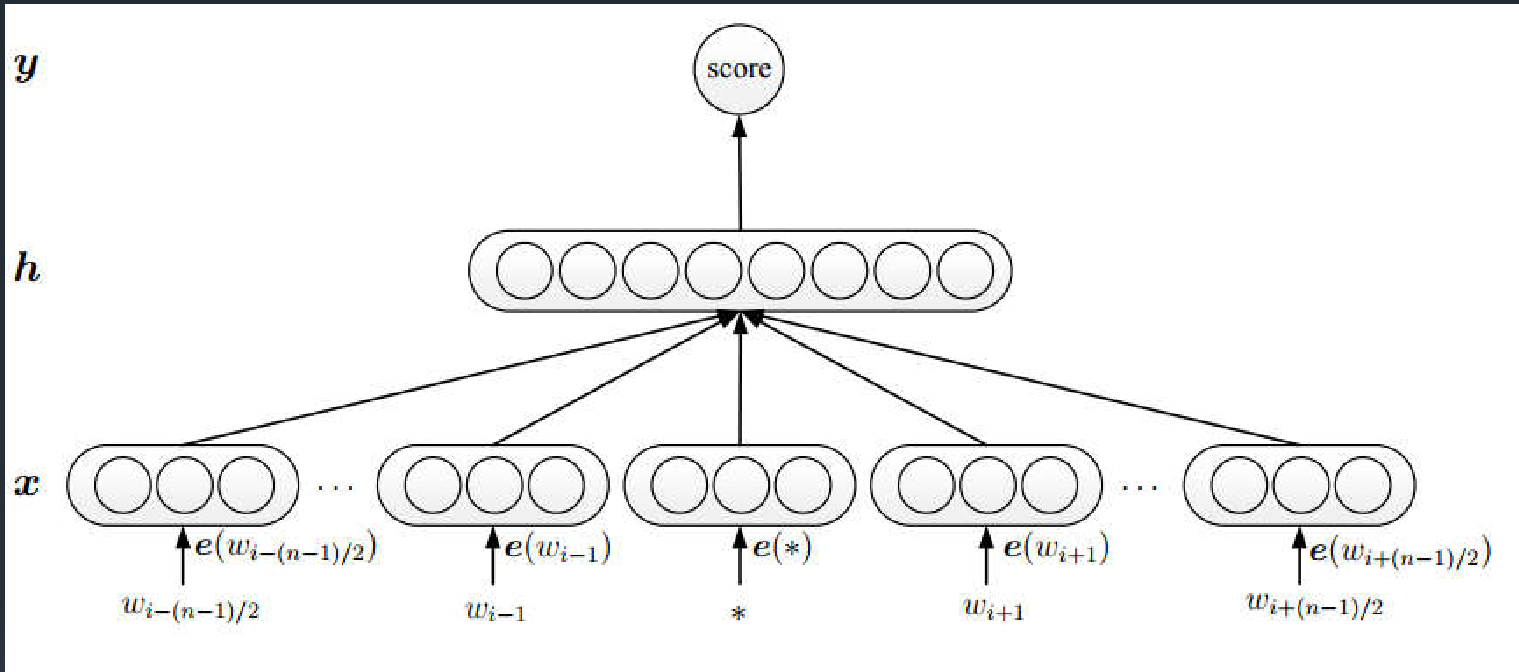
$$y_j = C(w_j)^T h$$

- Hierarchical LBL（HLBL）
  - Hierarchical softmax， *O*(log(|V|))
- ivLBL
  - NCE, *O*(c)

RNN based Language Model (RNNLM)

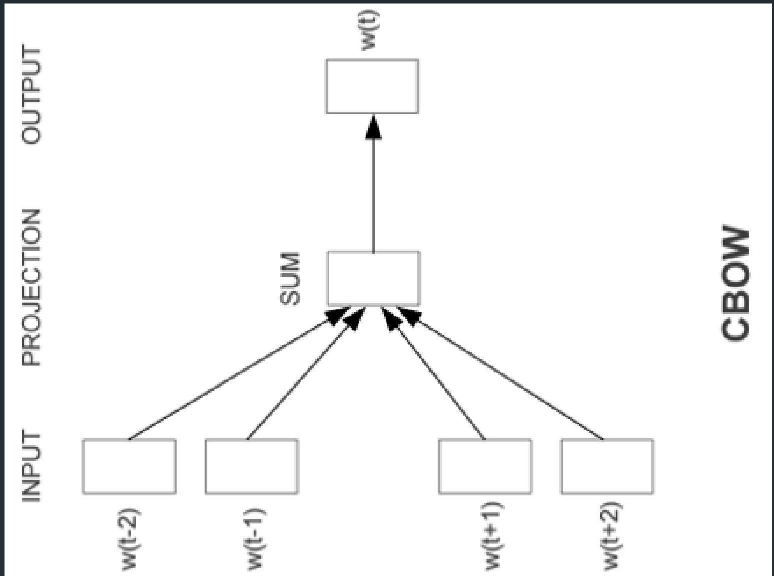


C&W



$$\textit{minimize} \sum_{(w,c)\in\mathbb{D}} \sum_{w'\in\mathbb{V}} \max(0, 1 - \text{score}(w,c) + \text{score}(w',c))$$

CBOW & Skip-gram



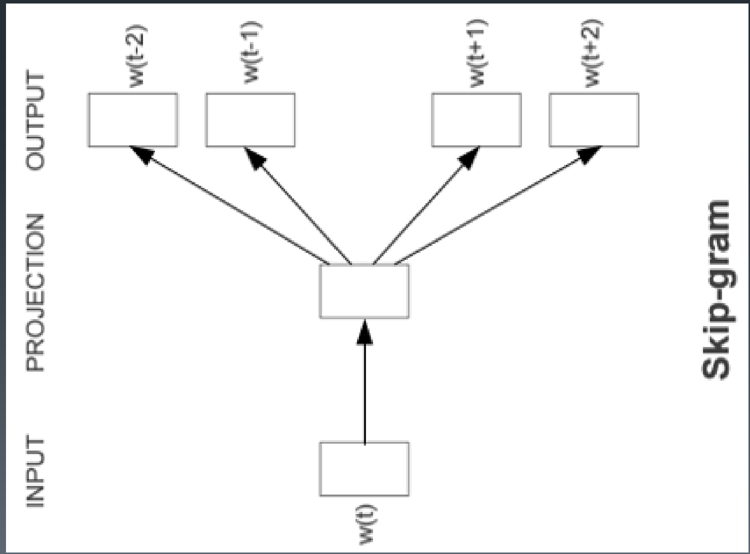
$$\textit{maximize} \sum_{(w,c)\in\mathbb{D}} \log P(w|c)$$

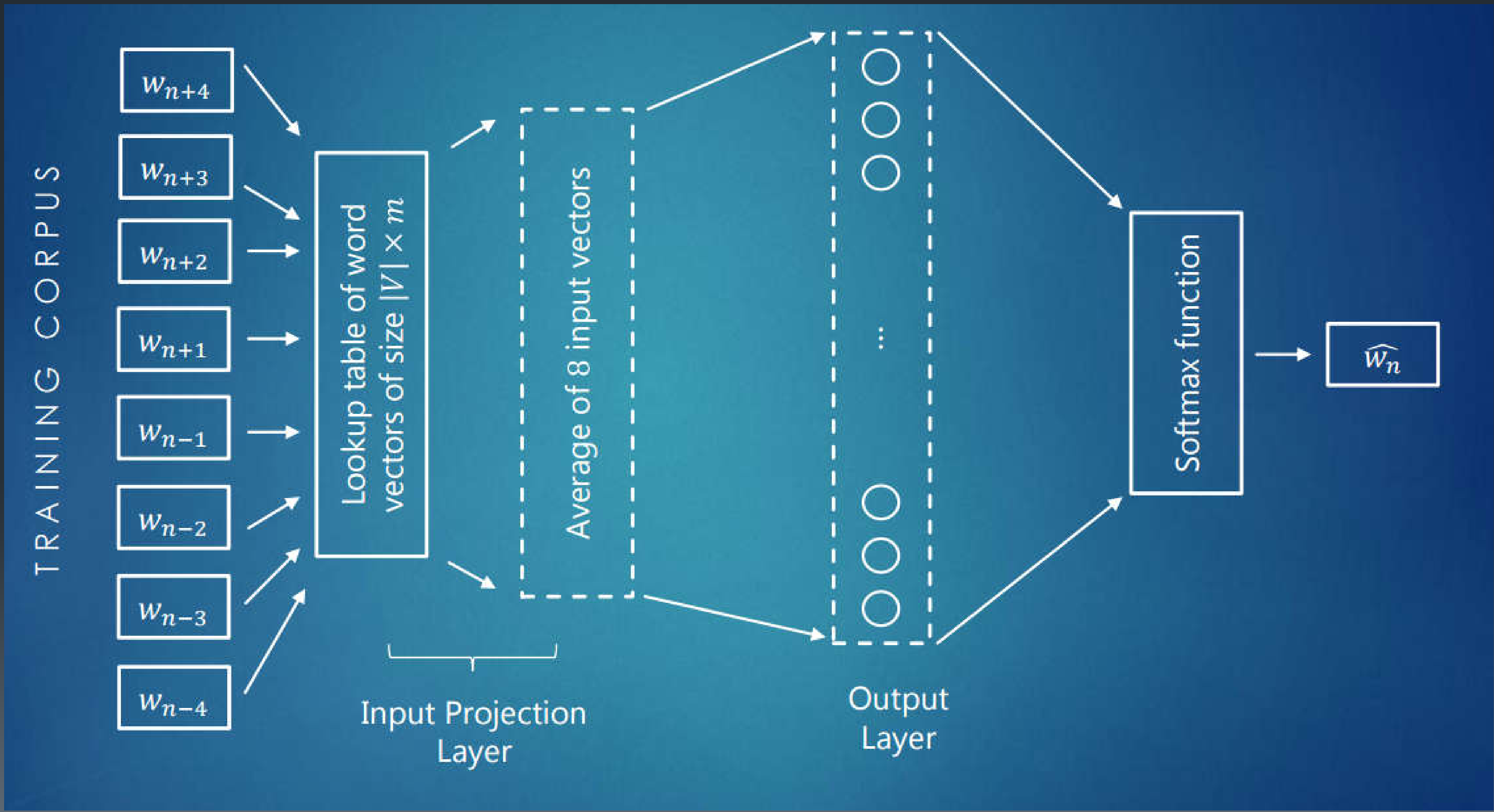
$$P(w|c) = \frac{\exp\left(\boldsymbol{e}'(w)^{\mathrm{T}}\boldsymbol{x}\right)}{\sum_{w'\in\mathbb{V}} \exp\left(\boldsymbol{e}'(w')^{\mathrm{T}}\boldsymbol{x}\right)}$$

$$\boldsymbol{x} = \frac{1}{n-1}\sum_{w_j\in c} \boldsymbol{e}(w_j)$$

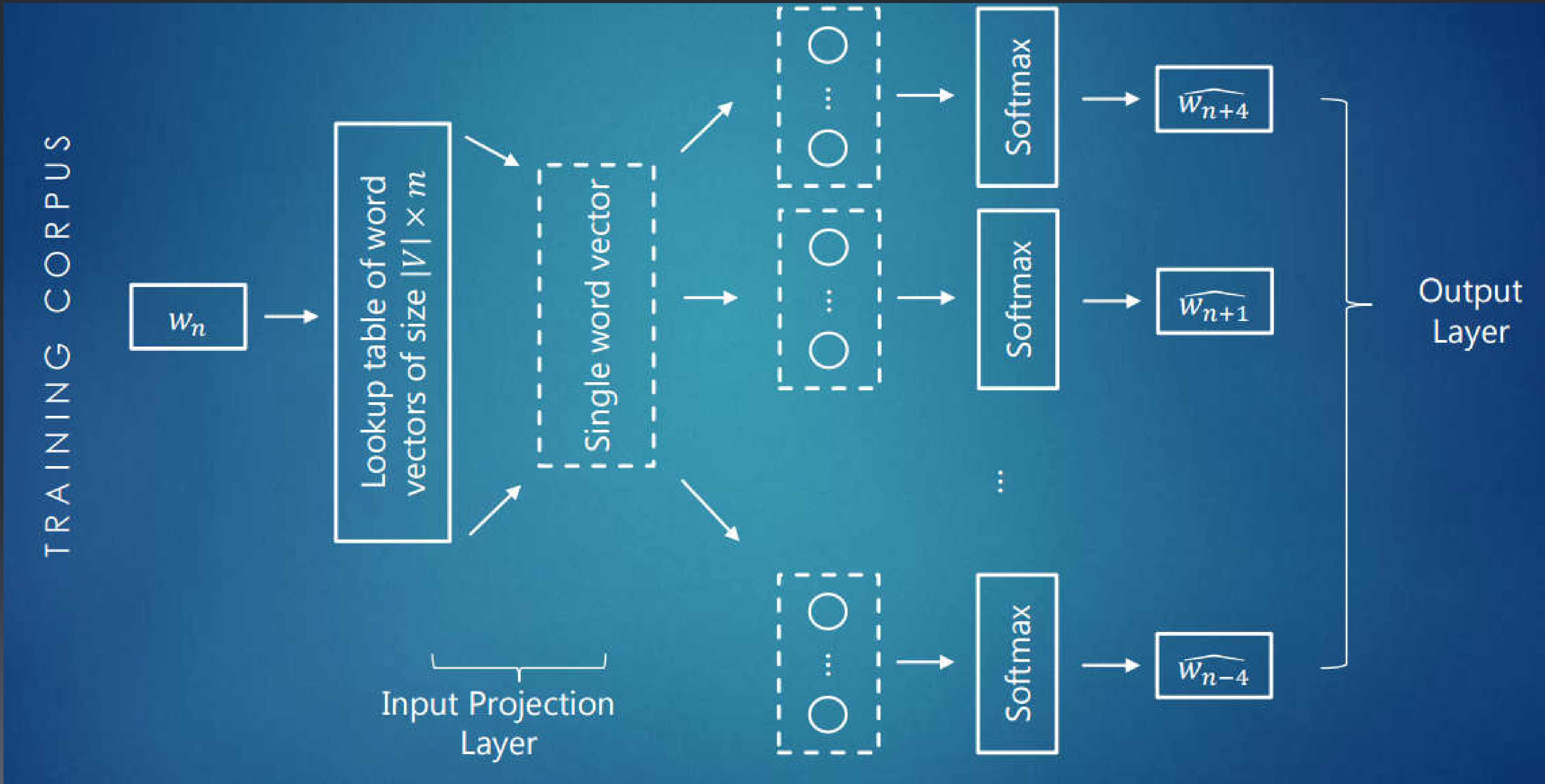
$$\textit{maximize} \ \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t)$$

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^{\top} v_{w_I}\right)}$$





TRAINING CORPUS
w_{n+4}
w_{n+3}
w_{n+2}
w_{n+1}
w_{n-1}
w_{n-2}
w_{n-3}
w_{n-4}
Lookup table of word vectors of size $|V| \times m$
Average of 8 input vectors
...
Softmax function
$\widehat{w_n}$
Input Projection Layer
Output Layer


TRAINING CORPUS
w_n
Lookup table of word vectors of size $|V| \times m$
Single word vector
Softmax
Softmax
Softmax
$\widehat{w_{n+4}}$
$\widehat{w_{n+1}}$
$\widehat{w_{n-4}}$
Output Layer
Input Projection Layer

Reduce the calculation of the last layer

- Hidden layer -> output layer O(m * |V|)
- Hierarchical softmax O(log(|V|)) $$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma\left([\![n(w,j+1) = \mathrm{ch}(n(w,j))]\!] \cdot {v'_{n(w,j)}}^\top v_{w_I} \right)$$
- NCE O(c)
- Group o(√|V|)
- Negative sampling $$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]$$
- subsampling $$P(w) = 1 - \sqrt{\frac{t}{f(w)}}$$ $$P(w) = \frac{f(w) - t}{f(w)} - \sqrt{\frac{t}{f(w)}}$$

Short summary

type	*content*	*modelling relationship between content and word based on*
LSA/LSI	document	matrix
HAL	word	
GloVe	word	
Jones & Mewhort	ngram	
Brown Clustering	word	clustering
Skip-gram	word	neural network
CBOW	n-gram （weighted ）	
LBL	n-gram （linear combination）	
NNLM	n-gram （non-linear combination ）	
C&W	n-gram （non-linear combination ）	

From word embedding to sense embedding

- Ambiguity in embedding
  - the resulting embeddings are dependent on the data on which they have been trained
  - If only a small corpus has been used, thus not all senses have been captured
  - words are captured in a single vector representation, which does not account for the possible polysemy or homonymy of the represented words

Recent research in embeddings

- tuning embeddings to various tasks with the help of extra information
  - Wang2Vec . Ling et al. (2015) seek to improve the quality of embeddings for syntactically-motivated tasks . Make a small modification to the original word2vec models try to include word order information
- exploit extra factors (or features) from supervised data to tailor embeddings for the intended tasks, using "context" or "world knowledge"
  - The main idea
    - unsupervised vectors do not distinguish between word senses
    - Not able to capture all aspects of language structure
    - structural features ought to be added for better performance
    - Using combined objective methods

Methods

- http://licstar.net/archives/328
- http://sebastianruder.com/word-embeddings-1/
- https://nlp.stanford.edu/projects/glove/
- https://nlp.stanford.edu/pubs/glove.pdf
- http://clic.cimec.unitn.it/marco/publications/acl2014/baroni-etal-countpredict-acl2014.pdf
- http://www.lix.polytechnique.fr/~anti5662/word_embeddings_intro_tixier.pdf
- http://hci-kdd.org/wordpress/wp-content/uploads/2016/06/T2-185A83-WORD-VECTOR-TUTORIAL-VO-2016.pdf
- http://sebastianruder.com/word-embeddings-softmax/index.html

OUTLINE

- Background
- Methods
- Evaluation
  - metrics
  - How to Generate a Good Word Embedding
- Tools
- Summary

METRICS

- Embedding’ s Semantic Properties
  - similarity task, wordsim353
  - Synonym detection, toefl
  - syntactic and semantic analogy task, A -B=C-D
- Embedding as Features
  - Classification
  - NER
  - POS
  - …
- Embedding as the Initialization of NNs

How to Generate a Good Word Embedding

- Model
- Corpus
- parameters

How to choose proper models?

- Analyze its semantic properties
  - “c predicts w” is better than “scores w , c”
  - C&W has no analogy information
- Use it as a feature for supervised Tasks
  - Simple models provide sufficient performance in most cases, such as Skip-gram, CBOW
- Use it to initialize neural networks
  - Simple models provide sufficient performance in most cases, such as Skip-gram, CBOW
- Corpus size
  - Small corpus, using simple models, such as skip-gram
  - Large corpus, using more complex models, such as CBOW

The Effect of the Training Corpus

- corpus size
  - using a larger corpus can yield a better embedding, when the corpora are in the same domain
- corpus domain
  - the influence of the corpus domain is dominant (except for the syn task)
  - In-domain corpus is helpful for the tasks
  - Out-domain corpus even may has a negative effect
- Which is More Important, Size or Domain?
  - When no sufficient in-domain data, keep the corpus pure or add the out-domain corpus?
  - The corpus domain is more important than the corpus size

The Choice of the Training Parameters

- Number of Iterations
  - early stopping
    - stop the iterator when the loss on the validation set peaks?
    - For specific tasks, the loss on the word embedding validation set may be inconsistent with the task performance
    - using the development set for that task to determine when to stop iteration
    - using a simple task to verify whether the word embedding has peaked on other tasks, when testing the task performance would be excessively time consuming
- Dimensionality of the Embedding
  - for the semantic property tasks, larger dimensions will lead to better performance
  - for the NLP tasks, a dimensionality of 50 is typically sufficient

OUTLINE

- Background
- Methods
- Evaluation
- Future works
  - **interpretable relations**
  - **lexical resources**
  - **beyond words**
  - **beyond English**
- Summary

- http://yanran.li/peppypapers/2015/08/17/post-word-embedding.html

OUTLINE

- Background
- Methods
- Evaluation
- tools
- Summary

tools

- SENNA
- Gensim
- Glove
- word2vec

- An optimal vector representation does not exists

Thanks!