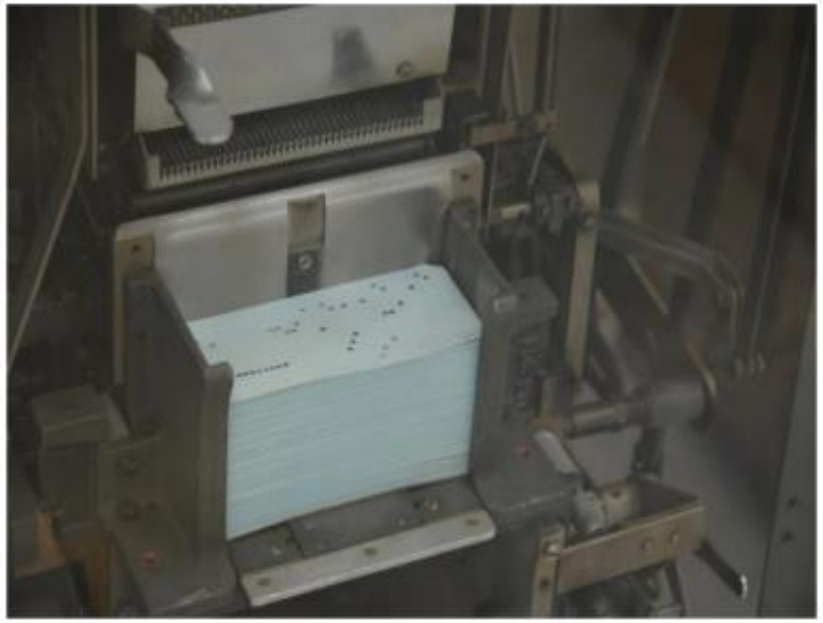
Introduction to NLTK and learning Text Representations

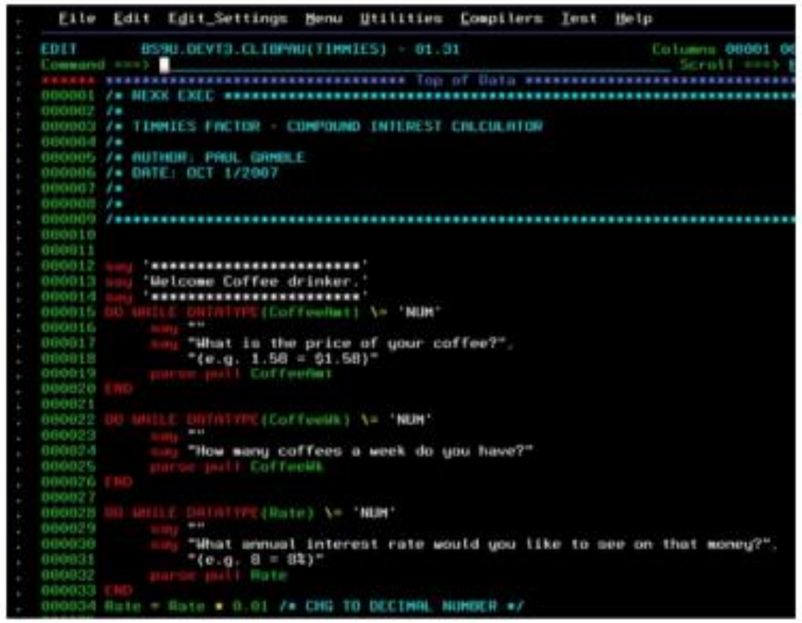
Ms. Raisa M. Shaikh

Data Scientist

Communication With Machines



~50-70s



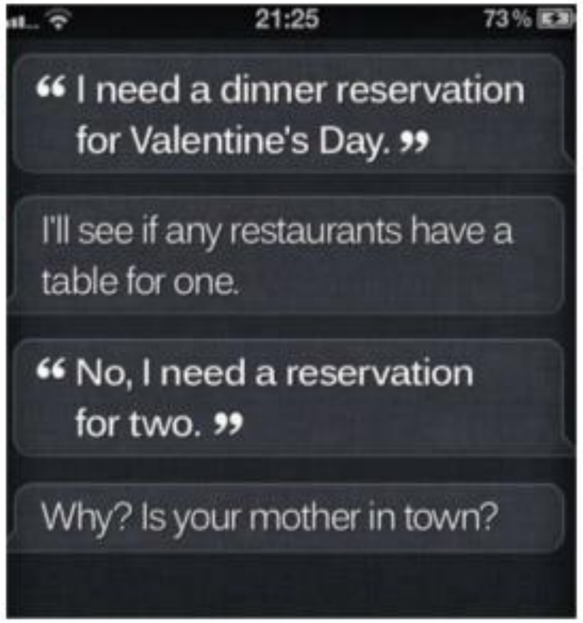
```
EDIT       BS9U.DEVT3.CLIBPAU(TIMMIES) - 01.31             Columns 00001 00
Command ===>                                                  Scroll ===>
****** ***************************** Top of Data ****************************
/* REXX EXEC ****************************************************************
/*
/* TIMMIES FACTOR - COMPOUND INTEREST CALCULATOR
/*
/* AUTHOR: PAUL GAMBLE
/* DATE: OCT 1/2007
/*
/*
/****************************************************************************


say '************************'
say 'Welcome Coffee drinker.'
say '************************'
DO WHILE DATATYPE(CoffeeAmt) \= 'NUM'
     say ""
     say "What is the price of your coffee?",
         "(e.g. 1.58 = $1.58)"
     parse pull CoffeeAmt
END

DO WHILE DATATYPE(CoffeeWk) \= 'NUM'
     say ""
     say "How many coffees a week do you have?"
     parse pull CoffeeWk
END

DO WHILE DATATYPE(Rate) \= 'NUM'
     say ""
     say "What annual interest rate would you like to see on that money?",
         "(e.g. 8 = 8%)"
     parse pull Rate
END
Rate = Rate * 0.01 /* CHG TO DECIMAL NUMBER */
```

~80s



“ I need a dinner reservation for Valentine's Day. ”

I'll see if any restaurants have a table for one.

“ No, I need a reservation for two. ”

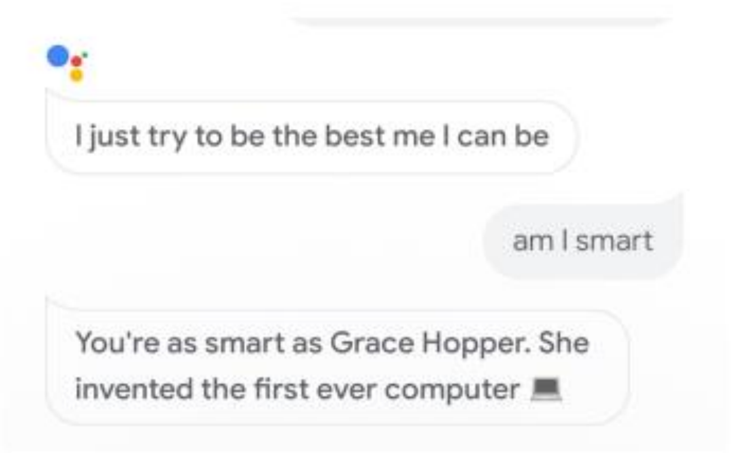
Why? Is your mother in town?

today

Conversational Agents

Conversational agents contain:

- Speech recognition
- Language analysis
- Dialogue processing
- Information retrieval
- Text to speech



I just try to be the best me I can be

am I smart

You're as smart as Grace Hopper. She invented the first ever computer 💻

Natural Language Processing

Applications

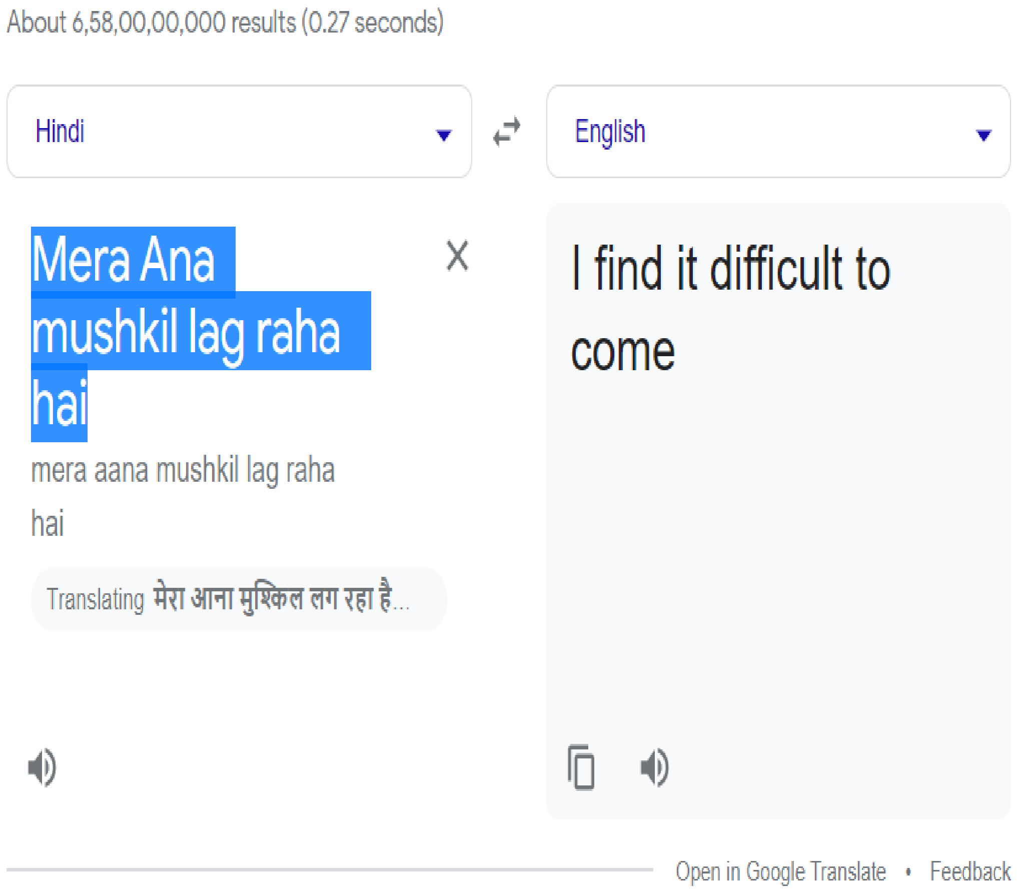
- Machine Translation
- Information Retrieval
- Question Answering
- Dialogue Systems
- Information Extraction
- Summarization
- Sentiment Analysis
- ...

Core Technologies

- Language modeling
- Part-of-speech tagging
- Syntactic parsing
- Named-entity recognition
- Word sense disambiguation
- Semantic role labeling
- ...

NLP lies at the intersection of computational linguistics and machine learning.

Machine Translations



About 6,58,00,00,000 results (0.27 seconds)

Hindi ⇄ English

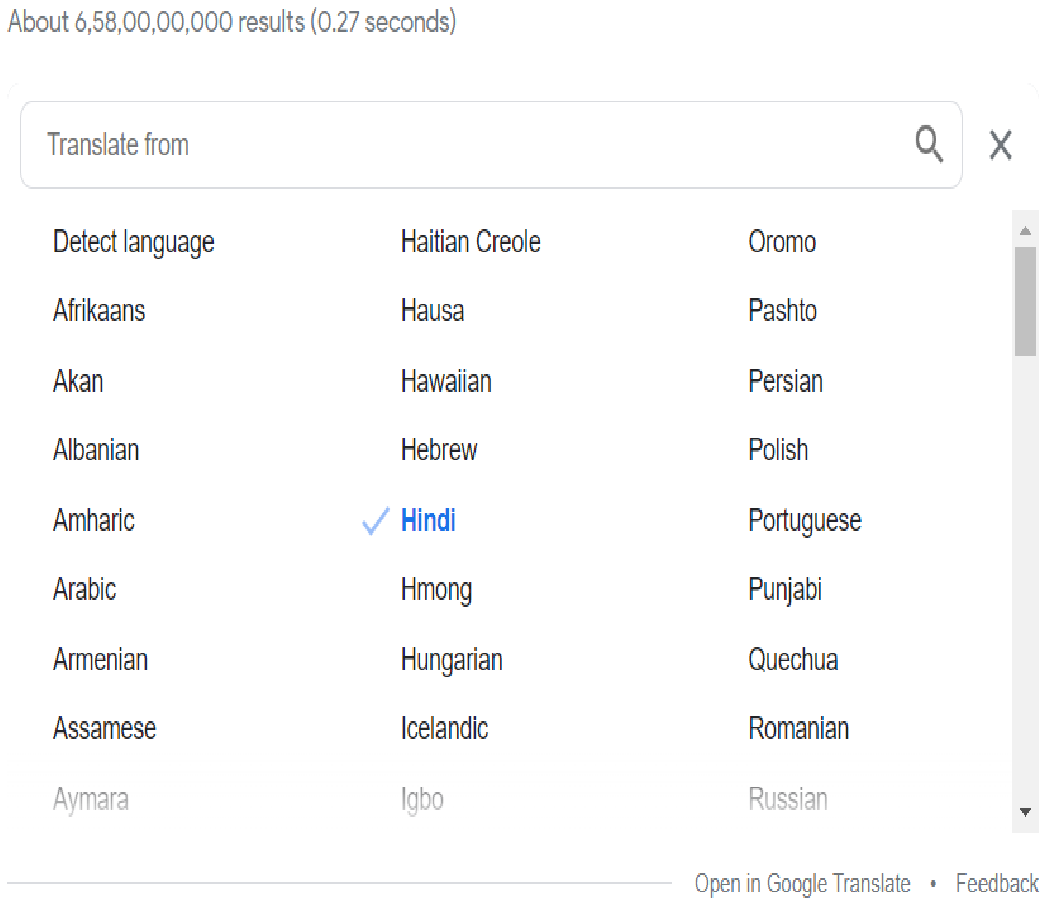
Mera Ana mushkil lag raha hai

mera aana mushkil lag raha hai

Translating मेरा आना मुश्किल लग रहा है...

I find it difficult to come

Open in Google Translate • Feedback



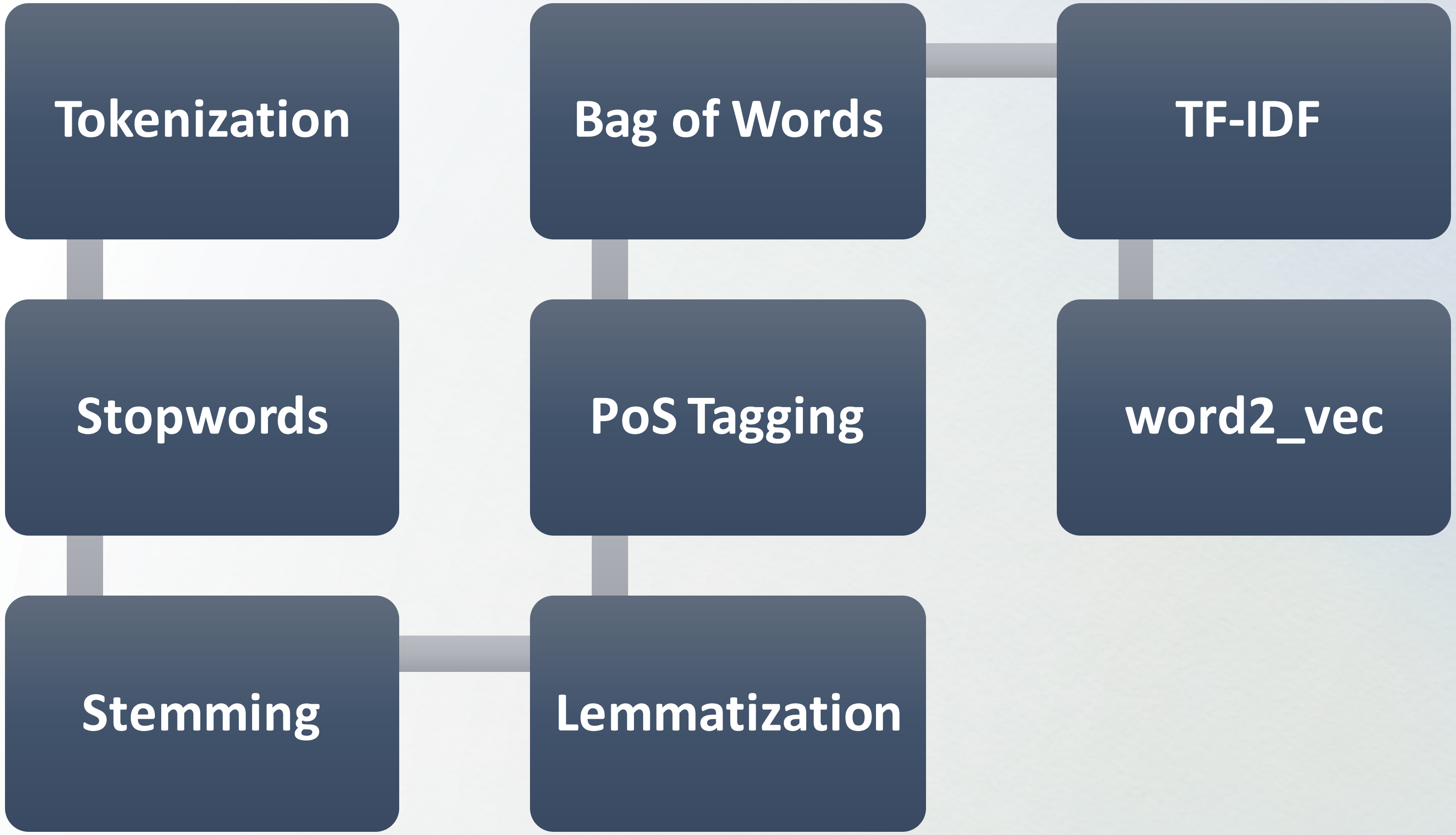
About 6,58,00,00,000 results (0.27 seconds)

Translate from

Detect language	Haitian Creole	Oromo
Afrikaans	Hausa	Pashto
Akan	Hawaiian	Persian
Albanian	Hebrew	Polish
Amharic	✓ Hindi	Portuguese
Arabic	Hmong	Punjabi
Armenian	Hungarian	Quechua
Assamese	Icelandic	Romanian
Aymara	Igbo	Russian

Open in Google Translate • Feedback

Outline

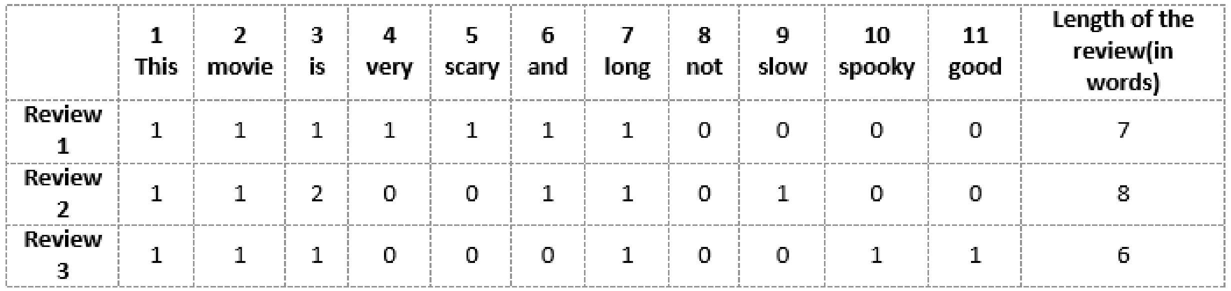

Tokenization
Stopwords
Stemming
Lemmatization
PoS Tagging
Bag of Words
TF-IDF
word2_vec

Bag of words

- 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

We will first build a vocabulary from all the unique words in the above three reviews. The vocabulary consists of these 11 words: 'This', 'movie', 'is', 'very', 'scary', 'and', 'long', 'not', 'slow', 'spooky', 'good'.

We can now take each of these words and mark their occurrence in the three movie reviews above with 1s and 0s. This will give us 3 vectors for 3 reviews:



	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]

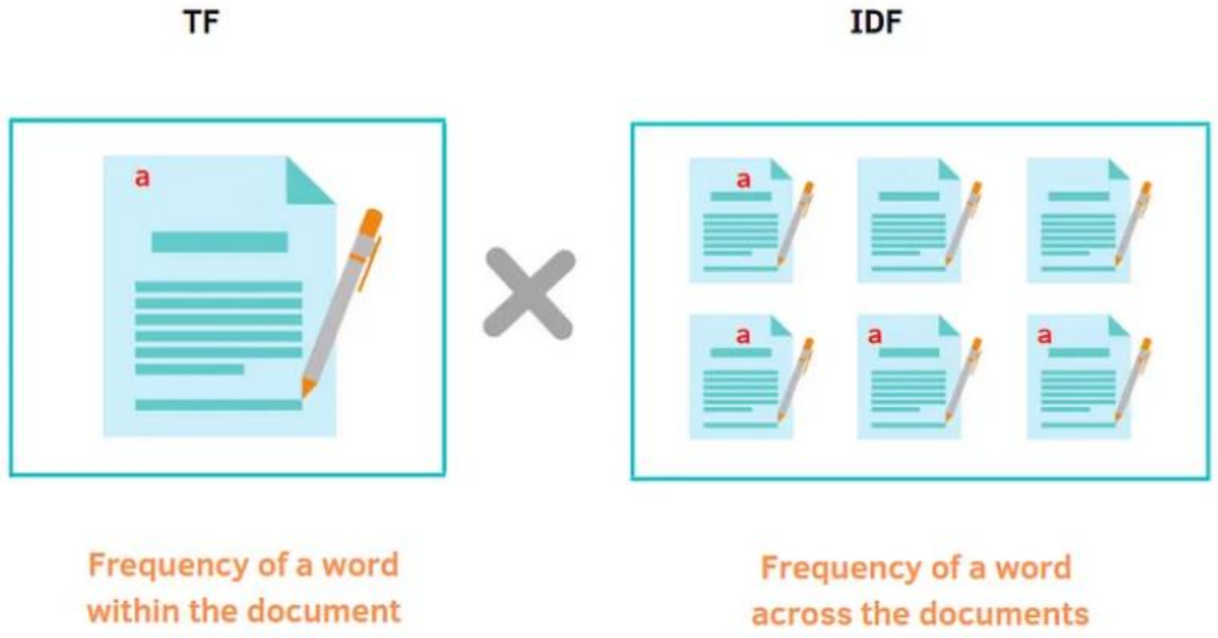
Vector of Review 2: [1 1 2 0 0 1 1 0 1 0 0]

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

Drawbacks of using a Bag-of-Words (BoW) Model

- In the above example, we can have vectors of length 11. However, we start facing issues when we come across new sentences:
- If the new sentences contain new words, then our vocabulary size would increase and thereby, the length of the vectors would increase too.
- Additionally, the vectors would also contain many 0s, thereby resulting in a sparse matrix (which is what we would like to avoid)
- We are retaining no information on the grammar of the sentences nor on the ordering of the words in the text.

TF-IDF


TF
IDF
a
Frequency of a word within the document
Frequency of a word across the documents

Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency (TF)

Let's first understand Term Frequent (TF). It is a measure of how frequently a term, t, appears in a document, d:

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

Here, in the numerator, n is the number of times the term "t" appears in the document "d". Thus, each document and term would have its own TF value.

We will again use the same vocabulary we had built in the Bag-of-Words model to show how to calculate the TF for Review #2:

Review 2: This movie is not scary and is slow

Here,

- 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('and') = 1/8
- TF('long') = 0/8 = 0
- TF('not') = 1/8
- TF('slow') = 1/8
- TF('spooky') = 0/8 = 0
- TF('good') = 0/8 = 0

We can calculate the term frequencies for all the terms and all the reviews in this manner:

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

Inverse Document Frequency (IDF)

IDF is a measure of how important a term is. We need the IDF value because computing just the TF alone is not sufficient to understand the importance of words:

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

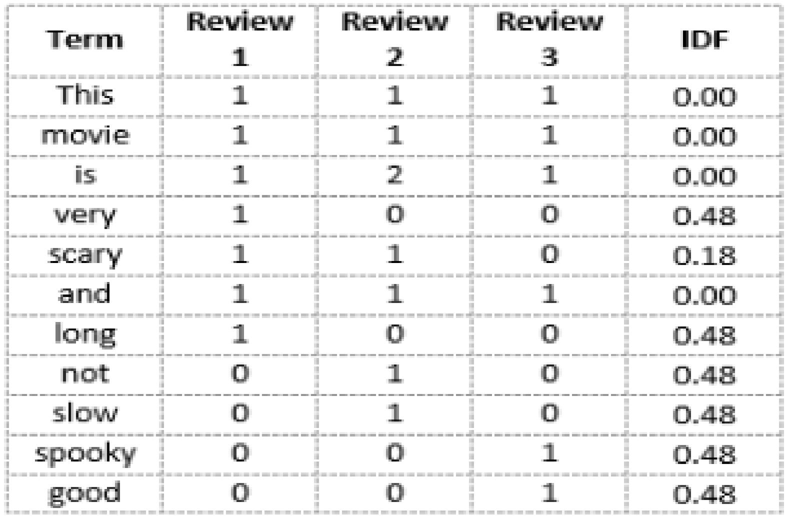
We can calculate the IDF values for the all the words in Review 2:

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

We can calculate the IDF values for each word like this. Thus, the IDF values for the entire vocabulary would be:



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

Hence, we see that words like "is", "this", "and", etc., are reduced to 0 and have little importance; while words like "scary", "long", "good", etc. are words with more importance and thus have a higher value.

We can now compute the TF-IDF score for each word in the corpus. Words with a higher score are more important, and those with a lower score are less important:

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

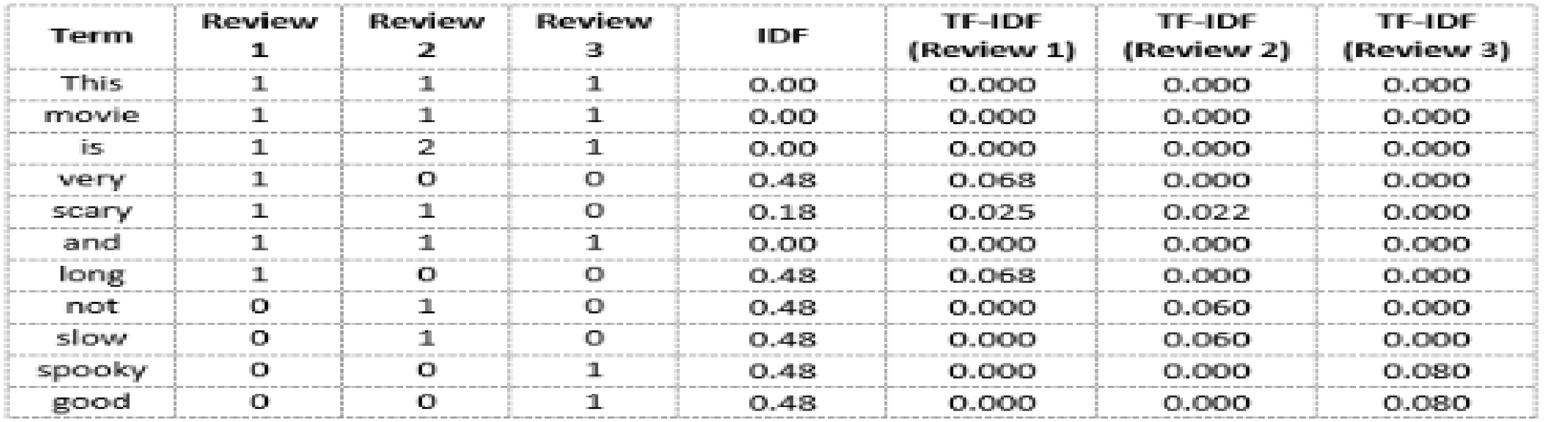
We can now calculate the TF-IDF score for every word in Review 2:

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

Similarly, we can calculate the TF-IDF scores for all the words with respect to all the reviews:



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

We have now obtained the TF-IDF scores for our vocabulary. TF-IDF also gives larger values for less frequent words and is high when both IDF and TF values are high i.e the word is rare in all the documents combined but frequent in a single document.

What is the problem with bag of words?

Huge amount of weights: Huge input vectors means a huge number of weights for a neural network.

Computationally intensive: More weights means more computation required to train and predict.

Lack of meaningful relations and no consideration for order of words: BOW is a collection of words that appear in the text or sentences with the word counts. Bag of words does not take into consideration the order in which they appear.

Word Embedding is solution to these problems

- **Embeddings translate large sparse vectors into a lower-dimensional space that preserves semantic relationships**.
- Word embeddings is a technique where individual words of a domain or language are represented as real-valued vectors in a lower dimensional space.
- **Sparse Matrix problem with BOW is solved by mapping high-dimensional data into a lower-dimensional space.**
- Lack of meaningful relationship issue of BOW is solved by placing **vectors of semantically similar items close to each other**. This way words that have similar meaning have similar distances in the vector space as shown below.

Word Embedding

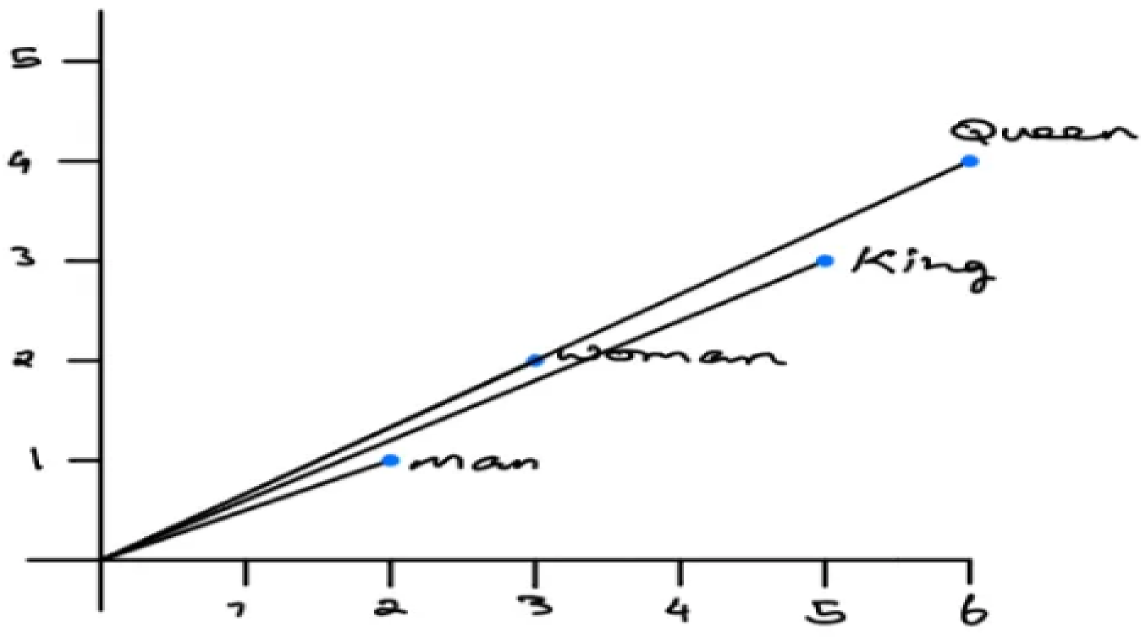
- One Hot encoding :

$$\left[\begin{array}{c}0\\0\\0\\1\\0\end{array}\right]\begin{array}{l}\\ \\ \\ \text{— 500 index}\\ \\ \end{array}$$

	BOY	Girl	king	Queen	Apple	
Gender	-1	1	-0.92	0.93	0	
Royal	0.01	0.02	0.95	0.96		
youth						
fruit	0	0	0	0	0.99	

Word2Vec

```
King      -    Man     +    Woman     =    Queen
[5,3]     -    [2,1]   +    [3, 2]    =    [6,4]
```



You can see that the words King and Queen are close to each other in position. (Image provided by the author)

Thank You