WORD EMBEDDING

SHUBHAM NAIK
SOFTWARE ENGINEER | SYNTEL PVT LTD,PUNE
DATA SCIENCE, ML ENTHUSIAST
naikshubham55@gmail.com

Word Embedding

- Before we start lets have a look at below examples:

1. You open google and search for ongoing FIFA WorldCup and get hundreds of search results.
2. Nate Silver analysed millions of tweets and correctly predicted the results of 49 out of 50 states in 2008 U.S Presidential Elections.
3. You type a sentence in google and say translate in chinese, you will get equivalent Chinese conversion.

❖So what do you find common in the above examples?

Text processing

- All of the above scenarios deal with huge amount of text to perform different task like :

1. Clustering in google search example
2. Classification in the second
3. Machine translation in the third

- Humans can do the above task but while millions of documents being generated daily, its not feasible nor effective.
- So , how do we tell computers to perform above tasks such as clustering, classification, etc on text data ,as we know that computers can deal with numbers and not text.

Word Embedding

- Computers can compare two strings and tell us whether they are same or not.
- But how do we make computer tell us about ***football or Ronaldo*** when we search for ***Messi***.
- Suppose we have a sentence “Apple is a tasty fruit”.
- How do we make computers understand that “Apple” in the above sentence is a fruit to be eaten and not a company.
- All these can be achieved by creating a representation of words that capture their *meanings*, *semantic relationship* etc.
- All of these are implemented by using Word Embeddings.

Word Embedding

- Why do we need word embedding?

➢ Many machine learning algorithms and almost all deep learning architectures are incapable of processing strings or plain text in their raw form.

➢ They require numbers as input.

➢ Word Embedding is a technique of converting text into numerical representation.

Different types of Word Embeddings

- Word Embeddings can be classified into below categories

1. One hot encoding
2. Frequency based Embedding
3. Prediction based Embedding

One hot encoding

- Consider below dataset, we need to create a model to predict the heights

Weight	Hair Color	Counrty	Height
75	Black	India	180
65	Brown	US	150
80	Red	Canada	190
77	Golden	UK	175
70	Grey	China	170

- We need to convert the haircolor into numeric values.
- Option 1: Integer encoding – Give a number to each of them.

Black	Brown	Red	Golden	Grey
1	2	3	4	5

Integer encoding

- But numbers have a order. By converting hair colors into numbers we are creating artificial order which is not correct.

e.g 3>2 , but Red > Brown is incorrect.

- Option 2: One hot encoding ,which is a better strategy.

➢ One hot encoding converts categorical features into binary vectors.

One hot encoding

- One hot encoding creates a binary vector which has the size of number of distinct values.
- So we will have vectors like:

➢ Black

Black	Brown	Red	Golden	Grey
1	0	0	0	0

➢ Brown

Black	Brown	Red	Golden	Grey
0	1	0	0	0

➢ Red

Black	Brown	Red	Golden	Grey
0	0	1	0	0

➢ And So on

Disadvantage of one hot encoding

- Country column can have 200 distinct values.
- So the dimension of binary vector will increase, as the dimension of binary vector is the number of distinct values.

Country 1	Country2	Country 3	…….	…….	……	……	…….	……	Country 200

- If the number of distinct values for a categorical feature is large then One hot encoding can create large and sparse vectors.

Imputing Country Column by mean replacement

- Our objective is to predict height.
- Replace country names in country column by average heights of people from that country.
- This way we have converted categorical names into numeric vectors.
- This are problem specific solutions.

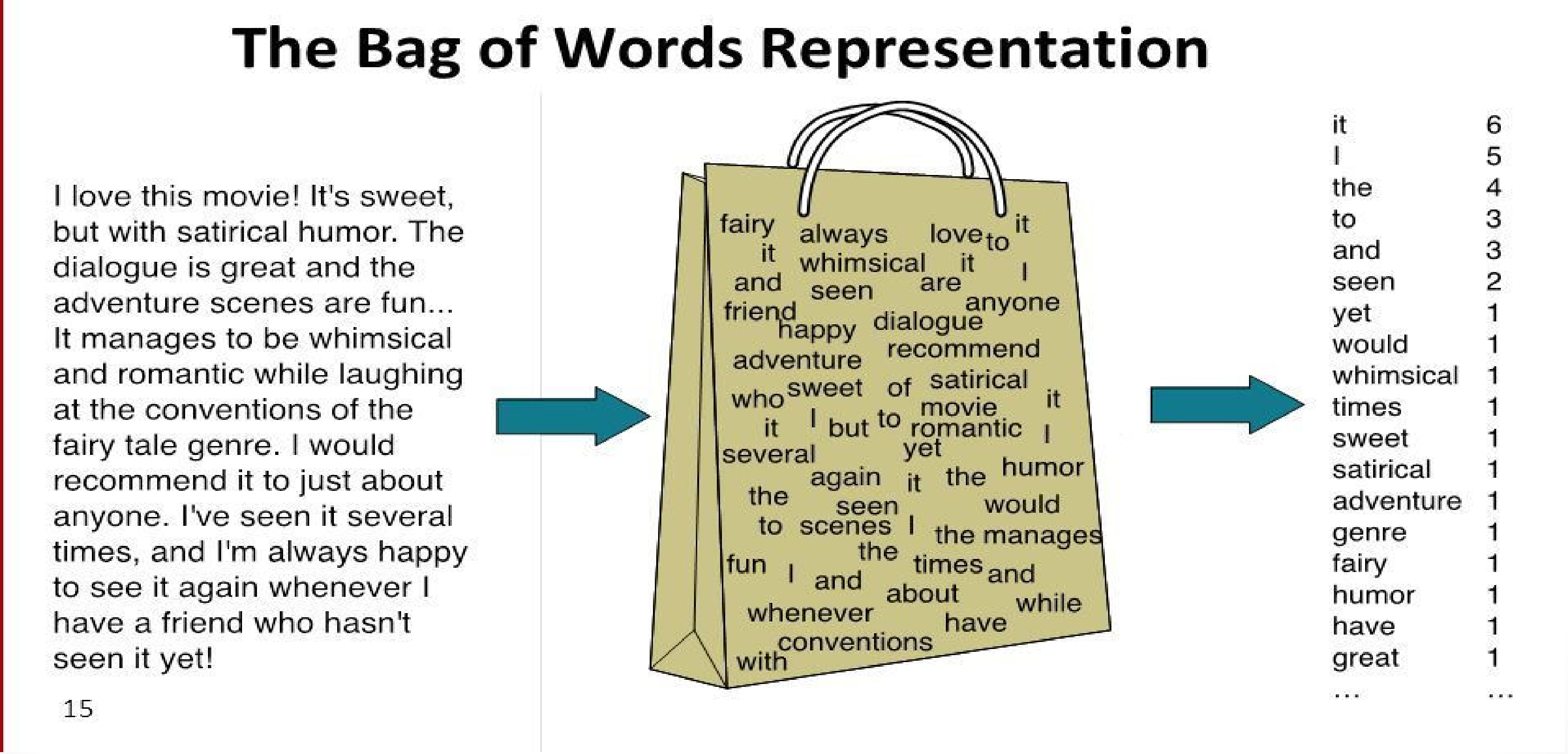
Frequency Based Embedding

- Below are frequency based embedding techniques

1. Bag of Words (Count Vector)
2. TF – IDF
3. Uni-gram/Bi-gram/Tri-gram/n-grams

Bag of Words (BoW)

- BoW is a technique used to convert text to vector.



The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

15

Why convert text to vectors?

- Once text is converted to vector we can leverage the whole power of linear algebra.
- Machine learning algorithms cannot work with raw text directly , the text must be converted to numbers.
- A problem with text is that it is messy and techniques like machine learning algorithms require well defined fixed-length inputs and outputs.

Let’s create a Bag of Words model

Consider below are reviews for a food product(pasta) on amazon.

- Review 1: This pasta is very tasty and affordable.
- Review 2: This pasta is not tasty and affordable.
- Review 3: This pasta is delicious and cheap.
- Review 4: Pasta is tasty and pasta tastes good.

Now task at hand is to classify whether the Review is positive or negative using Machine Learning Algorithm.

Step 1: Design the vocabulary ->Set of all unique words in the Reviews(ignoring punctuations)

- Below will be the Dictionary of all the unique words:

{'affordable', 'very', ‘pasta', 'good', 'tastes', 'tasty', 'not', 'delicious', 'cheap', 'is', 'and', this'}

- Text is called Document in NLP(Natural Language Processing)

Step 2: Create Document Vectors

- Next step is to count the frequency of words in each document(text).
- Objective is to convert each document(text) into vector that we can use as input or output to a machine learning model.
- Below is our **Document Corpus**

affordable	**very**	**pasta**	**good**	**tastes**	**tasty**	**not**	**delicious**	**cheap**	**is**	**and**	**This**

- Now create Document Vector for each document(Review text) by simply marking the frequency of words against the words in document corpus.

Creating Document Vectors.

- Review 1: “This pasta is very tasty and affordable.”

affordable	very	pasta	good	tastes	tasty	not	delicious	cheap	is	and	This
1	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**

- This is how we construct document vector.

“This pasta is very tasty and affordable”= [1,1,1,0,0,1,0,0,0,1,1,1]

“This pasta is not tasty and affordable” = [1,0,1,0,0,1,1,0,0,1,1,1]

“This pasta is delicious and cheap” = [0,0,1,0,0,0,0,1,1,1,1,1]

“Pasta is tasty and pasta tastes good” = [0,0,2,1,1,1,0,0,0,1,1,0]

- Such vectors having lot of zeros are called **Sparse Vectors**.

Similarity of vectors

- Review 1: This pasta is very tasty and affordable.

V1=

affordable	very	pasta	good	tastes	tasty	not	delicious	cheap	is	and	This
1	1	1	0	0	1	0	0	0	1	1	1

- Review 2: This pasta is not tasty and is affordable.

V2=

affordable	very	pasta	good	tastes	tasty	not	delicious	cheap	is	and	This
1	0	1	0	0	1	1	0	0	1	1	1

➢ To obtain similarity of vectors we subtract the two vectors.

Length(V1 – V2) =||V1 – V2||=sqrt(1^2 + 1^2) = sqrt(2)=1.414

- Although the two reviews have completely different meaning(very tasty) and (not tasty) ,semantic English meanings are different, still the distance is between the vectors is very small.

Managing Vocabulary

- As vocabulary size increases ,the vector size also increases.
- If we have thousands of Reviews then our Document Corpus will have thousands or millions of words, and accordingly the length of our document vectors will be thousands or millions.
- We need to decrease the size of vocabulary when using BOW model.
- We can use simple text cleaning techniques such as:

➢ Ignoring case

➢ Ignoring punctuation

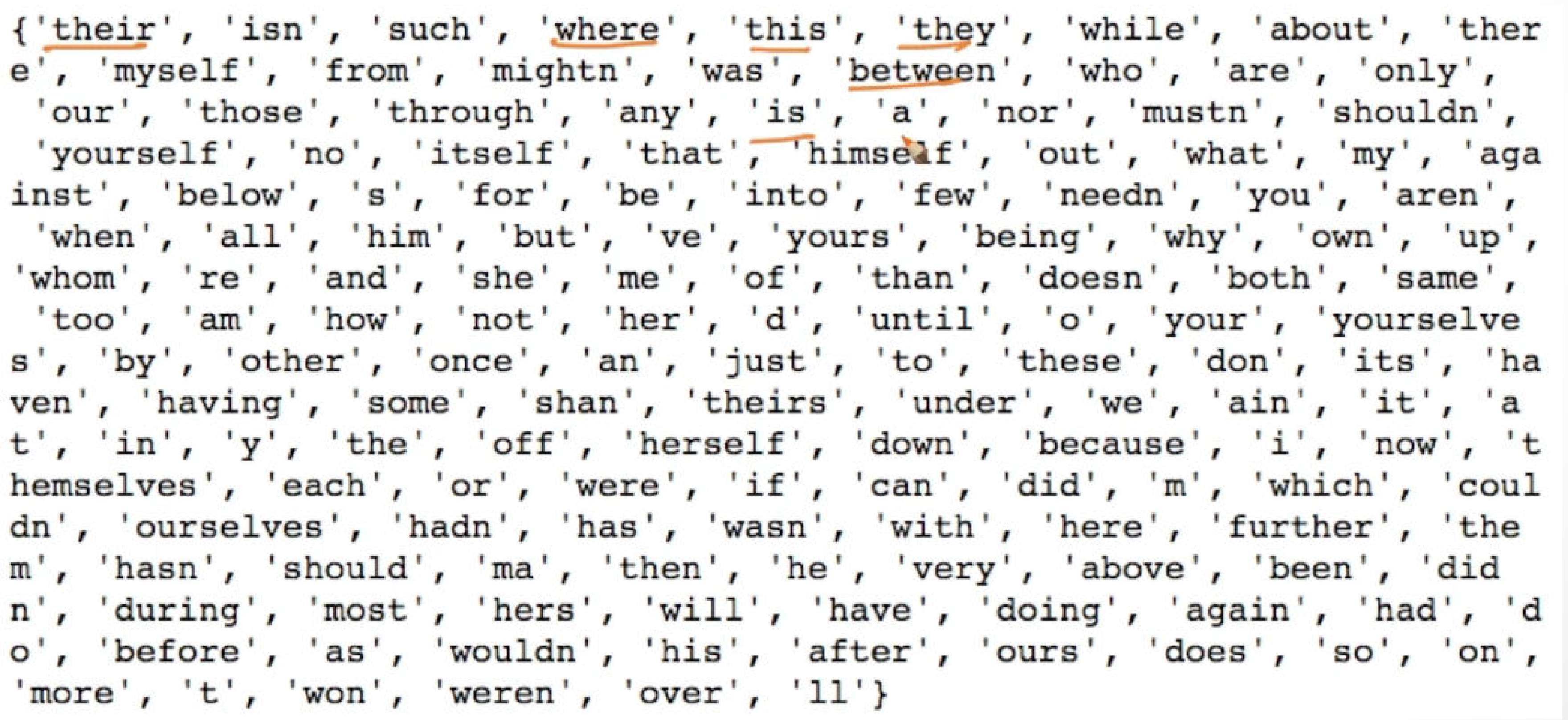
➢ Removing stopwords such as “a” “of” “and” etc

➢ Fixing misspelled words.

➢ Reducing words to their stem (e.g. “play” from “playing”) using stemming algorithms.

Stopwords in English

- If we remove stopwords ,we wil have smaller and meaningful vectors.



```
{'their', 'isn', 'such', 'where', 'this', 'they', 'while', 'about', 'ther
e', 'myself', 'from', 'mightn', 'was', 'between', 'who', 'are', 'only',
 'our', 'those', 'through', 'any', 'is', 'a', 'nor', 'mustn', 'shouldn',
 'yourself', 'no', 'itself', 'that', 'himself', 'out', 'what', 'my', 'aga
inst', 'below', 's', 'for', 'be', 'into', 'few', 'needn', 'you', 'aren',
 'when', 'all', 'him', 'but', 've', 'yours', 'being', 'why', 'own', 'up',
'whom', 're', 'and', 'she', 'me', 'of', 'than', 'doesn', 'both', 'same',
 'too', 'am', 'how', 'not', 'her', 'd', 'until', 'o', 'your', 'yourselve
s', 'by', 'other', 'once', 'an', 'just', 'to', 'these', 'don', 'its', 'ha
ven', 'having', 'some', 'shan', 'theirs', 'under', 'we', 'ain', 'it', 'a
t', 'in', 'y', 'the', 'off', 'herself', 'down', 'because', 'i', 'now', 't
hemselves', 'each', 'or', 'were', 'if', 'can', 'did', 'm', 'which', 'coul
dn', 'ourselves', 'hadn', 'has', 'wasn', 'with', 'here', 'further', 'the
m', 'hasn', 'should', 'ma', 'then', 'he', 'very', 'above', 'been', 'did
n', 'during', 'most', 'hers', 'will', 'have', 'doing', 'again', 'had', 'd
o', 'before', 'as', 'wouldn', 'his', 'after', 'ours', 'does', 'so', 'on',
'more', 't', 'won', 'weren', 'over', 'll'}
```

Uni –Gram/Bi-gram/n-gram

- We saw that (not,is,and,this,very) are stopwords and if we remove this stopwords from our Reviews then reviews will be exactly the same.
- Review 1: This pasta is very tasty and affordable.
- Review 2: This pasta is not tasty and is affordable.
- After removing stopwords vectors V1 and V2 are exactly the same, which is not true.
- To overcome this we use Bi-gram and Tri-gram.

Uni –gram ,Bi-grams

- In uni-gram each word is considered a dimension.
- In Bi-grams pairs of consecutive words are considered a dimension.
- Our Document Corpus for Bi-grams will be:

this pasta	pasta is	is not	not tasty	is very	very tasty	tasty and	and is	is affordable	and affordable

- Review 1: “This pasta is very tasty and affordable.”

this pasta	pasta is	is not	not tasty	is very	very tasty	tasty and	and is	is affordable	and affordable
1	1	0	0	1	1	1	0	0	1

- This way we can maintain the semantics of the sentence.

Tri-grams and n-grams

- In Tri-gram we take 3 consecutive words.
- In n-gram we take n –consecutive words.
- Advantage:

➢It retains the sequence information.

- Disadvantage:

➢ is that the number of dimensions increases with increase in n.

- No of tri-grams > No of bi-grams > No of uni-grams

Code for BOW

- Lets write and understand a code for BOW.

TF – IDF

- TF – term frequency
- IDF – Inverse Document Frequency
- Consider below BOW representation:

R1 : w1,w2,w3,w2,w5

W1	W2	W3	W4	W5	W6
1	2	1	0	1	0

R2 : w1,w3,w4,w5,w6,w2

W1	W2	W3	W4	W5	W6
1	1	1	1	1	1

Term frequency

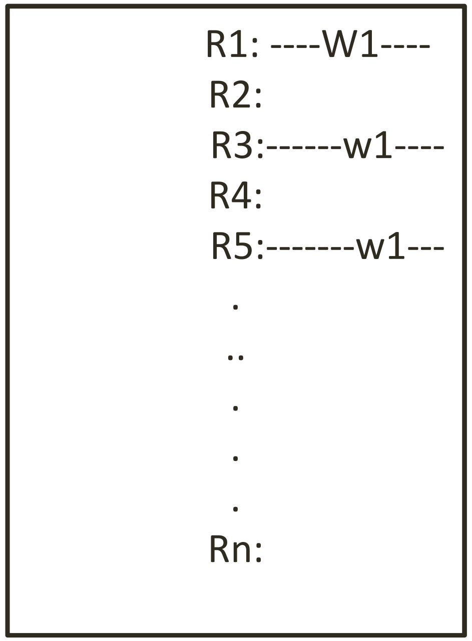
- Term frequency is always computed for a word and a given document.

- $$\mathrm{TF}(Wj,Di) = \frac{\text{(no of times Wj occurs in document Di)}}{\text{(no of words in Document Di)}}$$

- TF(W2,D1) = 2/5 = 0.4

Inverse Document Frequency (IDF)

- D = Document corpus (Contains all the documents)
- Document corpus



R1: ----W1----
R2:
R3:------w1----
R4:
R5:-------w1---
.
..
.
.
.
Rn:

IDF formula

- $\text{IDF(Wj , D)} = \log\left\{\frac{\text{(no of documents in D)}}{\text{(no of documents in D that contains Wj)}}\right\}$
- If Wj is frequent word ,then its IDF will be low.
- If Wj is rare word, then its IDF will be high.
- As per research papers log is taken in relation to Zipf’s law, which is a part of research.

TF-IDF vector

- Product of TF & IDF of word is used in the vector.
- W1 = TF(W1,D1) * IDF(W1,D)
- Similarlily ,we calculate TF-IDF values for all words and form a vector.

W1	W2	W3	W4	W5	W6
TF(W1,D1) * IDF(W1,D)	TF(W2,D1) * IDF(W2,D)	TF(W3,D1) * IDF(W3,D)	TF(W4,D1) * IDF(W4,D)	TF(W5,D1) * IDF(W5,D)	TF(W6,D1) * IDF(W6,D)

- In BOW we insert number of occurrences of Word(Wi) in Document(Di).

TF-iDF Pros and Cons

- Advantages:

➢ We give more importance to rarer words in Document corpus.

➢ Also we give more importance to frequent words to in a Document.

- Disadvantages:

➢ TF-IDF still doesn’t take the semantic meaning of words.

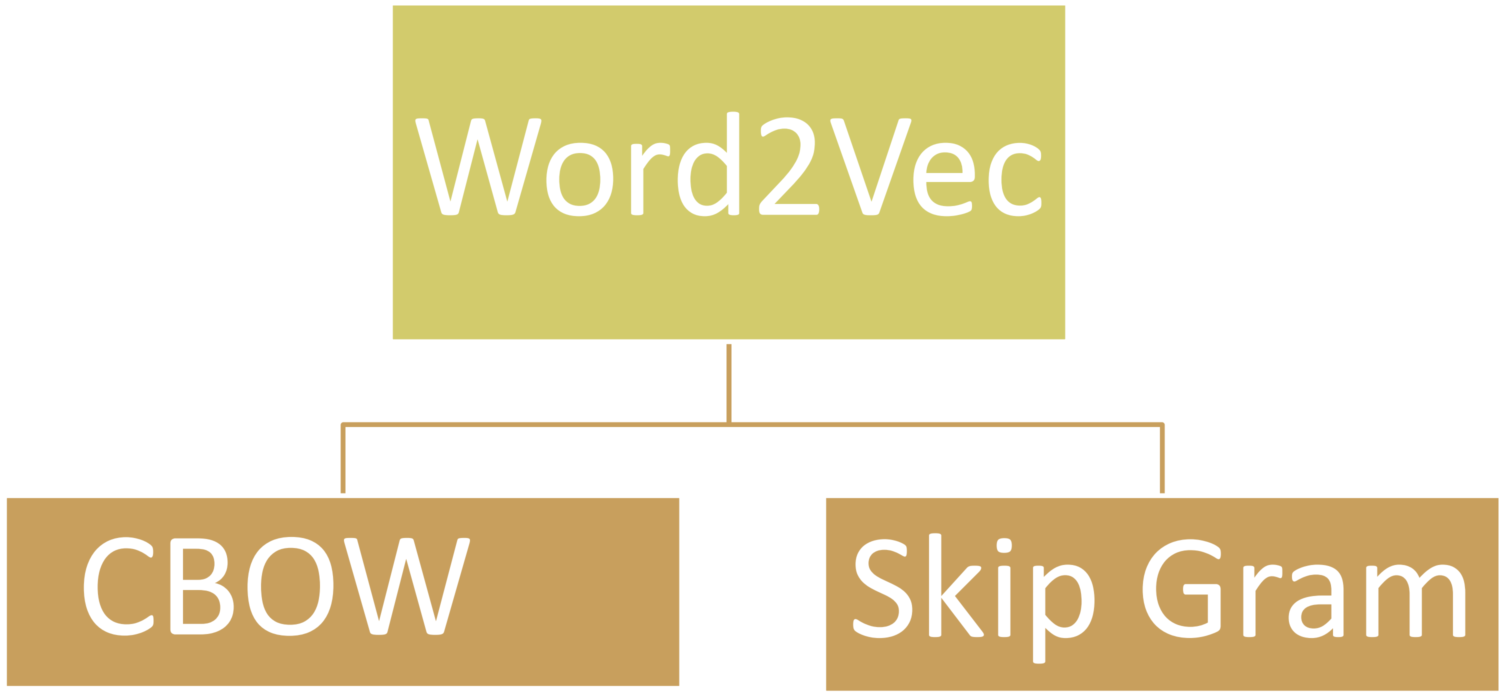
Code for TF - IDF

- Let’s understand TF – IDF with the help of a code.

Prediction based Embedding

Word2Vec

- Word2Vec is a Neural Network structure.
- It was developed by Tomas Mikolov at Google in 2013.
- There are two algorithms to achieve Word2vec



Word2Vec

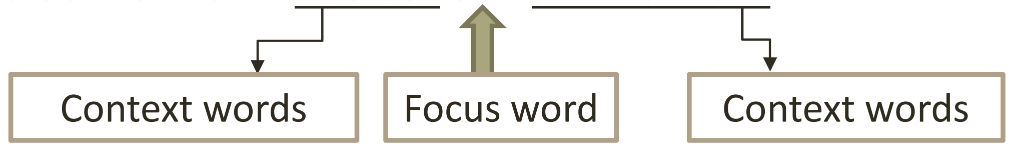
- Word2Vec is a text to vector conversion technique which takes semantic meaning of words into consideration.
- BOW and TF-IDF doesn't take semantic meaning of words into consideration.
- Word2Vec converts a word into d-dimensional vector, d is typically 50,100,200,300.
- Suppose we have 3 words:
- W1 = tasty ,W2 =delicious,W3=baseball
- If we create vectors for this words using Word2Vec

W1 → V1 , W2→ V2 , W3 → V3

- Since Word2Vec takes semantic meanings of words into consideration vectors V1 and V2 will be closer as compared to V1 and V3 or V2 and V3.

Continous Bag of Words (CBOW)

- Consider a sentence : “The cat sat on the wall”



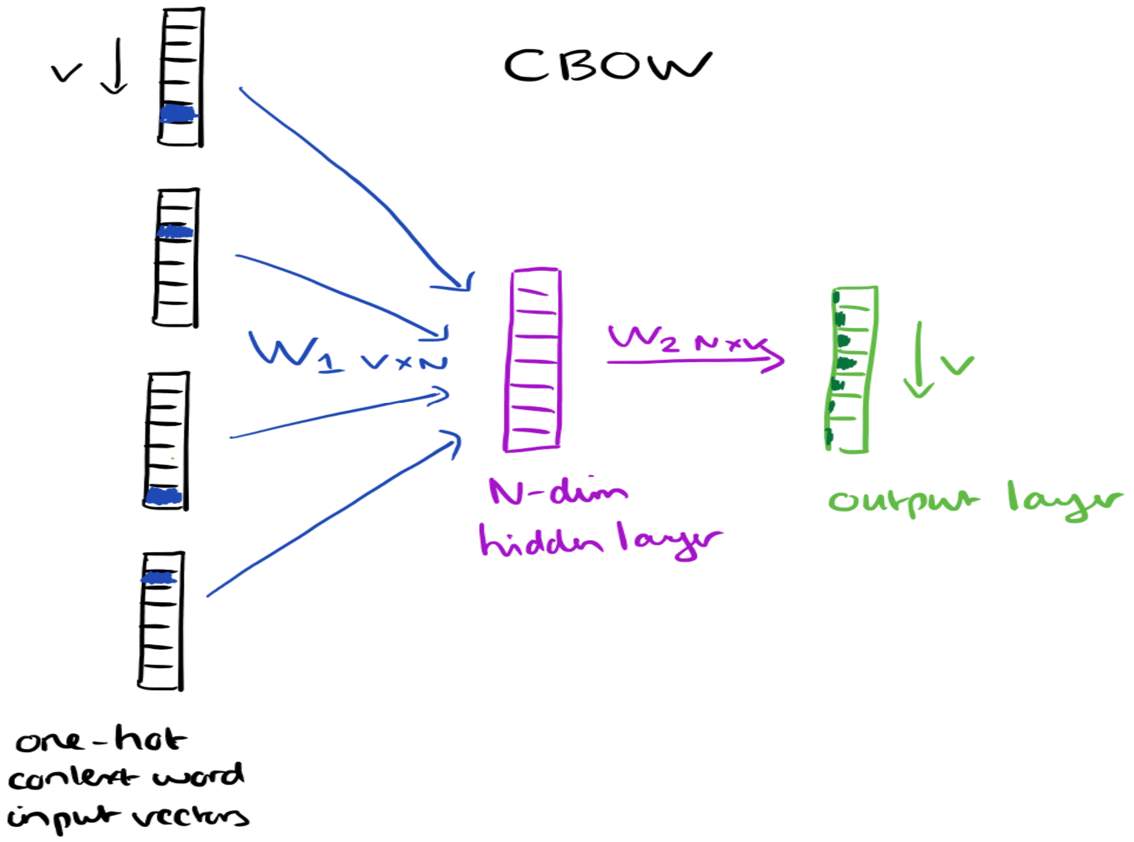
- If “sat” is a focus word ,the words surrounding focus word becomes context words.
- Context words are very useful in understanding focus words and vice versa.
- For CBOW create a vocabulary of all the words. Suppose V is the size of vocabulary.
- Suppose we use One hot encoding for each word.
- Then Wi will be, Wi: binary vector of V dimensions as we have total V words in our vocabulary.

CBOW structure

- CBOW consists of :

1. Input layer which consists of V- dimensional, One hot Encoded Vectors
2. N-dimensional Hiden Layer, where N can be 50,100,200,300 etc
3. Output layer: which predicts the focus word.

CBOW



V
W_1 V×N
W_2 N×V
V
N-dim hidden layer
output layer
one-hot context word input vectors

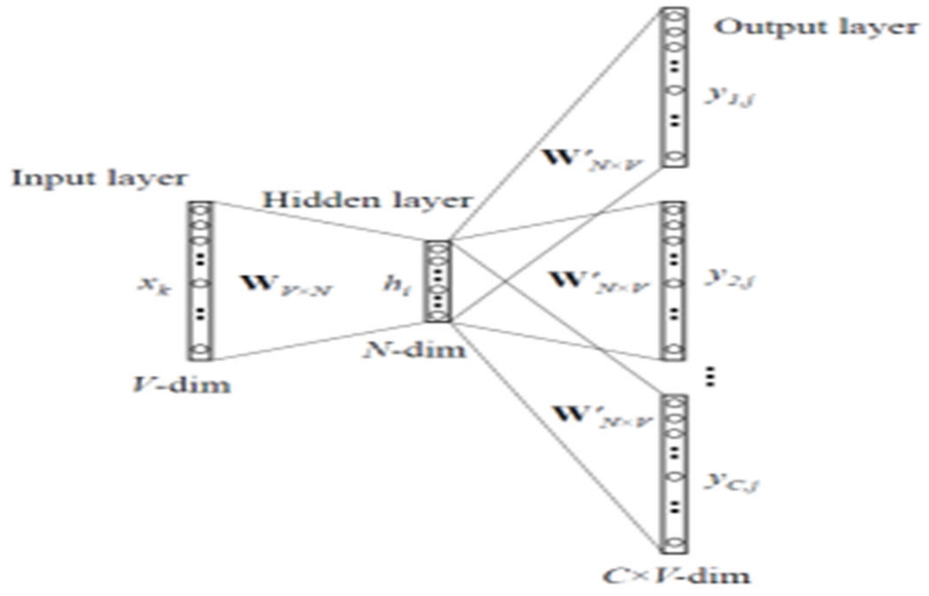
- The core idea of CBOW is given context words ,it should predict the focus word.
- We need to Train this CBOW:
- Given repository of text create a dataset of focus words and context words.
- Text : “The cat sat on the wall”

Focus Words	Context Words
Sat	The,cat,on,the,wall
Cat	The,sat,on,the,wall
on	The,cat,sat,the,wall
The	Cat,sat,wall
Wall	The,cat,sat,on,the

- We can then train the CBOW Neural Network using this dataset.

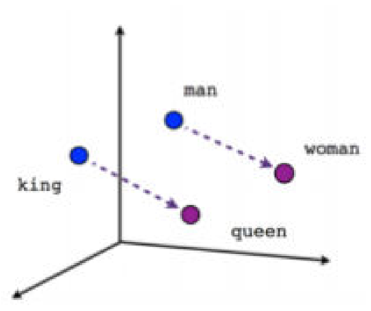
Skip Gram

- Skipgram is just opposite to CBOW, in Skipgram we predict the context words given a focus word.
- Here our input is a focus word and outputs are context words:

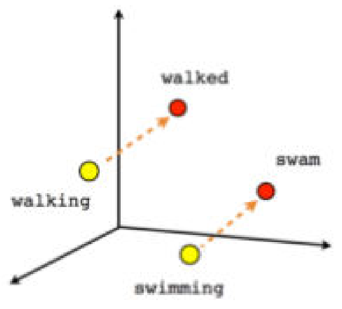


Word2Vec Features

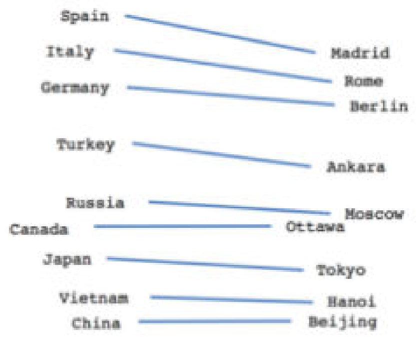
- If W1 and W2 are semantically similar then vectors V1 and V2 will be closer.
- Word2Vec studies relationships,Word2Vec understands the relationship of Male gender word and female gender word, which means (Vman – Vwoman) || (Vking – Vqueen)



Male-Female



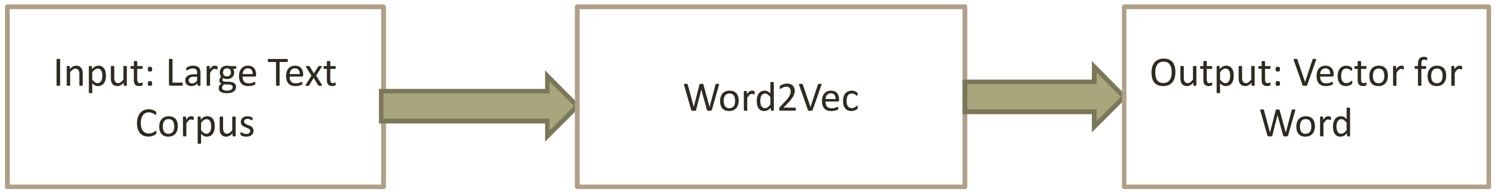
Verb tense



Country-Capital

Word2Vec Functionality

- Word2Vec learns all this relationships without being programmed to do so. It learns it automatically from Raw Text.



- Vector can be chosen to be 50,100,200,300 dimensional.
- Larger the dimensionality → More information rich the vector is.

Using pre-trained word vectors

- We will use google’s pretrained model.
- It contains vectors for a vocabulary of 3 million words trained on around 100 billions words from the google news dataset.
- Let’s see the code for this.

Training own Word2Vec

- We will be training our own Word2Vec on a custom corpus.
- For training the model we will be using genism.
- Lets see the code.