Introduction to Word Embeddings

Wednesday, September 30, 2020 3:11 PM

① Word Representation

- Word embeddings are used to represent words
- So far we have used one-hot vector
- It doesn't represent relations between words like man and woman, apple and orange, etc.
- The distance between any 2 one-hot vectors is same and inner product is zero
- Hence, we use featurized representations

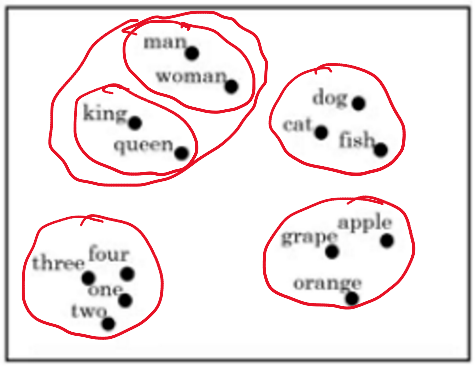
	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97
⋮						
size						
cost						
alive						
verb						

300

- We use a floating point number to declare a feature
  - eg:
    - each word will have 300 features (not explicitly representing a feature)
    - each word colum will be a 300 dimensional vector
    - eg: e_{5391} → describes man vector where 5391 is position in vocabulary

- Visualization of word embeddings using t-SNE algorithm to reduce the features to 2 dimensions (which makes it easy to

using t-SNE algorithm to reduce the features to 2 dimensions which makes it easy to visualize

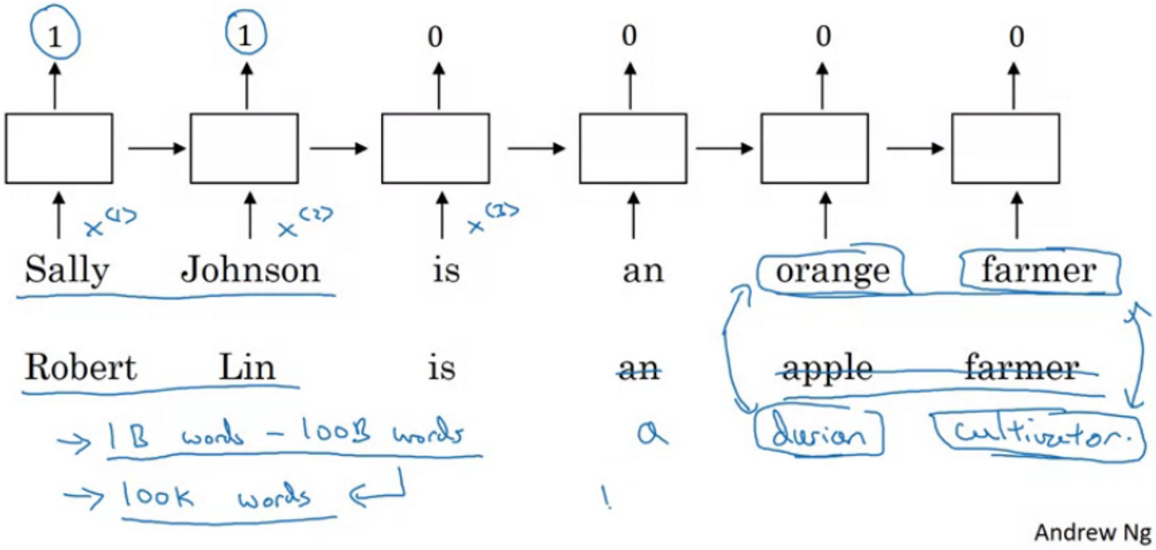


→ The related words are grouped together

② Using word Embeddings

- Eg: we use word embeddings in named entity recognition

Named entity recognition example



→ The network should be able to identify that Sally Johnson is a person's name
→ Apple and orange should have close representation
→ The algorithms are powerful enough to recognise less common words like "durian cultivator"

- Transfer learning is used to learn a huge corpus of text

Transfer learning and word embeddings

1. Learn word embeddings from large text corpus. (1-100B words)
   (A) lots of training examples
   (Or download pre-trained embedding online.)
   ↓
   (B) less traing examples
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.

Andrew Ng

- Size of input is now a feature vector (~300 features) instead of one-hot vectors.
- Word embeddings can be compared to face data encoding in siamese network
- In word embeddings, we are learning an embedding for every word in vocabulary. In face encoding, we map each new image to a vector

(3) Properties of word embeddings

- Word embeddings have interesting properties that help us to derive the relations between words.

	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_{Woman}	e_{King}	e_{Queen}		

- If we give this neural network
  Man => woman
  King => ?
- The Neural Network will be able to compute this using difference

- The Neural Network will be able to compute this using difference between the embeddings

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

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

$\Rightarrow e_? = e_{king} - \begin{bmatrix} -2 \\ 0 \\ 0 \end{bmatrix}$

↳ This is a 4 dimensional embedding. In practice ~300 dimensions are used

- Mathematically

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

e_w → required word embedding

sim → similarity function

- Cosine similarity

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

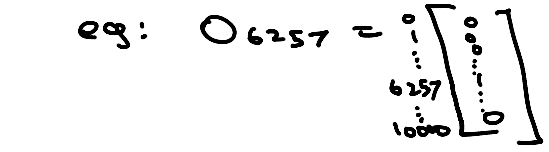
(inner product)

- Euclidean distance also works fairly well (although it needs to be inverted as it measures how far rather than how near they are)

④ Embedding matrix

- The embedding matrix (E) for our vocabulary will be (300, 10000) in shape (300 features & 10000 words)

- If we use one hot representation and multiply it with E, we get embedding of the particular word

eg: O_{6257} = 

$$E \cdot O_{6257} = \underset{(300,10000)}{[\quad]}\underset{(10000,1)}{[\quad]} = \underset{(300,1)}{[\quad]}$$

$$\Rightarrow np.dot(E, O_j) = e_j$$

- we randomly initialize E & learn parameters
- Using one-hot vectors to find embeddings isn't efficient. Hence we use special functions.