CO**NEXT**S
Data as a Service (DaaS)

NLP Training for Beginners

Session-2

05 October 2021

NOVARTIS | Reimagining Medicine

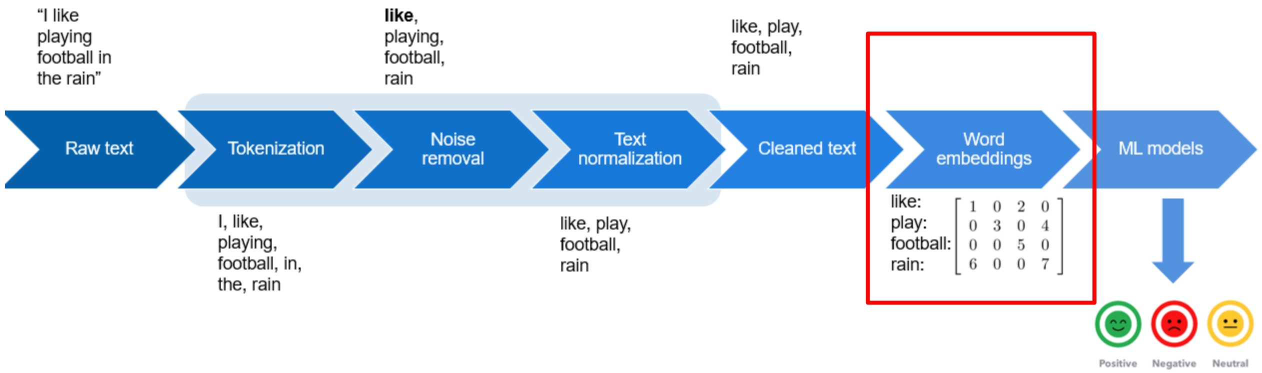
Previous session

1. Introduction to NLP
2. Applications of NLP
3. Tools and Python Libraries
4. Text Preprocessing
   – Tokenization
   – Noise Removal (stop words)
   – Stemming/Lemmatization
   – POS tagging

Agenda

1. Count Vectorizer - Frequency models
2. Unigram, Bigram, Trigram, N-grams
3. TF-IDF
4. Sneak-peak into next session

Feature extraction



	MARY	IS	HUNGRY	HAPPY	FOR	APPLES	NOT	JOHN	HE	
"Mary is hungry for apples."	1	1	1	0	1	1	0	0	0	[1, 1, 1, 0, 1, 1, 0, 0, 0]
"John is happy he is not hungry for apples."	0	2	1	1	1	1	1	1	1	[0, 2, 1, 1, 1, 1, 1, 1, 1]

Count Vectorization – Sparse Matrix representation

Feature extraction

- Consider a corpus of positive and negative tweets

Positive tweets

I am happy because I am learning NLP

I am happy

Negative tweets

I am sad, I am not learning NLP

I am sad

Vocabulary	PosFreq (1)	NegFreq (0)
I	3	3
am	3	3
happy	2	0
because	1	0
learning	1	1
NLP	1	1
sad	0	2
not	0	1

freqs: dictionary mapping from (word, class) to frequency

Creating a vector

Vocabulary	PosFreq (1)
I	3
am	3
happy	2
because	1
learning	1
NLP	1
sad	0
not	0

I am sad, I am not learning NLP

$$X_m = [1, \sum_w freqs(w,1), \sum_w freqs(w,0)]$$

$$\boxed{8}$$

Vocabulary	NegFreq (0)
I	3
am	3
happy	0
because	0
learning	1
NLP	1
sad	2
not	1

I am sad, I am not learning NLP

$$X_m = [1, \sum_w freqs(w,1), \sum_w freqs(w,0)]$$

$$\boxed{11}$$

This gives us the feature vector [1, 8, 11]

Count vectorizer algorithm creates BOW model

Bag of Words (BOW)

- Review 1: This movie is very scary and long
- Review 2: This movie is not scary and is slow
- Review 3: This movie is spooky and good
- Vocabulary consists of these 11 words: ‘This’, ‘movie’, ‘is’, ‘very’, ‘scary’, ‘and’, ‘long’, ‘not’, ‘slow’, ‘spooky’, ‘good’.

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

Vector of Review 1: [1 1 1 1 1 1 1 0 0 0 0]
Vector of Review 2: [1 1 2 0 0 1 1 0 1 0 0]
Vector of Review 3: [1 1 1 0 0 0 1 0 0 1 1]

Welcome to the Matrix

NOVARTIS | Reimagining Medicine

Term Frequency – Inverse Document Frequency (TF-IDF)

- TF-IDF tells us, how important a word is to a document in a corpus.

- **Term Frequency**

  - *Consider* ***Review 2: This movie is not scary and is slow***
  - Vocabulary**:** ‘This’, ‘movie’, ‘is’, ‘very’, ‘scary’, ‘and’, ‘long’, ‘not’, ‘slow’, ‘spooky’, ‘good’
  - Number of words in Review 2 = 8
  - TF for the word ‘this’ = (number of times ‘this’ appears in review 2)/(number of terms in review 2) = 1/8

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

Term	Review 1	Review 2	Review 3	TF (Review 1)	TF (Review 2)	TF (Review 3)
This	1	1	1	1/7	1/8	1/6
movie	1	1	1	1/7	1/8	1/6
is	1	2	1	1/7	1/4	1/6
very	1	0	0	1/7	0	0
scary	1	1	0	1/7	1/8	0
and	1	1	1	1/7	1/8	1/6
long	1	0	0	1/7	0	0
not	0	1	0	0	1/8	0
slow	0	1	0	0	1/8	0
spooky	0	0	1	0	0	1/6
good	0	0	1	0	0	1/6

Term Frequency – Inverse Document Frequency (TF-IDF)

- IDF is a measure of how important a term is
- **Inverse Document Frequency**

$$idf_t = \log \frac{number\ of\ documents}{number\ of\ documents\ with\ term\ 't'}$$

  - *Consider **Review 2: This movie is not scary and is slow***
  - IDF('this') = log(number of documents/number of documents containing the word 'this') = log(3/3) = log(1) = 0
- Similarly,
  - IDF('movie',) = log(3/3) = 0
  - IDF('is') = log(3/3) = 0
  - IDF('not') = log(3/1) = log(3) = 0.48
  - IDF('scary') = log(3/2) = 0.18
  - IDF('and') = log(3/3) = 0
  - IDF('slow') = log(3/1) = 0.48

Term	Review 1	Review 2	Review 3	IDF
This	1	1	1	0.00
movie	1	1	1	0.00
is	1	2	1	0.00
very	1	0	0	0.48
scary	1	1	0	0.18
and	1	1	1	0.00
long	1	0	0	0.48
not	0	1	0	0.48
slow	0	1	0	0.48
spooky	0	0	1	0.48
good	0	0	1	0.48

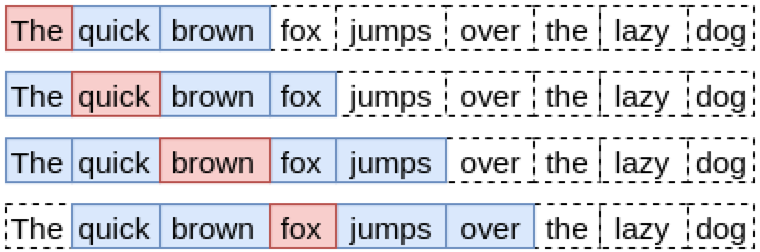
Term Frequency – Inverse Document Frequency (TF-IDF)

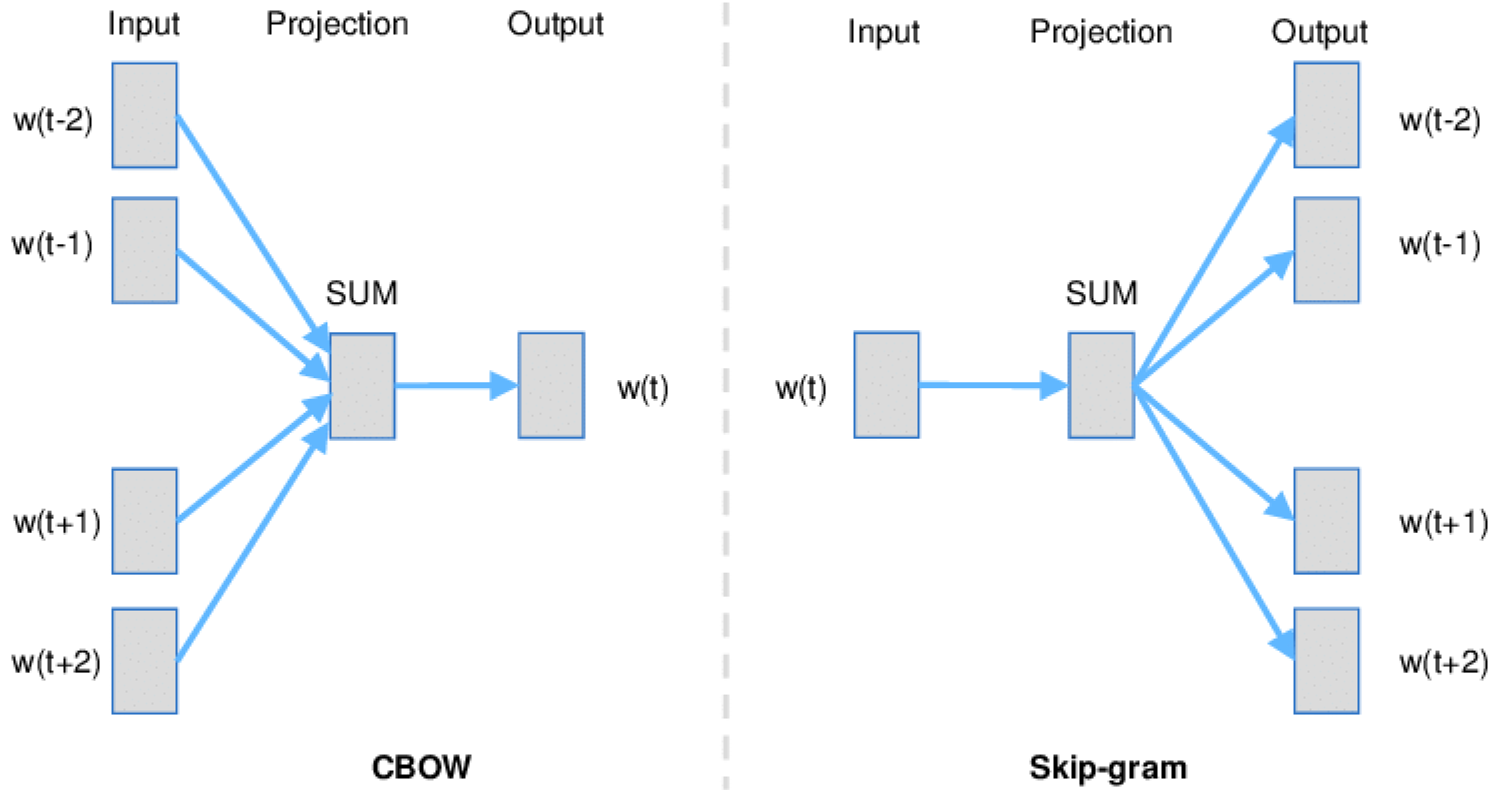
- We can now calculate the TF-IDF score for every word in Review 2:
  - TF-IDF(‘this’, Review 2) = TF(‘this’, Review 2) * IDF(‘this’) = 1/8 * 0 = 0
- Similarly,
  - TF-IDF(‘movie’, Review 2) = 1/8 * 0 = 0
  - TF-IDF(‘is’, Review 2) = 1/4 * 0 = 0
  - TF-IDF(‘not’, Review 2) = 1/8 * 0.48 = 0.06
  - TF-IDF(‘scary’, Review 2) = 1/8 * 0.18 = 0.023
  - TF-IDF(‘and’, Review 2) = 1/8 * 0 = 0
  - TF-IDF(‘slow’, Review 2) = 1/8 * 0.48 = 0.06

$$(tf\_idf)_{t,d} = tf_{t,d} * idf_t$$

Term	Review 1	Review 2	Review 3	IDF	TF-IDF (Review 1)	TF-IDF (Review 2)	TF-IDF (Review 3)
This	1	1	1	0.00	0.000	0.000	0.000
movie	1	1	1	0.00	0.000	0.000	0.000
is	1	2	1	0.00	0.000	0.000	0.000
very	1	0	0	0.48	0.068	0.000	0.000
scary	1	1	0	0.18	0.025	0.022	0.000
and	1	1	1	0.00	0.000	0.000	0.000
long	1	0	0	0.48	0.068	0.000	0.000
not	0	1	0	0.48	0.000	0.060	0.000
slow	0	1	0	0.48	0.000	0.060	0.000
spooky	0	0	1	0.48	0.000	0.000	0.080
good	0	0	1	0.48	0.000	0.000	0.080

CBOW and Skip-gram



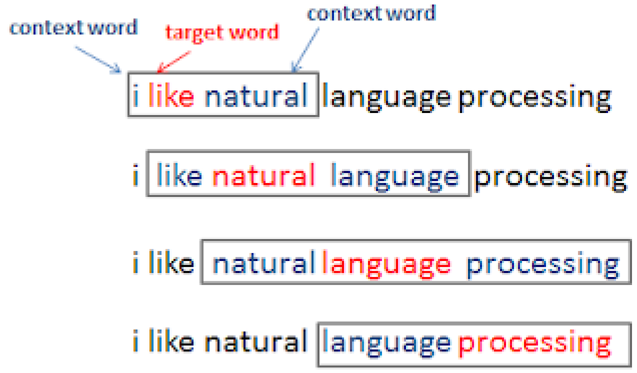


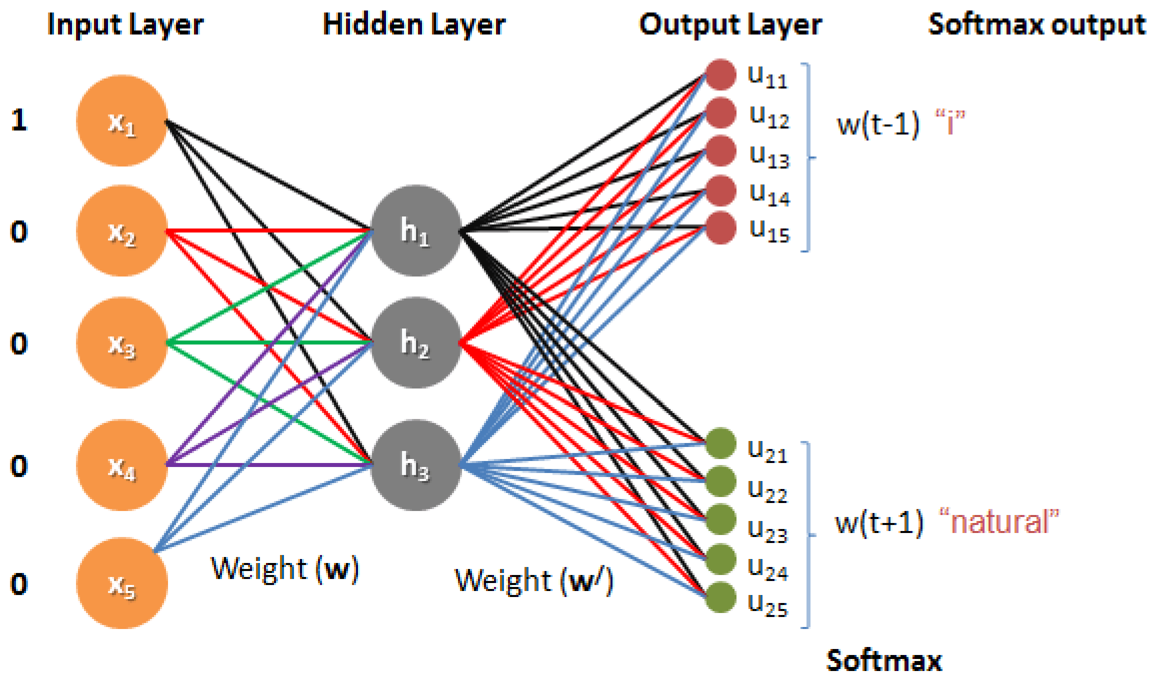
Skip-gram

Training Example	Context Word	Target Word
#1	(i, natural)	like
#2	(like, language)	natural
#3	(natural, processing)	language
#4	(language)	processing

	i	like	natural	language	processing
i	1	0	0	0	0
like	0	1	0	0	0
natural	0	0	1	0	0
language	0	0	0	1	0
processing	0	0	0	0	1

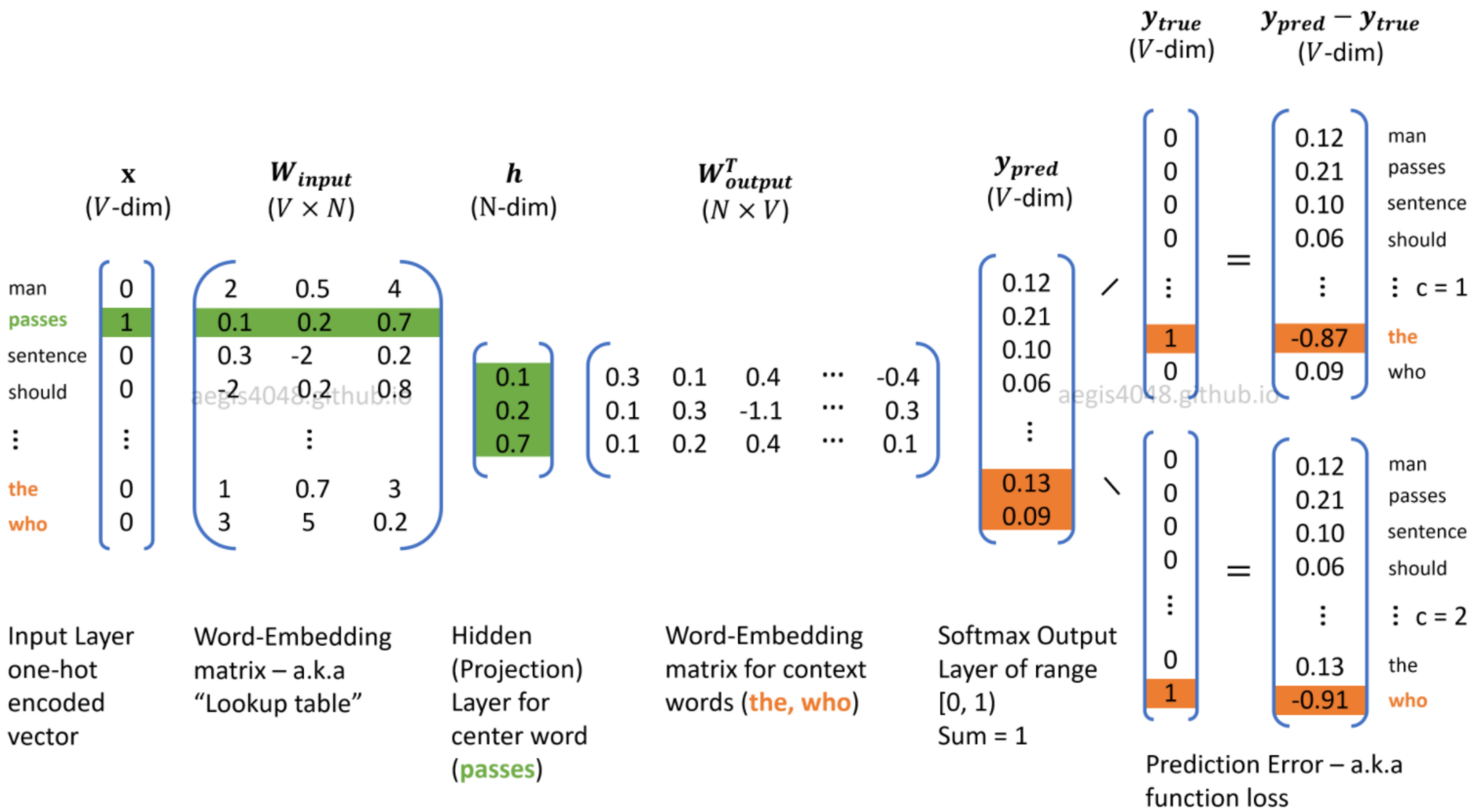
Training Example	Encoded Context Word	Encoded Target Word
#1	([1,0,0,0,0], [0,0,1,0,0])	[0,1,0,0,0]
#2	([0,1,0,0,0], [0,0,0,1,0])	[0,0,1,0,0]
#3	([0,0,1,0,0], [0,0,0,0,1])	[0,0,0,1,0]
#4	([0,0,0,1,0])	[0,0,0,0,1]





First training data point: The context words are "**i**" and "**natural**" and the target word is "**like**".

Skip-gram training



Source: https://aegis4048.github.io/demystifying_neural_network_in_skip_gram_language_modeling

Unigram, Bigram, Trigram (N-grams)

- **Sentence**: “Who let the dogs out”
- **Unigram**: [who, let, the dogs, out]
- **Bigram**: [(who, let), (let, the), (the, dogs), (dogs, out)]
- **Trigram**: [(who, let, the), (let, the, dogs), (the, dogs, out)]

Who let the dogs out

$$p(w1...ws) = p(w1) . p(w2 \mid w1) . p(w3 \mid w1\ w2) . p(w4 \mid w1\ w2\ w3) ..... p(wn \mid w1...wn\text{-}1)$$

- N-grams model works on the principle of Chain rule of probability or conditional probability.
- To simplify we will use the **Markov** assumption

$$p(wk \mid w1...wk\text{-}1) = p(wk \mid wk\text{-}1)$$

Unigram, Bigram, Trigram (N-grams)

- Machine Learning models are not just trained on unigrams, they are trained on bigrams and n-grams to get more accurate vector space representations.
- Words like:

1. San Francisco
2. New Delhi
3. Air Conditioner
4. Big Data Analytics

Need to be captured together, than as a unigram token.

Sneak-peak into next session

- Word Embeddings - SPACY
- Topic Modelling
- Text Matching – Fuzzy
- Coreference resolution

Let's write some code...!