Name: Mohammed Raamizuddin
Roll No: 19K41A0517
Subject: Natural language Processing
Class: IV CSE - A.

ASSIGNMENT - II

Word2Vec

Introduction:

→ Word embedding is one of the most Popular representation of document vocabulary. It is capable of capturing con--text of a word in a document, semantic and syntactic similarity, relation with other words, etc.

→ Word embeddings are vector representations of a par--ticular word.

→ Word2vec is one of the most Popular technique to learn word embeddings using shallow neural network. It was developed by Tomas Mikolov in 2013 at Google

→ The fundamental task to build a language model is to convert the words into numbers (vectors) to train the model and, get the output from it. There are number of ways you could do that, let's discuss the approach--es in breif;

(i) Assign numbers to words in the vocabulary:

The most significant approach anyone can think of is to assign numbers directly to the words in the vocabulary.

For example: vocabulary consists of a set of words like

[Ashes, Axar, bat, back, cover, ..., team, ..., zimbabwe]
1 2 3 4 5 1023 50000

They are numbered as mentioned above they are assig--ned unique numbers and these numbers will be used

For training the model. The problem with this conver-
-sion is that we cannot capture the similarities
between the words like Ashes and IPL, they both
are numbered differently and will be far apart but
the similarity between them is they are tournaments
this cannot be captured.

(ii) one-hot Encoding!

The second approach is to assign the numbers to a
two-dimensional format.

For example!

	Ashes	Axar	Bat	Back	Cover	....	zimbabwe
Ashes	1	0	0	0	0	--------	0
Axar	0	1	0	0	0	--------	0
Bat	0	0	1	0	0	--------	0
Back	0	0	0	1	0	--------	0
Cover	0	0	0	0	1	--------	0
⋮	⋮	⋮	⋮	⋮	⋮	--------	⋮
zimbabwe	0	0	0	0	0	--------	1

As we can see in the above representation the two
dimensional encoding of words is done based on the
similarity of the same words in the row or column
respectively. The issue with this encoding is that it
again Doesn't capture relationship with words and it
is computationally in efficient. The vectors will be of large
size.

(iii) Word Embeddings:

This is the most efficient technique among the two specified methods above. Word embeddings allow you to capture relationship between two words. The way this technique works is that for a given word certain features are crafted and vectors for those words are generated.

For example!

	Dhoni	Cummins	Australia
Player:	1	1	0
fitness:	0.9	0.8	0
location:	0	0	1
eyes:	1	1	0
gov:	0	0	1
	[1,0.9,0,1,0]	[1,0.87,0,1,0]	[0,0.7,1,0,1]

As we can see in the above representation the three words are converted to five-dimensional vectors by using features such as Player, fitness, location, eyes, government. This technique could resolve variety of problems in NLP tasks. The vectors could directly give similarities between words to some extent.

The word embedding technique is currently used to convert word to vectors there are two methods to do so, they are TF-IDF and Word2vec.

→ Let's see how word embeddings are calculated by those two techniques!

[illegible]:

1.) In a supervised learning technique for word embedding the vectors generated using the word embedding technique are multiplied ~~using~~ with the one-hot encoded vectors and these vectors are flattenned using the nueral networks and the output is predicted using these vectors. This vectors are finalized by numerous iterations and propogations through the nueral network. Term-Frequency - Inverse Document Frequency is an example of sepervised learning technique. It is a statistical measure used to determine the mathematical significance of words in documents. The vectorization process is similar to one hot encoding. Alternatively, the value corresponding to the word is assigned a TF-IDF value instead of 1. The TF-IDF value is obtained by multiplying the TF and IDF values.

2) In a self supervised learning technique (Word2vec) The entire corpus is scanned, and the vector crea-tion process is performed by determining which words the target word occurs with more often[3]. In this way, the semantic ~~cbo~~ closeness of the words to each other is also revealed. For example, let each lett-er in the sequences ...xyAzw..., ...xyBzk... and ....xlCdm... represent a word. In this case word-A will be closer to word_B than word_C. When this situation is taken into account in vector formation, the semantic closeness between words is expressed mathematically.

→ To find the similarities between words there should be features in order to accurately classify them. This cannot be done manually, cannot be hand crafted. Instead, they are learnt during nueral network train-ing. The steps for it are:

(i) Take a fake problem.

(ii) Solve it using nueral network.

(iii) You get word embeddings as a side effect.

Fake Problem: Fill in a missing word in a sentence. This will help us find feature vector

Taking a fake problem is something like predicting the

words which come into the picture.

<u>For example!</u>

NASA launched ________ last month.

[table, Rocket, apple, Pizza].

Meaning of word can be inferred by surrounding words

lets say we have a paragraph of 8-10 sentences

From that we create fake problems like after

kalinga battle, king ordered his. These are missing

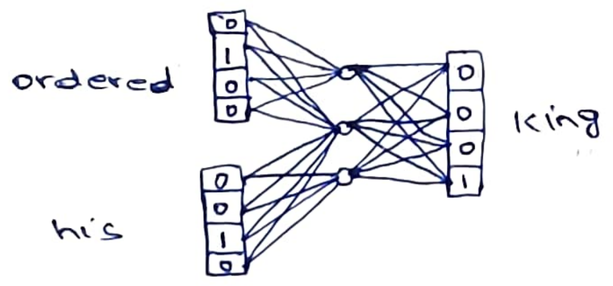
words but inferring to a single meaning. The sente-

-nces in the paragraph would be divided into part

using window sizes for an instance let's take the

Size of 3. then,

King → ordered his



In the above representation the words ordered and

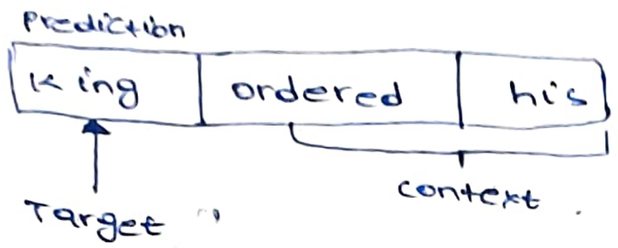
his are one-hot encoded and fed to a nueral net-

-work to find the actual target that is the king

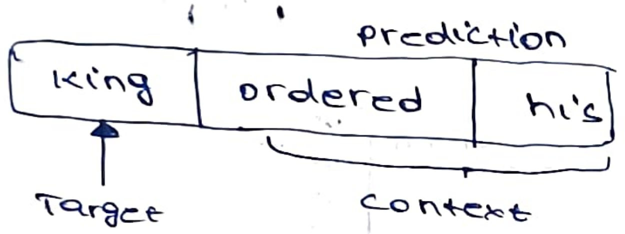
in the vector format. As a side effect of this we

will be getting the embedded word vectors.

This approach is called as continous Bag of words where the target word is Predicted based on the context given like.



The another approach would be skip gram where ba-sed on the target. word we need to predict the context.



To the above context those are fake problems we are not bothered about them, we are interested in the vectors that it creates which is the actual word embedding.

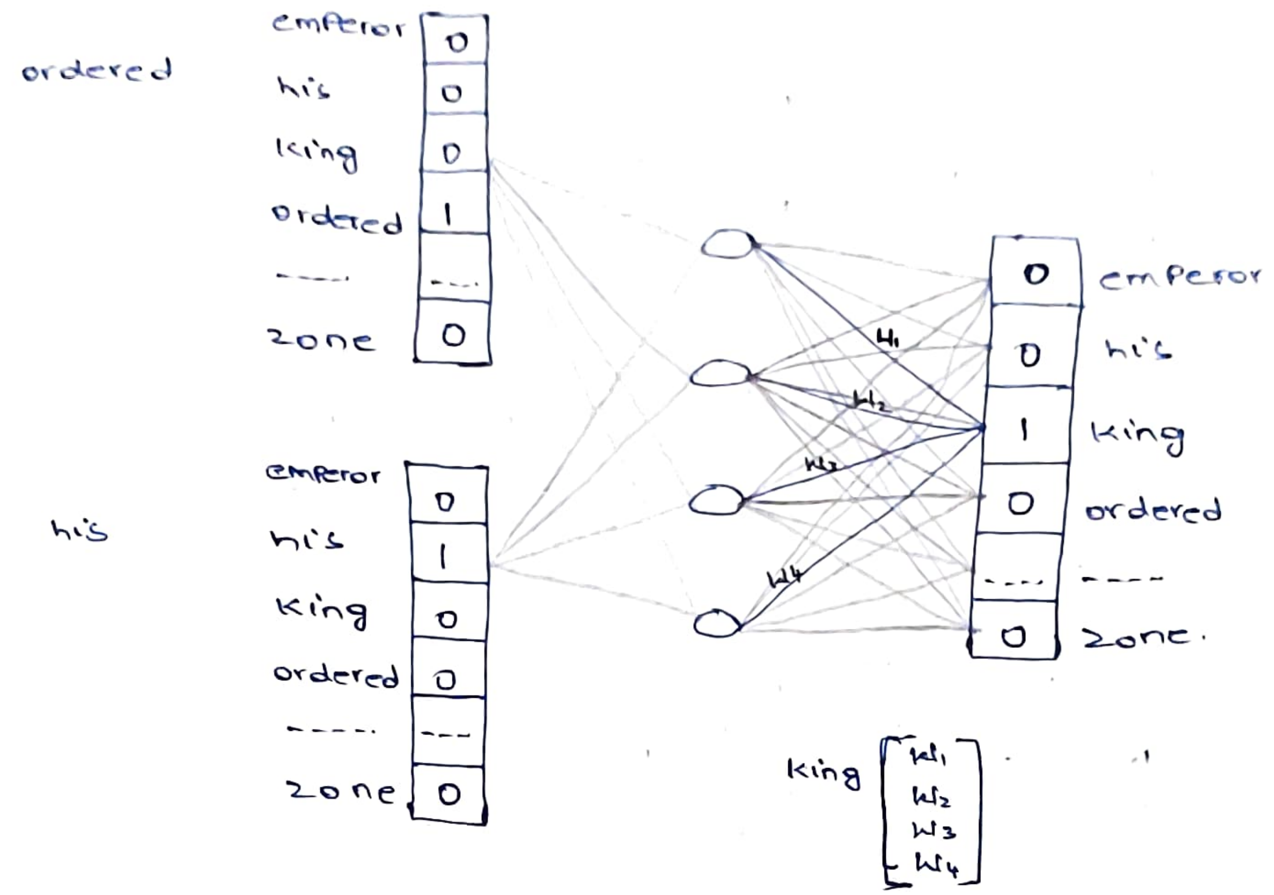
→ Word2vec is a revolutionary method in computer scie-nce where we can convert word to vectors and do mathematics on them. It is a technique which may utilize any one of the above mentioned pro-cess. i.e,

* continous Bag of words.

* Skip Gram

→ Continous Bag of words ~~&~~ & skip gram:

King ← ordered his (King - Man + woman = Queen)



$$\text{king}\begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix}$$

In the above representation we can see that the process of converting words into vectors is following steps in a neural network where one-hot encoded vector is fed into the networth and the output is predicted in that process we acquire the vector for the word "king". In the skip gram process the word "king" will be the input and the corresponding vectors will be extracted between the input and hidden layer. whereas here, it will be extracted from between the hidden layer and output layer.

General flow of the Algorithm (Word2Vec):

Step-1: Initially, we will assign a vector of random n-umbers to each word in the corpus.

Step-2: Then, we will iterate through each word of the document and grab the vectors of the nearest n-words on either side of our target word, and concatenate all these vectors, and then forward propagate these concatenated ve-ctors through a linear layer + softmax functio and try to predict what our target word was.

Step-3: In this step, we will compute the error between our estimate and the actual target word and then back propagate the error and then modifies not only the weights of the linear layer but also the vectors or embeddings of our neighbor's words.

Step-4: Finally, we will extract the weights from the hidden layer and by using these we-ights encode the meaning of words in the vocabulary.

→ Both the mentioned previous models i.e., CBOS and SG are basically shallow neural networks that map words to the target variable which is also a word. These techniques learn the wei-

-ghts that act as word vector representation
Both these techniques can be used to imple
-menting word embedding using word2 vec.

Conclusion:

→ Is it important for our model to represent rar
words? If so, we should choose skip-Gram
since when their vectors are not averaged
with the other background terms in the process
of making the predictions, the model will lea-
-rn better representations for the rare words
and it is better to use Skip-Gram model in
that case.

→ We don't have much time to train and rare
words are not that important for our solution
Then we should choose Continous Bag of wo
-rds model.