WORD EMBEDDINGS

Presented By:
Prasanna Adhikari
KCE074BCT029

WORD EMBEDDING

- Word embedding is a way of representing word in vectors of real numbers for language modeling and feature learning in Natural Language Processing(NLP).

- It is the basics for NLP applications like text simplifications, parsing, debiasing, sentiment analysis and many others.

- We first need to understand word representation.

Word Representation

- Suppose we have a Vector V which is a given as:

  V: [a, aaron, …., the, ……., Zyzzyva,<UNK>]

  such that |V| = 10000
- ‘the’ located at 2450 in V can be represented as:

$$O_{2450} = \begin{pmatrix} 0 \\ 0 \\ \dots \\ 1 \\ \dots \\ 0 \end{pmatrix} \begin{matrix} 1^{st} \\ 2^{nd} \\ 3^{rd} \text{ to } 2449^{th} \\ 2450^{th} \\ 2451^{th} \text{ to } 9999^{th} \\ 10000^{th} \end{matrix}$$

One hot representation for ‘the’ i.e. $O_{2450.}$

Word Representation

- For example:
  I want an apple juice. I want an orange _____.

- Predict of the word cannot be found form V.
- Relationship is built by featurized representation.

Man	Woman	King	Queen	Apple	Orange	
-1	1	-0.95	0.97	0	0.01	Gender
0.02	0.01	0.93	0.95	-0.01	0.00	Royal
		……..				…..
0.02	0.00	0.01	0.0	0.95	0.97	Food

- We create about 300D embeddings for each of the topics.
- We embed these following words into a single unit by using algorithms like t-SNE algorithm method.
- The each word are represented in a 300D as e_n, where n is the location of the given word in the vector V.

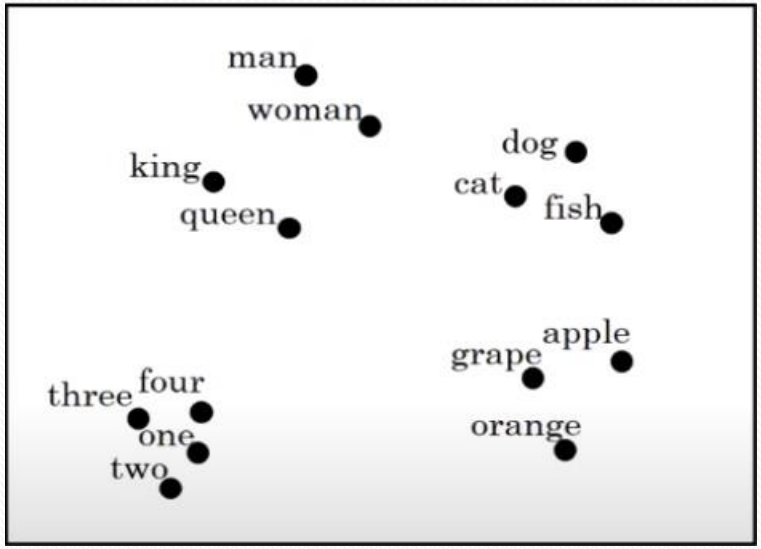


Figure 1: Conversion of 300D model to 2D model with t-SNE

Entity Recognition

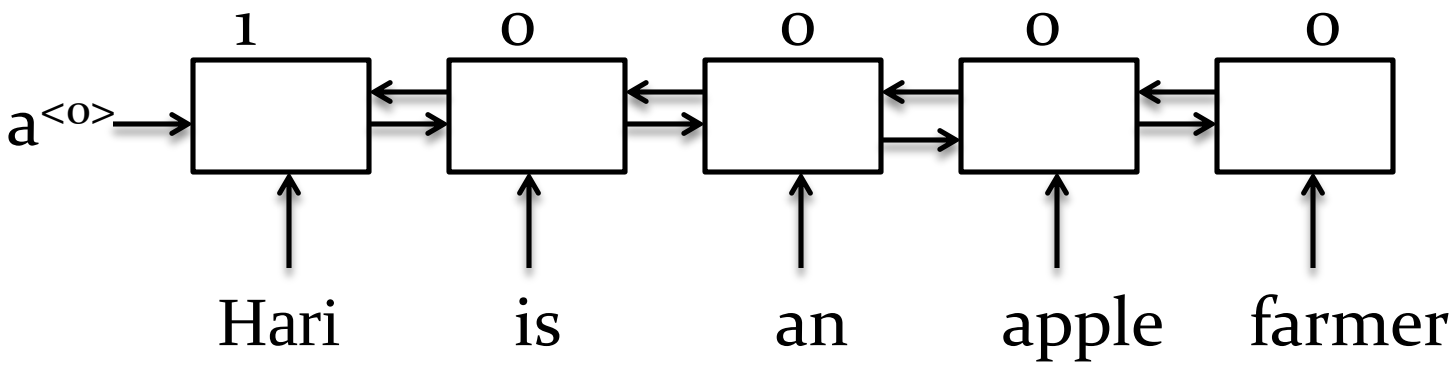


Figure 2: A bidirectional RNN model for Name entity recognition

- We have a bidirectional RNN based model that recognizes an name entity based on the context provided.
- We use <UNK> if our model doesn't find it.

- For this we extract words from a source that may have 1 billion to 100 billion of unlabelled text.
- We examine the text an figure out the similar words and group them together by entity recognition.

- This process can be facilitated by use of Transfer Learning mechanism where we take information we have learned form previous training of huge dataset of unlabelled text to gradually decrease the computational complexity.

- Then we learn the embedding vectors from e_0 , e_1, … e_{10000} for the given 10000 vectors only.

Transfer Learning and Word Embedding

SUMMARY

1. Learn word embedding form large text corpus. (1 – 100B words) or download pre-trained embedding model available at the internet.
2. Transfer embedding to new task with smaller training set. (reduced to 100K words) where we prefer the use of 300D dense vector for embedding or the 10000 one hot vector.
3. Optional :We continue to fine tune the word embedding with the new data.

Property of Embeddings

- From the embedding vector if we have

  e_{woman} : $[1\ 0\ \ldots..\ 0]_{300}$

  e_{man} : $[-1\ 0\ \ldots.\ 0]_{300}$

  Here the first element refers to gender where 1 represents female and -1 represents male.

  Then , $e_{man} - e_{woman}$: $[-2\ 0\ \ldots..\ 0]_{300,}$

  Similarly, $e_{king} - e_{queen}$: $[-2\ 0\ \ldots.\ 0]_{300}$

- Using this property we see that vector difference between two similar property can find analogies of a word like

MAN: WOMAN as KING : _?__.

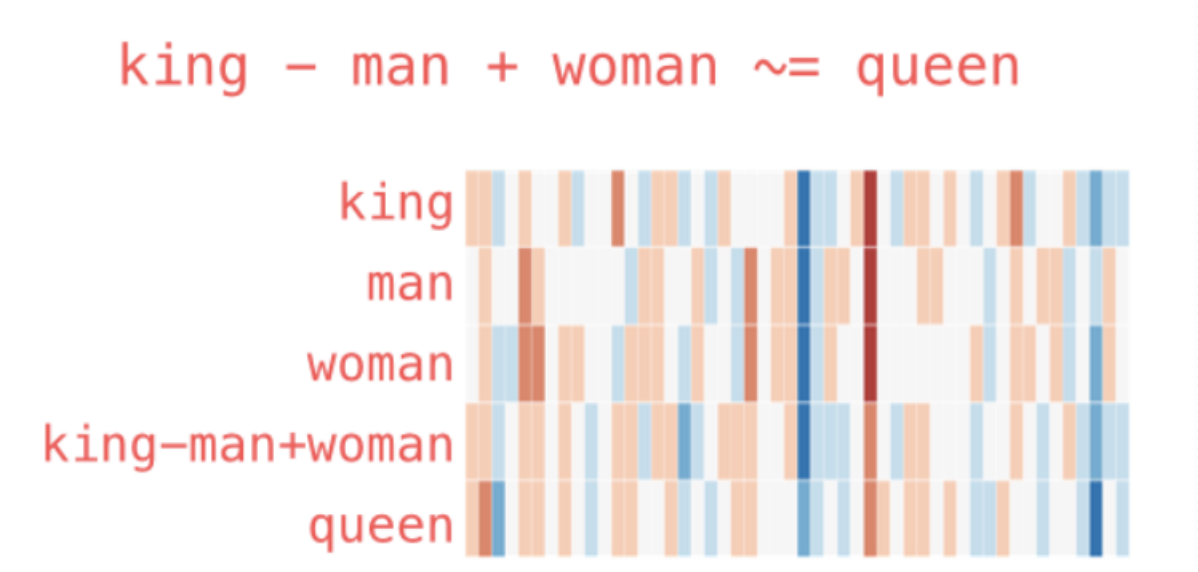
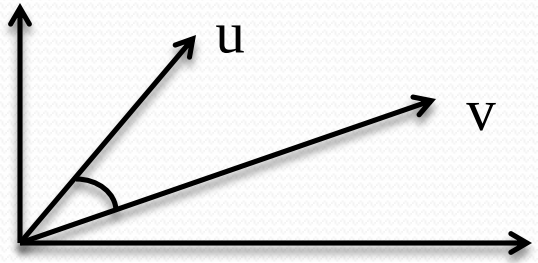


Figure 3 : Comparison of embedded matrix for the unknown word

- We use similarity function to find out the expected outcome by:

$$\arg\max \text{sim}(e_{?} , e_{king} - e_{man} + e_{woman})$$



- Here we use cosine similarity function which computes as:

$$\text{sim}(u,v) = \cos(\emptyset) = \frac{u \,.\, v}{\|u\| \,.\, \|v\|}$$

- We can also use eucledian distance between these two grouped wording but shows less efficient output.

$$\|u - v\|^2$$

EMBEDDING MATRIX

$$\begin{bmatrix} a & aaron & \ldots\ldots & orange & \ldots.. & Zyzzyva \\ 0 & 0 & \ldots\ldots & 0 & \ldots.. & 0 \\ 0 & 0 & \ldots\ldots & 0 & \ldots. & 0 \\ & & \ldots\ldots\ldots\ldots & & & \\ 0 & 0 & \ldots.. & 0 & \ldots\ldots & 0 \\ & & & & & \\ 0 & 0 & & 1 & & 0 \end{bmatrix} * \begin{bmatrix} O_{6257} \\ 0 \\ 0 \\ \ldots. \\ 1 \\ \ldots. \\ 0 \end{bmatrix}$$

Embedding matrix E (300, 10000) O(10000,1)

Embedded Matrix for Orange (O_{6257}) is $e_{6257} = E * O_{6257}$

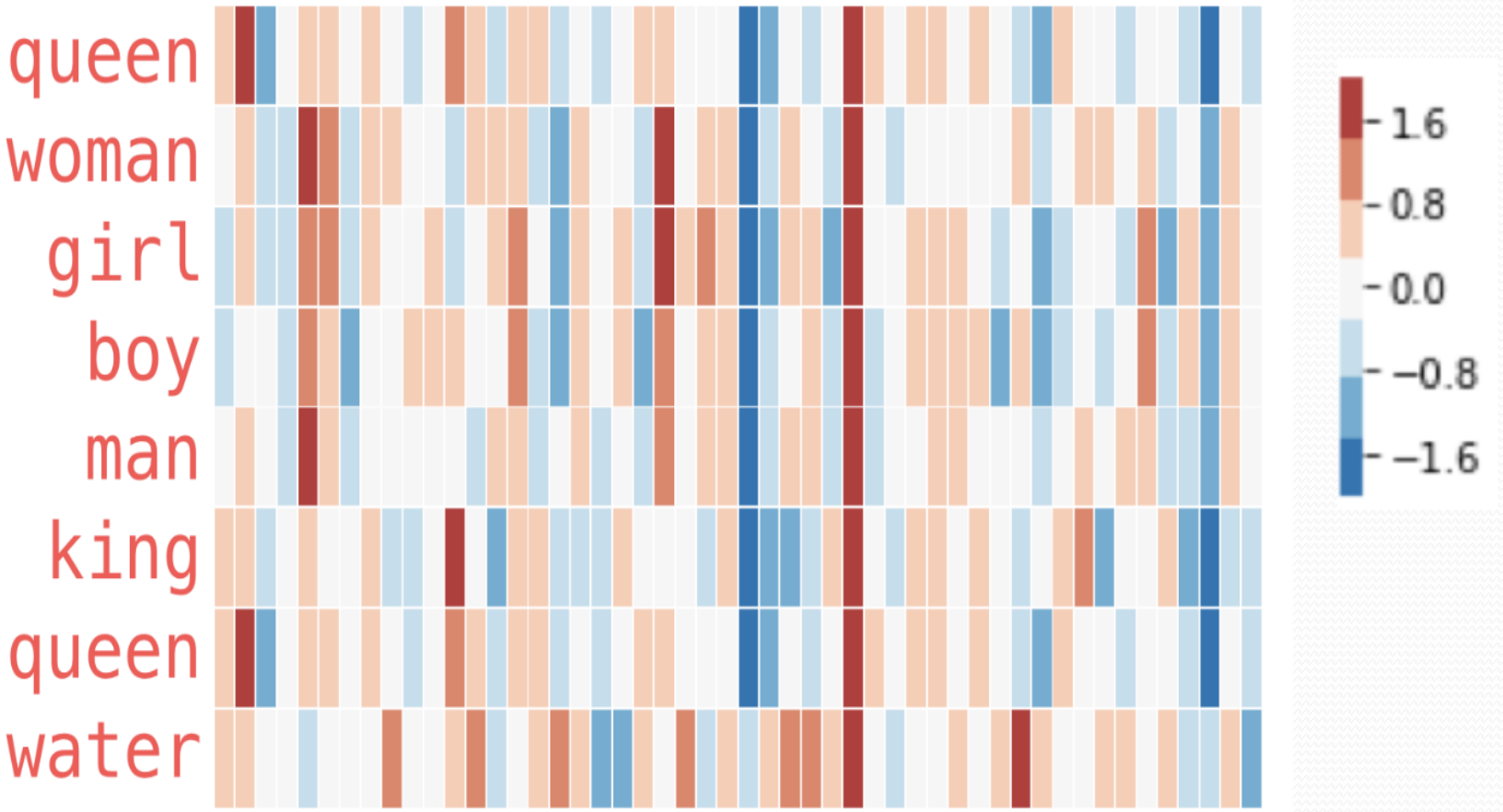


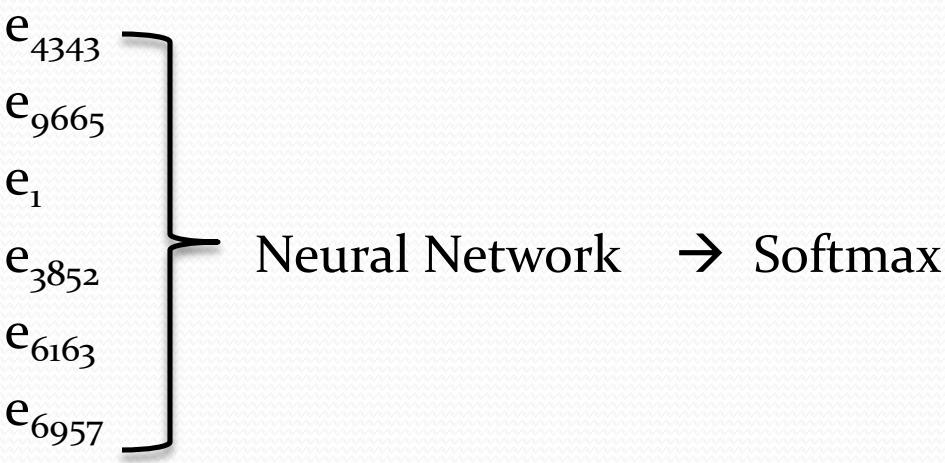
Figure 4: Colorful representation for a small embedded matrix

Natural Language Model

- Let us make a auto predictive sentence.

  I (O_{4343}) want (O_{9665}) a (O_1) glass (O_{3852}) of (O_{6163}) mango (O_{6957}) _____.

- We use: $e_n = E * O_{N,}$ where N is location number in V.



Word2Vec Skip-Gram

- This method uses the softmax function for output target.

  context c (‘mango’) → Target t (‘juice’)

  O_c → E * O_c → e_c → Softmax → Output (t)

- Probability for target, $p(w_O|w_I) = \frac{\exp\left(v'_{w_O}{}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left(v'_{w}{}^{\top} v_{w_I}\right)}$

w_0 is the target for context w_1.

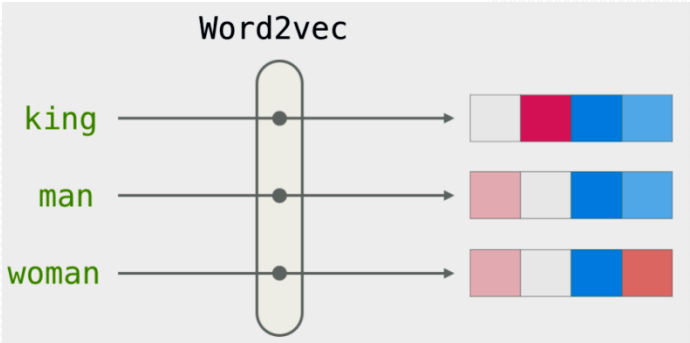
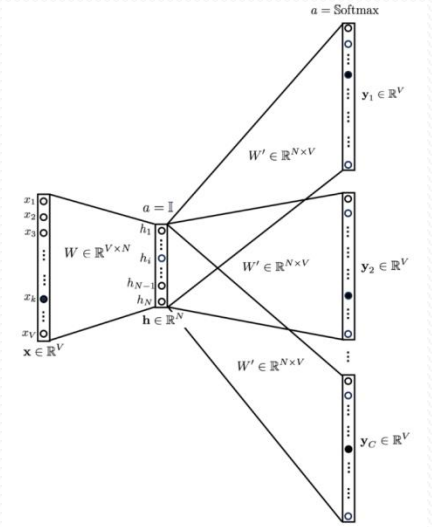


Figure 5 :Word2Vec Output



- Figue 6: Implementation of Word2Vec Skip Gram Model

- Sampling: Remove the frequent words like a, an, the, and, to etc and choose the noun, verb or any context related word present out there.
- ¸ The hierarchical softmax is carried out by dividing the whole part into n parts. For example if we take 10k of softmax output then we divide into batches of 2.5k each so that summing process get simplified.

Negative Sampling

- Optimized form for Word2Vec.

context	target word	target?
mango	juice	1
mango	the	0
mango	book	0

- Pick context and find probable target as in first row.
- For k time(2 to 20), we take same context but generate targets form vector using empirical frequency and label those as negative example using RNN models.
- Then we create training set for supervised learning method.

- Then we create a regression model with sigmoid activations and estimate the probabilities.

$$P(y = 1 \mid c, t) = \text{sigmoid}(\phi_t^T e_c)$$

- Here if we take empirical frequency with heuristic values, between the extremes of sampling then probability is given as:

$$P(w_i) = \frac{f(w_i))^{3/4}}{\Sigma f(w)^{3/4}}$$

Here $f(w_i)$ is the huristic value for the word.

GloVe Word Vectors

- Here the context gives target using an explicit context X_{ct} where X_{ct} = No. of times t appears in context of c
- We determine whether the c and t occurs in close proximity.

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

- Here $f(X_{ij})$ = weighting term which is 0 if X_{ij} =0 and gives more weight to empirically frequent words.

  b_i is for target and b_j‘ is for context
- We measure the relation of X_{ij} with our metrics.

Basic Sentiment Classification with Neural Network

- Example:

  Input : The film is excellent.

  Classifier generated ratings output : 4/5

The $(O_{8928}) \rightarrow e_{8928}$

film $(O_{4080}) \rightarrow e_{4080}$

is $(O_{4694}) \rightarrow e_{4694}$

excellent $(O_{3180}) \rightarrow e_{3180}$

Average of sentiments → Softmax 1 - 5 → Output y

- Averaging helps for any length of inputs whether it is short or a very long message by summing 300D with a RNN model and providing a output for representation.
- But it does ignore the word order for the given sentence.

IMPLEMENTATION

1. Split the given sentence into parts i.e. list.
2. Calculate the total of the words using Word2Vec mapping.
3. Divide the total to length of list to output the average.
4. Then pass the average through forward propagation.
5. Compute the cost.
6. Backpropagate to update the softmax paramaeters.
7. Carry this process for several epochs.

THANK YOU