deeplearning.ai

NLP and Word Embeddings

Word representation

Word representation

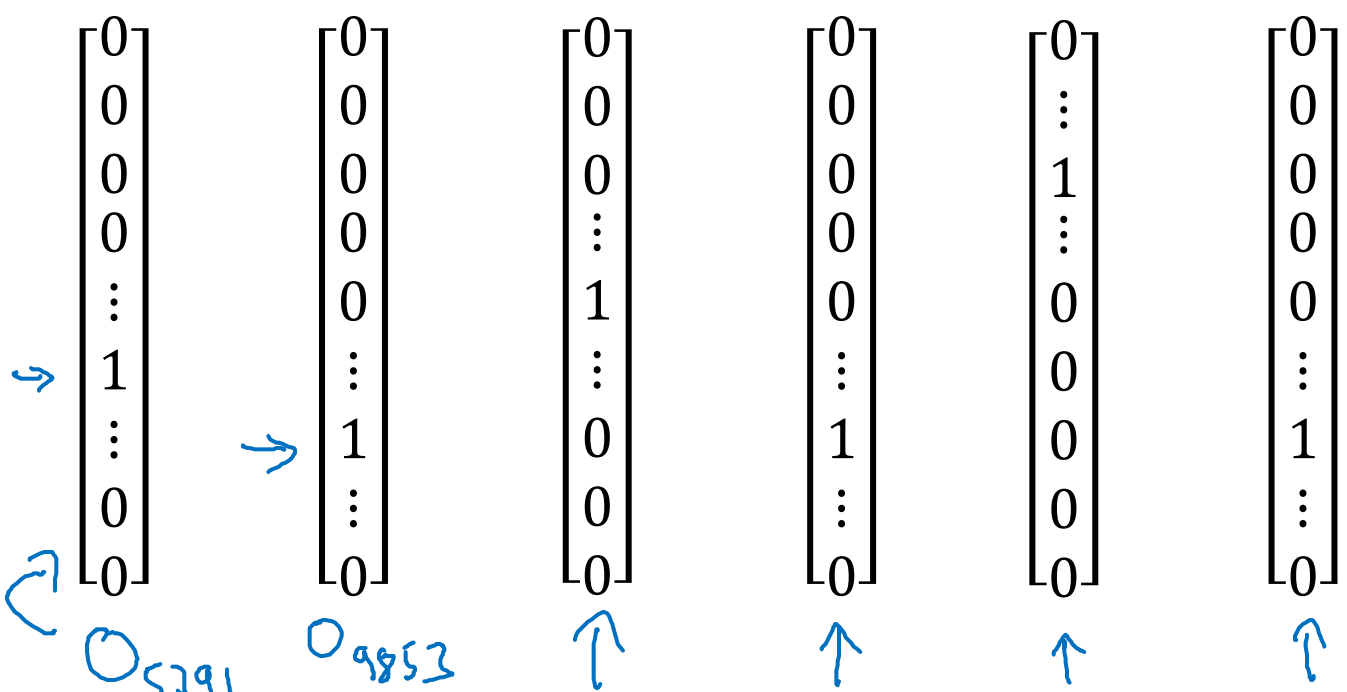
Limitation of 1-hot encoding of words: it doesn't encode similarity between words.

V = [a, aaron, …, zulu, <UNK>]

$|V| = 10{,}000$

1-hot representation

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



I want a glass of orange juice.

I want a glass of apple ?.

Featurized representation: word embedding

One possibility is to embed words in a feature space. This embedding requires a certain heuristic (a way to extract the feature value for each word).

	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	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97
⋮	⋮	⋮				
Size						
Cost						
Alive						
Verb						

300

e_{5391} embedding of word 5391

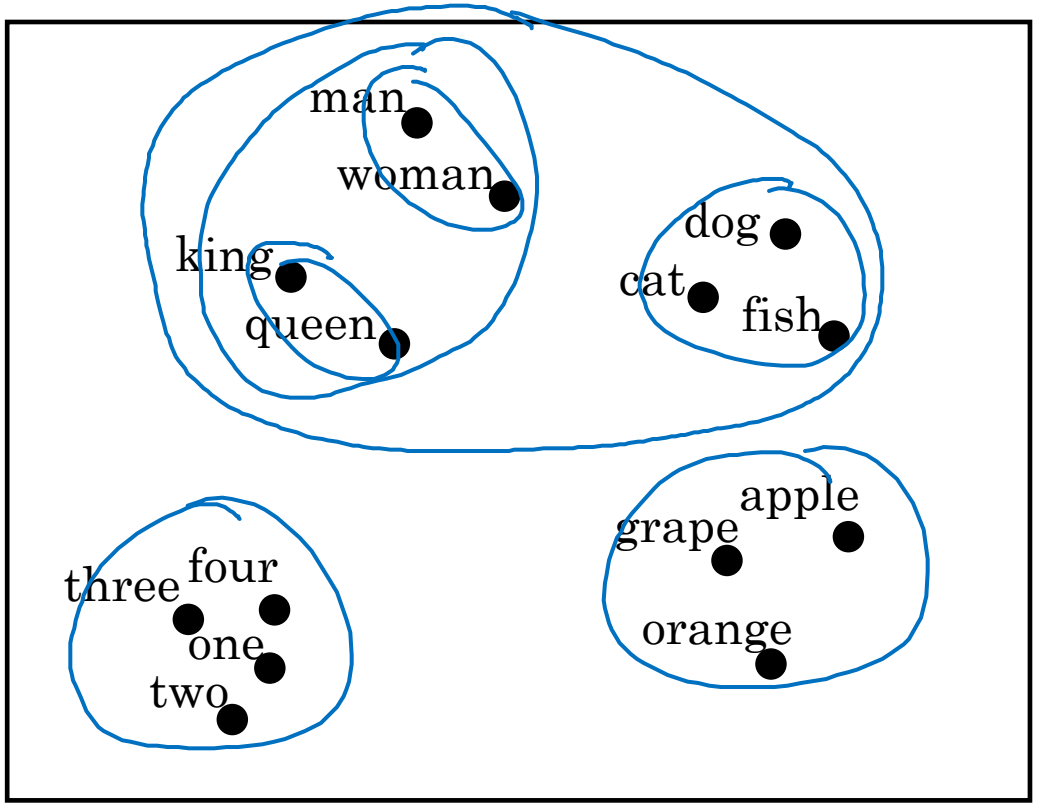
e_{9853}

I want a glass of orange juice.

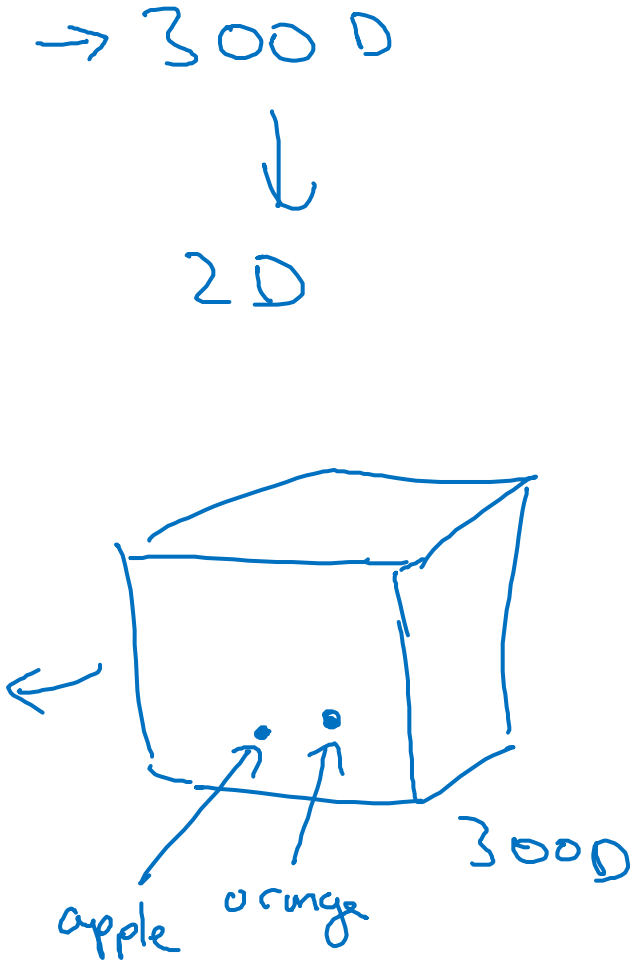
I want a glass of apple juice.

Andrew Ng

Visualizing word embeddings



t-SNE



[van der Maaten and Hinton., 2008. Visualizing data using t-SNE]

Using a word embedding in a task B implicitly corresponds to performing transfer learning from the task A of learning the embedding (normally done on very large text corpora, 1B to 100B words) to task B (where normally we have much less data).

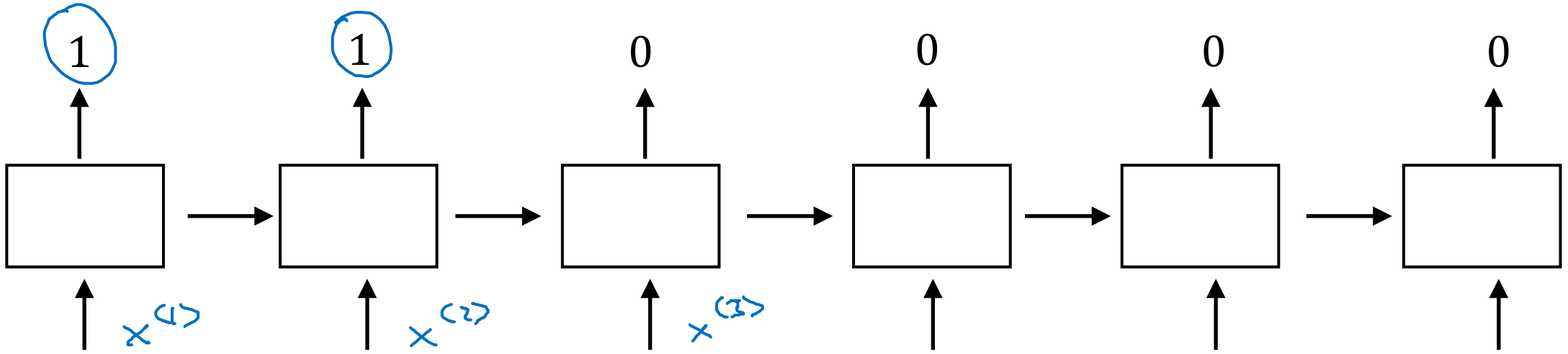
NLP and Word Embeddings

deeplearning.ai

In other words, By simply encoding the words in input with a certain word embedding for a certain task, we are transferring knowledge to that task.

Using word embeddings

Named entity recognition example



Robert Lin is ~~an~~ ~~apple~~ ~~farmer~~

durian cultivator.

→ 1B words – 100B words

→ 100k words

a BRNN

Transfer learning and word embeddings

1. Learn word embeddings from large text corpus. (1-100B words)

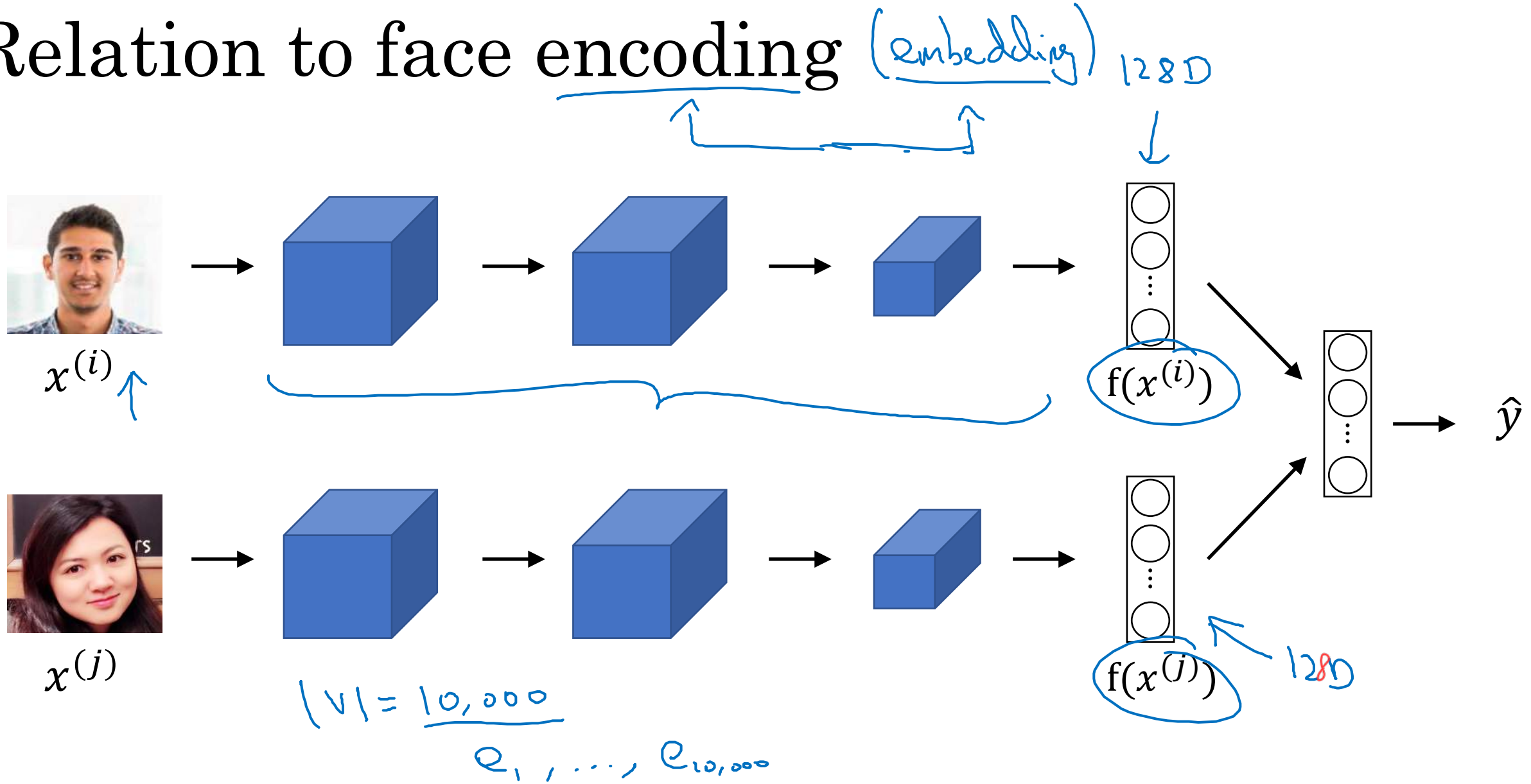
   (Or download pre-trained embedding online.)

2. Transfer embedding to new task with smaller training set. (say, 100k words)

   → 10,000 → 300

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

Relation to face encoding



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

One property of word embedding is that they can learn ANALOGY relationships. That is, in the embedding space, any 2 couples of words (A-A') and (B-B') that are related by an analogy (men-woman) (king-queen) have similar distance from each other.

deeplearning.ai

NLP and Word Embeddings

Properties of word embeddings

$$d(\text{men}, \text{woman}) \approx d(\text{king}, \text{queen})$$

$$\begin{bmatrix} 0.1 \\ 0 \\ 0.3 \\ \vdots \end{bmatrix} \qquad \begin{bmatrix} 0.11 \\ 0 \\ 0.29 \\ \vdots \end{bmatrix}$$

Often this holds even for linear similarity

$$\text{men} - \text{woman} \approx \text{king} - \text{queen}$$

which allows to solve analogy problems

like $\text{men} : \text{woman} = \text{king} : x \rightarrow \text{men} - \text{woman} = \text{king} - x \rightarrow x = \underset{x}{\text{argmin}} \left(x - \text{king} + (\text{men} - \text{woman}) \right)$

$$\rightarrow \underset{x}{\text{argmax}}\ d\left(x, \text{king} - (\text{men} - \text{woman})\right).$$

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}

Man → Woman as King → ? Queen

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

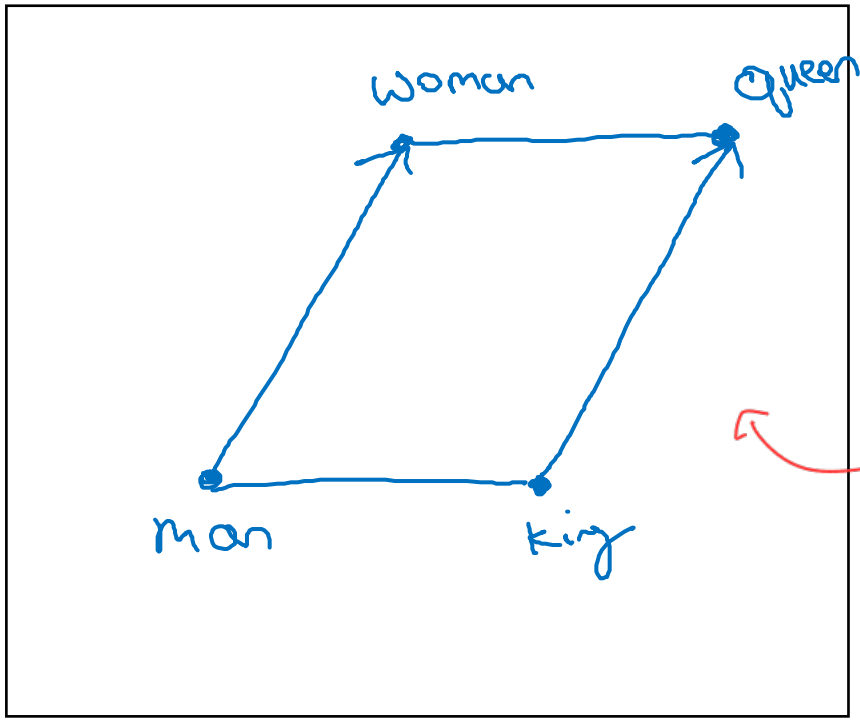
$e_{?} \approx e_{queen}$

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

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

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

Analogies using word vectors

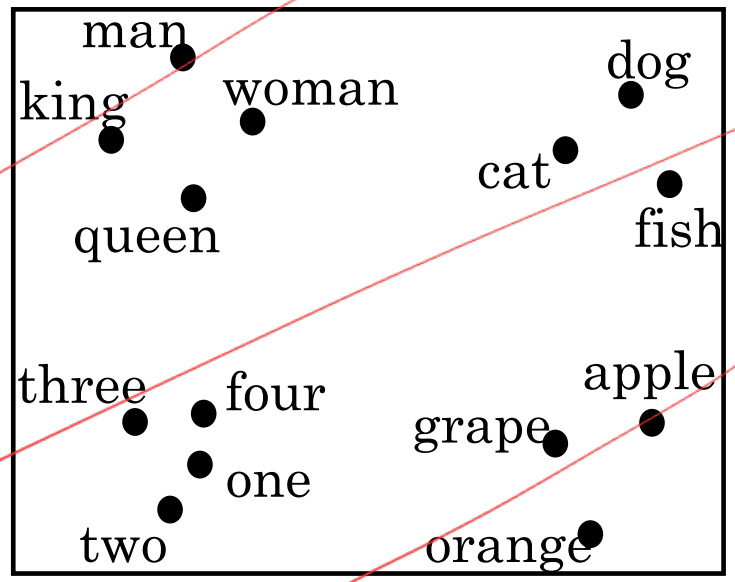


300D

Find word w: $\arg\max_w$ $\text{sim}(e_w, e_{king} - e_{man} + e_{woman})$

30-75%

3000 → 2D



t-SNE

Analogy relationship hold in the original embedding space.

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

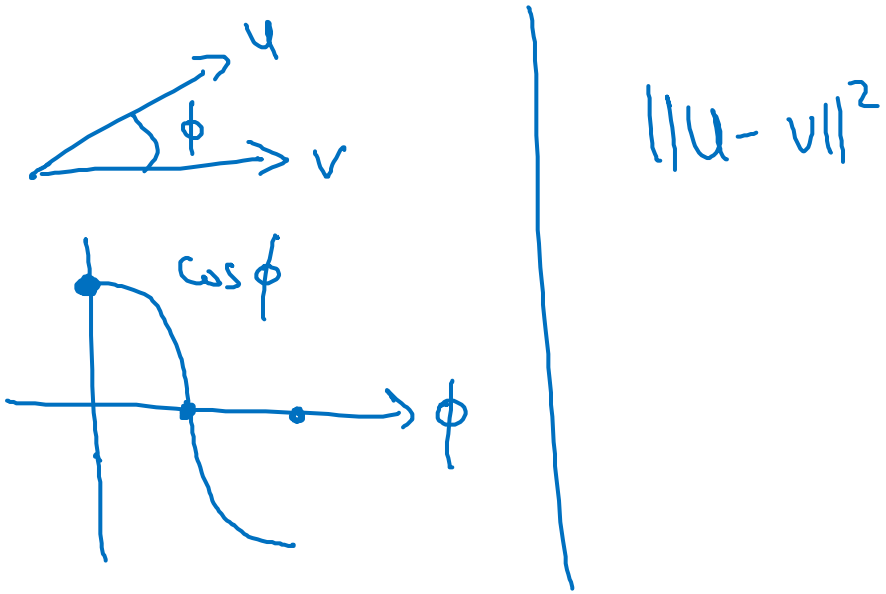
Andrew Ng

Cosine similarity

← most often used in NLP.

$$sim(e_w, e_{king} - e_{man} + e_{woman})$$

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



$\|u - v\|^2$

Man:Woman as Boy:Girl

Ottawa:Canada as Nairobi:Kenya

Big:Bigger as Tall:Taller

Yen:Japan as Ruble:Russia

NLP and Word Embeddings

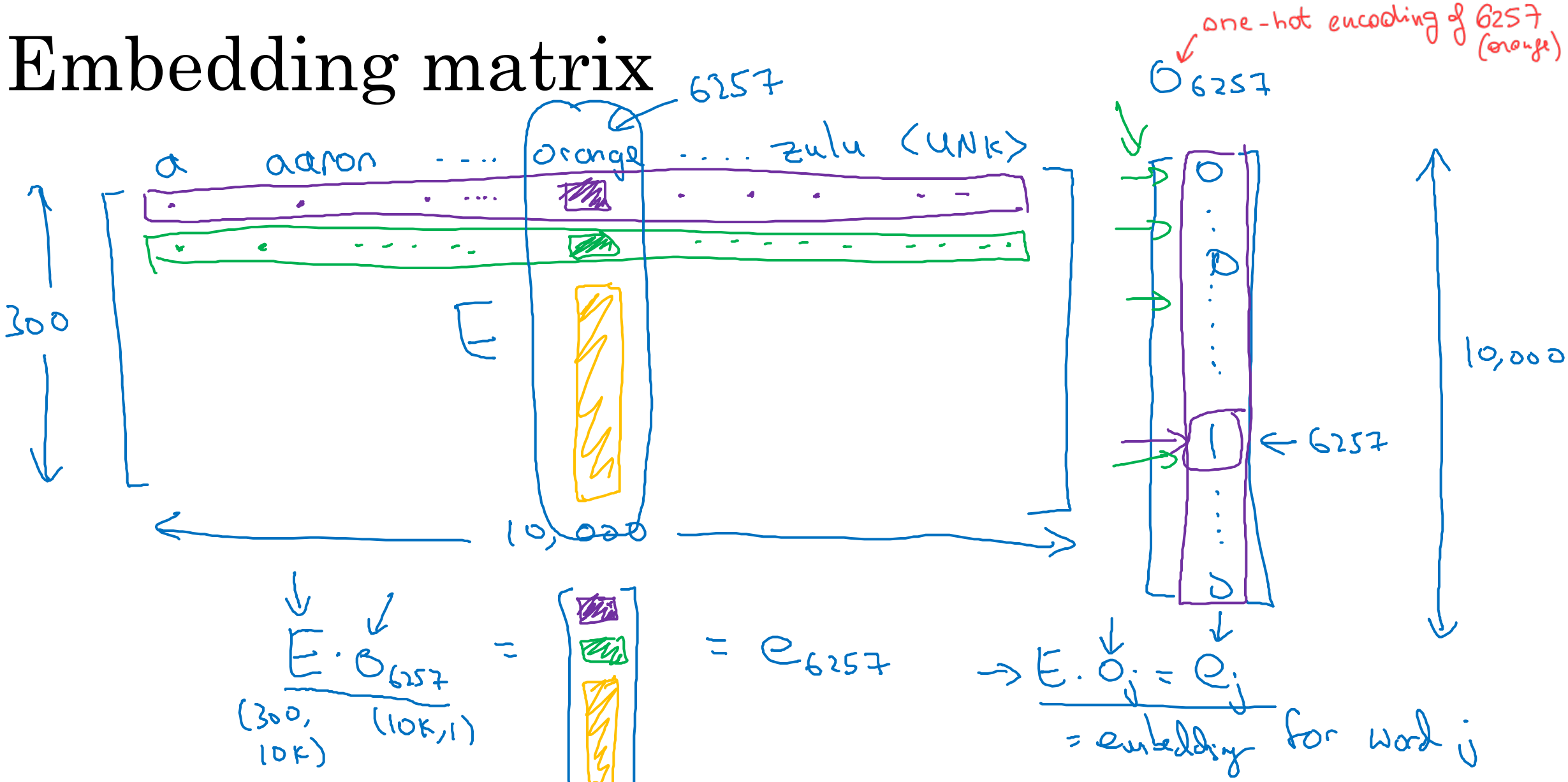
deeplearning.ai

Embedding matrix

Given a vocabulary of size 10000, an embedding of that vocabulary is represented as an embedding matrix E. If the embedding is 300-dim, then $E_{300 \times 10000}$ where each column i of E is the embedding of the word i.

Calling $\vec{e}_i$ and $\vec{O}_i$ the embedding and the one-hot encoding of the word i, we have that $\vec{e}_i = E \cdot \vec{O}_i$

Embedding matrix



In practice, use specialized function to look up an embedding.

Andrew Ng

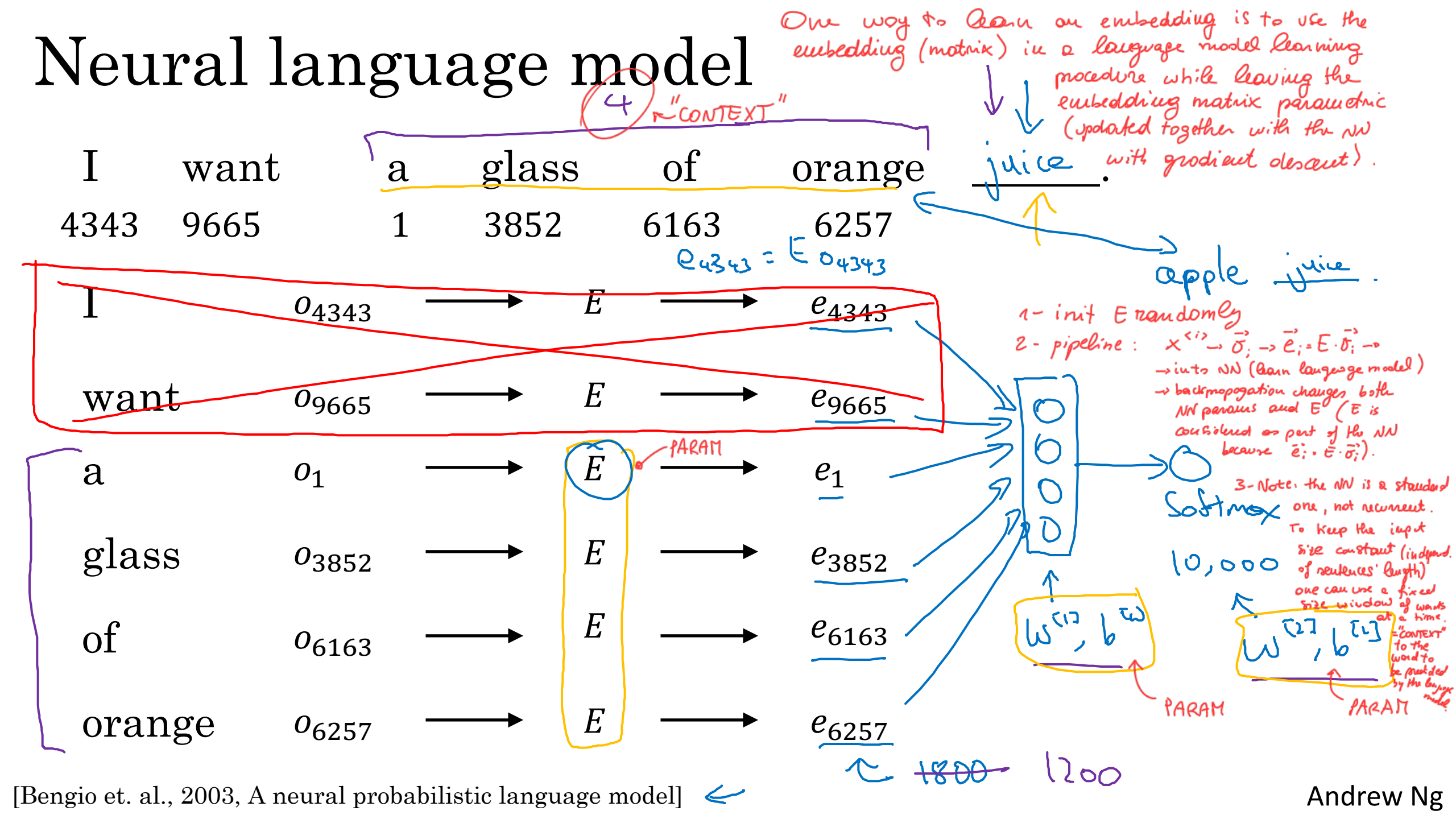
Now very efficient and simple ways exist to learn a word embedding. Ng goes through the history of these methods from old and inefficient to more recent ones.

NLP and Word Embeddings

deeplearning.ai

Learning word embeddings

Neural language model



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

Andrew Ng

Other context/target pairs

With the AIM OF LEARNING A WORD EMBEDDING (and not necessarily a good language model) other contexts can be used.

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

context

target

Context: Last 4 words.

- 4 words on left & right — a glass of orage ? to go alg with
- Last 1 word — orange ?
- Nearby 1 word (not necessarily neighboring). — glass ?

skip gram

An efficient word embedding, called WORD2VEC, is a word embedding built using a skip-gram model (another version is built with the continuous bag of words model but is not discussed here).

To learn a word2vec embedding based on the skip-gram model:

deeplearning.ai

NLP and Word Embeddings

Word2Vec

1- from the dataset sample pairs of words (CONTEXT, TARGET) such that TARGET is a word of the dataset and context is another word randomly sampled within n words before or after target. This sampling can use some heuristics to avoid sampling over frequent and not informative words as context (such as articles).

2- encode the pairs using the encoding E initialized at random (encoding matrix E initialized at random) whose entries will be updated by keeping E promotric in a skip-gram learning problem.

3- set a simple NN architecture to learn the skip-gram model context → target. This architecture only consists of a softmax node preceded by E.

$$c \rightarrow o_c \rightarrow e_c = E o_c \rightarrow y = \text{SOFTMAX}(\theta \cdot e_c) \text{ where } \theta_{\text{VOCABSIZE} \times \text{ENCODESIZE}} \rightarrow t = \underset{t}{\text{argmax}}(y)$$

CONTINUE

Skip-grams

h - In this model E and the softmax params are learned using the negative cross entropy loss

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

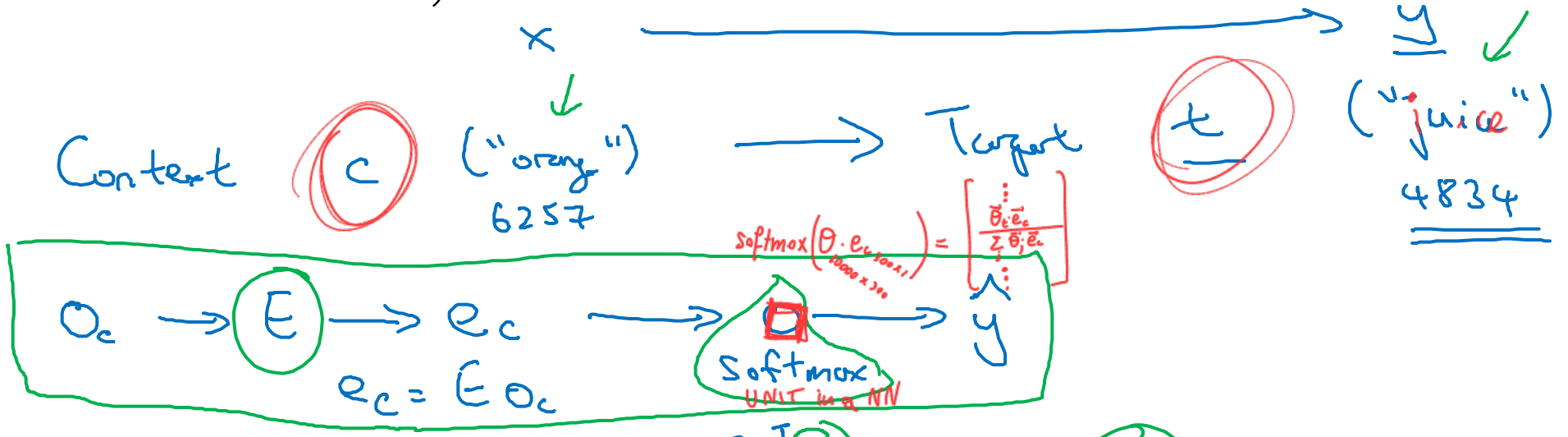
Context	Target
orange	juice
orange	glass
orange	my

NOTE: Softmax classification has the problem of requiring VOCABSIZE operations for each word (denom. of softmax) which doesn't scale well with the VOCABSIZE.

[Mikolov et. al., 2013. Efficient estimation of word representations in vector space.]

Model

Vocab size = 10,000k



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

θ_t = parameter associated with output t

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

$y = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow 4834$

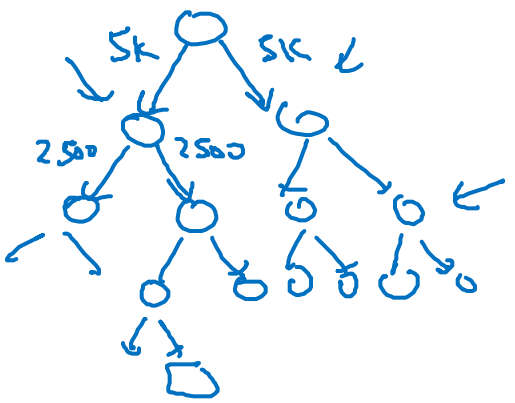
Andrew Ng

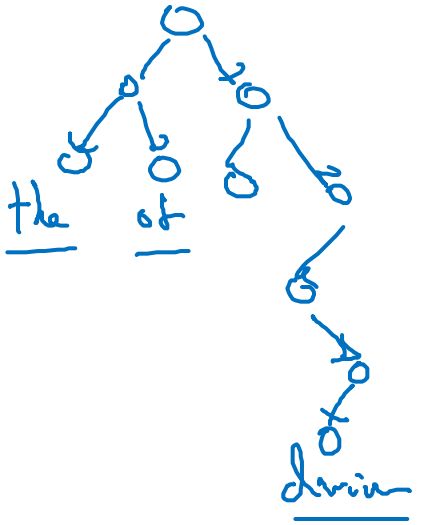
Problems with softmax classification

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

Hierarchical softmax.

$\log |V|$





How to sample the context c?

t

$c \to t$

avoid too frequent words → the, of, a, and, to, ...

→ orange, apple, durian

$P(c)$

$\frac{1}{2}$ durian

NLP and Word Embeddings

Negative sampling

Negative sampling is an alternative way to learn the skip-gram model that is more efficient than softmax classification. In softmax classification one wants to learn a mapping context -> target and to do that one needs to update the weights for all the words in the vocabulary. In the negative sampling this is avoided by rephrasing the problem as a binary classification one.

deeplearning.ai

1 - for each word in the corpus
create 1 POSITIVE EXAMPLE
<(word, context), 1>
K NEGATIVE EXAMPLES
<(word, wrong context), 0) obtained sampling random context words from the entire corpus.

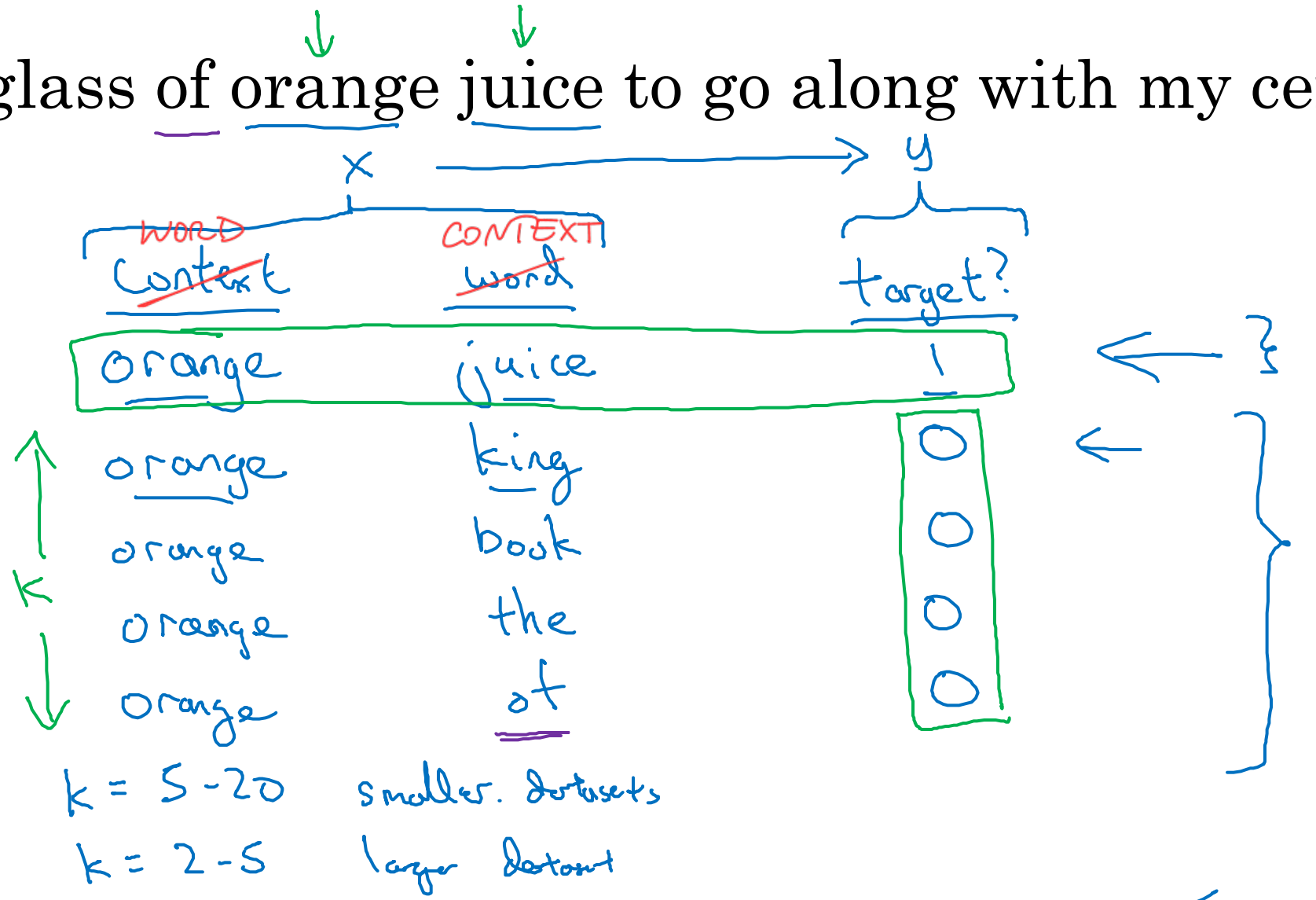
- K ~ 5 for large corpora and ~20 for small corpora.
- The sampling of random context words is made so to pick very frequent words with a lower probability.

---

2 - Instead of learning the usual skip-gram P(word|context) with softmax, we learn an approximated version of it P(1|word, context) with logistic regression. This allows to learn the word2vec more efficiently because for each sample you only update the weight between word and context (as opposed to updating the weights between a context and all possible words). It's unclear to me if you can then use the logistic model from negative sampling as a skip-gram. Anyway people use it to learn the word2vec encoding mainly.

Defining a new learning problem

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



[Mikolov et. al., 2013. Distributed representation of words and phrases and their compositionality]

Model

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

LOGISTIC REGRESSION PROBLEM

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

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

x y word context k c t y

Orange 6257

$o_{6257} \rightarrow E \rightarrow e_{6257}$

juice? king 10,000

10,000 binary classification problem

k+1

Selecting negative examples

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

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

the, of, and, ...

$\frac{1}{|V|}$

→ Empirically it's best to sample negative examples slightly inversely proportional to their frequency in the vocabulary.

Unclear to me... (esp. what are b_i and b_j ?)

deeplearning.ai

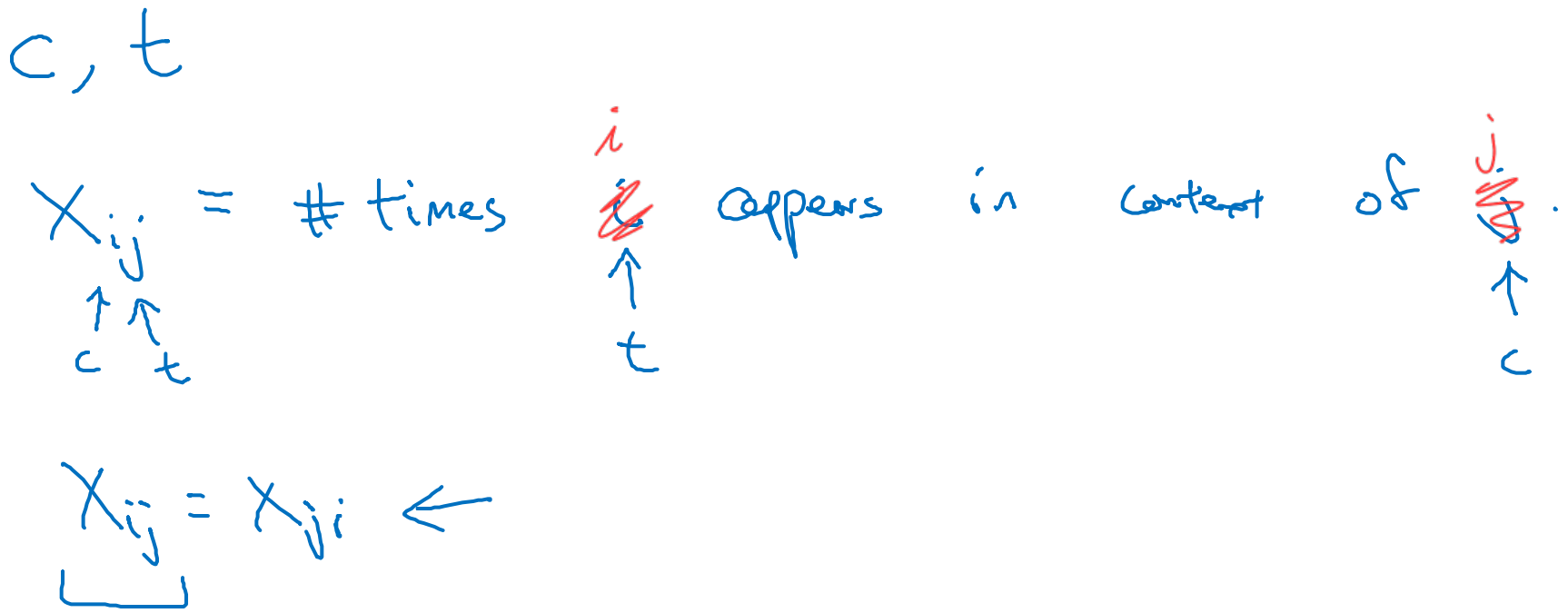
NLP and Word Embeddings

GloVe word vectors

- Computes the embedding of a word based on its global co-occurence with all the other words in the corpus (within a limited window).
- loss s.t. similarity between 2 embeddings proportional to co-occurence of those 2 words.
- Difference with word2vec: to my understanding is the procedure. While in word2vec we learn embedding through the task of learning context -> target, here we explicitly count cooccurences...

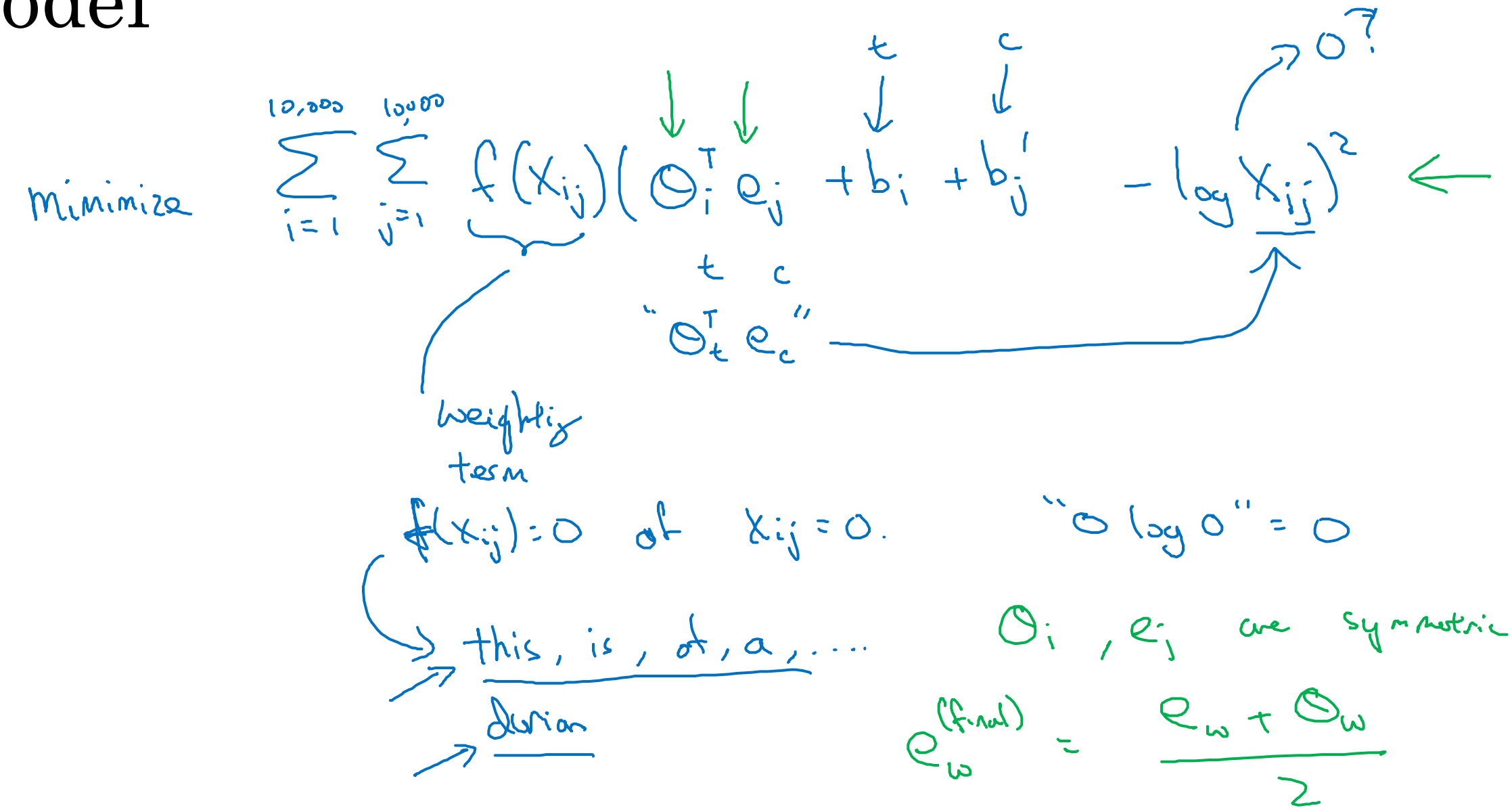
GloVe (global vectors for word representation)

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



[Pennington et. al., 2014. GloVe: Global vectors for word representation]

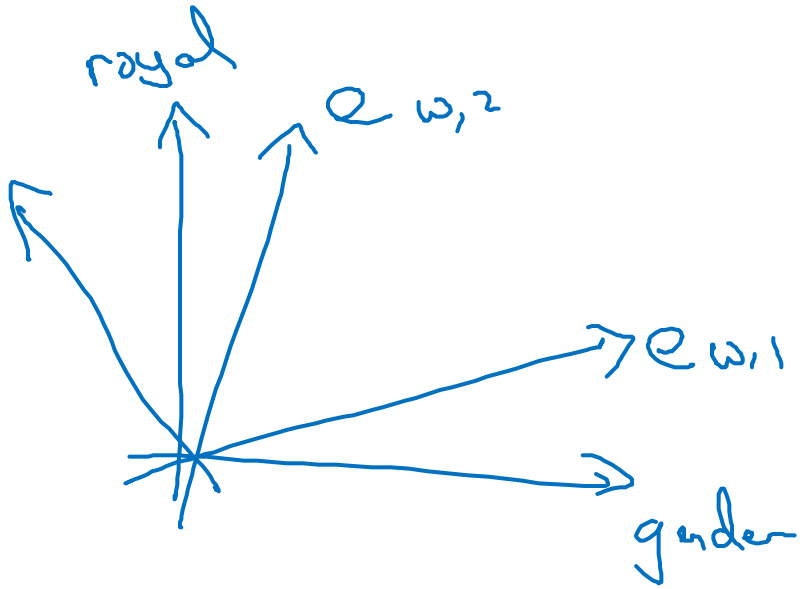
Model



Andrew Ng

A note on the featurization view of word embeddings

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



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

$\hookrightarrow (A\theta_i)^T (A^{-T} e_j) = \theta_i^T A^T A^{-T} e_j$

Andrew Ng

In general, use RNN for sentiment classification (input = embedded words)

NLP and Word Embeddings

deeplearning.ai

Sentiment classification

Sentiment classification problem

x → y

The dessert is excellent. ★★★★☆

Service was quite slow. ★★☆☆☆

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

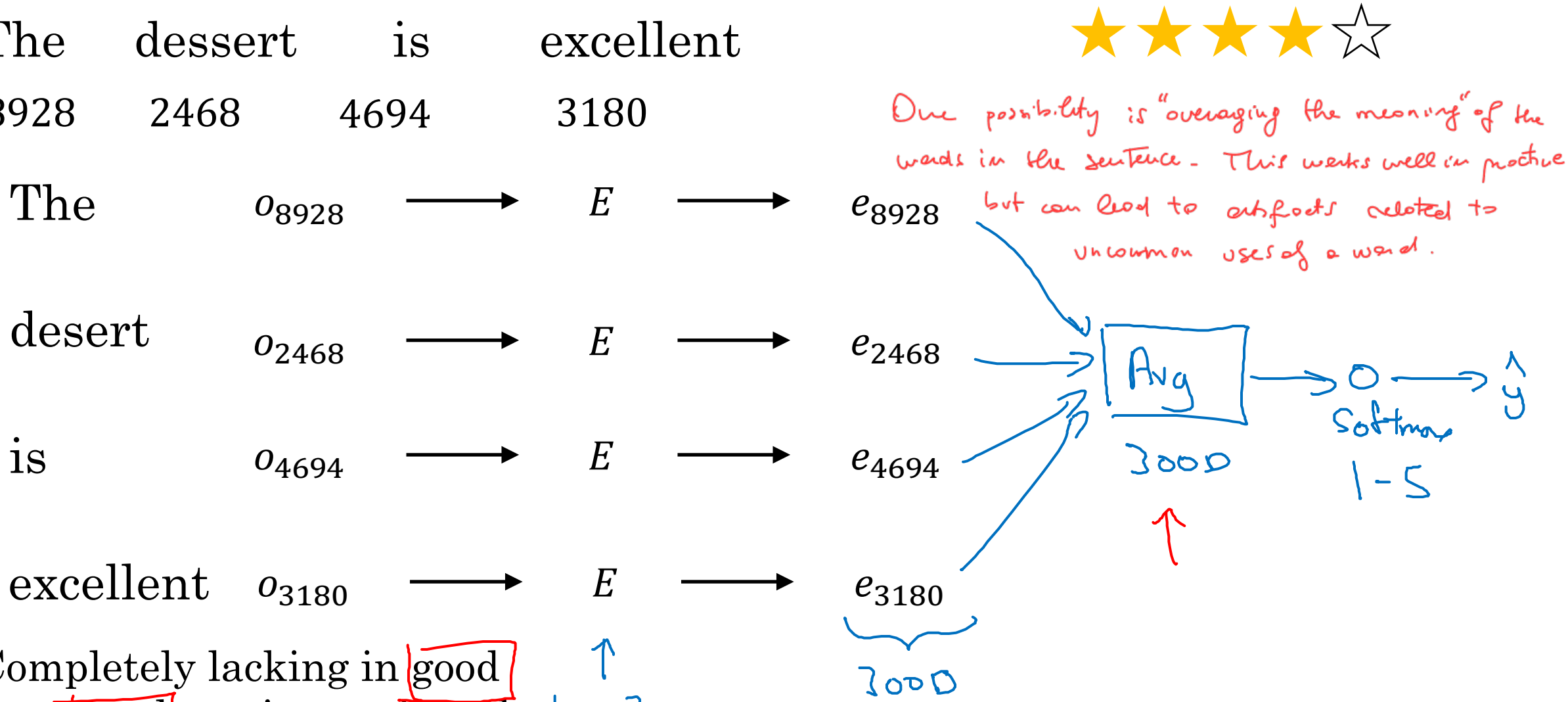
Completely lacking in good taste, good service, and good ambience. ★☆☆☆☆

10,000 → 100,000 words

Simple sentiment classification model

The dessert is excellent

8928 2468 4694 3180

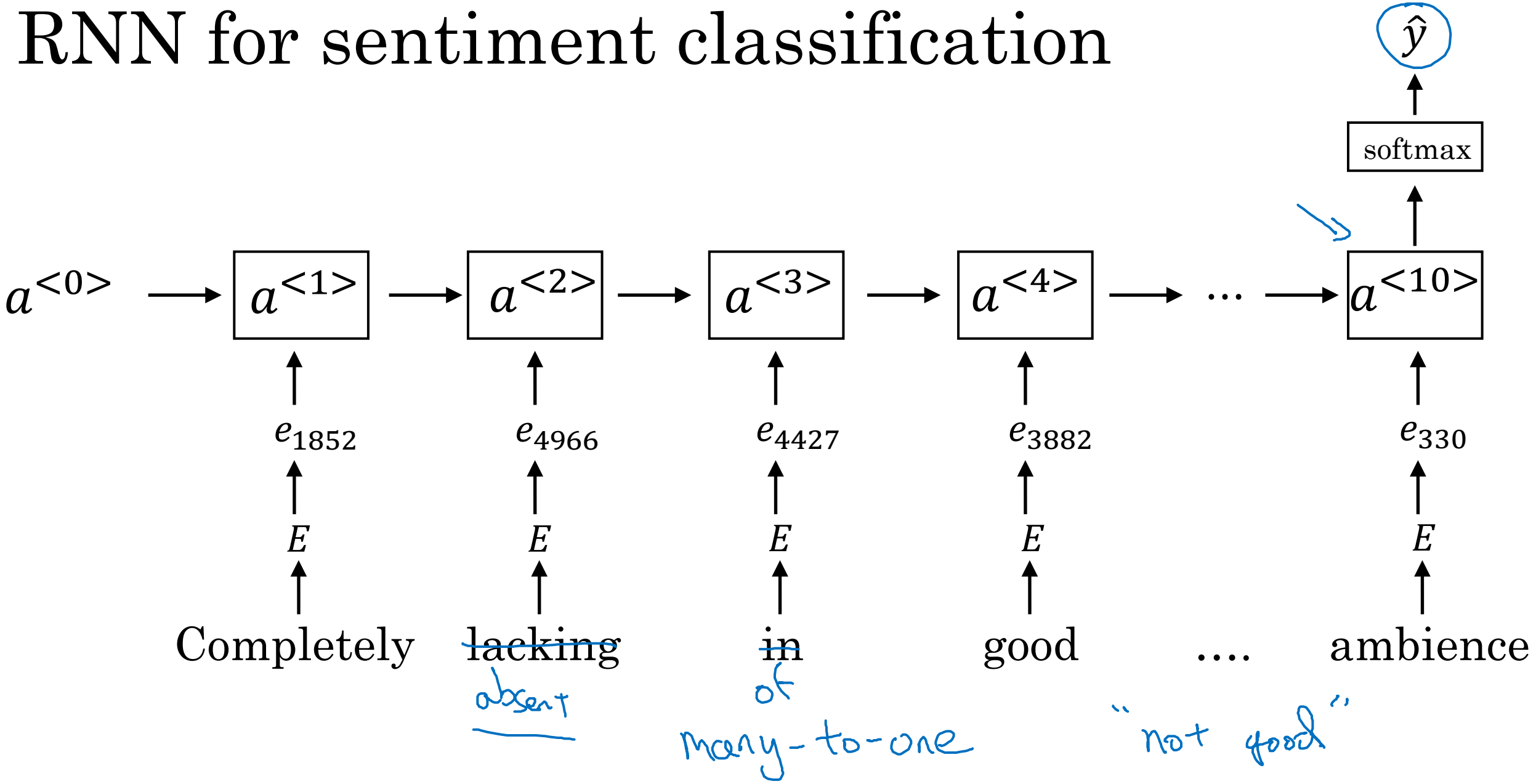


One possibility is "averaging the meaning" of the words in the sentence. This works well in practice but can lead to outputs related to uncommon uses of a word.

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

Andrew Ng

RNN for sentiment classification



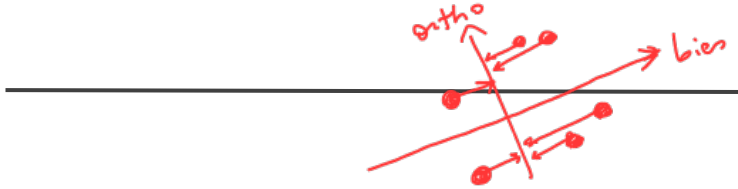
To remove bias from word encoding one pipeline could be:

1 - identify the bias direction in the embedding space by averaging the direction of pairs that are known to have that bias. e.g. average directions of $\overrightarrow{boy - girl}$, $\overrightarrow{man - woman}$, ... to find the direction of "gender" bias.

NLP and Word Embeddings

deeplearning.ai

2 - compress the embedding along the bias direction (often you cannot just eliminate it because the bias could be definitional for some words, i.e. employed in the definition of some words).



3 - for words for which the bias is definitional (e.g. man, woman, boy...) make them equally distant from bias-neutral words by operating a transformation of the space.

Debiasing word embeddings

The problem of bias in word embeddings

Man:Woman as King:Queen

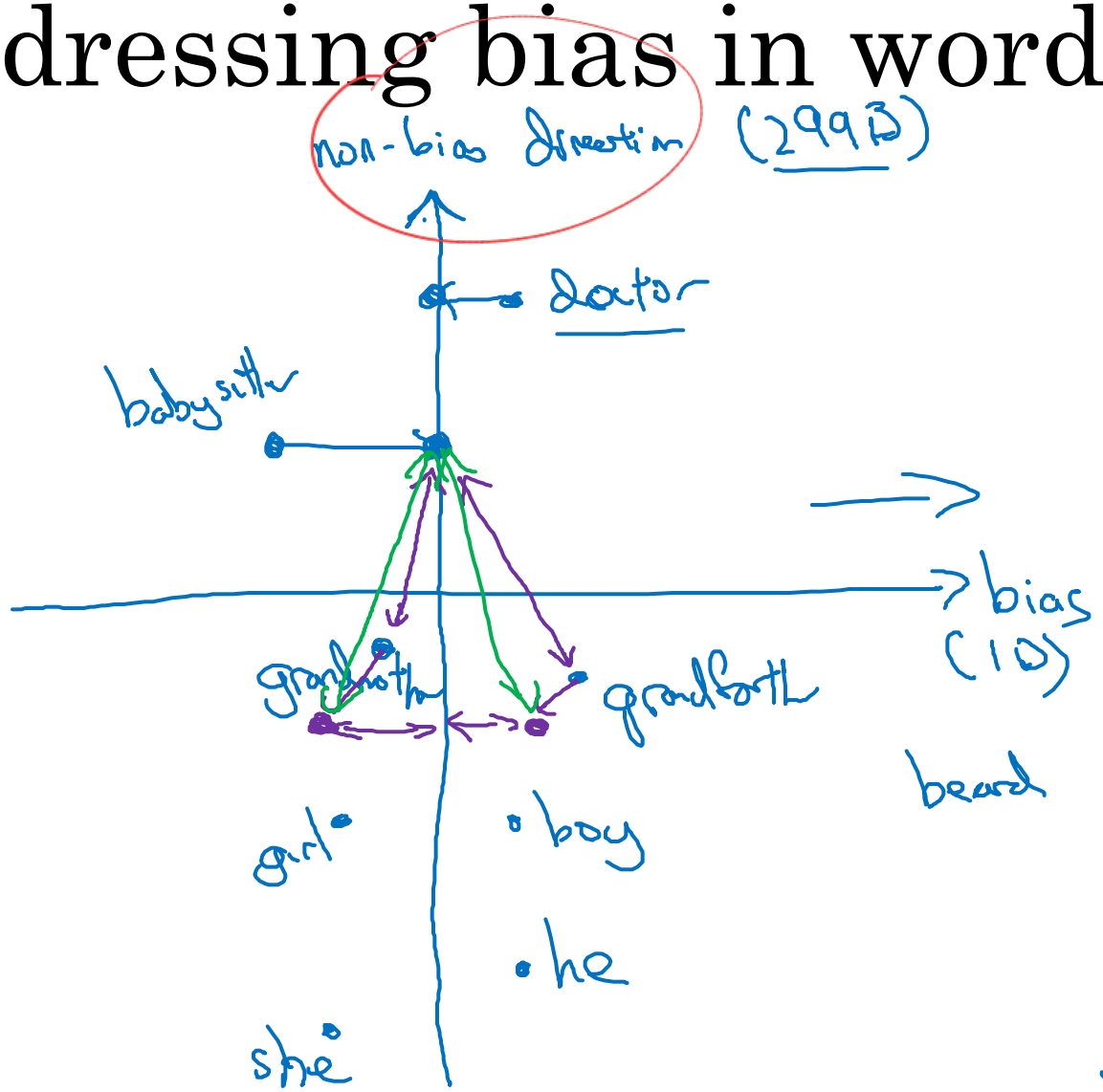
Man:Computer_Programmer as Woman:Homemaker

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.

$e_{he} - e_{she}$

$e_{male} - e_{female}$

⋮

average direction

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

non bias direction ⊥ "neutral axis"

↳ bias is part of the definition

3. Equalize pairs. using linear algebra

→ grandmother − grandfather

girl boy

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