C5W2

NLP & word embedding:-

Word representation:-

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

1-hot representation

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

O stands for one-hot.

Inner product of any two hot vector is zero. So in this way it is not possible to capture the relationship/similarities of any two words.

I want a glass of orange juice

I want a glass of apple ____.

We can also put juice in the blank if we know that apple & orange are similar

Featuralized representation:-

similar feature

Features	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						
verb						

300

e_{5391} e_{5391}

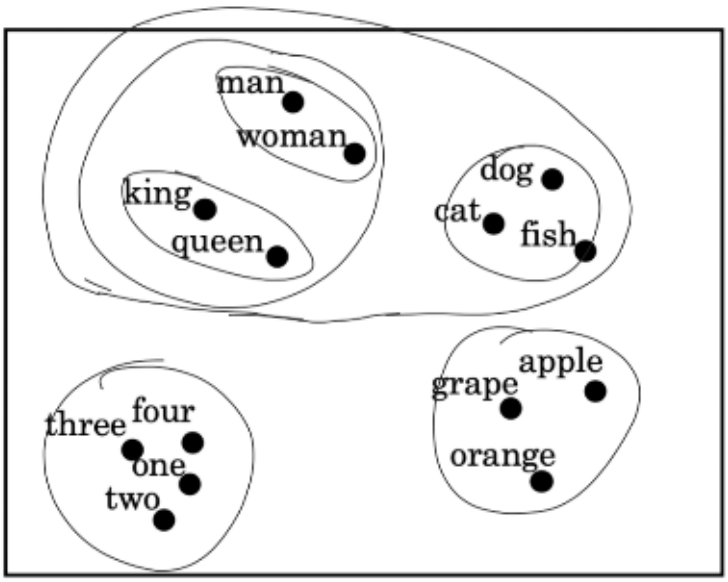
I want a glass of orange juice.

I want a glass of apple juice.

Let's say we have 300 feature. So each of the word

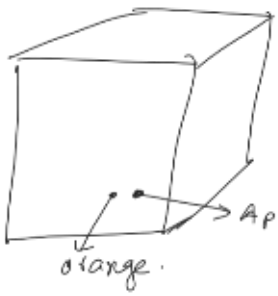
will be a vector of size 300.

Visualizing word embeddings:-



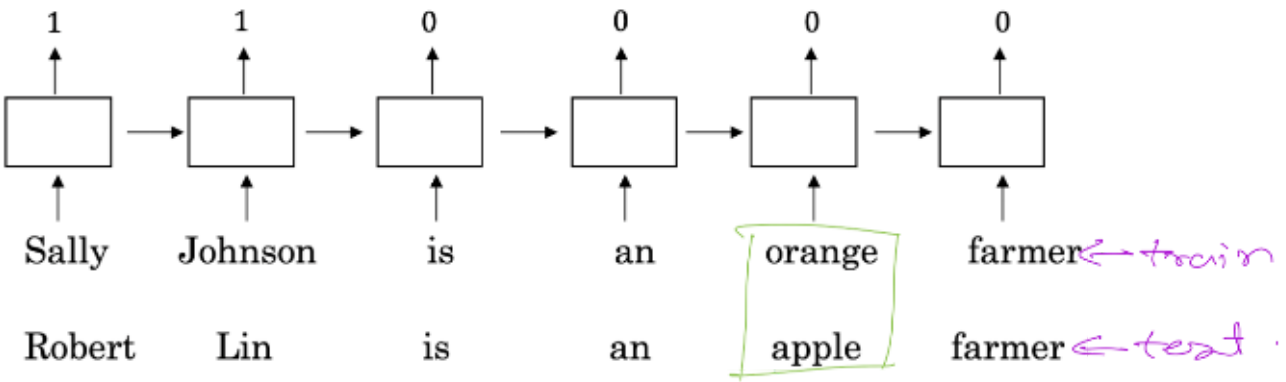
if the feature vector is 300D data then we can embed that in 2D data

t-SNE



Let this is a 300 dimensional vector. All the feature will have a points in this vector. t-SNE algorithm reduced this 300D to 2D for better visualization.

Name entity recognition example:-



Let say we train a model with the first sentence. Our

model learns that orange farmer indicate a person so it assign "Sally Johnson" as name. if we use word embedding it will know that orange & apple are similar so during test it will recognize "Robert Lin" as a name

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)

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

Properties of word embeddings:-

Man → woman as king → [?]

This relationship can be figured out by using properties of word embedding.

	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_{man}	e_{women}	e_{king}	e_{queen}.		

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

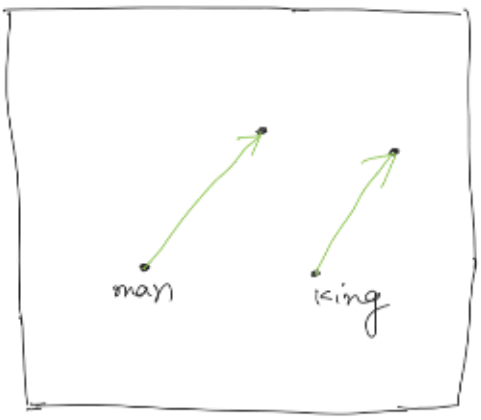
← 0 are approximate

we can find out:-

$e_{man} - e_{woman} \approx e_{king} - \boxed{e_?}$ ← e_{queen} will be the best match here.

Idea:

the vector difference between man & woman are pretty similar to the vector difference between king & queen.



300D

Find the word w, that:-

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

most commonly people use cosine similarities here.

euclidean distance is also used some cases.

if we use t-SNE here (2D data) instead of the 300D data. we might not a parallelogram like this. Because, t-SNE compress the dimensions so we might loose some data.

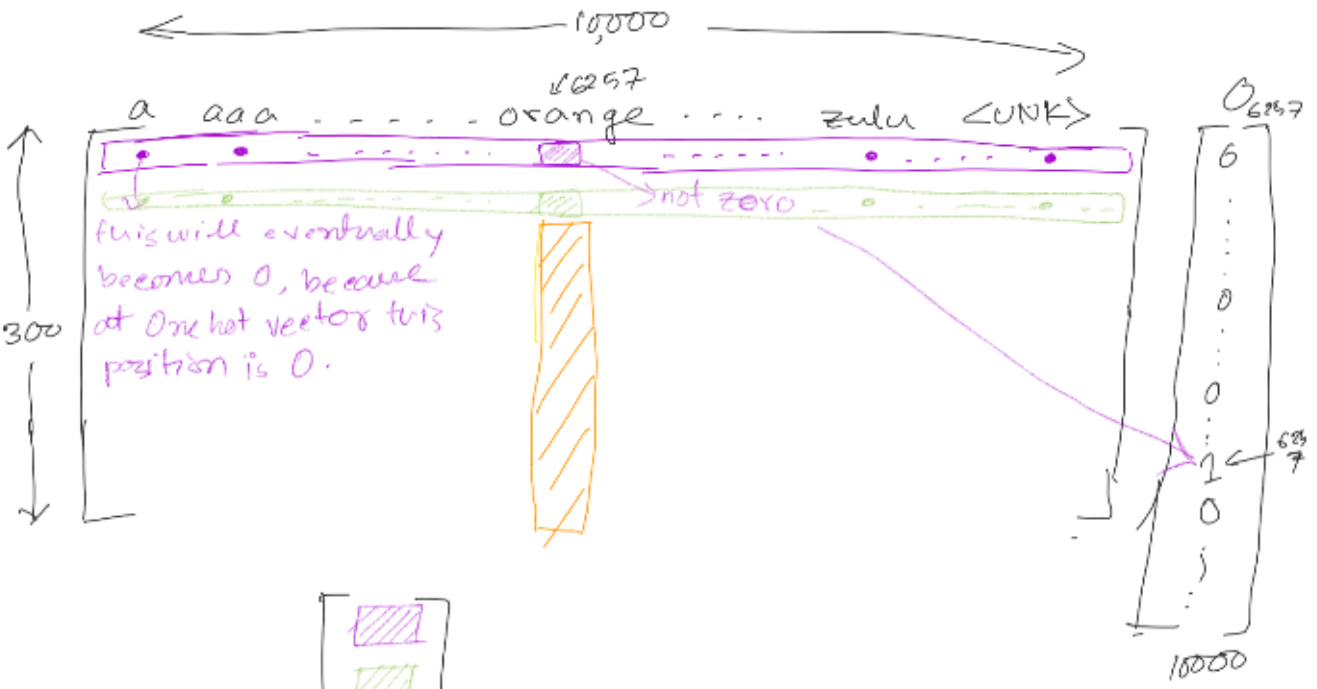
Cosine similarities:-

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

$\cos\phi$

Embedding matrix:-

Let we have 10000 words in our vocabulary & 300 feature. so embedding matrix will be 300 x 10,000 dimensional matrix. Let call it E.



$E \cdot O_{6257} = e_{6257}$

(300, 10K) (10K, 1) (300, 1)

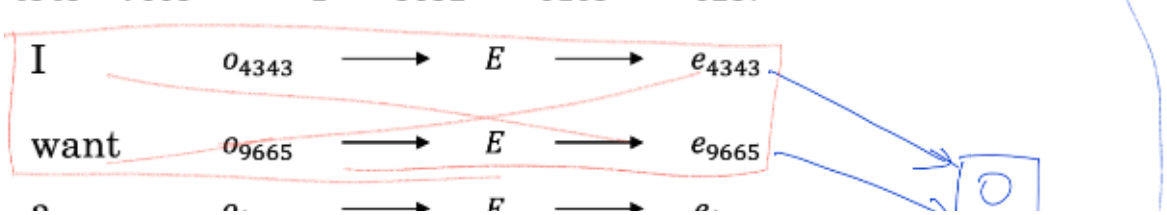
$EO_j = e_j$ = embedding vector for word j.

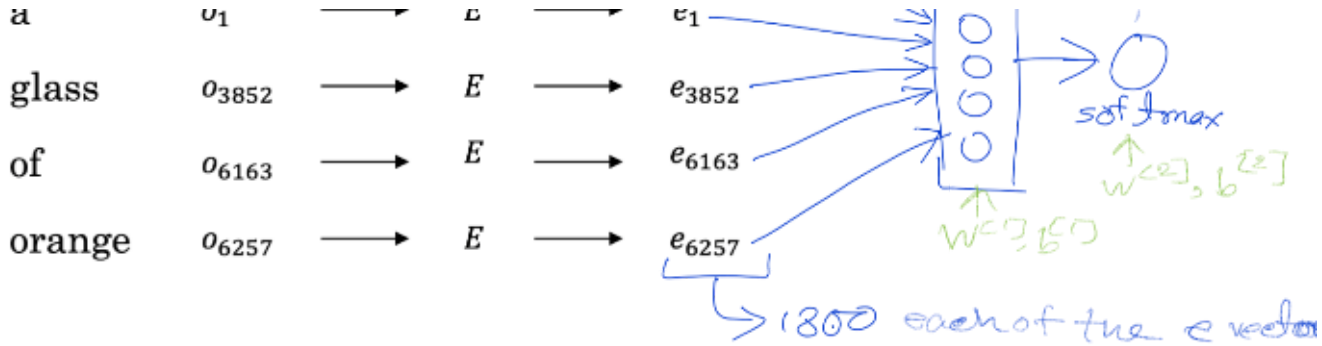
initialize E randomly, then use gradient descent to update it's value.

One-hot vector is pretty large & only one element of this vector is 1 rest are zero. so it is not efficient to apply matrix multiplications on one-hot vectors. So in practice we use specialized function to look up an embedding

Natural language model:-

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





We can omit the previous words & use last four words to predict.
↑ hyperparameter

Other context/target pairs:-

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

We can use different context to predict the target:-

Contexts:-

- Last 4 words.
- 4 words on left & right.
- Last 1 word
- Nearby one word → skip gram

Word2Vec:-

Skip-gram:- make a conlex-target pair.

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

Context	target:-
Orange	juice.

Orange juice

orange my.

Model:-

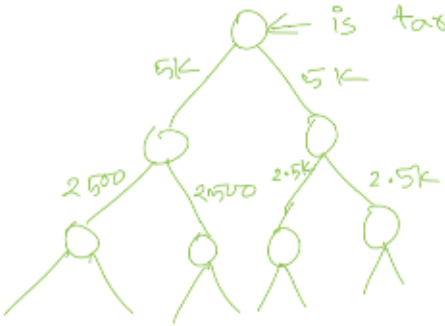
Context c ("orange") ⟶ Target t ("juice")

$O_c \rightarrow E \rightarrow e_c \rightarrow$ (softmax) $\rightarrow \hat{y}$

softmax:- $P(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$ → summing over 10,000 every time makes the algorithm slow

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

To solve the problem people uses hierarchical softmax:-



So we don't need to sum over all vocab size.

This hierarchical softmax is not a perfect binary tree or symmetry. Rather, it is developed in a way such that most common words are in the top of the tree. And uncommon words are at the very bottom.

How to sample context c?

→ we can use uniform randomly. If we do that we

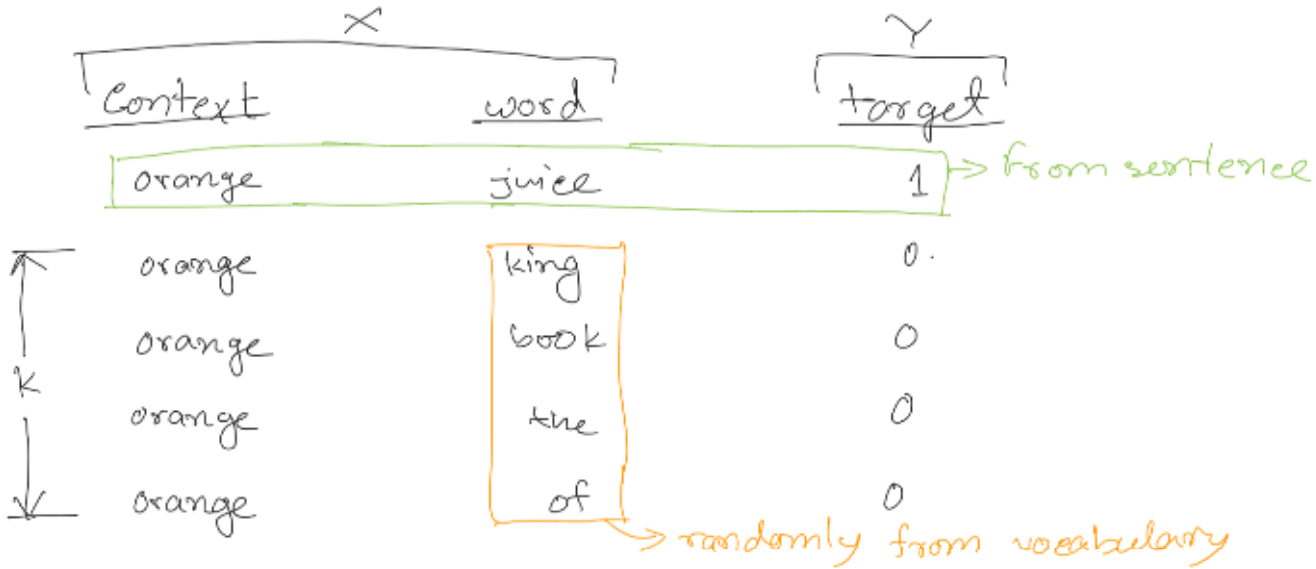
will see common words appear more often. (the, a, of, and etc).
And less common words doesn't appear often (orang, apple etc)
But this method is biased with the most frequent words.

Negative sampling:-

Generating dataset:-

we will select a context word from the sentence & related target word and set isTarget to 1. Then we will randomly pick K words from dictionary for the same context word and set their isTarget to 0.

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



$k = 5-20$ → smaller dataset.

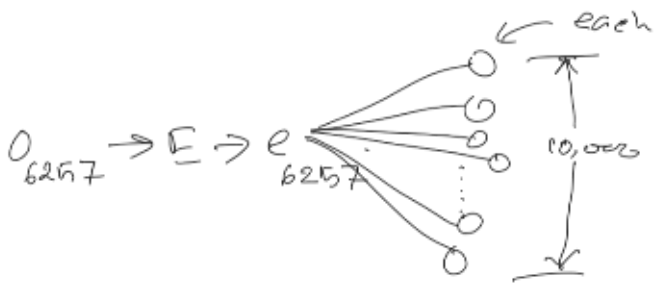
$k = 2-5$ → larger dataset.

Our positive to negative ratio is $1:k$.. so for every 1 positive example we have k negative example.

We can get a logistic regression model:-

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

if we have the context word orange 6257



each one of these is a logistic regression model.

In every iteration we will update k+1 logistic unit. which is computationally efficient than applying softmax which compute 10,000 unit.

How to select the negative sample?.

$f(w_i)$ ⇒ frequency of a word.

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

Glove (Global vectors for word representation):-

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

c → ij ← t

How many times does a word i appeas in the context of different word j. so most of the time

$X_{ij} = X_{ji}$

Models-

symmetry.

$$\text{Minimize} \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(x_{ij}) \left(\theta_i^T e_j + b_i + b_j - \log x_{ij}\right)^2$$

if $X_{ij} = 0$ then $\log X_{ij} = -\infty$. so in this case it will

be impossible to minimize. In order to avoid this a weighting term $f(X_{ij})$ is used where $f(X_{ij}) = 0$ if $X_{ij} = 0$. Also, it use a hueristic approach to weight - so that the common words (this, a, the ...) don't get too much weight & less frequent word don't get less weight.

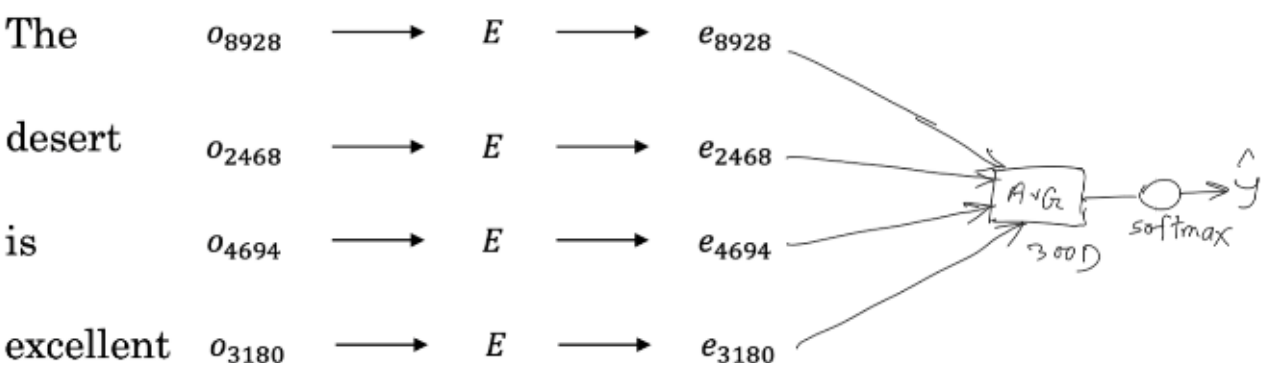
$e_w^{(final)} = \frac{e_w + \theta_w}{2}$ / θ & e plays the same role so we can average them.

It is hard to interpret the relationship between the feature matrix.

Sentiment classifications:-

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.	★☆☆☆☆
The dessert is excellent	★★★★☆

8928 2468 4694 3180



300D

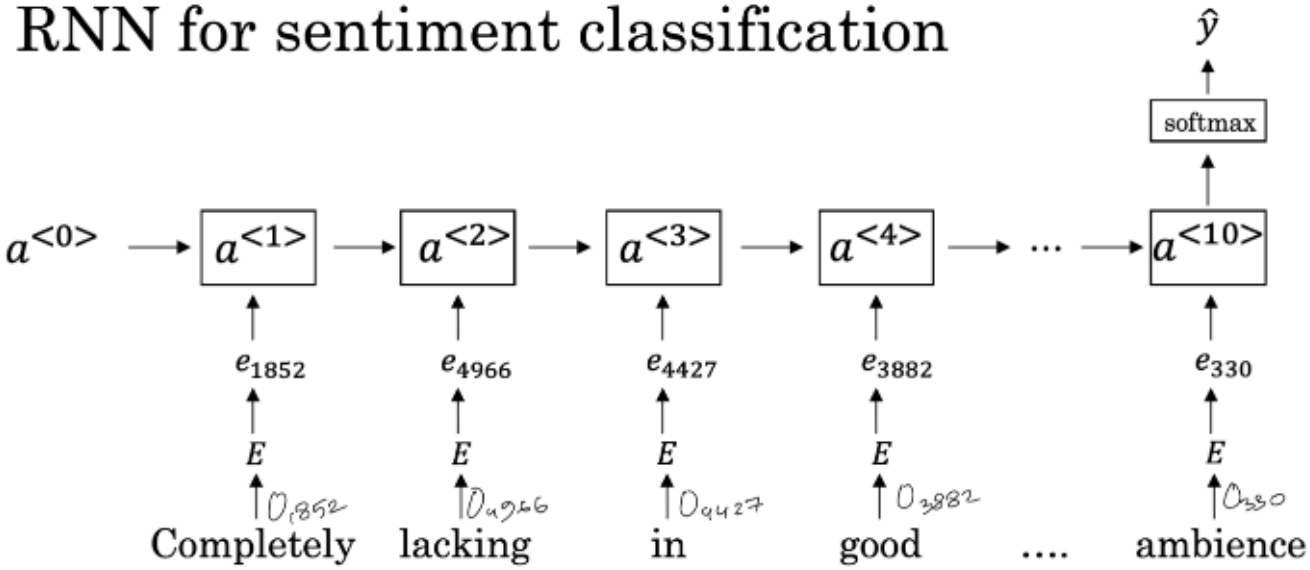
This approach has a problem. Let our review is :

"Completely lacking in good test, good service, & good ambiance".

there is lot of "good" in this review. So if we use the average technique then we might end up giving it a 4+ or 5-star review

To avoid this we can implement RNN:-

RNN for sentiment classification



Debiasing word embedding:-

The problem of bias in word embeddings

Man:Woman as King:Queen

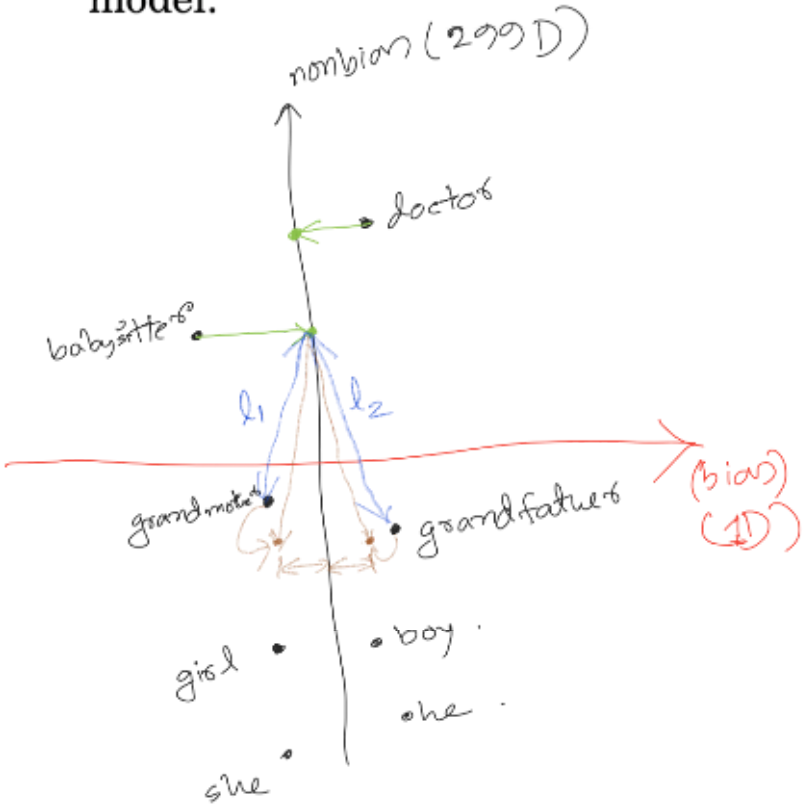
Biased to gender

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.



1. Identify bias direction
   - $e_{he} - e_{she}$
   - $e_{male} - e_{female}$
   - ⋮
   - → average.
2. Neutralize :- For every word that is not definational ((girl, boy), (she, he)), project to get rid of bias.
3. Equalize pair.

For this example only the gender biased is shown. Averaging is simple to find the bias direction we can use SVD (singular value decomposition) to find the bias direction. that might give us more direction instead of 1-dimensions.

Geoandmother–Geoandfather, She–he, Girl–Boy these are gender dependent by definition so we won't remove gender bias for these word.

$l_2 > l_1$. so the word babysitter is biased towards the grandmother. To remove the bias move the words geoandmother & grandfather is a way so that $l_2 = l_1$.