CMPE 258, Deep Learning

Sequence learning & NLP

May 01, 2018

DMH 149A

Taehee Jeong
Ph.D., Data Scientist

SJSU SAN JOSÉ STATE UNIVERSITY

Group Project schedule

Presentation date : 5/8, 5/10

Report (including code) due date : 5/6

Number of members : 1 to 4

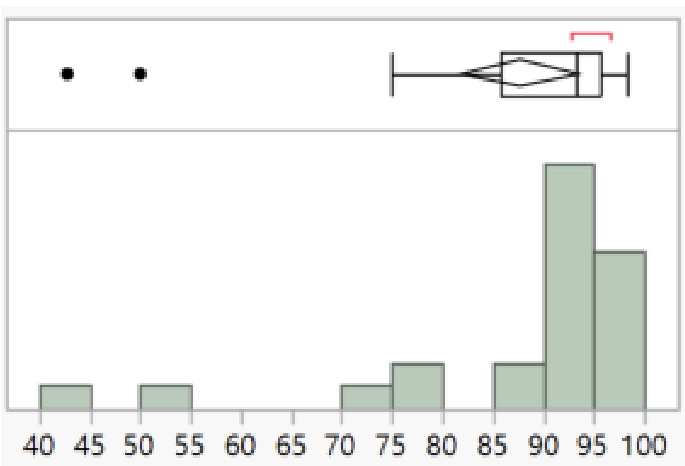
Content: DNN, CNN, RNN related

Platform : Pandas, Numpy, tensorflow, keras (please discuss with me for others)

Grading policy:

- Content : 40 pts
  ; Creativity in data collection, Neural network architecture / algorithm, application (same quality as a conference paper)
- Presentation : 20 pts
- Report : 20 pts
- Code : 20 pts

Mid-term exam 2 score



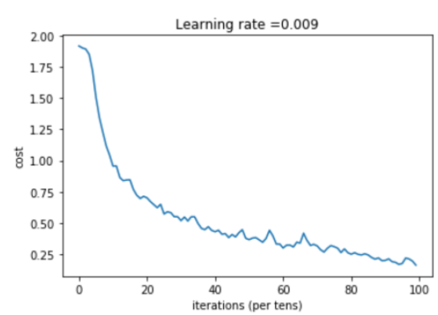
Median = 93.33

Mean = 87.67

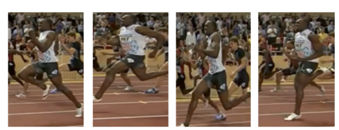
The submitted ipynb should be executable without any extra work and supposed be finished within 60 minutes.
If extra effort is needed to get reasonable result (whatever it is), 5 to 20 points for each event will be deducted.

Score = accuracy of testing data -20 + 10 (print out of CNN architecture) + 10 (plot of cost versus number of iteration)

Layer	Type	Size	Channels	Kernel size	Stride	Padding	Function
0	Input	64 x 64	3				
1	Convolution (C1)	32 x 32	8	4 x 4	2	1	ReLU
1	Pooling (P1)	28 x 28	8	5 x 5	1	0	max
2	Convolution (C2)	13 x 13	16	4 x 4	2	0	ReLU
2	Pooling (P2)	9 x 9	16	5 x 5	1	0	Avg
3	Flatten (F3)	1296					
4	Fully connected (F4)	108					ReLU
5	Fully connected (F5)	6					Sigmoid



Applications of Recurrent Neural Networks

Speech recognition		→	"The quick brown fox jumped over the lazy dog."
Music generation	∅	→	
Sentiment classification	"There is nothing to like in this movie."	→	
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger.

Coursera: Deep learning Specialization, Andrew Ng

Natural Language Process (NLP)

- Speech recognition
- Machine translation
- Chatbots (question answering)
- Sentiment classification
- Name entity recognition

Sentiment analysis

- Text is a sequence of words
- Word is a sequence of characters

How to separate words from a sentence?

Tokenization

- Tokenization is a process that splits an input sequence into tokens.
- We can split token by space, punctuation, a set of rule.

Python tokenization example

```
import nltk
text = "This is Andrew's text, isn't it?"
```

```
tokenizer = nltk.tokenize.WhitespaceTokenizer()
tokenizer.tokenize(text)
```

```
['This', 'is', "Andrew's", 'text,', "isn't", 'it?']
```

```
tokenizer = nltk.tokenize.TreebankWordTokenizer()
tokenizer.tokenize(text)
```

```
['This', 'is', 'Andrew', "'s", 'text', ',', 'is', "n't",
'it', '?']
```

```
tokenizer = nltk.tokenize.WordPunctTokenizer()
tokenizer.tokenize(text)
```

```
['This', 'is', 'Andrew', "'", 's', 'text', ',', 'isn',
 "'", 't', 'it', '?']
```

Coursera: Natural Language Processing, National Research University Higher School of Economics

Token normalization

Same token for different forms of words

- Examples
  - wolf, wolves → wolf
  - talk, talks → talk
- Stemming
  - removes and replaces suffixes to get to the root form of a word, which is called as stem.
- Lemmatization
  - returns the base or dictionary form of a word, which is known as lemma.

Coursera: Natural Language Processing, National Research University Higher School of Economics

Python stemming example

```
import nltk
text = "feet cats wolves talked"
tokenizer = nltk.tokenize.TreebankWordTokenizer()
tokens = tokenizer.tokenize(text)
```

```
stemmer = nltk.stem.PorterStemmer()
" ".join(stemmer.stem(token) for token in tokens)
```

```
u'feet cat wolv talk'
```

```
stemmer = nltk.stem.WordNetLemmatizer()
" ".join(stemmer.lemmatize(token) for token in tokens)
```

```
u'foot cat wolf talked'
```

Coursera: Natural Language Processing, National Research University Higher School of Economics

Transforming tokens into features

Bag of words (BOW)

For each token, we have a feature column, which is called text vectorization.

good movie
not a good movie
did not like

good	movie	not	a	did	like
1	1	0	0	0	0
1	1	1	1	0	0
0	0	1	0	1	1

Coursera: Natural Language Processing, National Research University Higher School of Economics

Preserve some ordering

N-grams: Token pairs, triplets, etc.

good movie
not a good movie
did not like

good movie	movie	did not	a	…
1	1	0	0	…
1	1	0	1	…
0	0	1	0	…

Coursera: Natural Language Processing, National Research University Higher School of Economics

Remove some n-grams

- High frequency n-grams
  - Articles, prepositions, etc. (example: and, a, the)
  - They are called stop-words. They do not help to discriminate texts.
- Low frequency n-grams
  - Typos, rare words

Coursera: Natural Language Processing, National Research University Higher School of Economics

Word Embedding

- Convert texts into numbers
- Map a word to a vector using a dictionary
- Applications
  - Sentiment analysis of reviews (amazon, movie review)
  - Document or news classification or clustering (google)

https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

Frequency based Embedding

- Count Vector
  - Frequency : Number of times a word has appeared in the document
  - Presence : Has the word appeared in the document?
- TF-IDF Vector
- Co-occurrence Vector

https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

Word Matrix

	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7	Document 8
Term(s) 1	10	0	1	0	0	0	0	2
Term(s) 2	0	2	0	0	0	18	0	2
Term(s) 3	0	0	0	0	0	0	0	2
Term(s) 4	6	0	0	4	6	0	0	0
Term(s) 5	0	0	0	0	0	0	0	2
Term(s) 6	0	0	1	0	0	1	0	0
Term(s) 7	0	1	8	0	0	0	0	0
Term(s) 8	0	0	0	0	0	3	0	0

Word Vector (Passage Vector)

Document Vector

How to make Term (word) features

1. All words in a dictionary
2. Unique words in corpus(all documents)

https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

TF-IDF

- N-gram with smaller frequency can be more discriminating because it can capture a specific issue in the text
- Term frequency (TF)
  - Frequency for term (or n-gram) t in document d

term frequency	$f_{t,d} / \sum_{t' \in d} f_{t',d}$

- Inverse document frequency (IDF)
  - N = D: total number of documents in corpus
  - $\{d \in D: t \in d\}$: Number of documents where the term t appears

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D: t \in d\}|}$$

Coursera: Natural Language Processing, National Research University Higher School of Economics

TF-IDF

Tfidf(t,d,D) = tf(t,d) x idf(t,D)

A high weight in TF-IDF means a high term frequency (in a given document) and a low document frequency of the term in a all collection of documents

good movie
not a good movie
did not like

good movie	movie	did not	…
0.17	**0.17**	0	…
0.17	**0.17**	0	…
0	0	**0.47**	…

Coursera: Natural Language Processing, National Research University Higher School of Economics

Term frequency & Inverse document frequency

1-gram

					term frequency					
text	good	movie	not	a	did	not	like	I	it	one
good movie										
not a good movie										
did not like										
I like it										
good one										

					inverse document frequency					
text	good	movie	not	a	did	not	like	I	it	one
good movie										
not a good movie										
did not like										
I like it										
good one										

Python TF-IDF example

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
texts = [
    "good movie", "not a good movie", "did not like",
    "i like it", "good one"
]
tfidf = TfidfVectorizer(min_df=2, max_df=0.5, ngram_range=(1, 2))
features = tfidf.fit_transform(texts)
pd.DataFrame(
    features.todense(),
    columns=tfidf.get_feature_names()
)
```

	good movie	like	movie	not
0	0.707107	0.000000	0.707107	0.000000
1	0.577350	0.000000	0.577350	0.577350
2	0.000000	0.707107	0.000000	0.707107
3	0.000000	1.000000	0.000000	0.000000
4	0.000000	0.000000	0.000000	0.000000

Coursera: Natural Language Processing, National Research University Higher School of Economics

Problem of one-hot representation

- It treats each word as a thing unto itself, and it doesn't allow an algorithm to easily generalize the cross words.
- Example
  - Sally Johnson is an orange farmer.
  - Robert Lin is an apple farmer.
  - Robert Lin is a durian cultivator.
- product of any two word vector is zero.

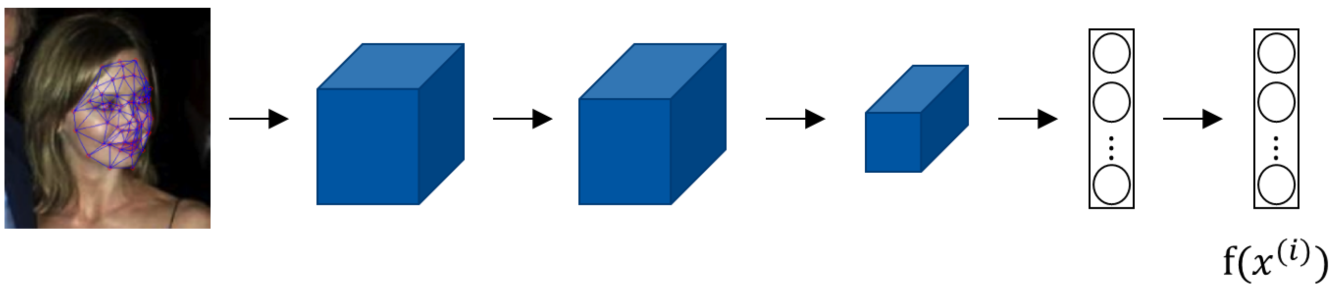
Coursera: Deep learning Specialization, Andrew Ng

Featured representation: word embedding

	Man	Woman	King	Queen	Apple	Orange
Gender	-1	1	-0.95	0.97	0	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97
Size						
Cost						
alive						
verb						

Word embedding vector

Deep face feature vector



If $x^{(i)}, x^{(j)}$ are same person, $d\left(f(x^{(i)}) - f(x^{(j)})\right)$ is small.

If $x^{(i)}, x^{(j)}$ are different persons, $d\left(f(x^{(i)}) - f(x^{(j)})\right)$ is large.

Taigman et. al., 2014. DeepFace closing the gap to human level performance

Transfer learning and word embeddings

- Learn word embeddings from large text corpus. (1-100B words)
  (Or download pre-trained embedding online.)
- Transfer embedding to new task with smaller training set. (say, 100k words)
- Continue to fine-tune the word embeddings with new data.

Coursera: Deep learning Specialization, Andrew Ng

Co-occurrence Matrix

- Hypothesis: Similar words tend to occur together and will have similar context.
- Example
  - Apple is a fruit. Mango is a fruit.
- Co-occurrence
  - For a given corpus, the co-occurrence of a pair of words is the number of times they have appeared together in a context window
- Context window
  - Context window is specified by a number and the direction

Quick	Brown	Fox	Jump	Over	The	Lazy	Dog

Quick	Brown	Fox	Jump	Over	The	Lazy	Dog

https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

Co-occurrence Matrix

Corpus = He is not lazy. He is intelligent. He is smart.

	He	is	not	lazy	intelligent	smart
He	0	4	2	1	2	1
is	4	0	1	2	2	1
not	2	1	0	1	0	0
lazy	1	2	1	0	0	0
intelligent	2	2	0	0	0	0
smart	1	1	0	0	0	0

Context windows = 2

He	is	not	lazy	He	is	intelligent	He	is	smart
He	is	not	lazy	He	is	intelligent	He	is	smart
He	is	not	lazy	He	is	intelligent	He	is	smart
He	is	not	lazy	He	is	intelligent	He	is	smart

https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

Co-occurrence Matrix

Matrix size

- V x V
  - Not practical
- V x N
  - N is a subset of V and can be obtained by removing irrelevant words like stop words
- V x k
  - k is k principal components out of V using PCA

PCA to decompose Co-occurrence matrix

X = U·S · V^T (Singular value decomposition)

U and S represent word vector

V presents word context

U is principal component.

$$\underset{m \times n}{\overset{\hat{X}}{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}}} \approx \underset{m \times r}{\overset{U}{\begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}}} \underset{r \times r}{\overset{S}{\begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}}} \underset{r \times n}{\overset{V^{\mathsf{T}}}{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}}}$$

m x m　　m x n　　n x n

https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

Frequency based Embedding

- Count Vector
  - Frequency : Number of times a word has appeared in the document
  - Presence : Count if the word appeared in the document
- TF-IDF Vector
- Co-occurrence Vector

https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

Analogies

	Man	Woman	King	Queen	Apple	Orange
Gender	-1	1	-0.95	0.97	0	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97

Man → Woman vs. King → ? (Queen)

$e_{Man} - e_{Woman} \approx e_{King} - e_{Queen}$

Linguistic regularities in continuous space word representations, Mikolov et. al., 2013,

Coursera: Deep learning Specialization, Andrew Ng

Analogies using word vectors

Man → Woman vs. King → ? (Queen)

$e_{Man} - e_{Woman} \approx e_{King} - e_{Queen}$

$e_{Man} - e_{Woman} = \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $e_{King} - e_{Queen} = \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

Man:Woman as Boy:Girl
Ottawa:Canada as Nairobi:Kenya
Big:Bigger as Tall:Taller
Yen:Japan as Ruble:Russia

$e_{Man} - e_{Woman} \approx e_{King} - e_{w}$

Find word w : arg max (e_w , $e_{King} - e_{Man} + e_{Woman}$)

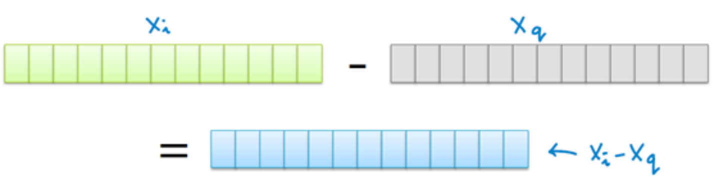
Coursera: Deep learning Specialization, Andrew Ng

Distance between two word vectors

(non-scaled) Euclidean distance

Defined in terms of inner product

$$\text{distance}(\mathbf{x}_i, \mathbf{x}_q) = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)^T(\mathbf{x}_i - \mathbf{x}_q)}$$
$$\sqrt{(\mathbf{x}_i[1]-\mathbf{x}_q[1])^2 + \ldots + (\mathbf{x}_i[d]-\mathbf{x}_q[d])^2}$$



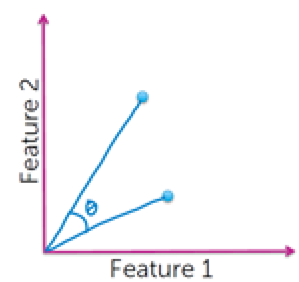
Cosine similarity – normalize

$$\textbf{Similarity} = \frac{\sum_{j=1}^{d} \mathbf{x}_i[j]\,\mathbf{x}_q[j]}{\sqrt{\sum_{j=1}^{d}(\mathbf{x}_i[j])^2}\sqrt{\sum_{j=1}^{d}(\mathbf{x}_q[j])^2}}$$

$a^T b = \|a\|\,\|b\| \cos(\theta)$

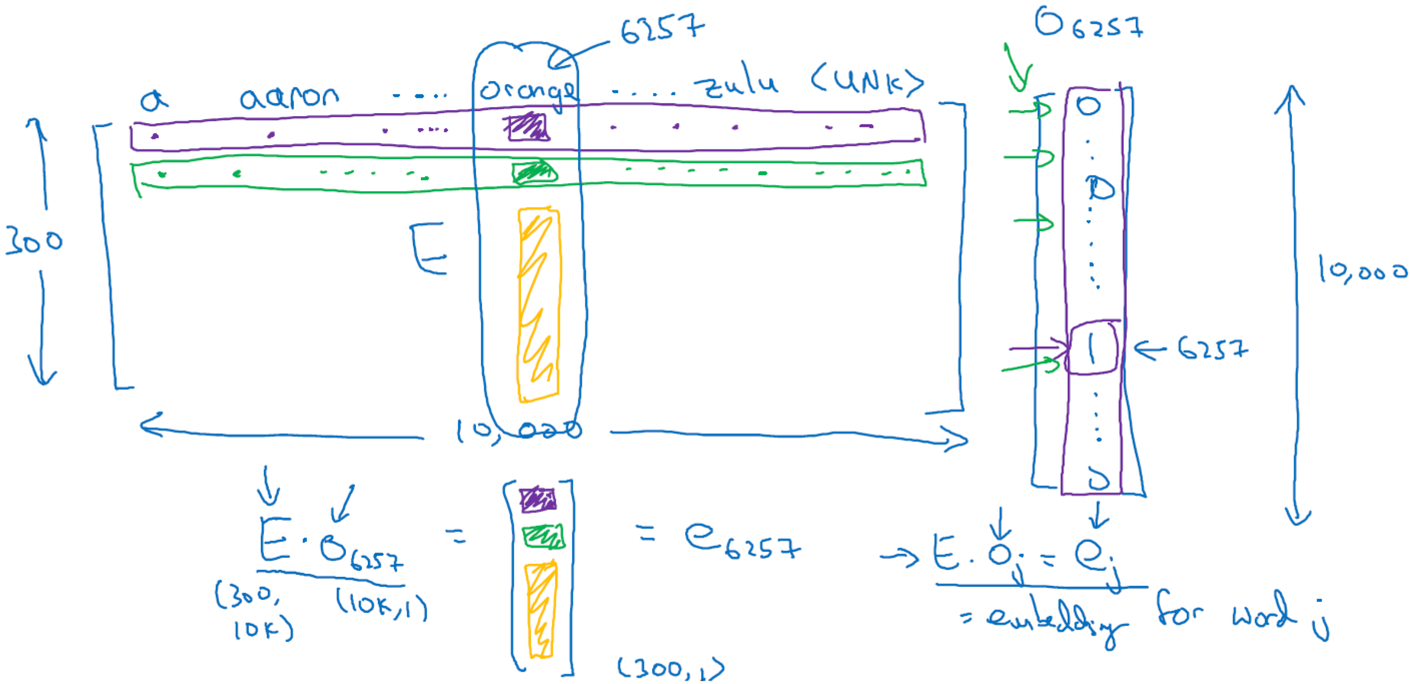
$$\frac{\mathbf{x}_i^T \mathbf{x}_q}{\|\mathbf{x}_i\|\,\|\mathbf{x}_q\|} = \cos(\theta)$$

$$= \left(\frac{x_i}{\|x_i\|}\right)^T \left(\frac{x_q}{\|x_q\|}\right)$$ first normalize



Coursera: Machine Learning, Emily Fox & Carlos Guestrin

Pull out word vector from Embedding matrix



Coursera: Deep learning Specialization, Andrew Ng

SJSU SAN JOSÉ STATE UNIVERSITY

Language model using word vector

I	want	a	glass	of	orange	______.
4343	9665	1	3852	6163	6257	

$e_{4343} = E\, o_{4343}$

I	o_{4343}	$\longrightarrow$	E	$\longrightarrow$	$\underline{e_{4343}}$
want	o_{9665}	$\longrightarrow$	E	$\longrightarrow$	$\underline{e_{9665}}$
a	o_{1}	$\longrightarrow$	E	$\longrightarrow$	$\underline{e_{1}}$
glass	o_{3852}	$\longrightarrow$	E	$\longrightarrow$	$\underline{e_{3852}}$
of	o_{6163}	$\longrightarrow$	E	$\longrightarrow$	$\underline{e_{6163}}$
orange	o_{6257}	$\longrightarrow$	E	$\longrightarrow$	e_{6257}

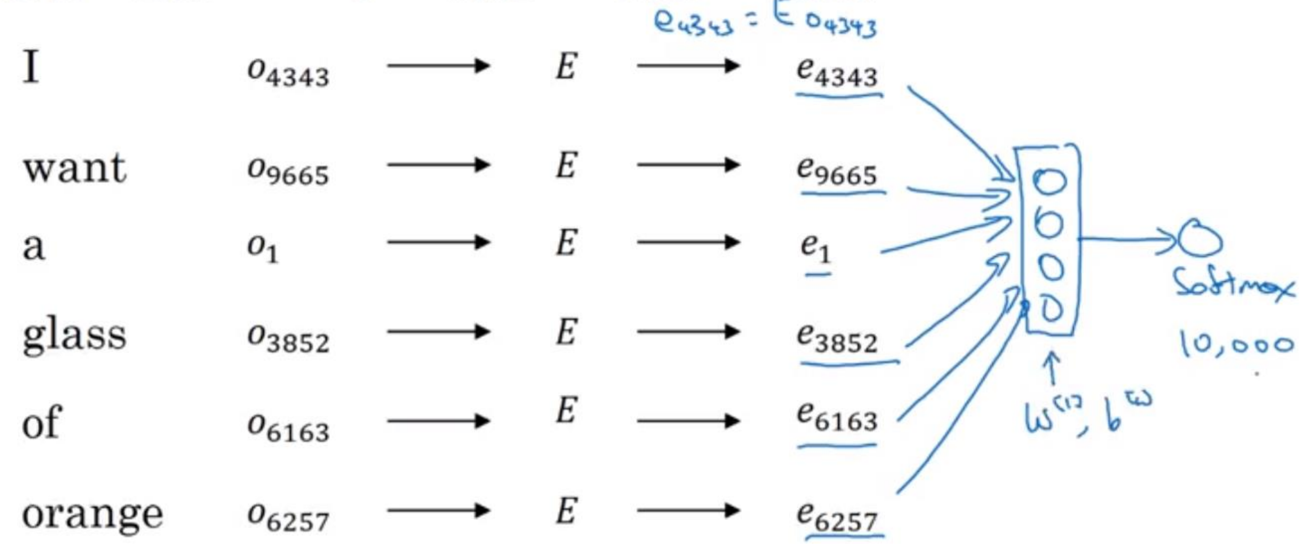
Coursera: Deep learning Specialization, Andrew Ng

Language model using word vector

A neural probabilistic language model , Bengio et. al., 2003

$$\hat{P}(w_1^T) = \prod_{t=1}^{T} \hat{P}(w_t | w_1^{t-1})$$

I	want	a	glass	of	orange	juice .
4343	9665	1	3852	6163	6257	

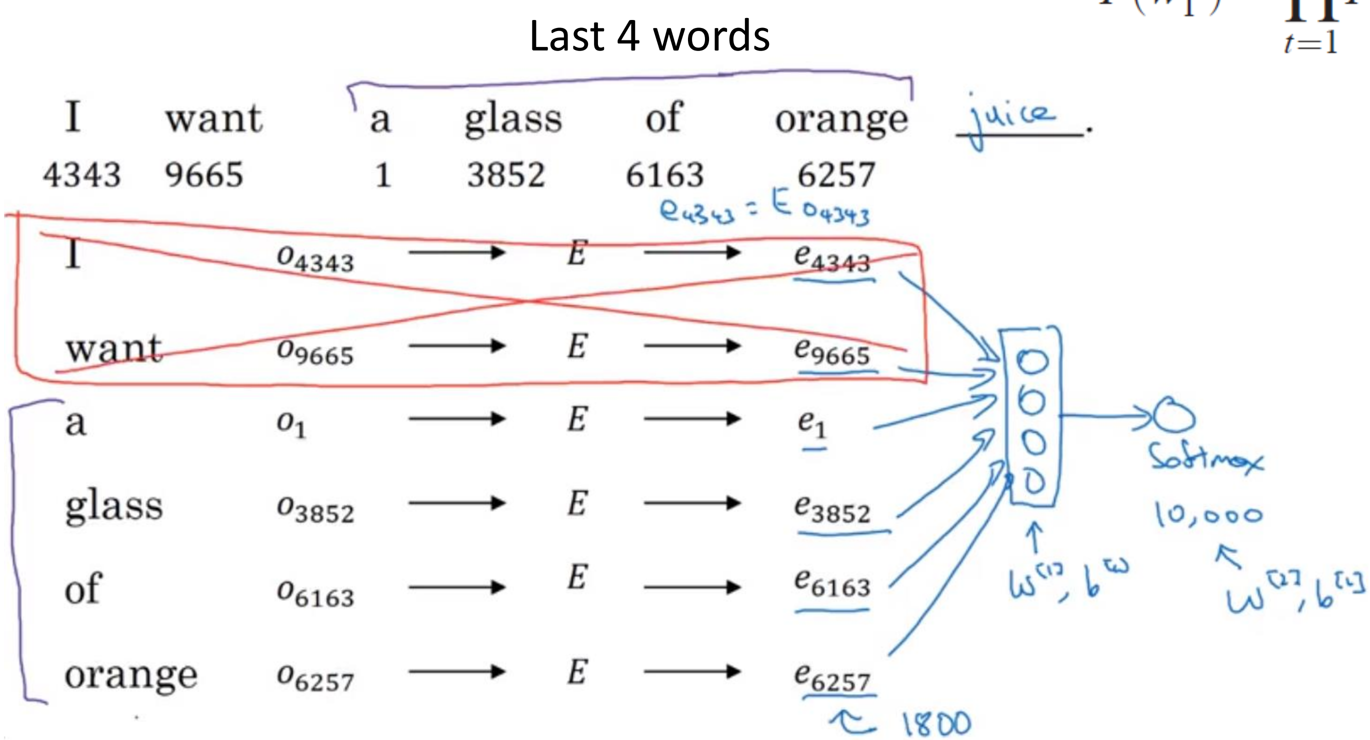


Coursera: Deep learning Specialization, Andrew Ng

Language model using word vector

A neural probabilistic language model , Bengio et. al., 2003

$$\hat{P}(w_1^T) = \prod_{t=1}^{T} \hat{P}(w_t | w_1^{t-1})$$



Coursera: Deep learning Specialization, Andrew Ng

Word2vec : Prediction based on Embedding

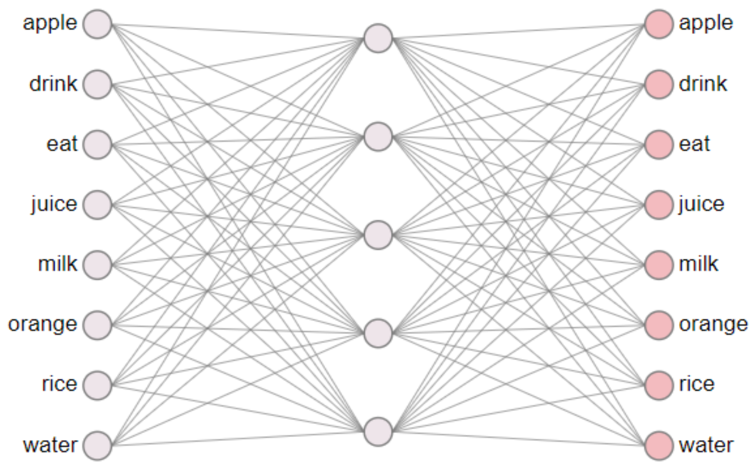
- CBOW (Continuous Bag of words)
  - Predicts the current word based on the context.
- Skip-gram
  - Predicts surrounding words given the current word.

Efficient Estimation of Word Representations in Vector Space, Tomas Mikolov et al.,2013

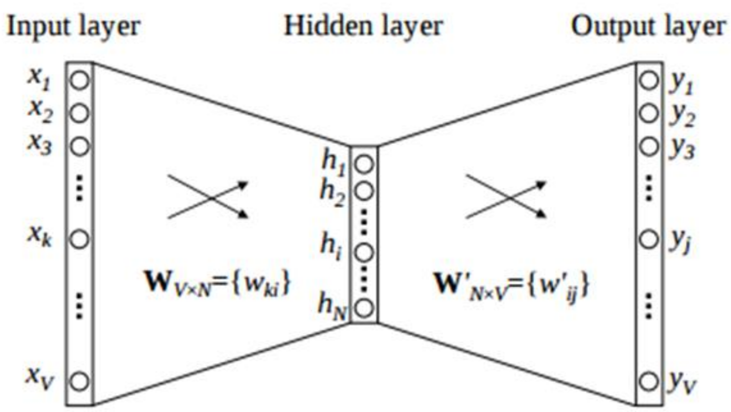
CBOW(Continuous Bag of words)

Predicts the current word based on the context

word vector :
the weight between the hidden layer and the output layer



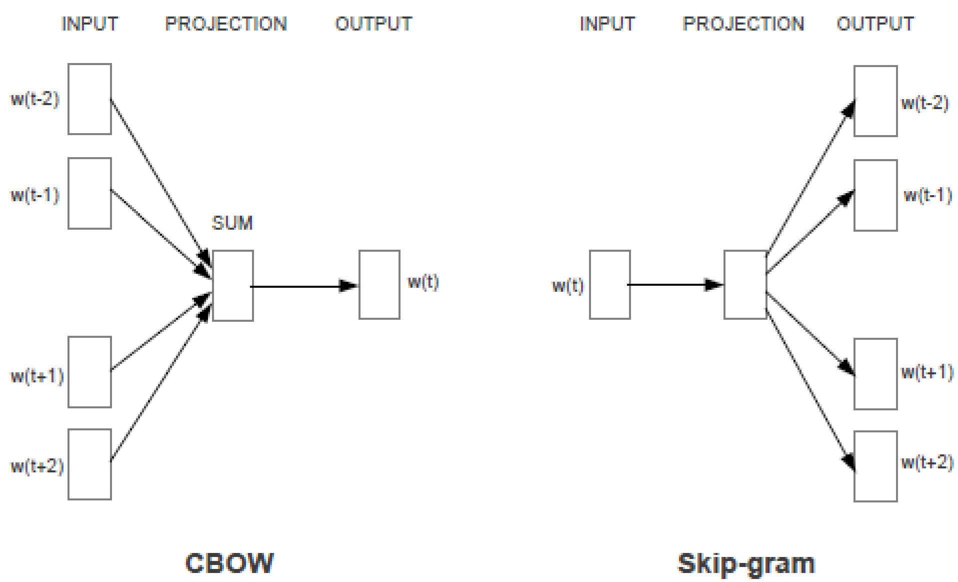
https://ronxin.github.io/wevi/



Linear activation (no activation function between any layers)

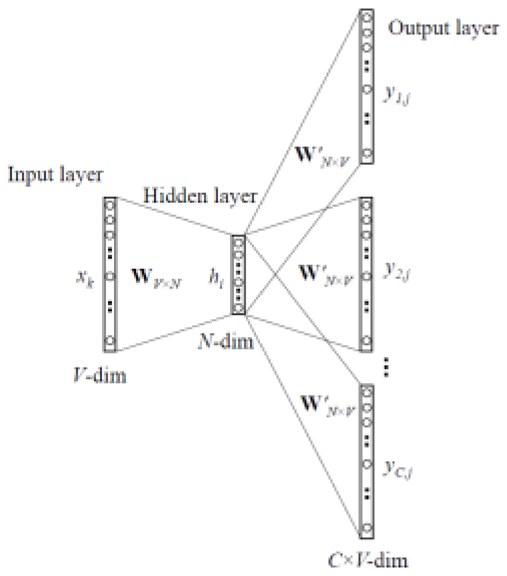
https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

Word2vec : Prediction based on Embedding



Efficient Estimation of Word Representations in Vector Space, Tomas Mikolov et al.,2013

Skip-gram



Predicts surrounding words given the current word

https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

Context/target pair

I want a glass of orange juice to go along with my cereal.

target

Context:

Last 4 words : a glass of orange

4 words on left & right : a glass of orange, to go along with

Last 1 word : orange

Nearby 1 word → skip-gram

Coursera: Deep learning Specialization, Andrew Ng

Skip-gram

I want a glass of orange juice to go along with my cereal.

Content	Target	
Orange	Juice	Next 1 word
Orange	Glass	Left/right 2 windows
Orange	to	

Coursera: Deep learning Specialization, Andrew Ng

Skip-gram

How to sample content word?

- Randomly select → there is high chance to pick up most of stop-words (a, the, and, etc)
- Select from less-frequent words

Coursera: Deep learning Specialization, Andrew Ng

Softmax-classification

Vocab size = 10,000k

$x \longrightarrow y$

Context c ("orange") 6257 $\longrightarrow$ Target t ("juice") 4834

$O_c \rightarrow E \rightarrow e_c \rightarrow \underset{\text{softmax}}{O} \rightarrow \hat{y}$

$e_c = E O_c$

Softmax: $$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$$

Θ_t : parameter associated with target t

Cost: $$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{10,000} y_i \log \hat{y}_i$$

Coursera: Deep learning Specialization, Andrew Ng

Negative sampling

I want a glass of orange juice to go along with my cereal.

x $\qquad\qquad$ y

context	word	target?
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0

k

c $\qquad$ t $\qquad$ y

Coursera: Deep learning Specialization, Andrew Ng

Binary classification

$$P(y=1 \mid c,t) = \sigma(\theta_t^T e_c)$$

Sigmoid function

Randomly pick up from dictionary

or

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{10,000} f(w_j)^{3/4}}$$

Word2vec

vocabulary_size=7 and *embedding_size=3*

$anarchism$	0.5	0.1	-0.1
$originated$	-0.5	0.3	0.9
as	0.3	-0.5	-0.3
a	0.7	0.2	-0.3
$term$	0.8	0.1	-0.1
of	0.4	-0.6	-0.1
$abuse$	0.7	0.1	-0.4

	Man	Woman	King	Queen	Apple	Orange
Gender	-1	1	-0.95	0.97	0	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97

http://adventuresinmachinelearning.com/word2vec-tutorial-tensorflow/

GloVe: Global Vectors for Word Representation

GloVe: Global Vectors for Word Representation, Jeffrey Pennington et al.,2014

I want a glass of orange juice to go along with my cereal.

X_{ij} = # times j appears in content of i

content target target content

Shallow Window-based methods (making predictions within local context windows)

GloVe: Model

GloVe: Global Vectors for Word Representation, Jeffrey Pennington et al.,2014

Loss function $$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

minimize $$\sum_{i=1}^{10,000}\sum_{j=1}^{10,000} f(X_{ij})(\theta_i^T e_j + b_i + b_j' - \log X_{ij})^2$$

1. f (0) = 0. If f is viewed as a continuous function, it should vanish as x ! 0 fast enough that the limx!0 f (x) log2 x is finite.
2. f (x) should be non-decreasing so that rare co-occurrences are not overweighted.
3. f (x) should be relatively small for large values of x, so that frequent co-occurrences are not overweighted.

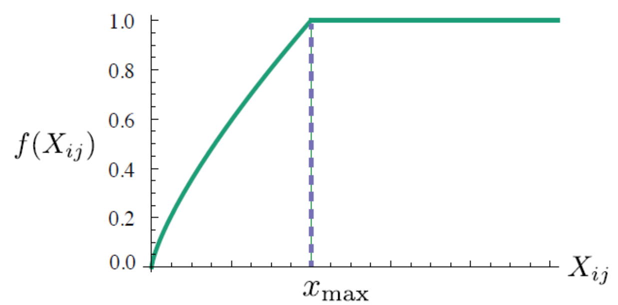


Figure 1: Weighting function f with $\alpha = 3/4$.

GloVe: Global Vectors for Word Representation

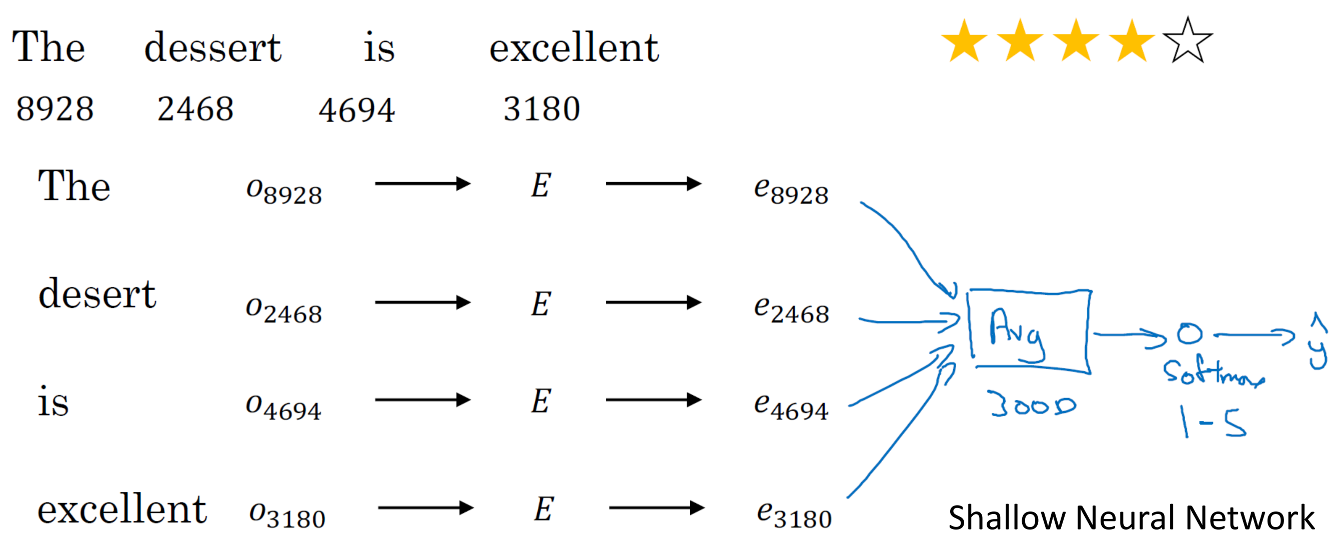
GloVe: Global Vectors for Word Representation, Jeffrey Pennington et al.,2014

- Unsupervised word representation
- Combined
  - Count-based method
  - Prediction-based method
- Outperforms
  - Word analogies
  - Word similarity
  - Named entity recognition

Sentiment classification problem

x	y
The dessert is excellent.	★★★★☆
Service was quite slow.	★★☆☆☆
Good for a quick meal, but nothing special.	★★★☆☆
Completely lacking in good taste, good service, and good ambience.	★☆☆☆☆

Simple sentiment classification model

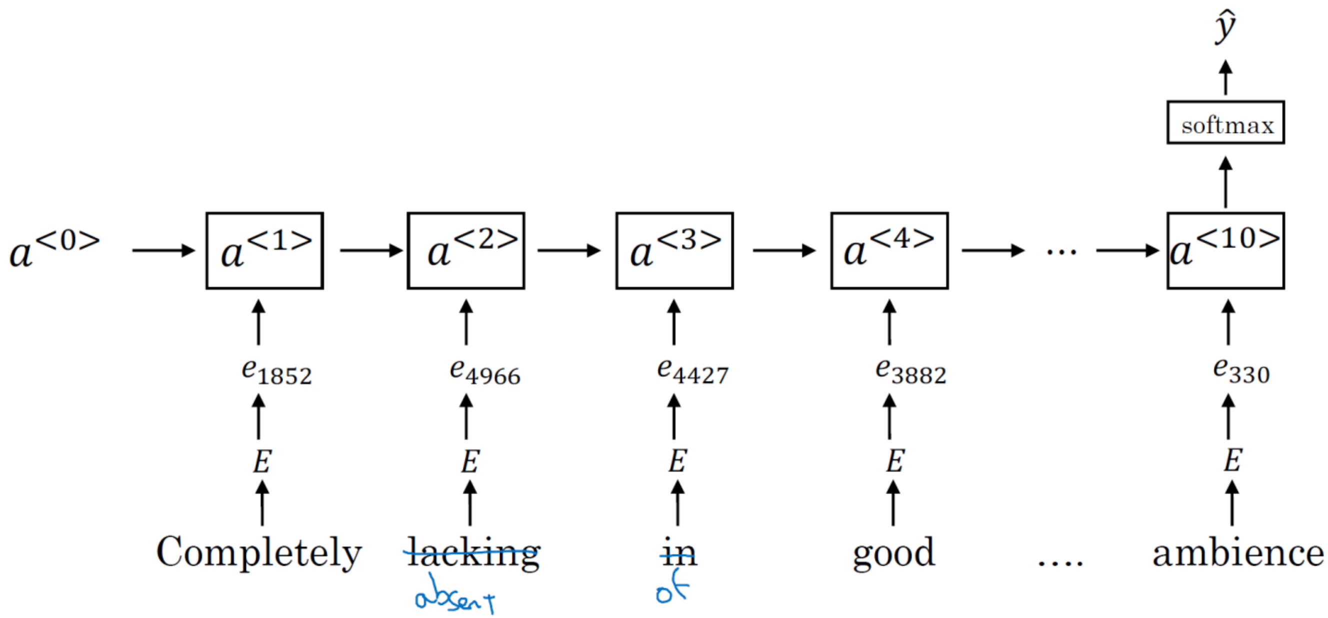


Problem in Simple sentiment classification model

Completely lacking in good taste, good service, and good ambience.

★☆☆☆☆

RNN for sentiment classification



Summary

- Character-based token → 1D convolution(word) + pooling(n-gram) → Neural Network → softmax
- Count vector or TF-IDF→ Neural Network → softmax
- Word vector (word2vec, glove) → Neural Network → softmax
- Word-embedding (word2vec, glove) → RNN