words to GPT

natural language processing - 01

How NLP Course works

How NLP Course works

What should we do first?

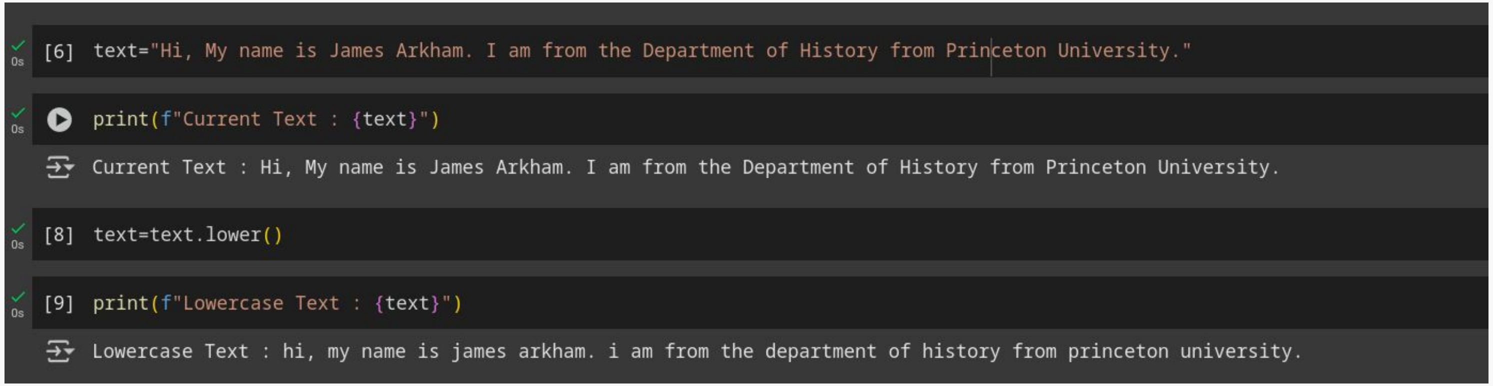
Let’s preprocess some text!

Let’s preprocess some text!

Lowercasing

Let’s preprocess some text!

Lowercasing



```
[6] text="Hi, My name is James Arkham. I am from the Department of History from Princeton University."

    print(f"Current Text : {text}")

    Current Text : Hi, My name is James Arkham. I am from the Department of History from Princeton University.

[8] text=text.lower()

[9] print(f"Lowercase Text : {text}")

    Lowercase Text : hi, my name is james arkham. i am from the department of history from princeton university.
```

Let’s preprocess some text!

Tokenization

Let’s say we have the sentence earlier.

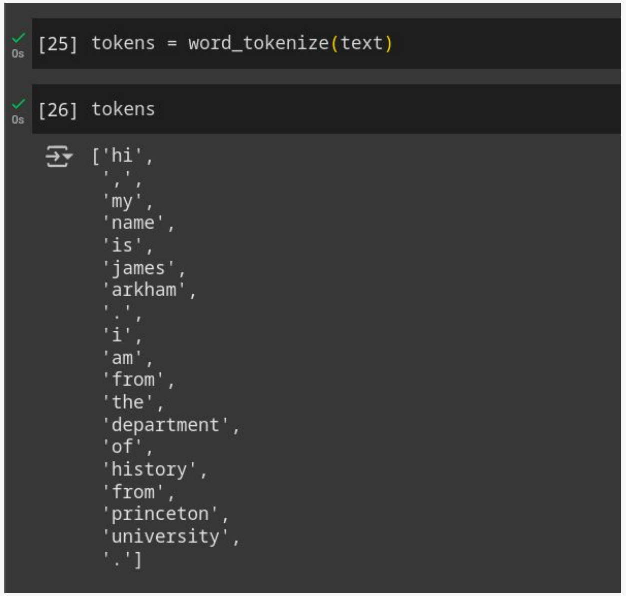
Let’s preprocess some text!

Tokenization

Let’s say we have the sentence earlier.
We can break it down to singular words . This is word tokenization, which breaks long sentences into individual words.

Let’s preprocess some text!

Tokenization



```
[25] tokens = word_tokenize(text)
```

```
[26] tokens
```

```
['hi',
 ',',
 'my',
 'name',
 'is',
 'james',
 'arkham',
 '.',
 'i',
 'am',
 'from',
 'the',
 'department',
 'of',
 'history',
 'from',
 'princeton',
 'university',
 '.']
```

Let’s preprocess some text!

Stopword Removal

What is a stopword?

Let’s preprocess some text!

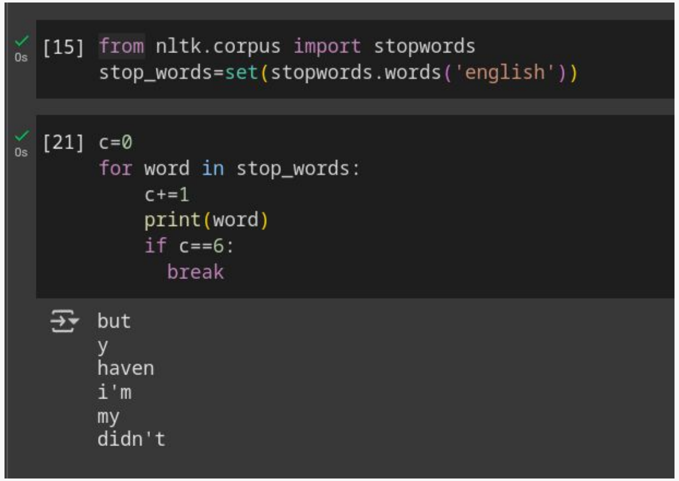
Stopword Removal

What is a **stopword**?

They are usually the words that don’t contribute any meaning or context to the sentence.

Let’s preprocess some text!

Stopword Removal



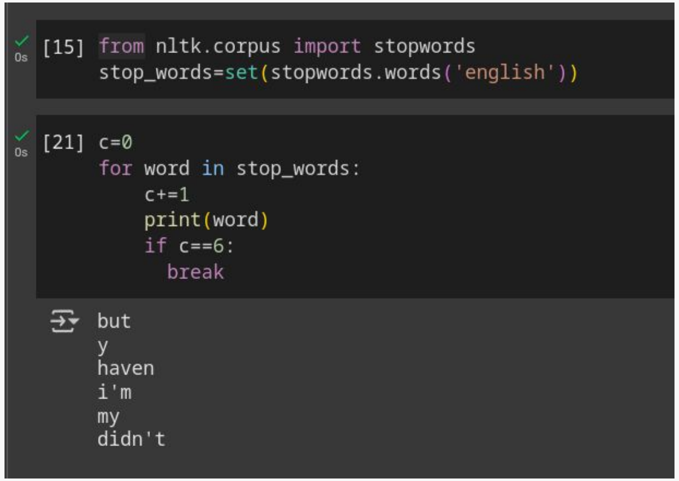
```
[15] from nltk.corpus import stopwords
     stop_words=set(stopwords.words('english'))
```

```
[21] c=0
     for word in stop_words:
         c+=1
         print(word)
         if c==6:
           break
```

```
but
y
haven
i'm
my
didn't
```

Let’s preprocess some text!

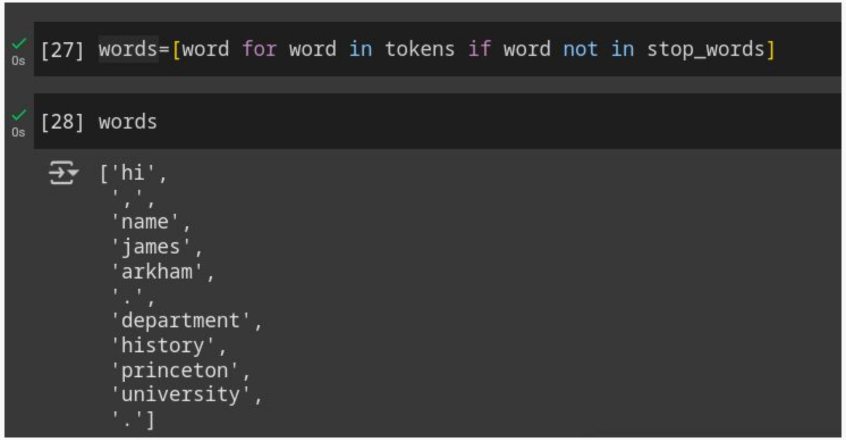
Stopword Removal



```
[15] from nltk.corpus import stopwords
     stop_words=set(stopwords.words('english'))

[21] c=0
     for word in stop_words:
         c+=1
         print(word)
         if c==6:
           break

     but
     y
     haven
     i'm
     my
     didn't
```



```
[27] words=[word for word in tokens if word not in stop_words]

[28] words

     ['hi',
      ',',
      'name',
      'james',
      'arkham',
      '.',
      'department',
      'history',
      'princeton',
      'university',
      '.']
```

Let’s preprocess some text!

Stemming

Let’s preprocess some text!

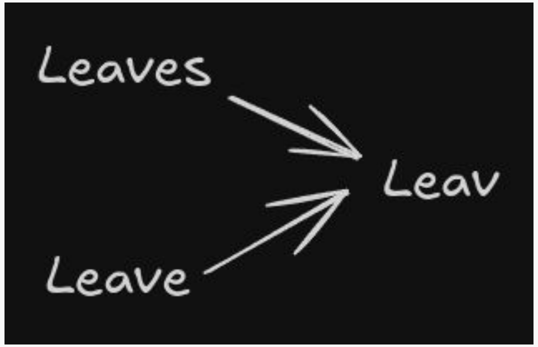
Stemming

With stemming, what we do is chop down the sentence into their root forms, which might not make any sense and lose the meaning of the word.

Let’s preprocess some text!

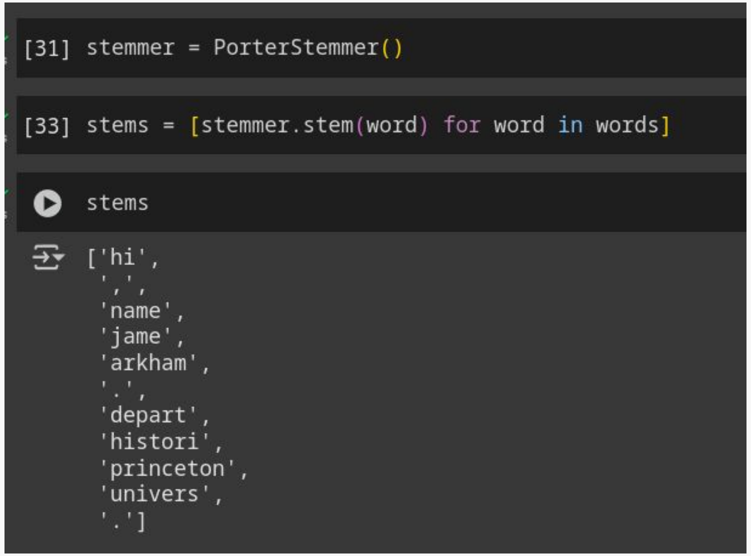
Stemming

With stemming, what we do is chop down the sentence into their root forms, which might not make any sense and lose the meaning of the word.



Let’s preprocess some text!

Stemming



```
[31] stemmer = PorterStemmer()
```

```
[33] stems = [stemmer.stem(word) for word in words]
```

```
stems
```

```
['hi',
 ',',
 'name',
 'jame',
 'arkham',
 '.',
 'depart',
 'histori',
 'princeton',
 'univers',
 '.']
```

Let’s preprocess some text!

Lemmatization

Let’s preprocess some text!

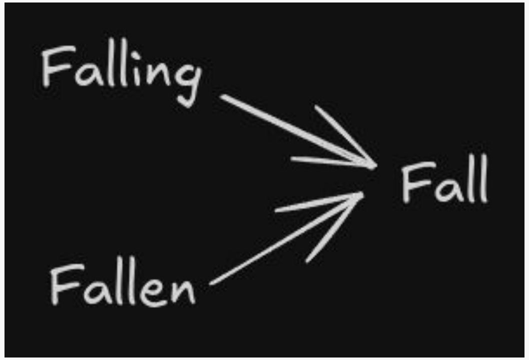
Lemmatization

What we do here, is we chop down the sentence into their base dictionary word, which considers the meaning and context of the word, hence not making a word, that makes no sense.

Let's preprocess some text!

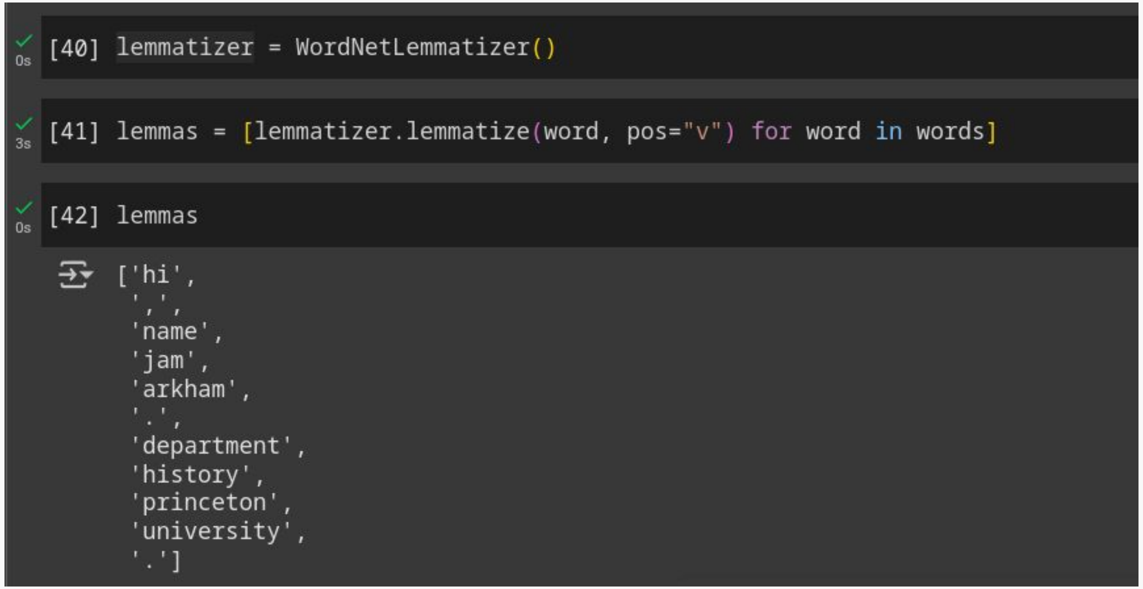
Lemmatization

What we do here, is we chop down the sentence into their base dictionary word, which considers the meaning and context of the word, hence not making a word, that makes no sense.



Let’s preprocess some text!

Lemmatization



```
[40] lemmatizer = WordNetLemmatizer()
```

```
[41] lemmas = [lemmatizer.lemmatize(word, pos="v") for word in words]
```

```
[42] lemmas
```

```
['hi',
 ',',
 'name',
 'jam',
 'arkham',
 '.',
 'department',
 'history',
 'princeton',
 'university',
 '.']
```

Task for 1 KitKat

Build a BPE Tokenizer

Bag of Words

Bag of Words

What’s it?

Bag of Words

What's it?

It is nothing but a simple way to represent text data. We don't care about the order of words not the grammar, we just care about what words appear in the sentence. It's like putting words in a bag and then randomly counting each type of word.

Bag of Words

What's it?

Let's say we have two sentences

S1 : I like pizza

S2 : I do not like pizza

Bag of Words

What’s it?

Let’s say we have two sentences

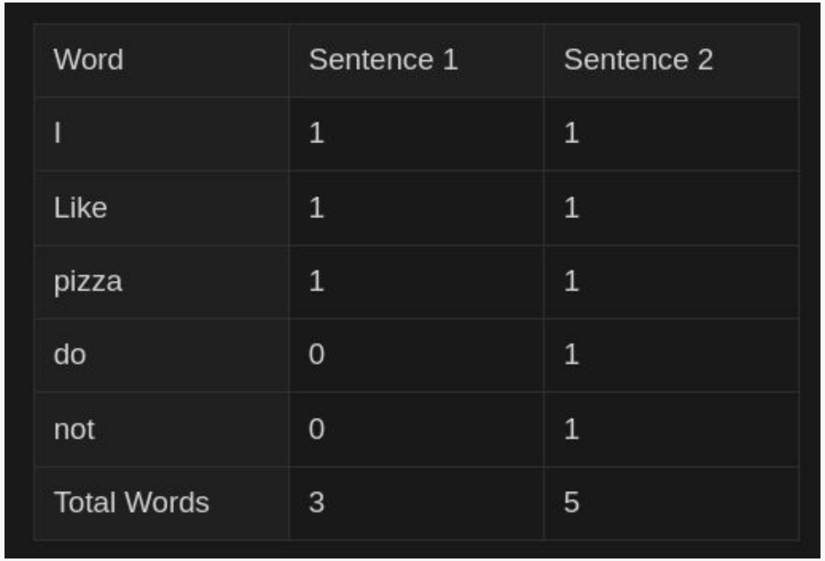
S1 : I like pizza

S2 : I do not like pizza

Our vocabulary will be [”I”, ”like”, ”pizza”, ”do”, ”not”]

Bag of Words

Now we count appearances



Word	Sentence 1	Sentence 2
I	1	1
Like	1	1
pizza	1	1
do	0	1
not	0	1
Total Words	3	5

Bag of Words

Representation

The BoW representation is :

1. Sentence 1 : [1,1,1,0,0]
2. Sentence 2 : [1,1,1,1,1]

Term Frequency (TF)

TF

What’s it?

TF

What’s it?

It measures how often a specific word appears in a document, compared to the total number of words in that document. Instead of counting like in BoW, TF normalises the counts, thus for larger datasets, we can see how important a word is in context of the document. It sets the stage for more advanced models to come.

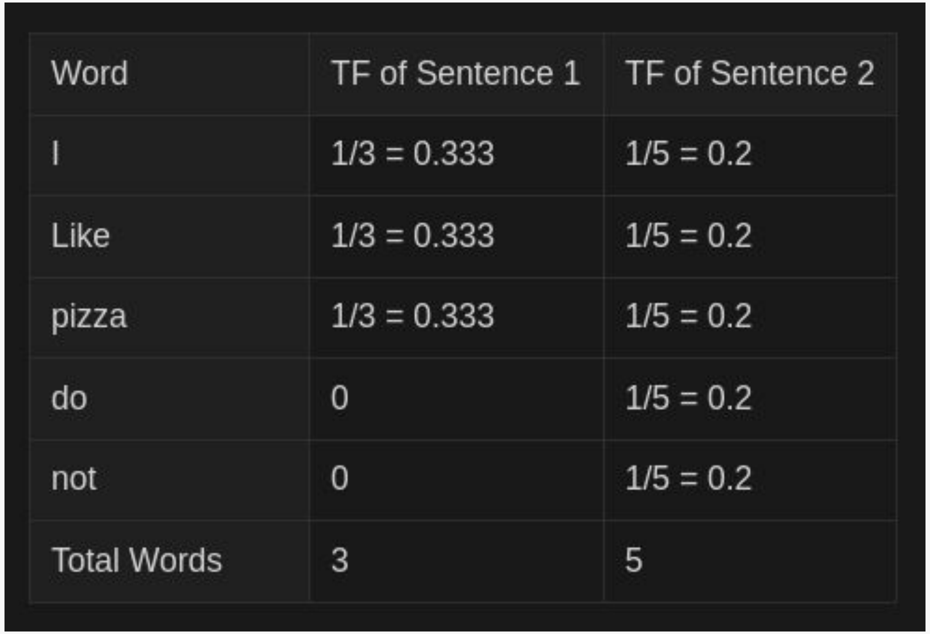
TF

What’s it?

$$TF(w) = \frac{count\ of\ w\ in\ document}{Total\ number\ of\ words\ in\ document}$$

TF

Revisit our old example



Word	TF of Sentence 1	TF of Sentence 2
I	1/3 = 0.333	1/5 = 0.2
Like	1/3 = 0.333	1/5 = 0.2
pizza	1/3 = 0.333	1/5 = 0.2
do	0	1/5 = 0.2
not	0	1/5 = 0.2
Total Words	3	5

Inverse Document Frequency (IDF)

IDF

What's it?

While TF told us about the importance of a word in a document, IDF tells us about how rare or unique a word is in the document. Using this, we can often eliminate words that carry very less context, rather than the rare words which might carry huge context. Words that appear in every document get a lower IDF (close to 0). Words that appear in fewer documents get a higher IDF.

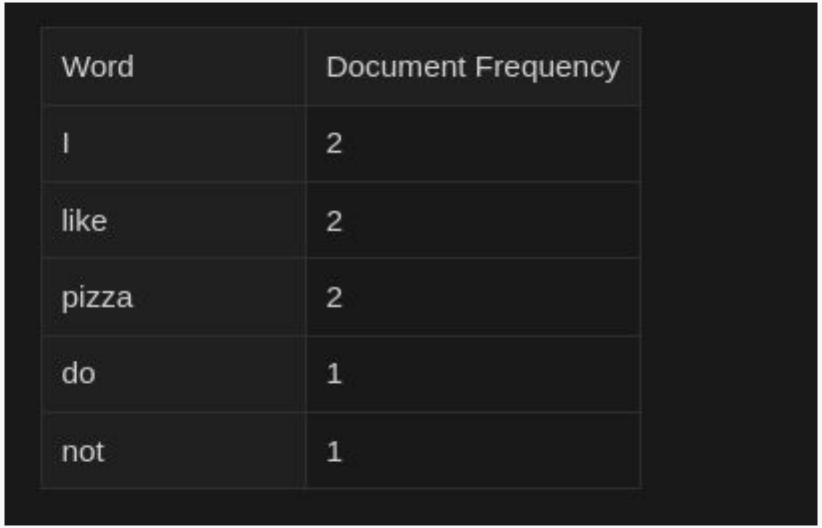
IDF

What’s it?

$$IDF(w) = log(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ having\ w})$$

IDF

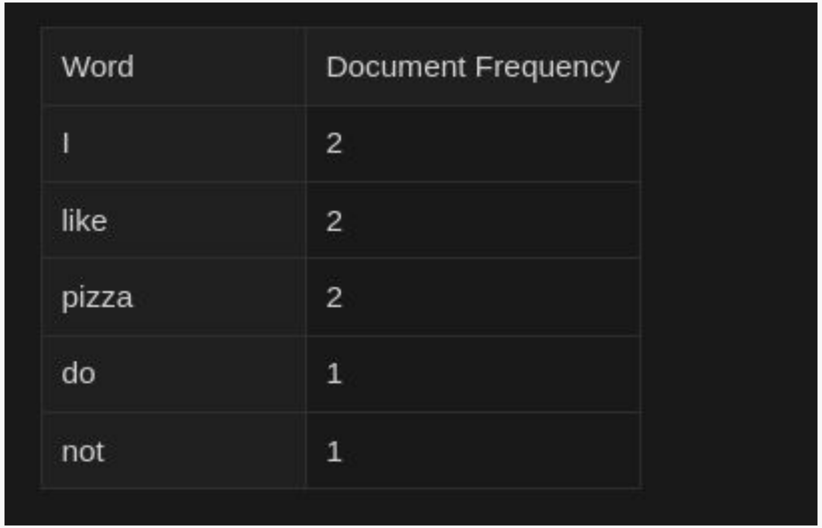
Revisit our old example



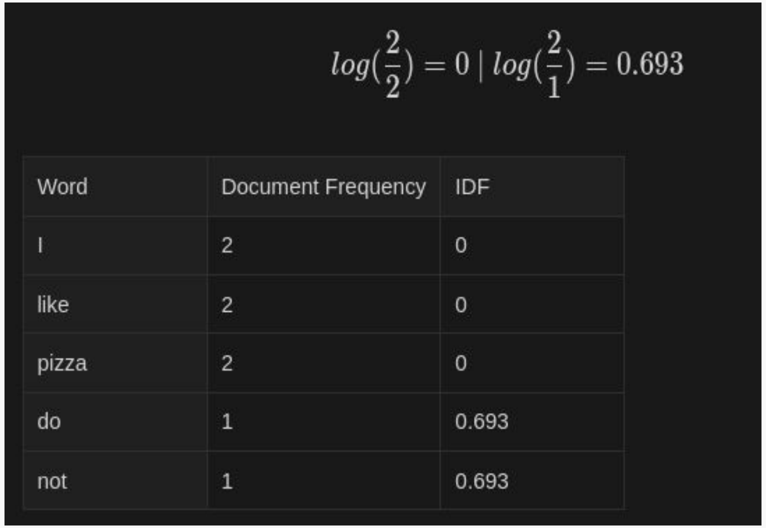
Word	Document Frequency
I	2
like	2
pizza	2
do	1
not	1

IDF

Revisit our old example



Word	Document Frequency
I	2
like	2
pizza	2
do	1
not	1



$$log(\frac{2}{2}) = 0 \mid log(\frac{2}{1}) = 0.693$$

Word	Document Frequency	IDF
I	2	0
like	2	0
pizza	2	0
do	1	0.693
not	1	0.693

TF-IDF

TF-IDF

Let’s combine

TF tells us how important a word is in a document, IDF tells us how rare that word is. Combining them gives a score that tells us how important a word is in a document, while leaving behind common words.

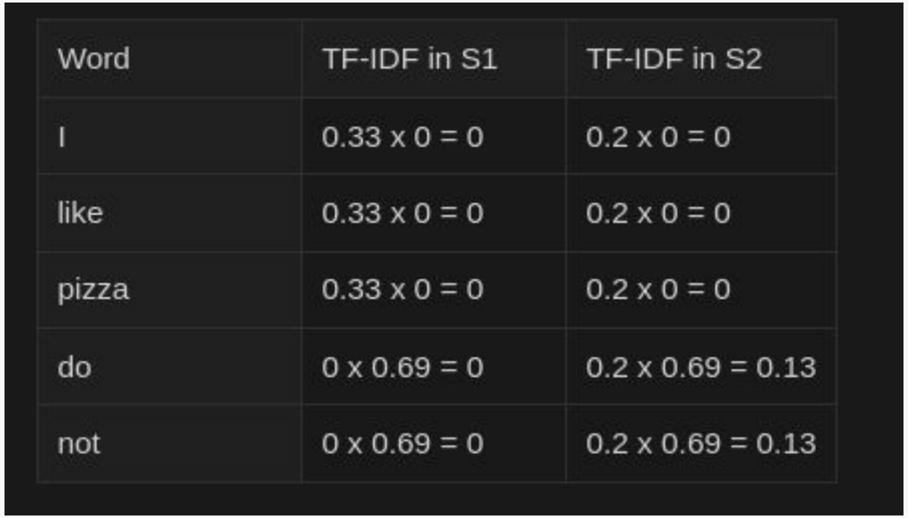
TF-IDF

Formula

$$TF - IDF(w) = TF(w) \times IDF(w)$$

TF-IDF

Revisit our old example



Word	TF-IDF in S1	TF-IDF in S2
I	0.33 x 0 = 0	0.2 x 0 = 0
like	0.33 x 0 = 0	0.2 x 0 = 0
pizza	0.33 x 0 = 0	0.2 x 0 = 0
do	0 x 0.69 = 0	0.2 x 0.69 = 0.13
not	0 x 0.69 = 0	0.2 x 0.69 = 0.13

Word Embeddings

Word Embeddings

Why do we need it?

Word Embeddings

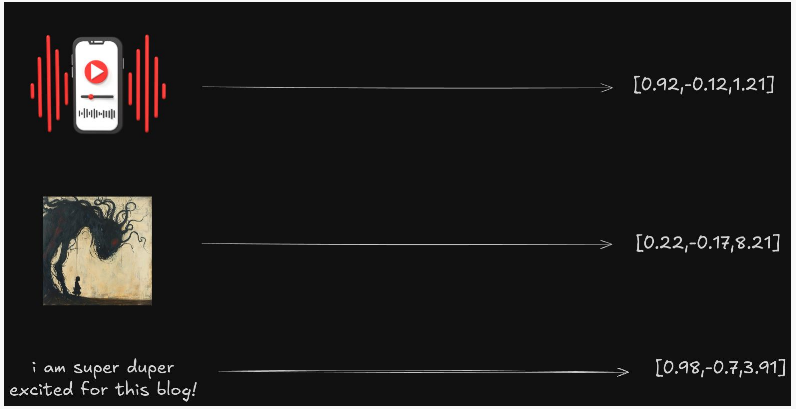
Why do we need it?

The model can’t understand any input we give. Whatever form it is in, we must convert it into some vector for the model to understand it. By this process, we do embedding!

Word Embeddings

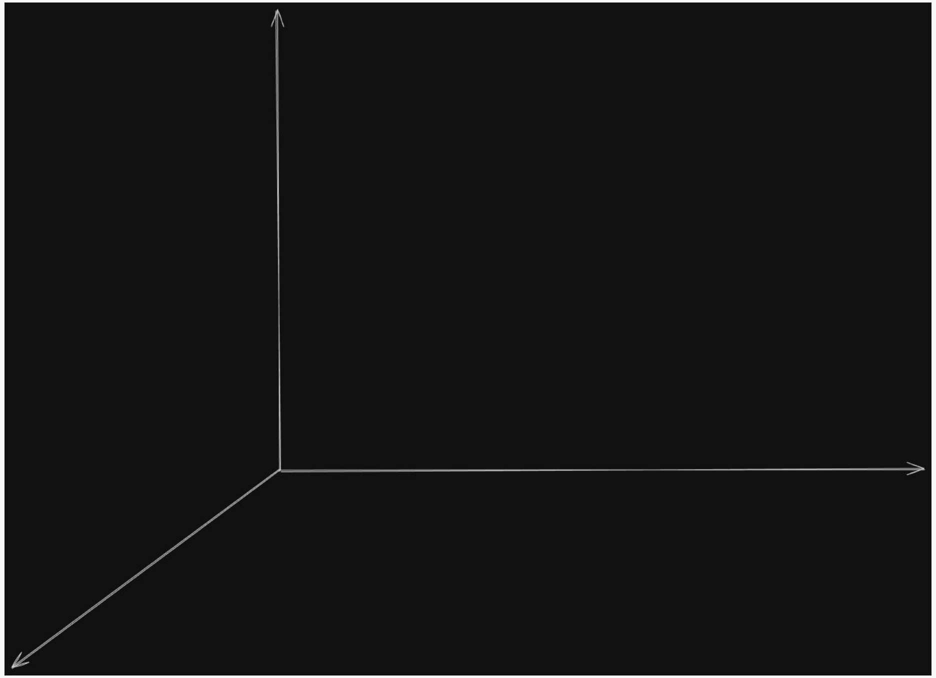
Why do we need it?

The model can't understand any input we give. Whatever form it is in, we must convert it into some vector for the model to understand it. By this process, we do embedding!



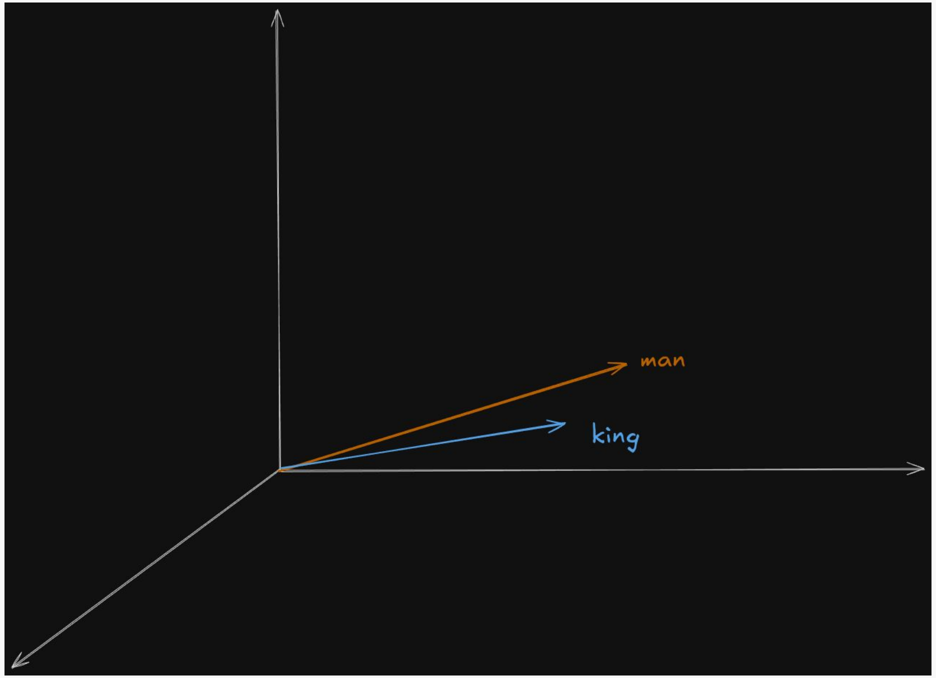
Word Embeddings

A fun example



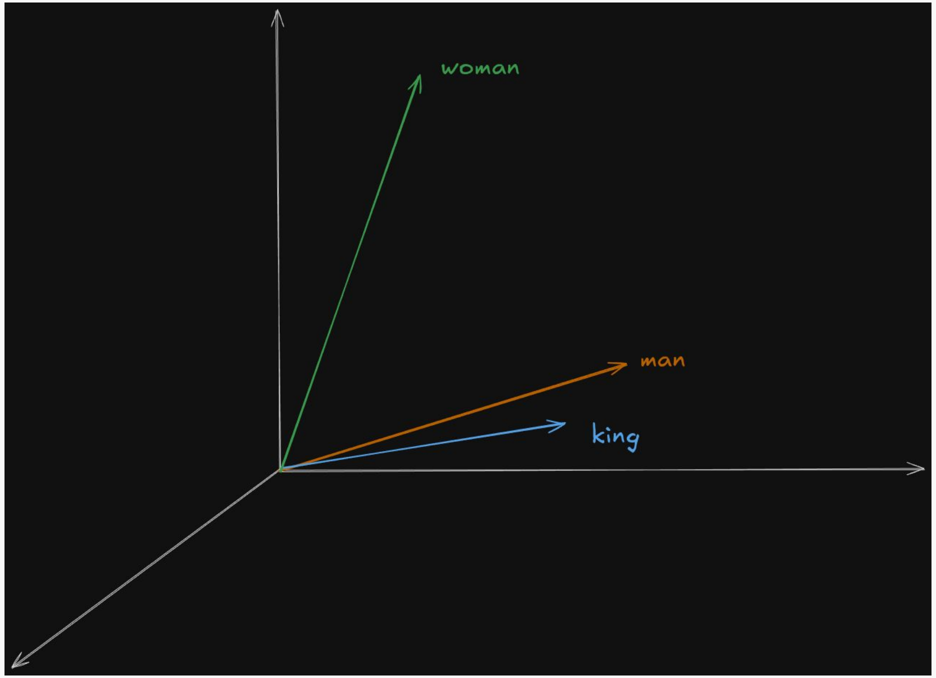
Word Embeddings

A fun example



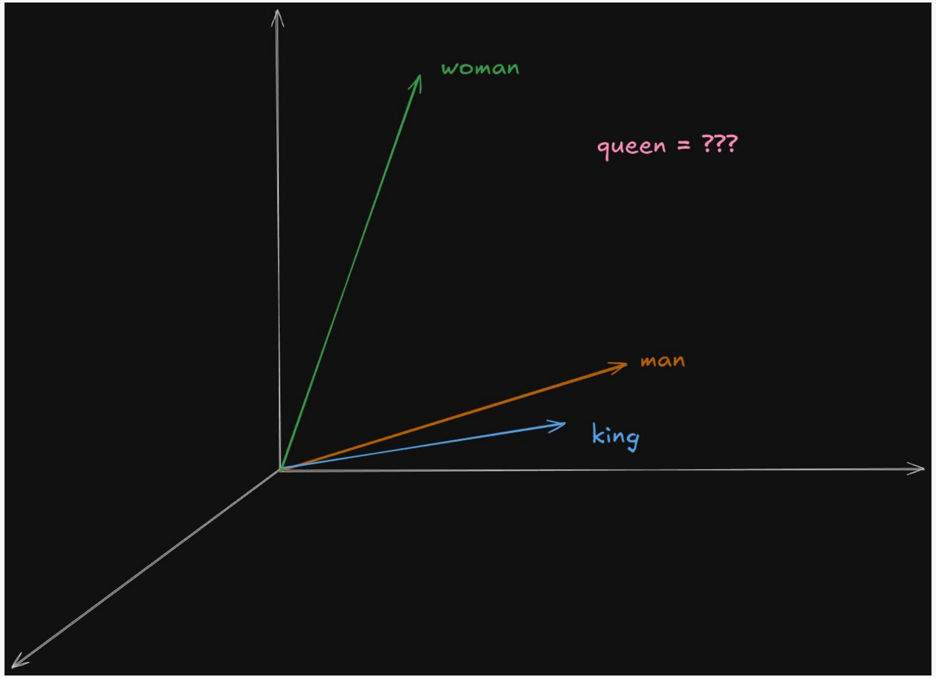
Word Embeddings

A fun example



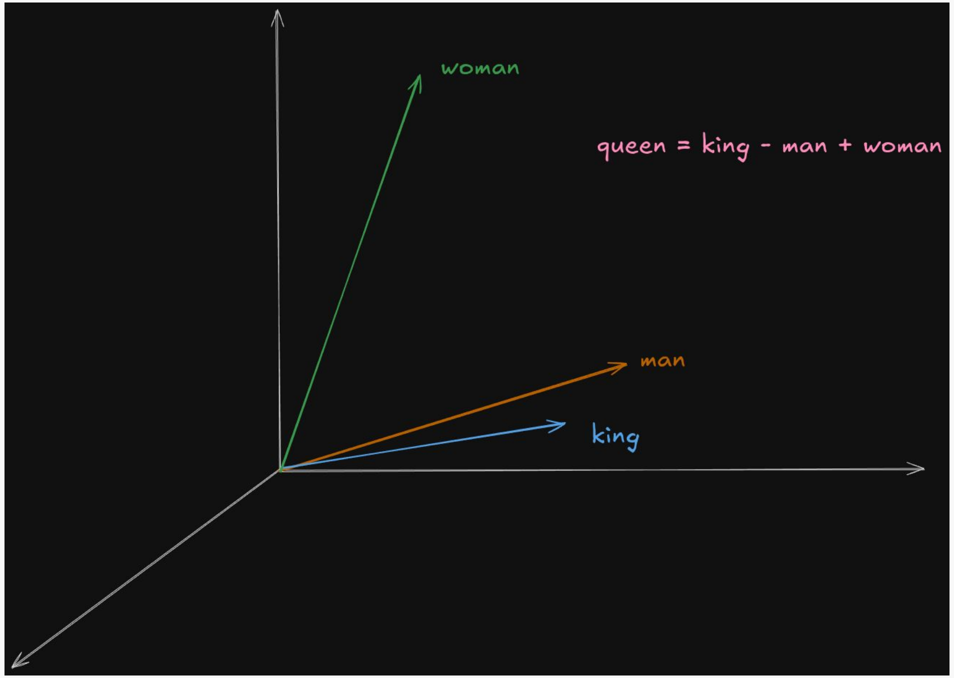
Word Embeddings

A fun example



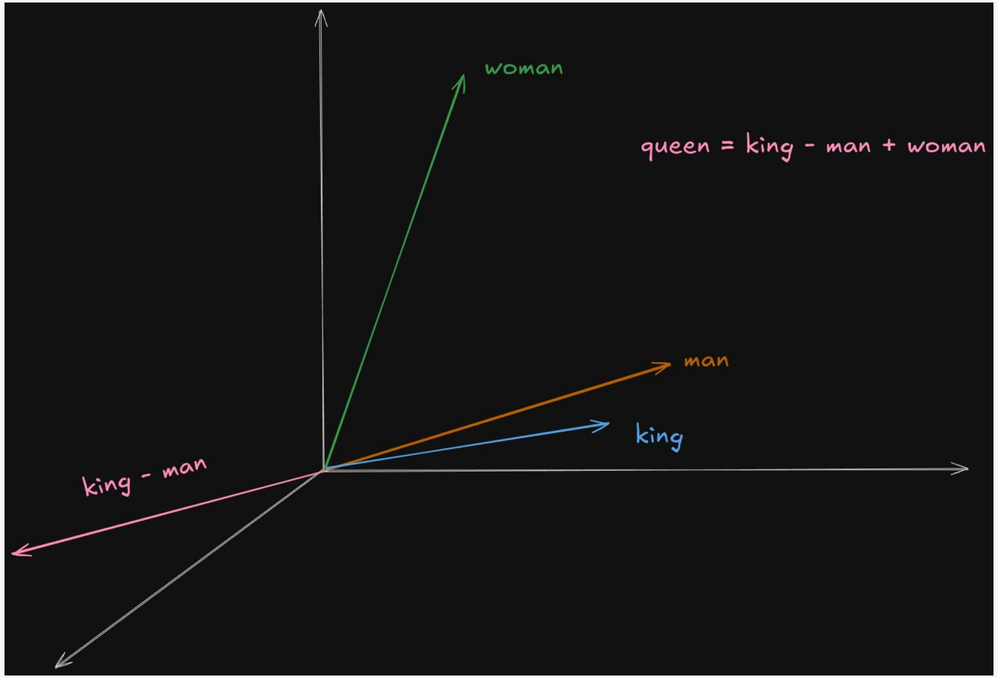
Word Embeddings

A fun example



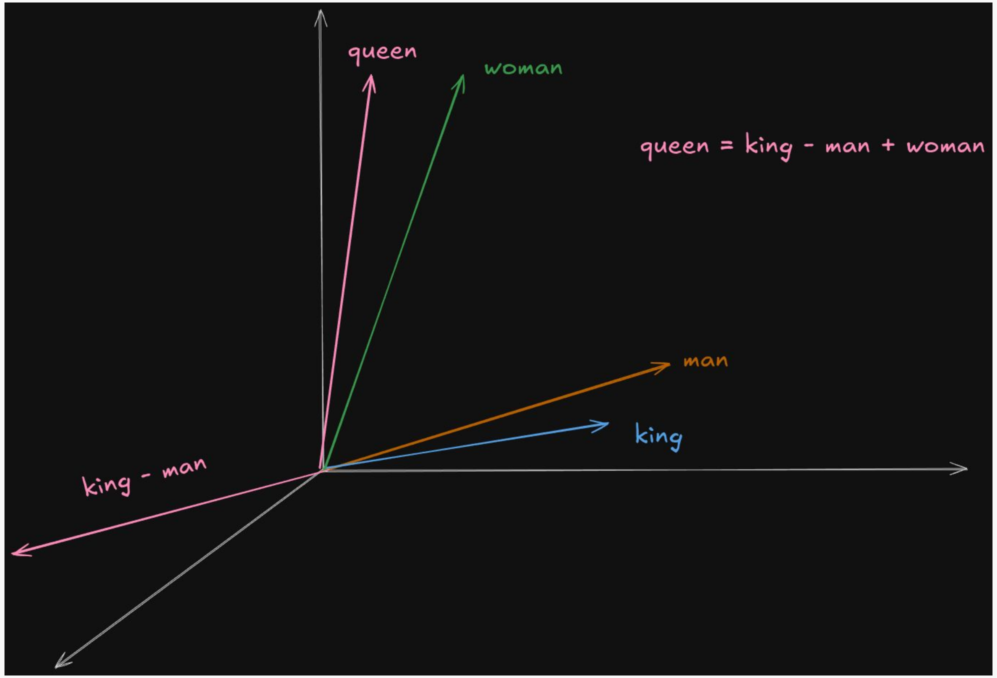
Word Embeddings

A fun example



Word Embeddings

A fun example



Word2Vec

Word2Vec

What is it?

Word2Vec

What is it?

What word2vec works in such a way, that it causes words that tend to occur in same type of contexts, to have embedding values similar to each other.

Word2Vec

What is it?

Let’s consider two sentences.

S1 : The child threw the ball across the park.

S2 : The kid threw the ball across the park.

Word2Vec

What is it?

Let’s consider two sentences.

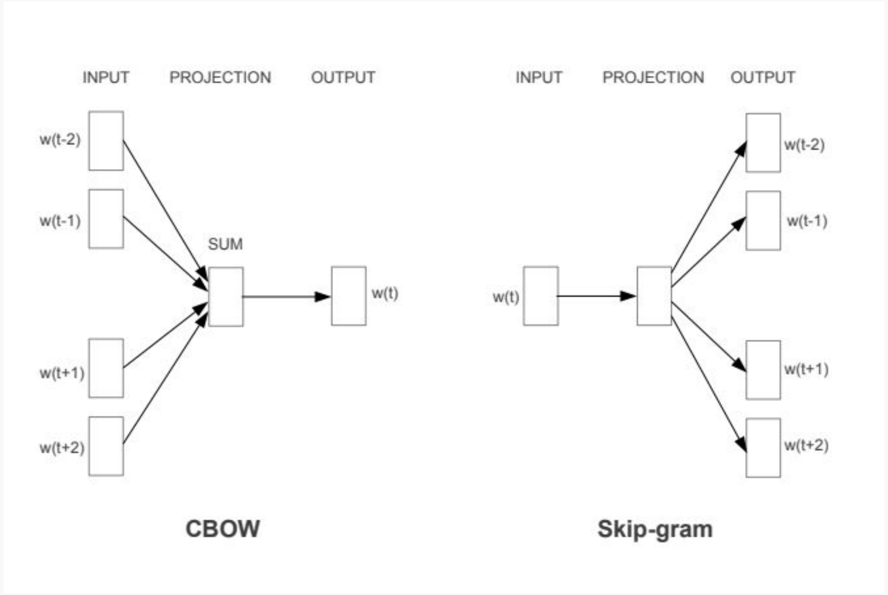
S1 : The child threw the ball across the park.

S2 : The kid threw the ball across the park.

Here, due to word2vec, the words child and kid will have similar embedding vectors, as they have similar context in these sentences.

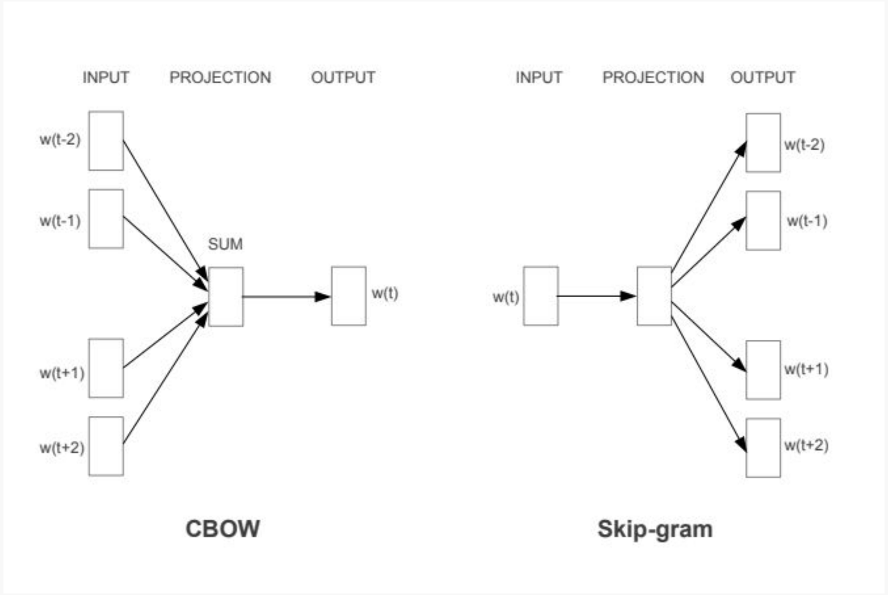
Word2Vec

How does it look?



Word2Vec

How does it look?



Let’s dissect it!

Word2Vec

CBOW

Word2Vec

CBOW

Let’s use the example sentence: *What is life*. The goal of CBOW is to predict the middle word (*is*) given its surrounding words (*What* and *life*)

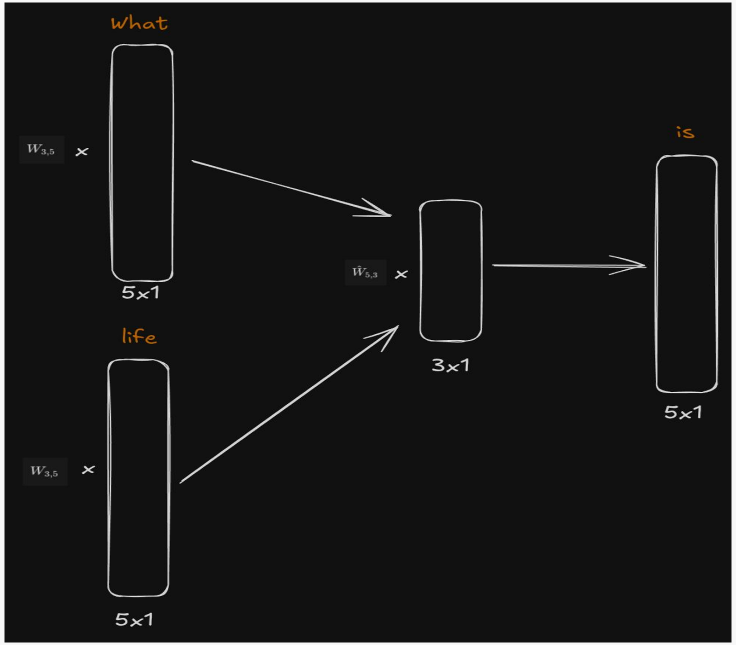
Word2Vec

CBOW

One hot encode *What* and *Life*

Word2Vec

CBOW



Word2Vec

Skipgram

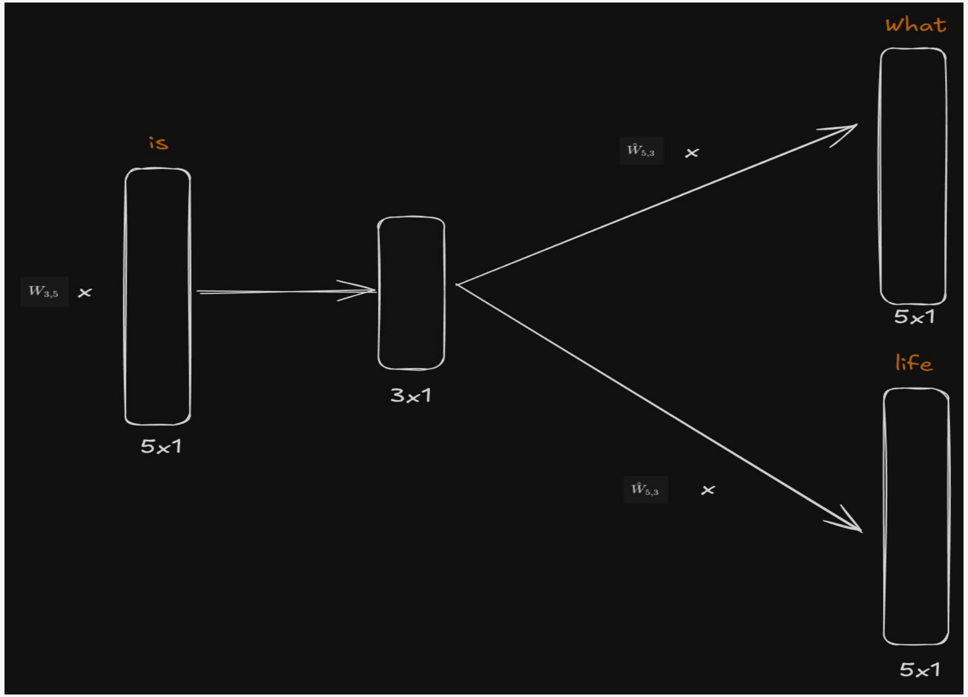
Word2Vec

Skipgram

Skip-gram works in reverse compared to CBOW. Instead of predicting a word from its context, it predicts surrounding context words given a center word.

Word2Vec

Skipgram



GLoVe

GLoVE

What is it?

GLoVE

What is it?

It is an unsupervised learning algorithm that is designed to learn word embeddings, by making statistical relations between words in a large corpus. It tries to find the co-occurrence patterns in a corpus

GLoVE

How does it work?

GLoVE

How does it work?

Let us consider a set of sentences

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

GLoVE

How does it work?

Let us consider a set of sentences

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

From these sentences, let's list the unique words: [I, enjoy, flying, like, NLP, deep, learning]

GLoVE

How does it work?

We need to fix a context window, here too like Word2Vec. Let’s make it 1. That means we will only check co-occurrence with the adjacent word, aka the word earlier and the word later

GLoVE

How does it work?

$$X = \begin{array}{c|cccccccc}
 & I & \text{like} & \text{enjoy} & \text{deep} & \text{learning} & \text{NLP} & \text{flying} & . \\
\hline
I & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\
\text{like} & 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
\text{enjoy} & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\text{deep} & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
\text{learning} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\text{NLP} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
\text{flying} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
. & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0
\end{array}$$

source : CS224N

Task for 1 KitKat (right now)

Why was the matrix symmetric?

POS Tagging

POS Tagging

What is it?

POS Tagging

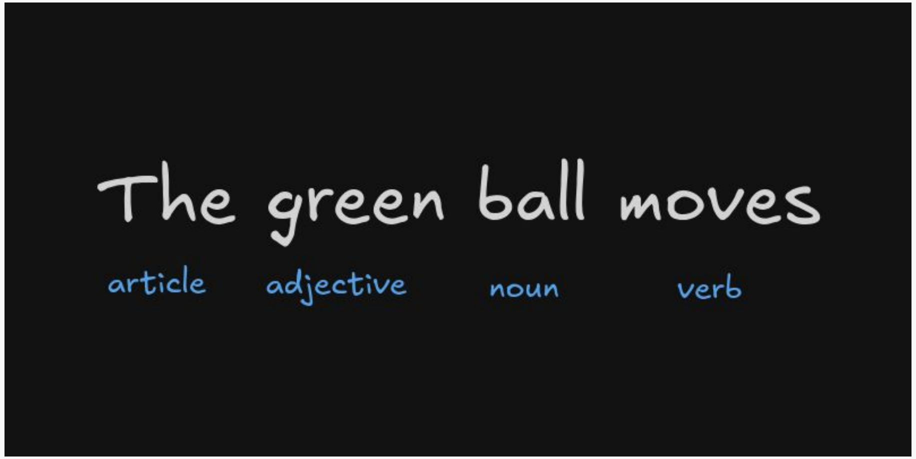
What is it?

The basic understanding is that we label each word in the sentence with its corresponding parts of speech.

POS Tagging

What is it?

The basic understanding is that we label each word in the sentence with its corresponding parts of speech.

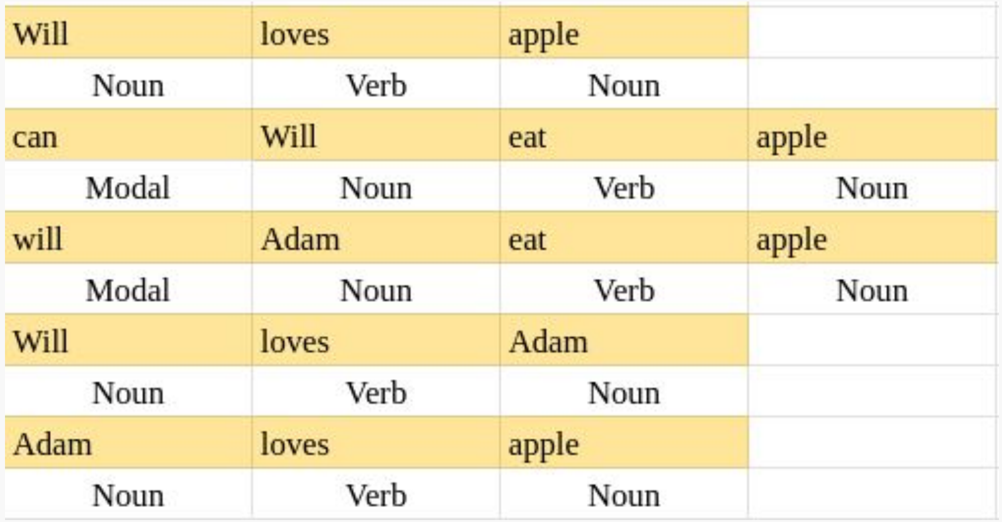


POS Tagging

How does it work?

POS Tagging

How does it work?



Will	loves	apple	
Noun	Verb	Noun	
can	Will	eat	apple
Modal	Noun	Verb	Noun
will	Adam	eat	apple
Modal	Noun	Verb	Noun
Will	loves	Adam	
Noun	Verb	Noun	
Adam	loves	apple	
Noun	Verb	Noun	

POS Tagging

How does it work?

Now we need to make an emission probability table first

POS Tagging

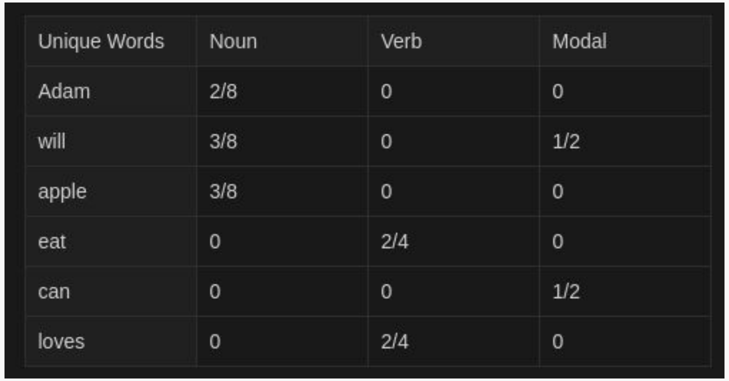
How does it work?

Now we need to make an emission probability table first.
We just count the frequencies of in what context they have been used in the sentences and find their probabilities concerning the part of speech (column-wise).

POS Tagging

How does it work?

Now we need to make an emission probability table first. We just count the frequencies of in what context they have been used in the sentences and find their probabilities concerning the part of speech (column-wise).



Unique Words	Noun	Verb	Modal
Adam	2/8	0	0
will	3/8	0	1/2
apple	3/8	0	0
eat	0	2/4	0
can	0	0	1/2
loves	0	2/4	0

POS Tagging

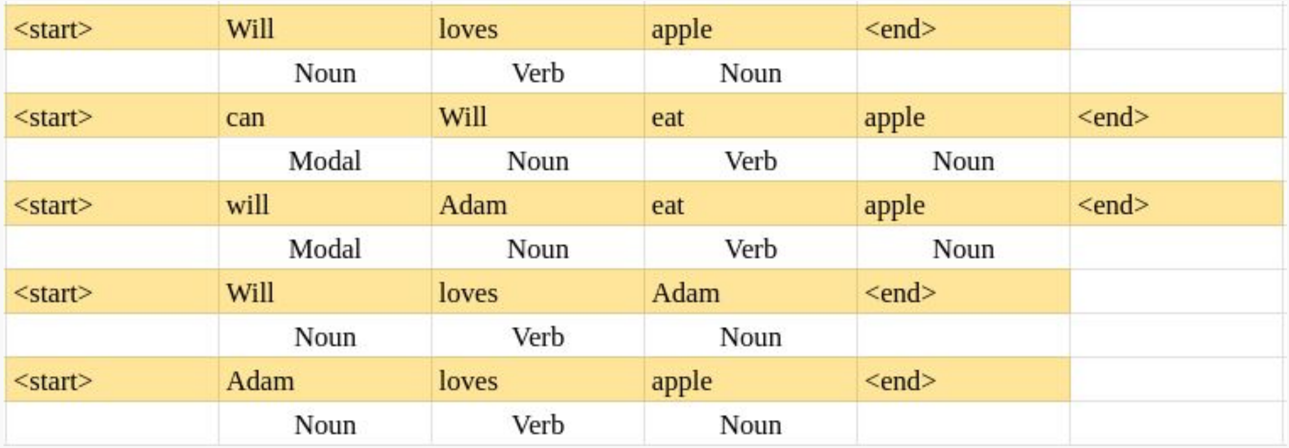
How does it work?

Now we build a transition probability table. For this, we add a start and end tag before the sentences.

POS Tagging

How does it work?

Now we build a transition probability table. For this, we add a start and end tag before the sentences.

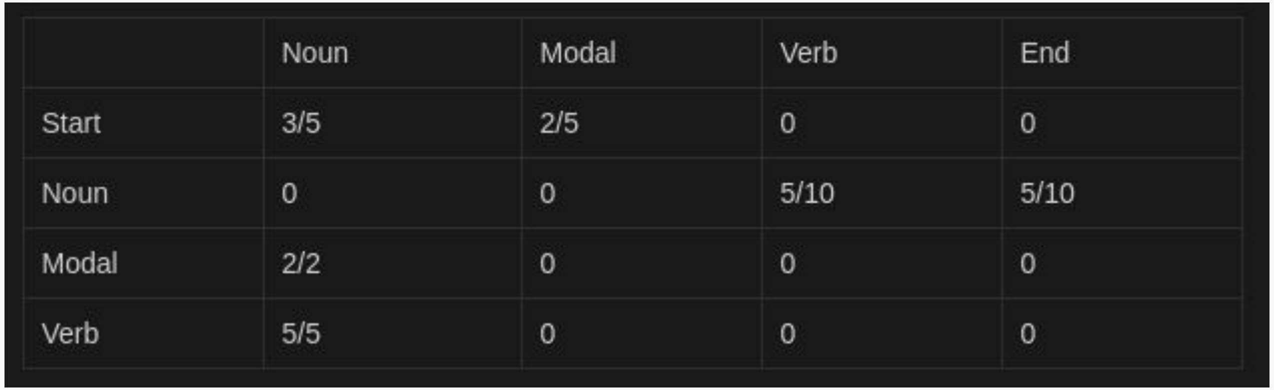


<start>	Will	loves	apple	<end>	
	Noun	Verb	Noun		
<start>	can	Will	eat	apple	<end>
	Modal	Noun	Verb	Noun	
<start>	will	Adam	eat	apple	<end>
	Modal	Noun	Verb	Noun	
<start>	Will	loves	Adam	<end>	
	Noun	Verb	Noun		
<start>	Adam	loves	apple	<end>	
	Noun	Verb	Noun		

POS Tagging

How does it work?

Now we build a transition probability table. For this, we add a start and end tag before the sentences.



	Noun	Modal	Verb	End
Start	3/5	2/5	0	0
Noun	0	0	5/10	5/10
Modal	2/2	0	0	0
Verb	5/5	0	0	0

POS Tagging

How does it work?

let’s visit our statement, Will will eat apple

POS Tagging

How does it work?

let's visit our statement, Will will eat apple

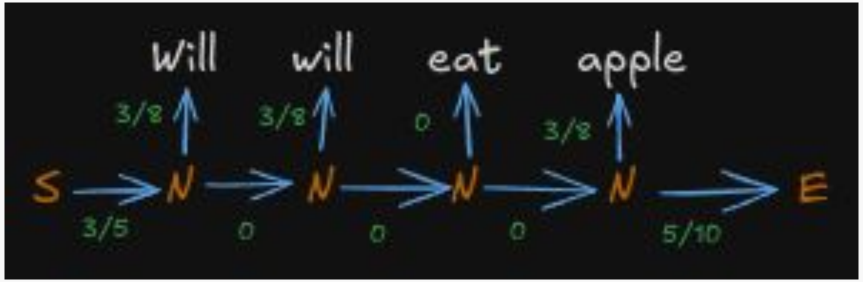
Let us consider all of them nouns and fill in the emission and transition probabilities.

POS Tagging

How does it work?

let's visit our statement, Will will eat apple

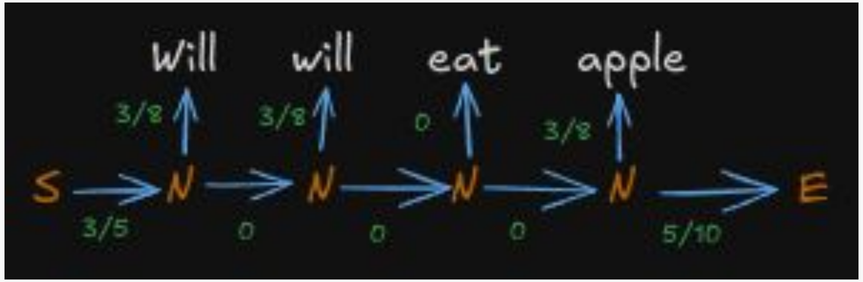
Let us consider all of them nouns and fill in the emission and transition probabilities.



POS Tagging

How does it work?

let's visit our statement, Will will eat apple
Let us consider all of them nouns and fill in the emission and transition probabilities.



The total probabilities come out to be 0. Hence such a part of speech tagging is completely impossible.

POS Tagging

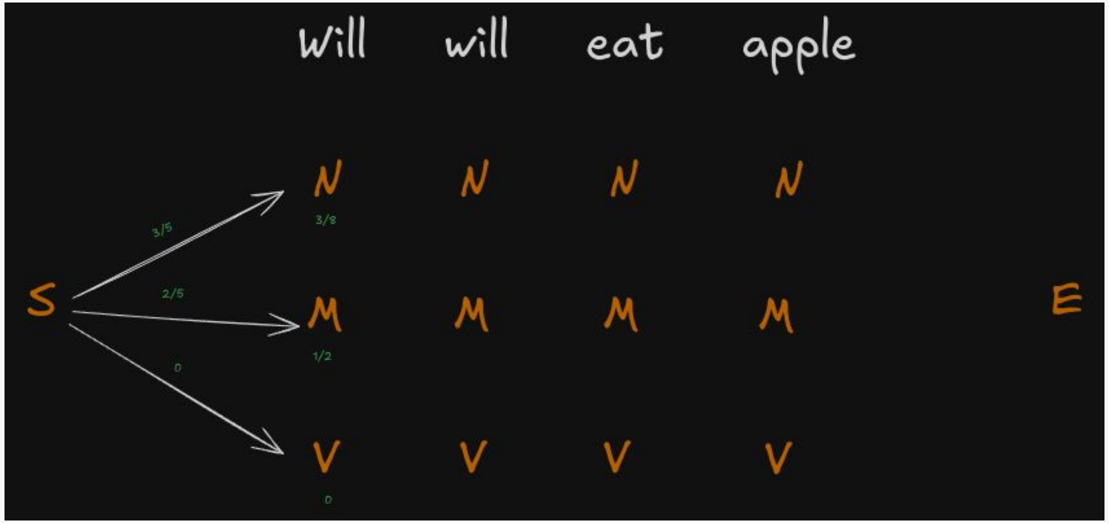
WHAT IS THE PROBLEM WITH THIS PROCEDURE

POS Tagging

Viterbi Algorithm

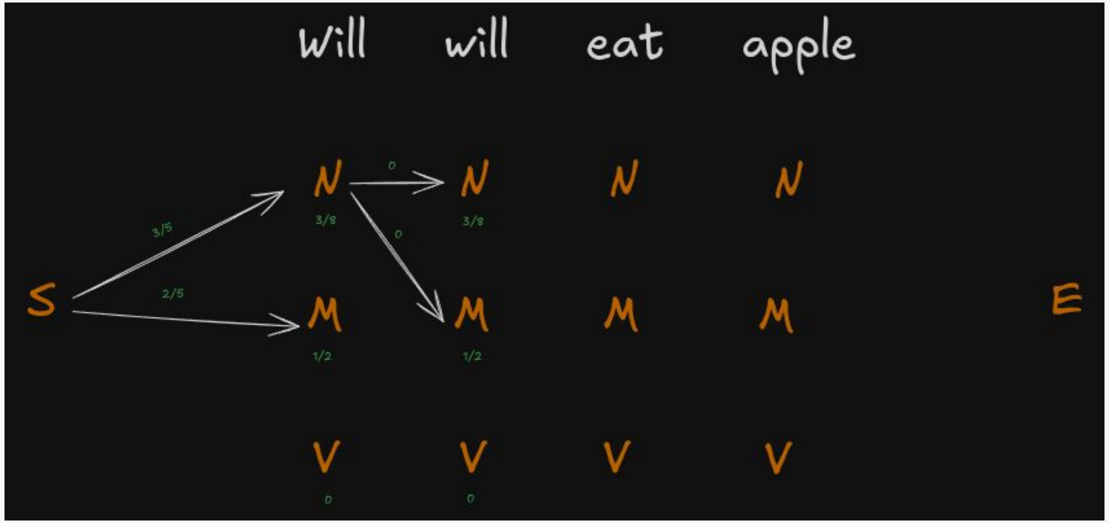
POS Tagging

Viterbi Algorithm



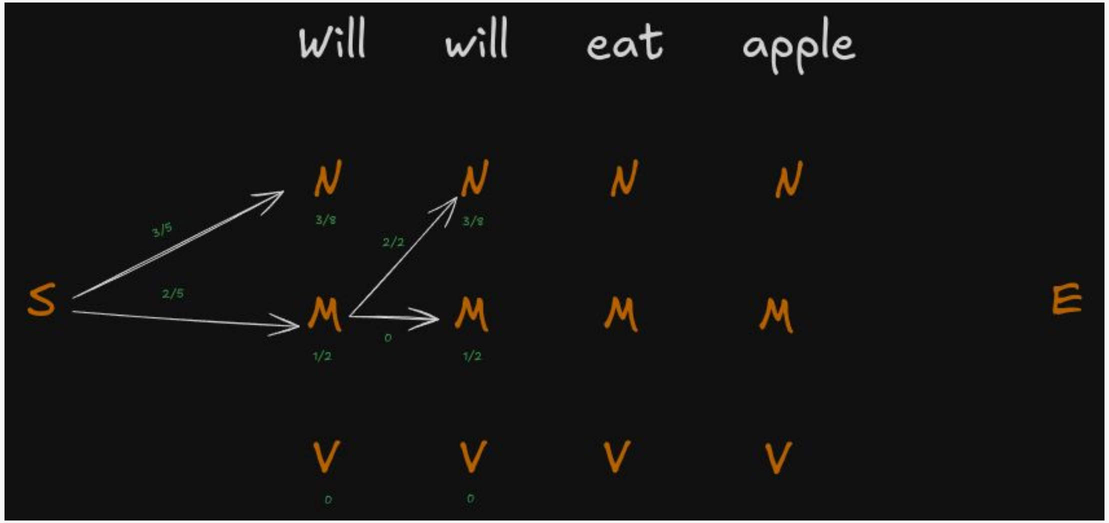
POS Tagging

Viterbi Algorithm



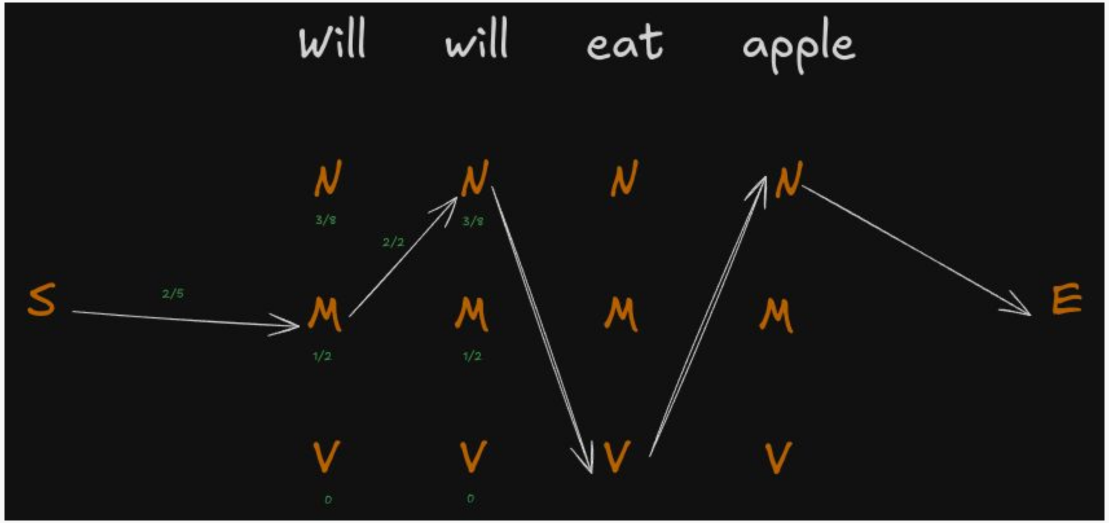
POS Tagging

Viterbi Algorithm



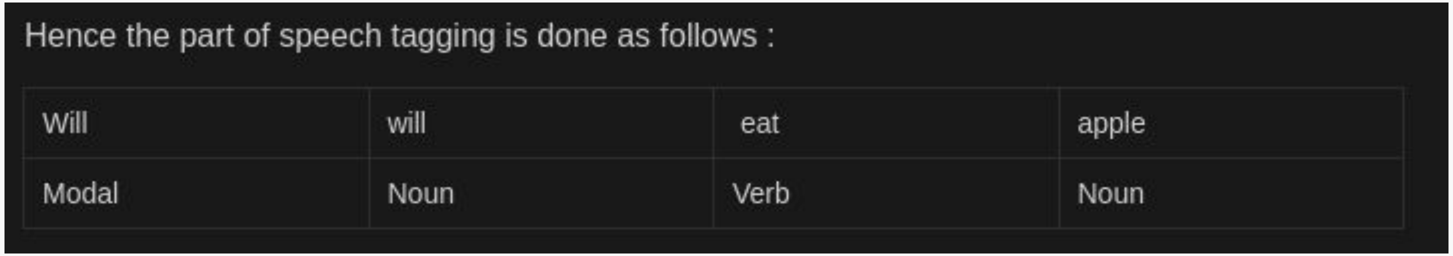
POS Tagging

Viterbi Algorithm



POS Tagging

Viterbi Algorithm



Hence the part of speech tagging is done as follows :

Will	will	eat	apple
Modal	Noun	Verb	Noun

Sentiment Analysis

Sentiment Analysis

Should I go to this movie?

Sentiment Analysis

Should I go to this movie?

we try to find out what the sentence is feeling overall.

Sentiment Analysis

Should I go to this movie?

we try to find out what the sentence is feeling overall.

I am feeling great is a positive sentiment, while I am dying is a negative one.

Sentiment Analysis

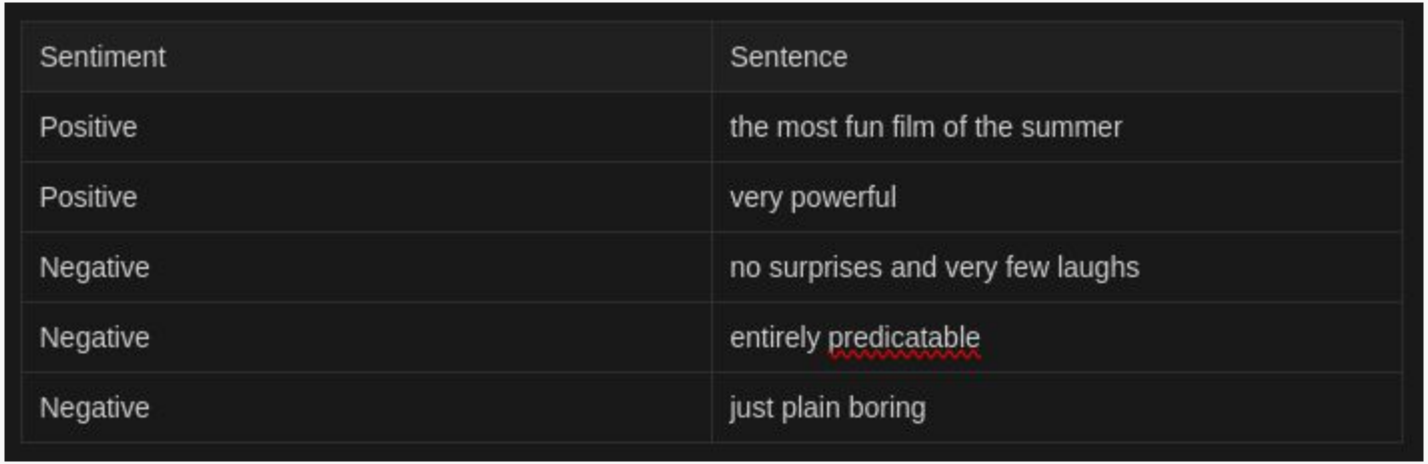
Should I go to this movie?

Let us try to find out the sentiment of the sentence

predictable with no fun

Sentiment Analysis

Training Samples



Sentiment	Sentence
Positive	the most fun film of the summer
Positive	very powerful
Negative	no surprises and very few laughs
Negative	entirely predicatable
Negative	just plain boring

Sentiment Analysis

Preprocessing

After some preprocessing we get

predictable no fun

Sentiment Analysis

Probabilities

try to just see the probability of a sentence being negative or positive

Sentiment Analysis

Probabilities

try to just see the probability of a sentence being negative or positive

$$P(positive) = \frac{2}{5}$$
$$P(negative) = \frac{3}{5}$$

Sentiment Analysis

Probabilities

we first calculate the probability likelihood of each case, using this formula.

$$P(word_i|sentiment) = \frac{count(word_i, sentiment) + 1}{\sum_{word \in V} count(word, sentiment) + |V|}$$

Sentiment Analysis

What even was that formula

Bayes Theorem looks like this

$$P(w_i|c) = \frac{count(w_i, c)}{\sum count(w, c)}$$

Sentiment Analysis

What even was that formula

now sometimes the count can be zero, thus meaning the probability of a test document belonging to positive/negative is 0, which is impossible. Hence we added some Laplace smoothing, and the formula is updated

Sentiment Analysis

Probabilities

Now we find all the probabilities of predictable no fun with respect to all sentiments.

Sentiment Analysis

Probabilities

Now we find all the probabilities of predictable no fun with respect to all sentiments.

$$\begin{aligned}
P(\text{“predictable”} \mid -) &= \frac{1+1}{14+20} & P(\text{“predictable”} \mid +) &= \frac{0+1}{9+20} \\
P(\text{“no”} \mid -) &= \frac{1+1}{14+20} & P(\text{“no”} \mid +) &= \frac{0+1}{9+20} \\
P(\text{“fun”} \mid -) &= \frac{0+1}{14+20} & P(\text{“fun”} \mid +) &= \frac{1+1}{9+20}
\end{aligned}$$

Sentiment Analysis

Probabilities

Now we find out the scores of the sentence S now

Sentiment Analysis

Probabilities

Now we find out the scores of the sentence S now

$$P(-)P(S \mid -) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$
$$P(+)P(S \mid +) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

<u>Task for 1 KitKat</u>

Create a sentiment analysis tool that divides sentences into sentiments of happy, sad, angry, and no emotion.

N-Gram Models

N-Gram Models

Let’s think

N-Gram Models

Let’s think

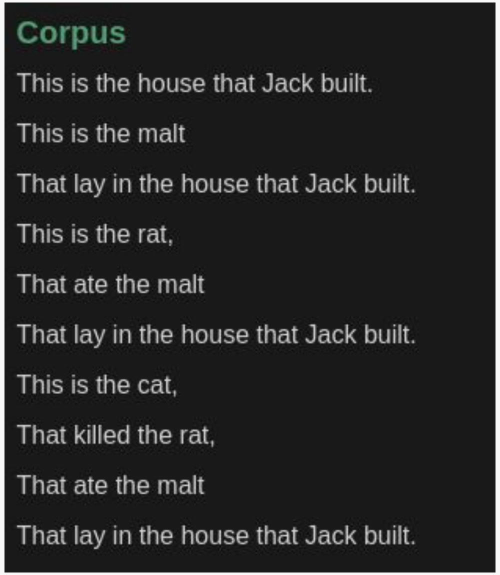
Let’s say we have a part of a sentence -

This is a ___

What will be the next word here?

N-Gram Models

Let’s make our own



Corpus

This is the house that Jack built.

This is the malt

That lay in the house that Jack built.

This is the rat,

That ate the malt

That lay in the house that Jack built.

This is the cat,

That killed the rat,

That ate the malt

That lay in the house that Jack built.

N-Gram Models

Let’s think

What is $P(\text{house} \mid \text{This is the})$?

N-Gram Models

Let’s think

What is P(house | This is the)?

This is the will be in Sentence 1,2,4,7, and This is the house is in Sentence 1.

N-Gram Models

Let’s think

What is P(house | This is the)?

This is the will be in Sentence 1,2,4,7, and This is the house is in Sentence 1.

So the answer is 1/4

N-Gram Models

How does it work?

Let’s predict the probability of a sequence of words.

$$w = (w_1 w_2 w_3 \ldots w_k)$$

N-Gram Models

How does it work?

We will apply the chain rule to this :

$$p(w) = p(w_1)p(w_2|w_1)\dots p(w_k|w_1\dots w_{k-1})$$

N-Gram Models

What’s the problem?

N-Gram Models

How does it work?

we use the Markov Assumption. It tells us we don't need to calculate from w_1 to w_{k-1} as it is very unlikely that all of the text is connected to itself.

N-Gram Models

How does it work?

we use the Markov Assumption. It tells us we don’t need to calculate from w_1 to w_{k-1} as it is very unlikely that all of the text is connected to itself.

$$p(w_i|w_1 \ldots w_{i-1}) = p(w_i|w_{i-n+1} \ldots w_{i-1})$$

Bigram Models

How does it work?

Bigram Models

How does it work?

Equation :

$$p(w) = p(w_1)p(w_2|w_1)\ldots p(w_k|w_{k-1})$$

Bigram Models

How does it work?

Let’s find our initial probability!

Bigram Models

How does it work?

Let’s find our initial probability!

$$p(\text{this is the house}) = p(\text{this})p(\text{is}|\text{this})p(\text{the}|\text{is})p(\text{house}|\text{the})$$
$$= \frac{1}{12}.1.1.\frac{1}{2} = \frac{1}{24}$$

Bigram Models

Problems

We should not use the probability of w_1 distributed throughout the corpus, but instead only use the start words.

Bigram Models

Problems

We should not use the probability of w_1 distributed throughout the corpus, but instead only use the start words.

$$p(w) = p(w_1|start)p(w_2|w_1)\dots p(w_k|w_{k-1})$$

Bigram Models

Problems

The same needs to be done at the end too.

Bigram Models

Problems

The same needs to be done at the end too.

$$p(w) = p(w_1|start)p(w_2|w_1)\dots p(w_k|w_{k-1})p(end|w_k)$$

Thanks for Attending!

Send the solutions to deeponh.2004@gmail.com