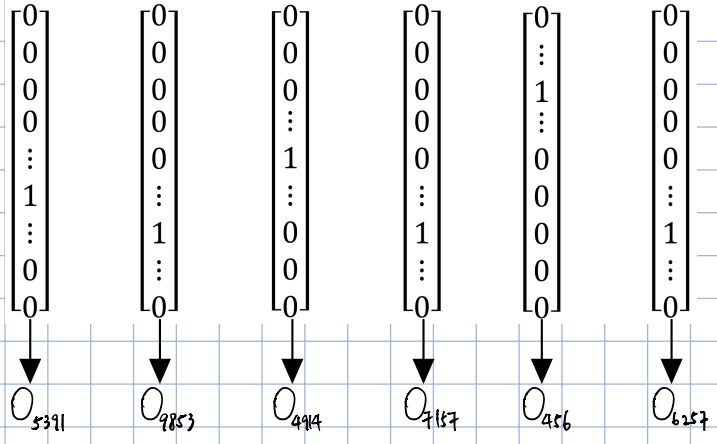
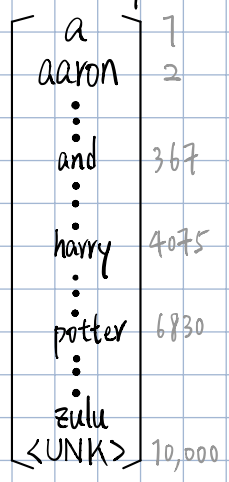
Word Representation

Given a Vocabulary V, we can use one-hot representation.

Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)



Vocabulary:



$\text{len(Vocabulary)} = 10^5$

The problem with one-hot representation is that it cannot represent the "distance" or "similarity" between words. Applying dot product or subtraction won't work.

Eg:

I want a glass of orange juice .

I want a glass of apple _____ .

If the answer to the first question is juice, then it's highly likely the second answer should also be juice because orange and apple are similar words.

Thus, a better representation is Featurized Representation: Word Embedding

Feature	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.0	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.0
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97
⋮						
	e_{5391}	e_{9853}			e_{456}	e_{6257}

features

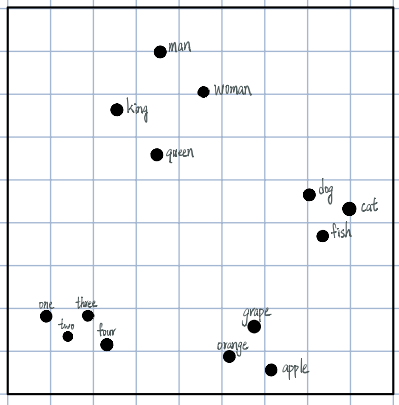
Eg: 300

With word embeddings, Apple and Orange are close.

Visualize Word Embedding

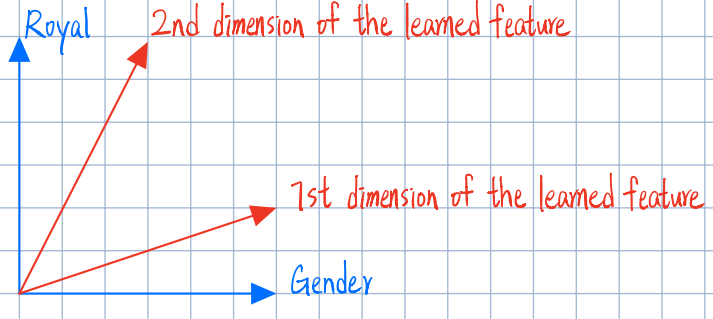
Read "Visualizing Data using t-SNE."

The word embedding vectors are 300 dimensional vectors. We can use t-SNE to project 300D into 2D plane.



The above example represents words by using meaningful features (Gender, Royal, Age, etc). However, features in word embeddings are automatically learned (learning word embedding will be discussed later). Thus, we can't really interpret meaningful information from automatically learned features.

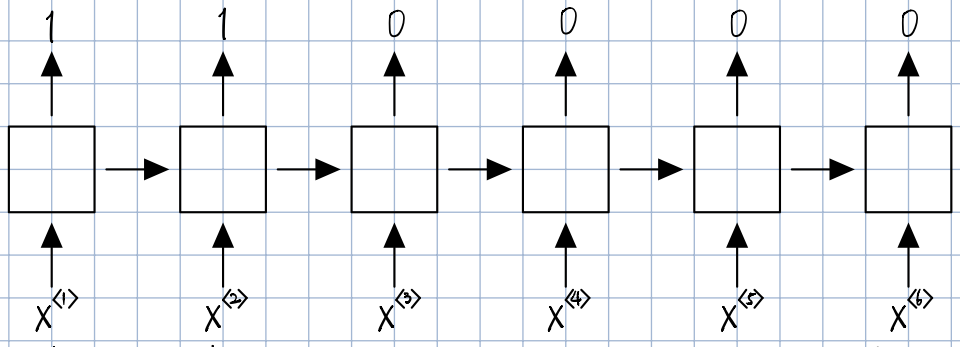
A note on the featurization view of word embeddings:



The learned features are not orthogonal.

Using Word Embeddings

Named Entity Recognition example



Sally	Johnson	is	an	orange	farmer.	→ Training data
Robert	Lin	is	an	apple	farmer.	→ Testing data
Amelie	Jordan	is	a	durian	cultivator.	→ Not in training data.

durian and cultivator are not seen in the labelled training set of Named Entity Recognition tasks, but it can still figure out Amelie and Jordan are names due to the words, durian and cultivator, are close to apple and farmer in the sense of word embeddings.

Word embeddings are learned from huge unlabelled corpus (1 billion to 100 billions) so durian and apple, and orange are grouped in high dimensional feature space.

Don't you see. It's Transfer Learning. Thus, I can do the following:

1. Learn word embeddings from large text corpus. (1 billion to 100 billions words).

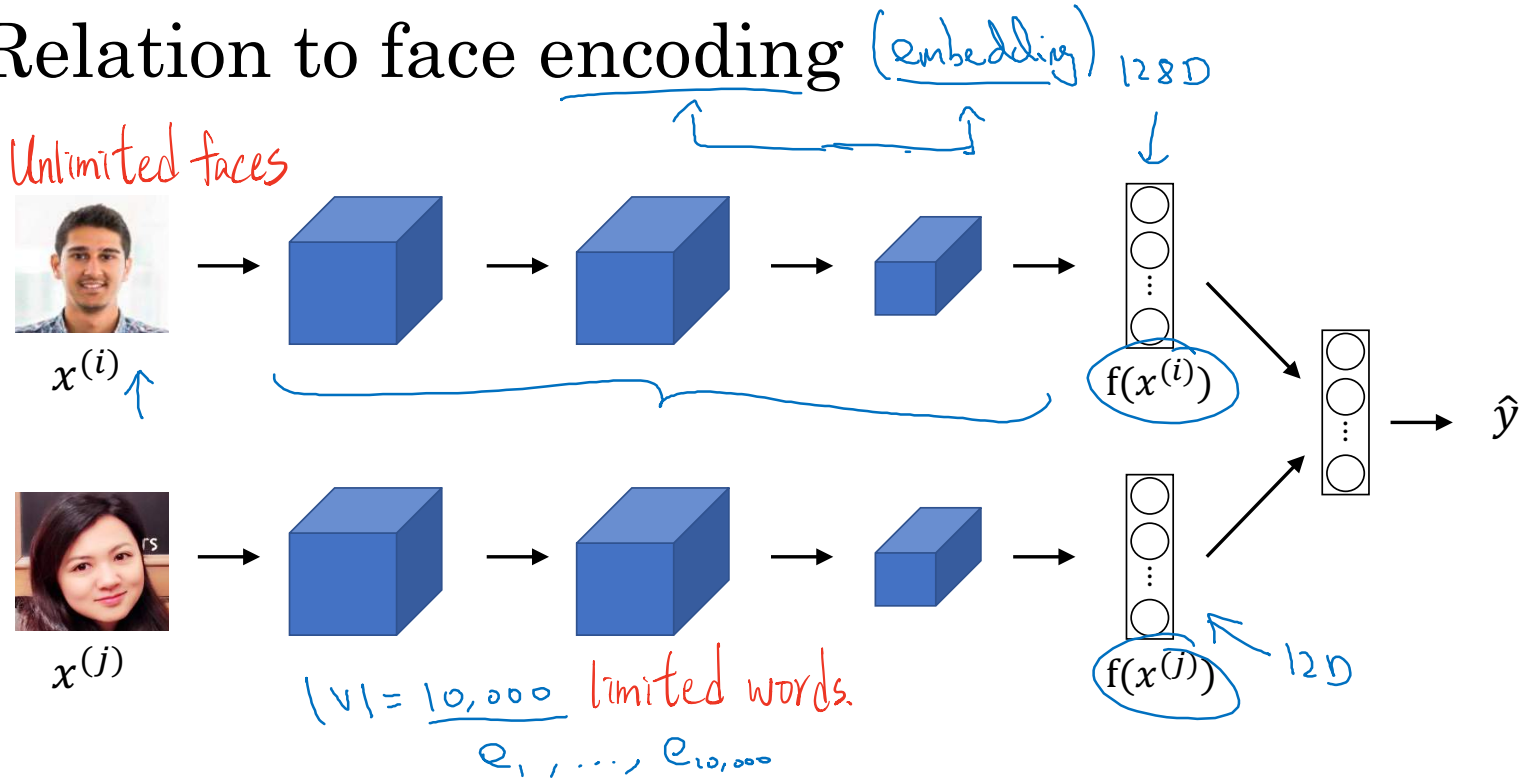
   Or download pre-traind word embeddings online.

2. Transfer word embeddings to new tasks (eg: Named Entity Recognition) with smaller labelled training data. (eg: 100k words)

3. Optional: Continue to finetune word embeddings with new data.

We can use dense 300D feature vectors, instead of sparse 10^5D one-hot vectors.

Relation to face encoding



[Taigman et. al., 2014. DeepFace: Closing the gap to human level performance]

Properties of Word Embeddings

Question: man is to woman as king is to ?

Analogies

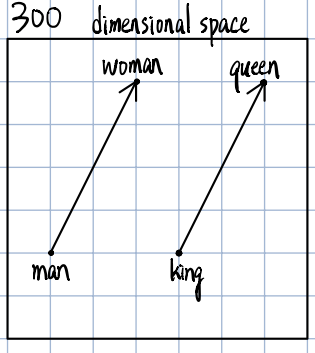
	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

e_{5391} / e_{man} e_{woman} e_{king} e_{queen}

$$e_{man} - e_{woman} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad e_{king} - e_{queen} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The solution is: first calculate $e_{man} - e_{woman}$, then find $e_?$ to make

$$e_{man} - e_{woman} \approx e_{king} - e_?$$



$$e_{man} - e_{woman} \approx e_{king} - e_w$$

Use similar vector to find the word w.

Find the word w : $\underset{w}{\text{argmax}}$ similarity$(e_w, e_{man} - e_{woman} + e_{king})$

Cosine Similarity

$sim(e_w, e_{man} - e_{woman} + e_{king})$ L2 norm can also be used.

$$sim(u, v) = \frac{u^T v}{\|u\|_2 \|v\|_2}$$

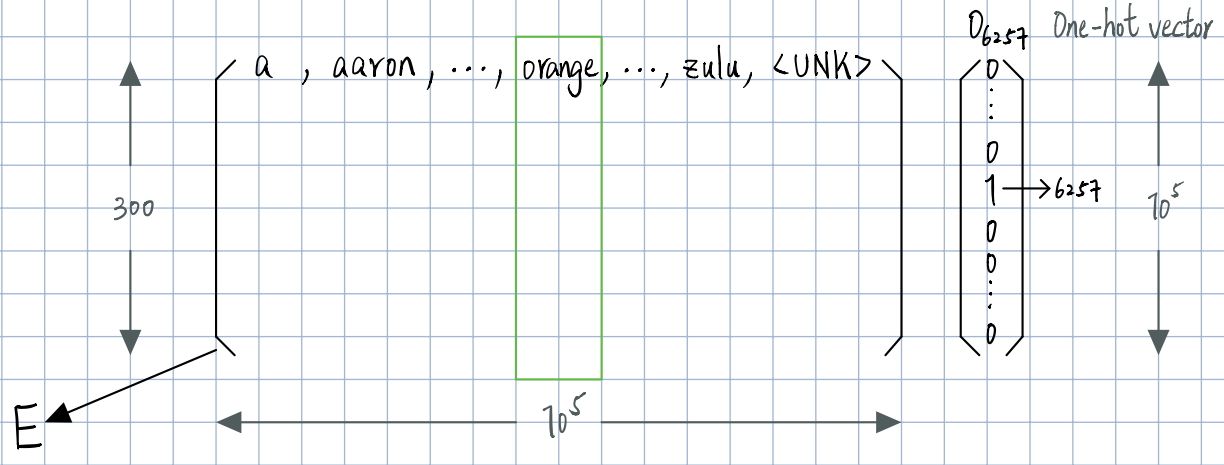
When $sim(u,v)$ is larger, then u and v are more similar.

Read 2013 "Linguistic regularities in continuous space word representation"

Embedding Matrix E

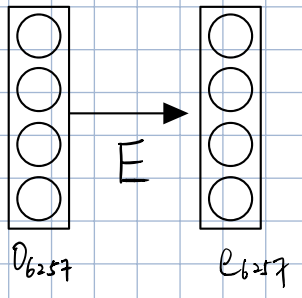
Vocabulary = a , aaron , ..., orange, ..., zulu, <UNK>. length = 10^5

6257



$$\underset{(300,10^5)}{E} \cdot \underset{(10^5,1)}{O_{6257}} = \underset{(300,1)}{\begin{bmatrix} \\ \\ \end{bmatrix}}$$

Embedded matrix E is just like a fully connected layer. Recall $a = wx + b$



Learning Word Embeddings

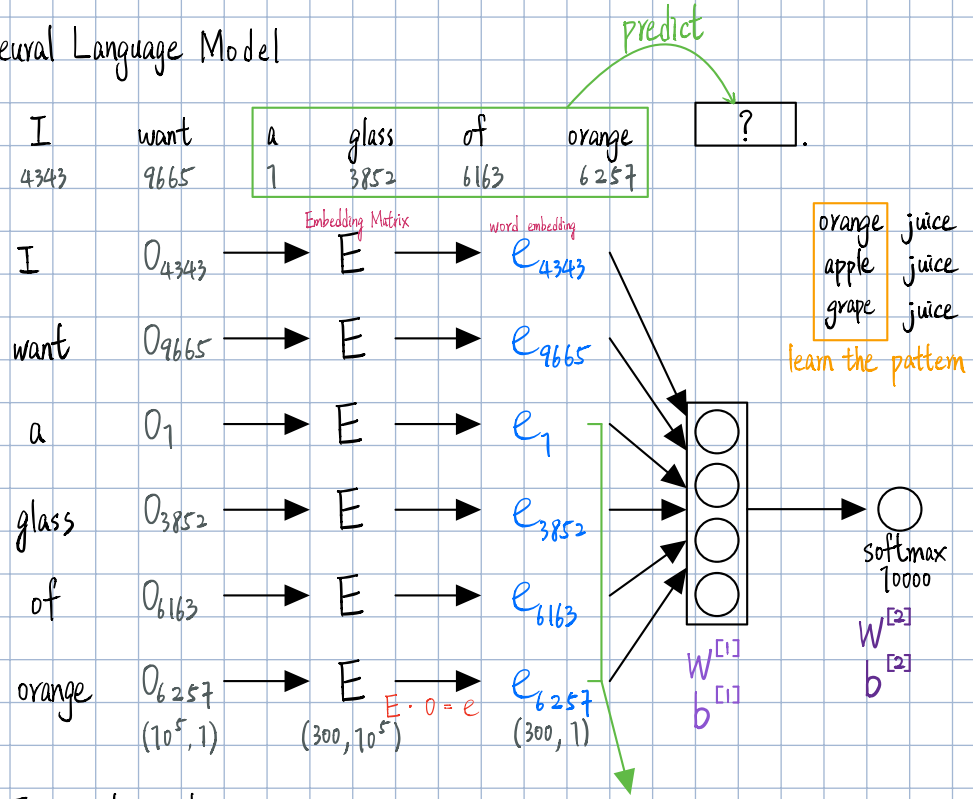
Read 2003 "A Neural Probablistic Language Model."

We've beening using word embeddings to solve various tasks, but we haven't talked about how to create word embeddings at the first place. Now, let's talk about it.

Vocabulary:

a	1
aaron	2
⋮	
and	367
⋮	
harry	4075
⋮	
potter	6830
⋮	
zulu	10,000

Neural Language Model



E is learned.

Input layer: In order to have a fixed size of an input layer, we can choose to use the fixed number of previous words. Eg, use 4. The input layer would be $(300 \times 4, 1)$.

Other context / target pairs.

Another example:

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

context: a glass of orange

target: juice

Context: Last 4 words.

4 words on left and right.

Last 1 word.

Nearby 1 word.

Word2Vec

Read 2013 "Efficient Estimation of Word Representations in Vector Space."

Skip-grams

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

Context	Target	
orange	juice	} randomly chosen from a fixed size window
orange	glass	
orange	go	

Model:

Vocabulary size = 10000

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

$O_c \longrightarrow E \longrightarrow e_c \longrightarrow \text{softmax} \longrightarrow \hat{y}$

$e_c = E \cdot O_c$ $\quad \theta_t^T e_c$

softmax: $p(t|c) = \dfrac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10^5} e^{\theta_j^T e_c}}$ (target, context)

θ_t: parameter associated with target output t (softmax)

θ_t and $e_c \in \mathbb{R}^{300}$

$\mathcal{L}(\hat{y}, y) = \sum_{i=1}^{10^5} y_i \log \hat{y}_i$ $\qquad y = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ $\rightarrow 4834$ $\quad 10^5$

Problems with softmax classification

$$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10^5} e^{\theta_j^T e_c}}$$

takes too much time to compute. Use Hierarchical Softmax

How to sample the context c?

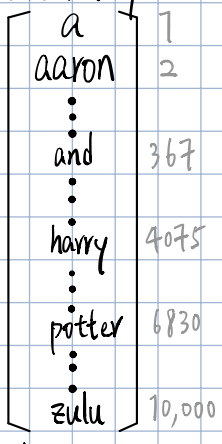
Note: It's OK if we do poorly on this artificial prediction task. The more important by-product of this task is that we learn word embeddings.

Negative Sampling

Previously, skip-gram model constructs a supervised learning task, mapping from context to target and learning word embeddings. But the downside was that the softmax was slow to compute.

Read "Distributed Representation of Words and Phrases and their Compsitionality"

Vocabulary:



Define a new learning problem.

Training sentence:

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

input X | output y

context	word	target?	
orange	juice	1	One positive sample from the training sentence.
orange	king	0	k negative samples sampling from Vocabulary.
orange	book	0	
orange	the	0	
orange	of	0	
c	t	y	

$k = 5 \sim 20$ smaller dataset

$k = 2 \sim 5$ large dataset

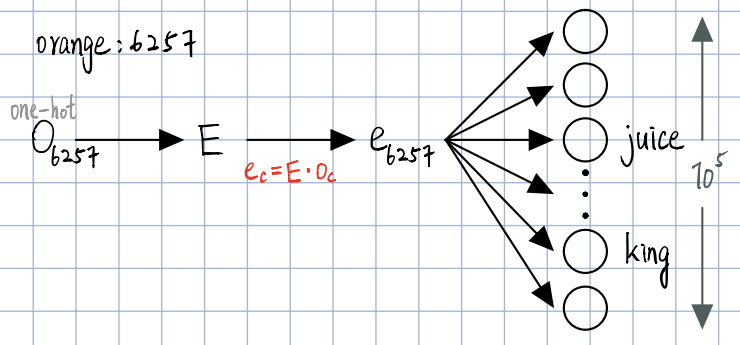
Model:

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

sigmoid

orange: 6257

O_{6257} (one-hot) $\rightarrow E \rightarrow e_{6257}$, $e_c = E \cdot O_c$



10^5 binary classification problems

Train k+1 samples on every iteration.

How to select negative samples?

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

Sampling averagely will get a lot of "the", "of", "and", etc, which is bad.

$$P(\text{freq}_i) = \frac{P(\text{freq}_i)^{\frac{3}{4}}}{\sum_{j=1}^{10^5} P(\text{freq}_j)^{\frac{3}{4}}}$$

GloVe Word Vectors

Read 2014 "GloVe: Global Vectors for Word Representation."

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

X_{ij} : the number of times that word j appears in the context of word i.

(X_{ij}: context, target; j: target; i: context)

Model:

$$\text{minimize} \sum_{i=1}^{10^5} \sum_{j=1}^{10^5} f(X_{ij}) \left(\theta_i^T e_j + b_i + b_j' - \log X_{ij} \right)^2$$

($\theta_i^T e_j$: context, target)

$\theta_t^T e_c$ We want to know how related are words i and j as measured by how often they occur with each other, which is affected by X_{ij}.

weighting term

$f(X_{ij}) = 0$ if $X_{ij} = 0$

θ_i^T and e_j are symmetric. $e_{word}^{(final)} = \frac{e_{word} + \theta_{word}}{2}$

θ_t and e_c should be initialized randomly at the beginning of training.

deeplearning.ai

NLP and Word Embeddings

Sentiment classification

Sentiment classification problem

x → y

The dessert is excellent. ★★★★☆

Service was quite slow. ★★☆☆☆

Good for a quick meal, but nothing special. ★★★☆☆

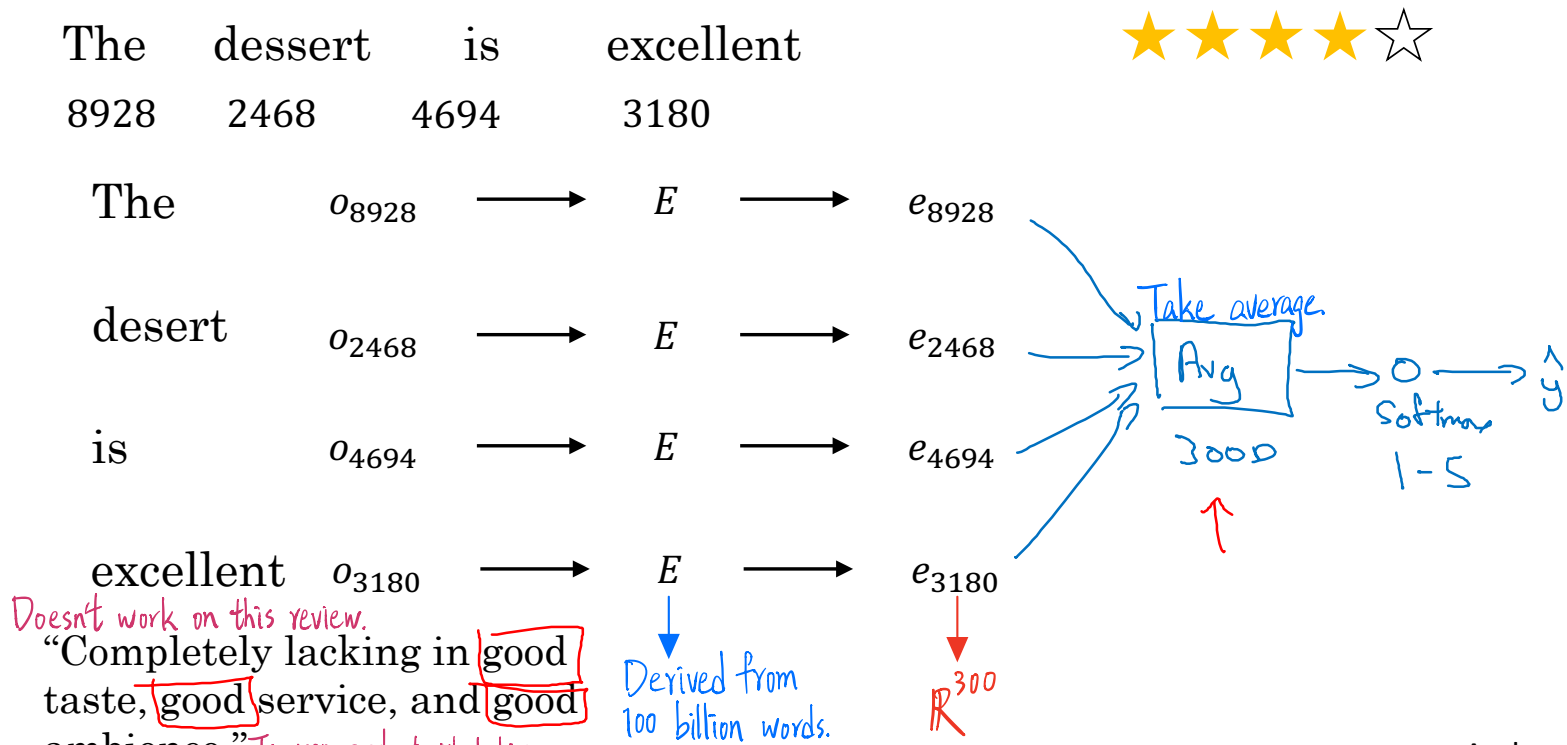
Completely lacking in good taste, good service, and good ambience. ★☆☆☆☆

10,000 → 100,000 words

Simple sentiment classification model

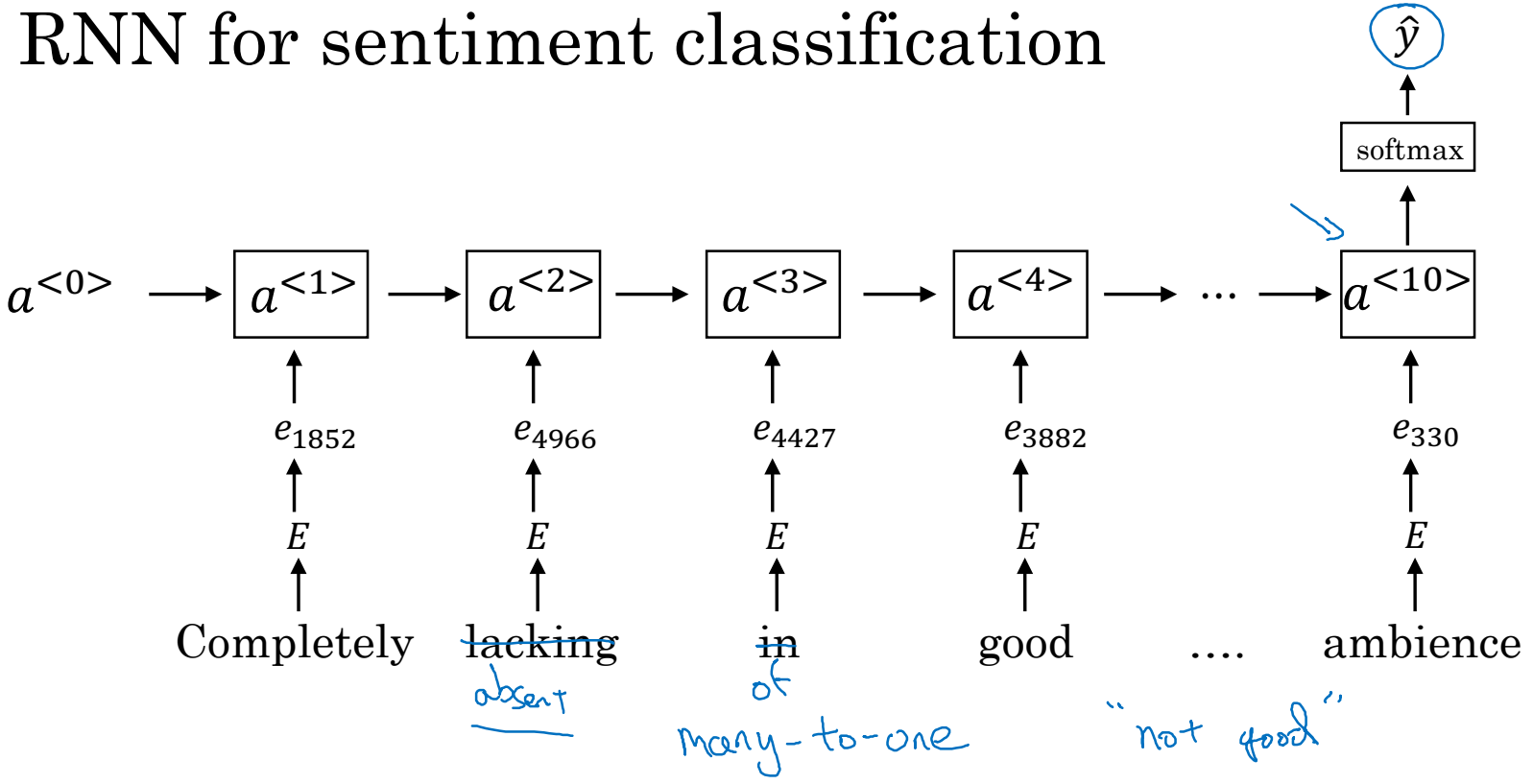
The dessert is excellent

8928 2468 4694 3180



Andrew Ng

RNN for sentiment classification



deeplearning.ai

NLP and Word Embeddings

Debiasing word embeddings

The problem of bias in word embeddings

Man:Woman as King:Queen

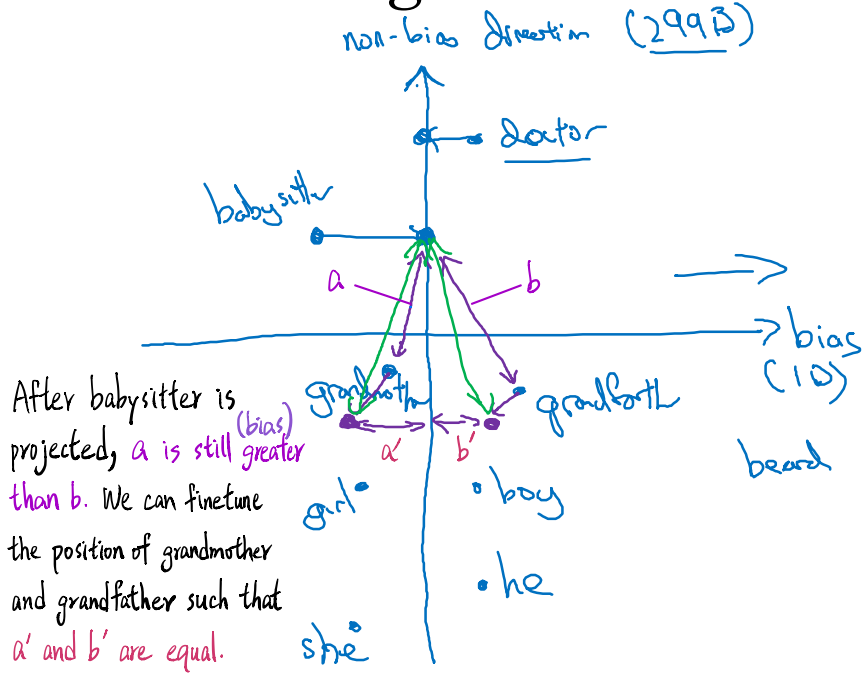
Man:Computer_Programmer as Woman:Homemaker

Father:Doctor as Mother:Nurse

Word embeddings can reflect gender, ethnicity, age, sexual orientation, and other biases of the text used to train the model.

[Bolukbasi et. al., 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings]

Addressing bias in word embeddings



1. Identify bias direction.

$e_{he} - e_{she}$

$e_{male} - e_{female}$

$\vdots$

average

2. Neutralize: For every word that is not definitional, project to get rid of bias.

Eg: doctor, babysitter, etc

3. Equalize pairs.

Make the distance equal so there's no bias.

grandmother − grandfather

girl boy

[Bolukbasi et. al., 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings]

Andrew Ng