Week 2

Word embeddings:

Trying to find relationships between words like man : woman :: king : queen

word representation:

V = [a, aaron, ..., zulu, <UNK>] |V| = 10,000

1-hot representation

Man (5391)	Woman	Apple	Orange
	O_{9853}	O_{456}	6257

O_{5391} (a 1 at position 5391, 0 elsewhere)

I want a glass of ~~orange juice~~ orange juice

I want a glass of apple ?

It can't decide apple juice even though it has learned to recognise orange juice since it doesn't know apple and orange are related

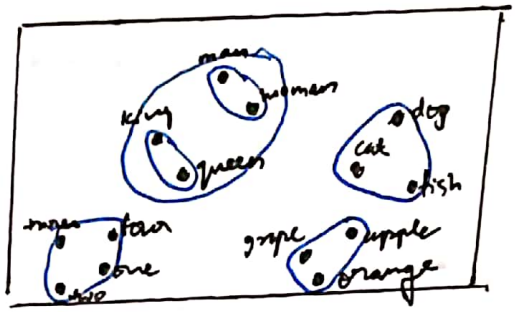
So we use featurised representation:

	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	0.09	0.01	0.02	0.01	0.95	0.97
⋮						
size						
cost						
verb						
etc						

300

Since apple and orange are similar, it'll now be able to find out I want a glass of apple juice

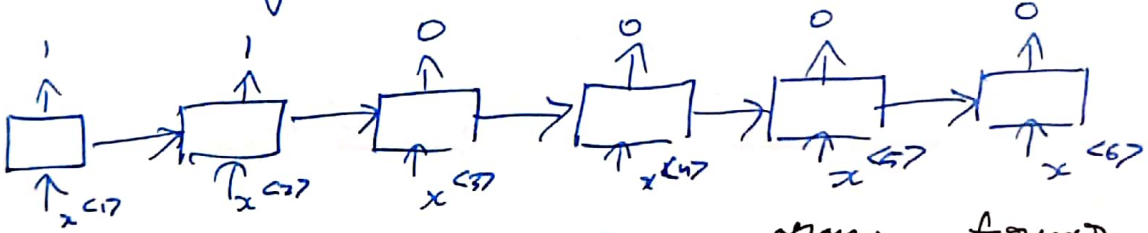
You can use the t-SNE algorithm to map the 300 dimension into 2D.



It is called embedding since each word is similarly embedded into a 300D space.

Using word embeddings:

Named entity recognition example:



Sally Johnson is an orange farmer

Robert Lin is a durian cultivator

name

How to learn words it hasn't seen before and associate it with orange farmer?

-> Use 1B - 100B words (unlabelled) from the internet along with your 100K labelled words

Transfer learning and word embeddings:

1. Learn word embeddings from large text corpus (1-100B onwards)
   (or download pre-trained embedding online)
2. Transfer embedding to new task with smaller training set (say, 100K onwards)
   Now you can use a smaller 10K or 30K word vocabulary (the training set).
3. Optional: Continue to finetune the word embeddings with new data.

Properties of word embeddings:

They can help with analogy reasoning.

Man -> Woman as King -> ?

	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

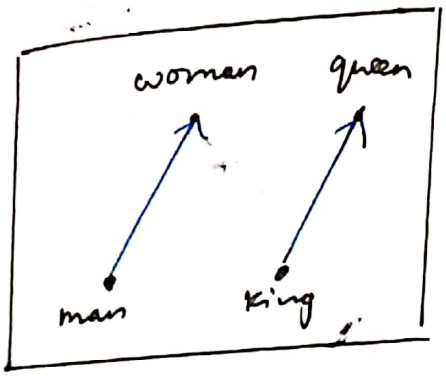
$e_{man} - e_{woman} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $e_{king} - e_{queen} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

∴ Gender is the relation

∴ It's queen

Analogies using word vectors:

So the algorithm we use to find the word e_w (queen in this example) is as follows:



Find a word w such that

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

$\therefore = \arg\max_w$ (...)

$\therefore \arg\max_w \text{sim}(e_w, e_{king} - e_{man} + e_{woman})$

Find the word with the maximum

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

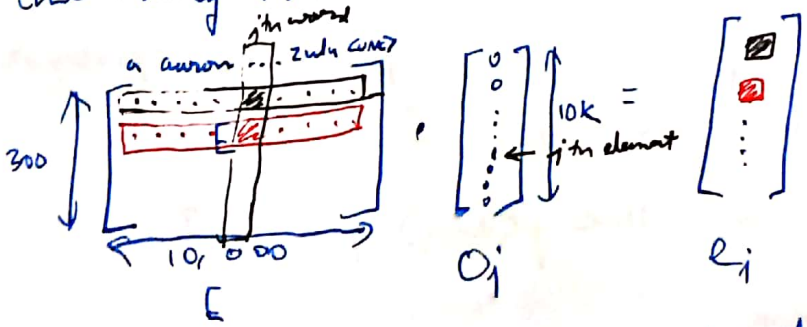
sim is called cosine similarity

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

It find the cosine of the angle ϕ between the 2 vector.

Embedding matrix:

Embedding for word j: $e_j = E \cdot O_j$



In practice we use a specialized function (embedding() in keras) to look up an embedding since matrix multiplication is costly since most are 0s (zeroes)

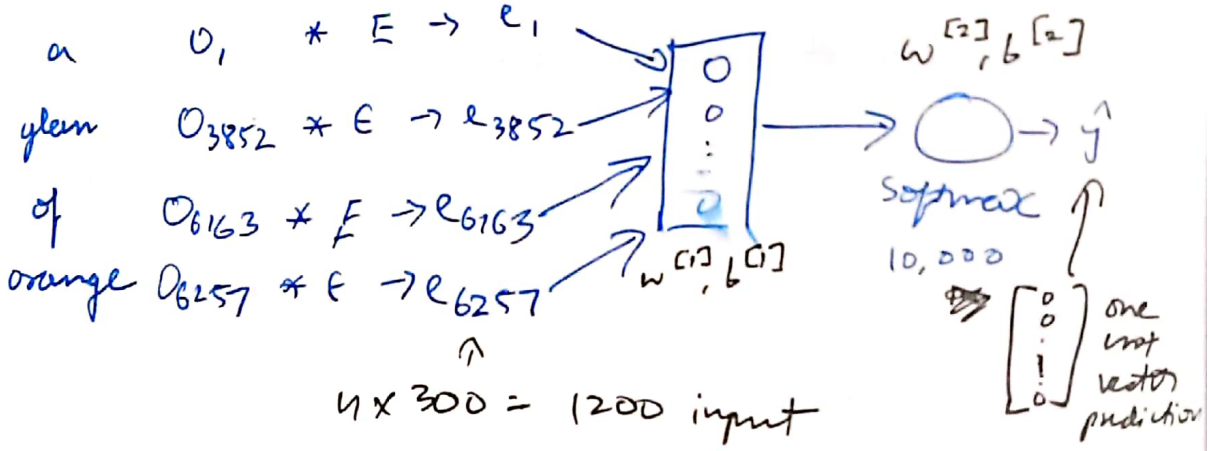
Learning word embeddings:

It started off with complex algorithms, but over time simpler algorithms have been found that does the job better. We will start learning from the complex algorithm to develop a good intuition:

I) A normal neural network:

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

We take the last 4 words into consideration.



Here we used a 'Last 4 words' context, other contexts that work better are:

- 4 words on left & right a glass of orange ? to go along with
- Last 1 word orange ?
- Nearby 1 word (like glass) glass ?

skip grams

context word — Target word

Word2Vec:

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

→ Choose a random context word

→ Then choose a target word randomly in a ~~4 or 5~~ word range (coming before)

Context	Target
orange	juice
orange	glass
orange	my

. . . .

model:

Vocal size = 10,000

Context c ("orange") → Target t ("juice")
6257 4834

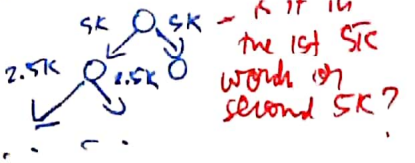
$O_c \to E \to e_c \to O \to \hat{y}$
Softmax

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 output t

→ $\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{10,000} y_i \log \hat{y}_i$ $y = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow 4834$

Problem with softmax classification:

→ It gets computationally expensive to sum up over 10,000 elements everytime. So use hierarchical softmax



Common words are on top while rare words are deeper →

However negative sampling is a better method to make softmax faster.

How to sample context c?

→ the, of, a. . . . ∈ common words

→ orange, apple . . . ∈ less frequent

we try to train on those less frequent words

Negative Sampling: 2 words

Given ~~a target and context pair~~ we will predict if they are a target and context pair.

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

Context c	word t	target? y
orange	juice	1
orange	king	0
orange	the	0
orange	of	0 ← even though it comes beside orange
orange	.	0
orange	.	0 ← pick k random words from the dictionary with orange and put 0
orange	.	

k (×) — y

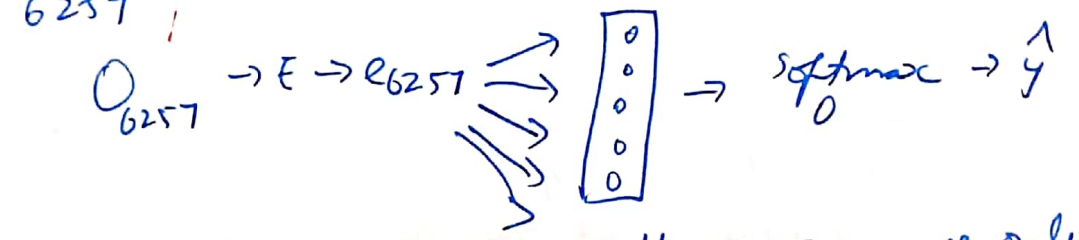
k = 5-20 smaller datasets

k = 2-5 larger datasets

Model:

context	word	target	
orange	juice	1	– I correct example
orange	king	0	} 4 randomly picked words to make incorrect examples
orange	book	0	
orange	the	0	
orange	of	0	

Orange
6257



∴ Rather than training all 10,000, we only train 5 (1 + 4)

How to choose the negative examples:

→ Choose words that have a high probability of occuring.

Problem: words like the, of, and will come

→ The other extreme is to choose rare words but that isn't good either

→ The best is to choose in between given by

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{10000} f(w_j)^{3/4}}$$

GloVe Word Vectors:

GloVe → global vectors for word representation

Here we use

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

X_{ij}: i → like c, j → like t

model:

$$\text{minimize} \sum_{i=1}^{10000} \sum_{j=1}^{10000} f(X_{ij}) (\theta_i^T e_j + b_i + b_j - \log X_{ij})^2$$

$f(X_{ij})$: weighting term; $\theta_i^T e_j$: we train this

→ Here we use $f(X_{ij})$ to prevent computation if $\log X_{ij} = 0$

i.e. $f(X_{ij}) = 0$ if $X_{ij} = 0$ → $0 \log 0 = 0$

like its anyways gonna be 0, so skip this training example

We can even use it to give less priority to words like this, is, of, a ... and more priority to words like durian so they kind of balance out

→ θ_i, e_j are symmetrical so at the end we can take their average

$$e_w^{(\text{final})} = \frac{e_w + \theta_w}{2}$$

Applications using Word Embeddings:

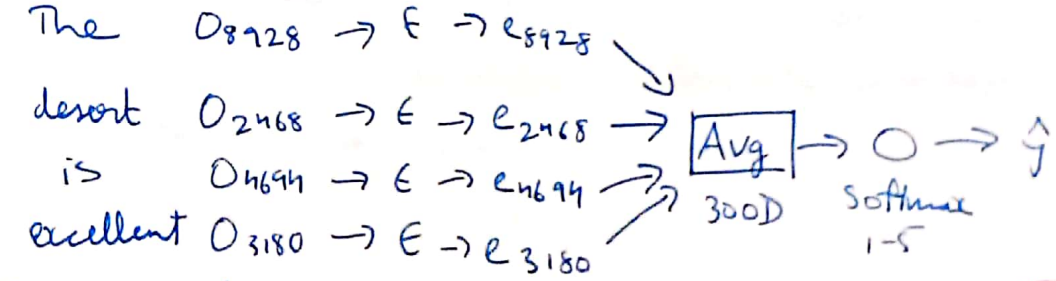
Sentiment Classification:

The task of looking at a piece of text and telling if someone likes or dislikes the thing they're talking about.

→ Challenge is that we may not have a huge labelled training set for it but with word embedding we can build a good classifier.

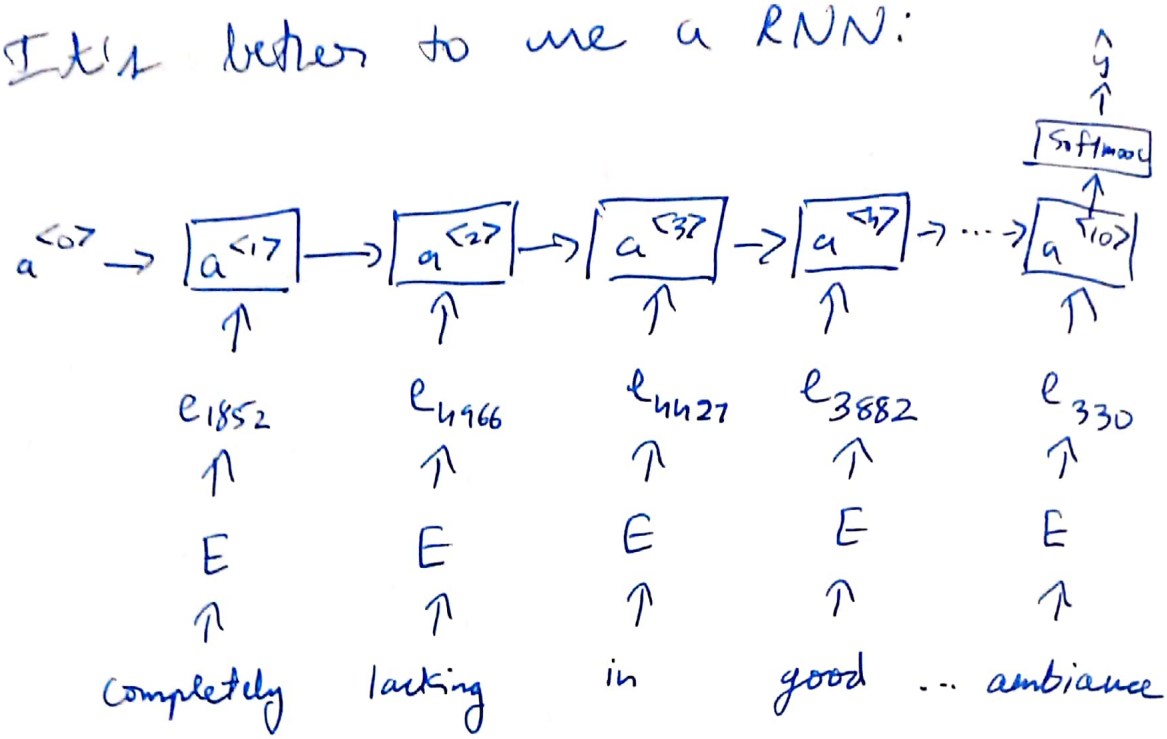
x	y
1) The dessert is excellent	★★★★☆
2) Service was quite slow	★★☆☆☆
3) Good for a quick meal, but nothing special	☆☆
4) Completely lacking in good taste, service & ambience	★★★☆☆
	★☆☆☆☆

Simple model: (for example 1)



For a sentence like "completely lacking in good taste, good service and good ambience", it'll think its a positive message if we use the above model.

It's better to use a RNN:



many-to-one model

Even if a word wasn't in the training example (like we use absent instead of lacking), it'll still perform good.

Debiasing word embeddings:

A few researchers found that the AI learnt biased stuff like:

→ Man: Computer_Programmer as Woman: Homemaker

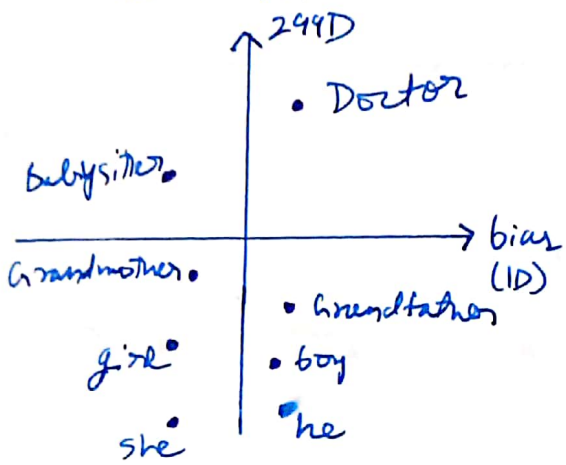
→ Father: Doctor as Mother: Nurse

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

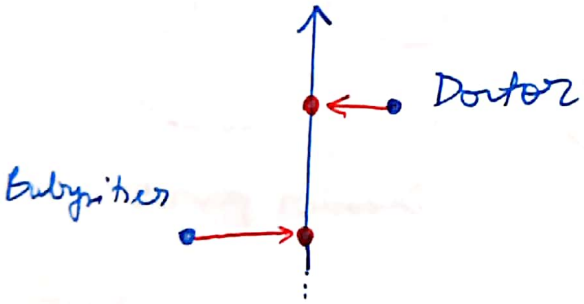
Steps to addren it (gender an example):

1. Identify bias direction

average { e_he - e_she
e_male - e_female
...



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



3. Equalize points