Text Representation in NLP

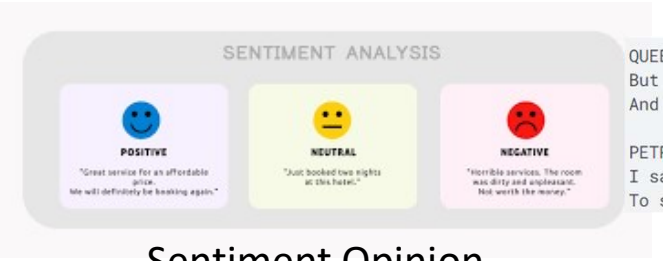
What is NLP?

- Natural Language Processing is an interdisciplinary field of computer science, artificial intelligence and computational linguistics, concerned with the interactions between computers and human languages

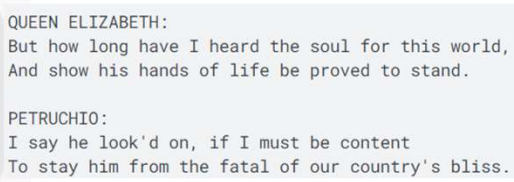
- NLP helps to analyzes, processes, efficiently retrieve information from text data and is applied in a huge range of real-world problems

Text Classification

Machine Translation



Sentiment Opinion



Text Generation

Some more NLP applications:

- **Text-based applications:**
  - Finding documents on certain topics (document classification)
  - Information extraction: extract information related events, relations, concepts
  - Complete understanding of texts, requires a deep structure analysis
  - Translation from a language to another,
  - Knowledge acquisition, Question-Answering
- **Dialogue-based applications (involves human-machine communication):**
  - Conversational Agents
  - Tutoring systems
- **Speech Processing**

Why is NLP hard?

1. **Ambiguity** at many levels:

- Include your children when baking cookies
- Local High School Dropouts Cut in Half
- Hospitals are Sued by 7 Foot Doctors
- Iraqi Head Seeks Arms
- Safety Experts Say School Bus Passengers Should Be Belted
- I saw the woman with the telescope wrapped in paper

Why is NLP hard?

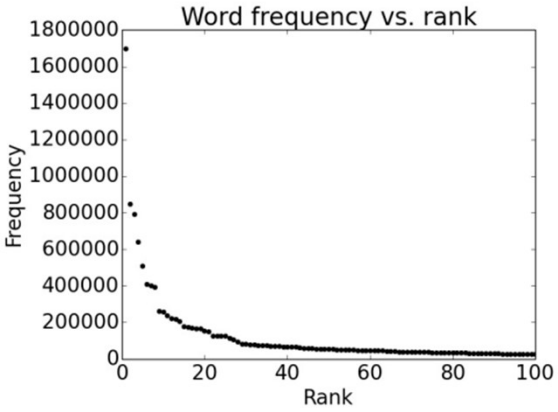
2. Sparse data due to **Zipf's Law:** $f \times r \approx k$

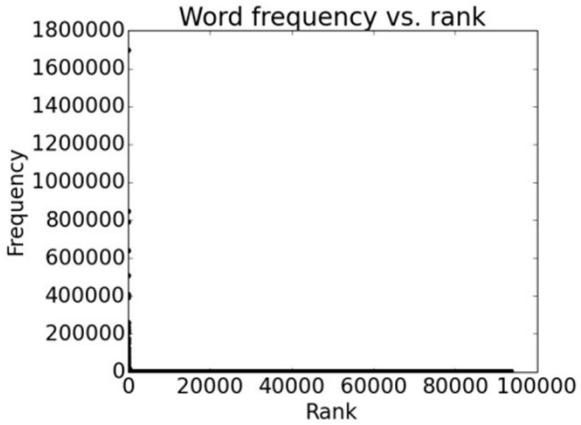
But also, out of 93638 distinct word types, 36321 occur only once.

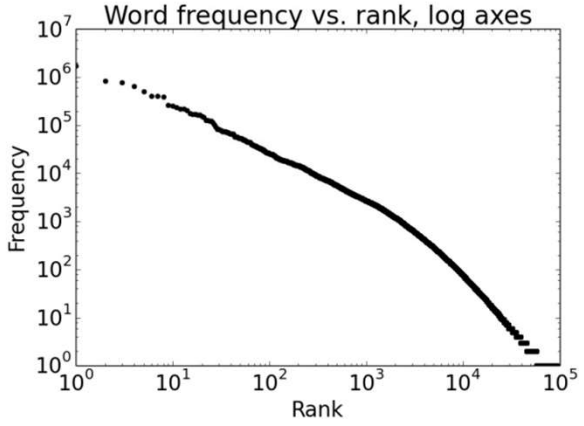
- Cornflakes, mathematicians, fuzziness, jumbling etc.

any word

Frequency	Type
1,698,599	the
849,256	of
793,731	to
640,257	and
508,560	in
407,638	that
400,467	is
394,778	a
263,040	I



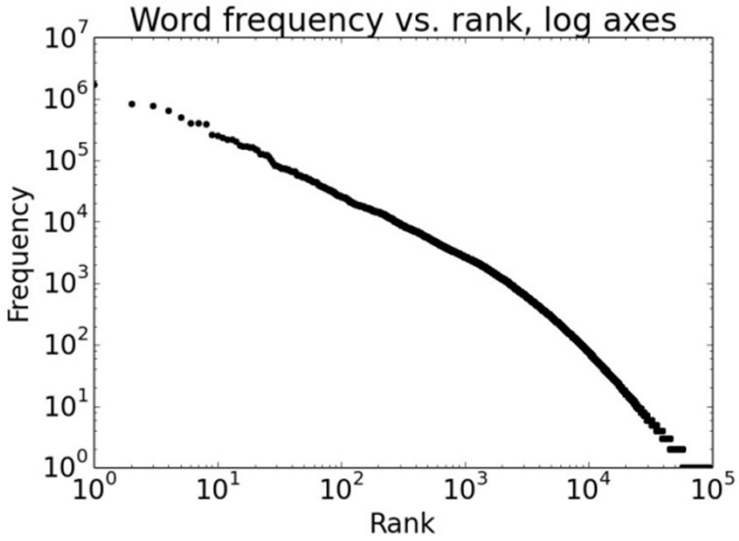




Why is NLP hard?

2. Sparse data due to **Zipf's Law:** $f \times r \approx k$

- Regardless of how large our corpus is, there will be a lot of infrequent (and zero-frequency!) words.
- This means we need to find clever ways to estimate probabilities for things we have rarely or never seen during training.



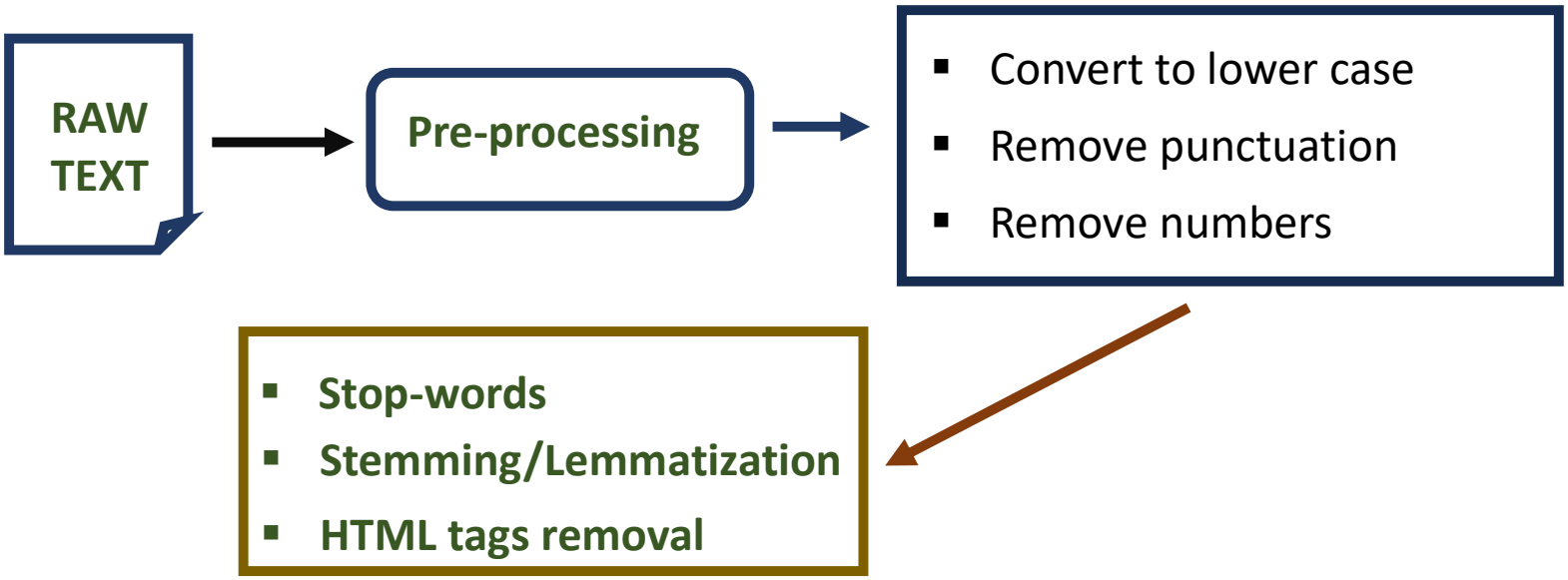
Why is NLP hard?

3. **Expressivity**: Same meaning can be expressed with different forms, or a single form can have different meaning

- Is the window still open vs. Please close the window
- She will arrive at the airport at 3 PM vs. If she leaves now, she will arrive at the airport at 3 PM
- The house is on fire vs. The team is on fire

- Overall, NLP is hard as human language is messy, ambiguous, and constantly changing. We must understand:
  - What is the “meaning” of a word or sentence?
  - How to model context?

NLP Pipeline



Stop-words

- Stop-words are words that from non-linguistic view do not carry information
  - …they have mainly functional role
  - …usually, we remove them to help the methods to perform better

- Stop-words are language dependent – examples:
  - **English**: A, ABOUT, ABOVE, ACROSS, AFTER, AGAIN, AGAINST, ALL, ALMOST, ALONE, ALONG, ALREADY, ...
  - **Hindi**: मैं, मुझको, मेरा, अपने आप को, हमने, हमारा, अपना, हम, आप, आपका, तुम्हारा, अपने आप, स्वयं, वह, इसे, उसके, …

Stemming, Lemmatization

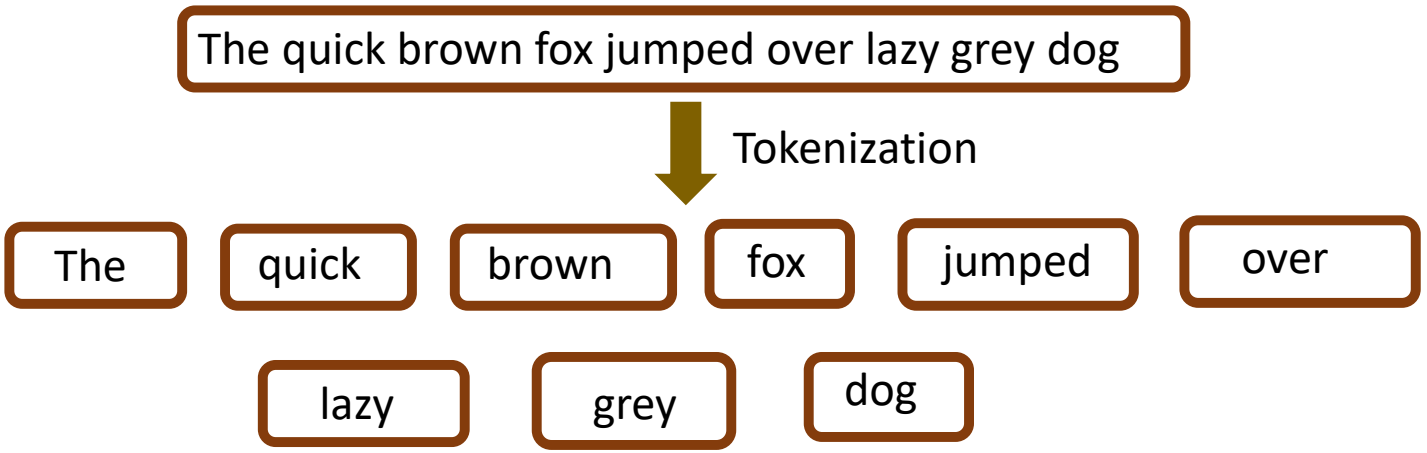
- Techniques used to reduce the texts to their root forms, as documents use different form of the same word for grammatical reasons. For e.g.: play, plays, playing, played
- Stemming is the process of producing morphological variants of a root/base word, by chopping off letters from the end until the stem is reached
  - Sleeping -> Sleep, Sleeps -> Sleep, Poorly -> Poorli
- Lemmatization reduces the words to their root stem but differs in the way that it makes sure the root word belongs to the language

	Stemming	Lematization
Improve	Improv	Improve
Improving		
Improvements		
Improved		
Improver		

NLP Pipeline

Tokenization

- Tokenization is an important pre-processing step in NLP, which involves splitting the text into minimal meaningful units/chunks.

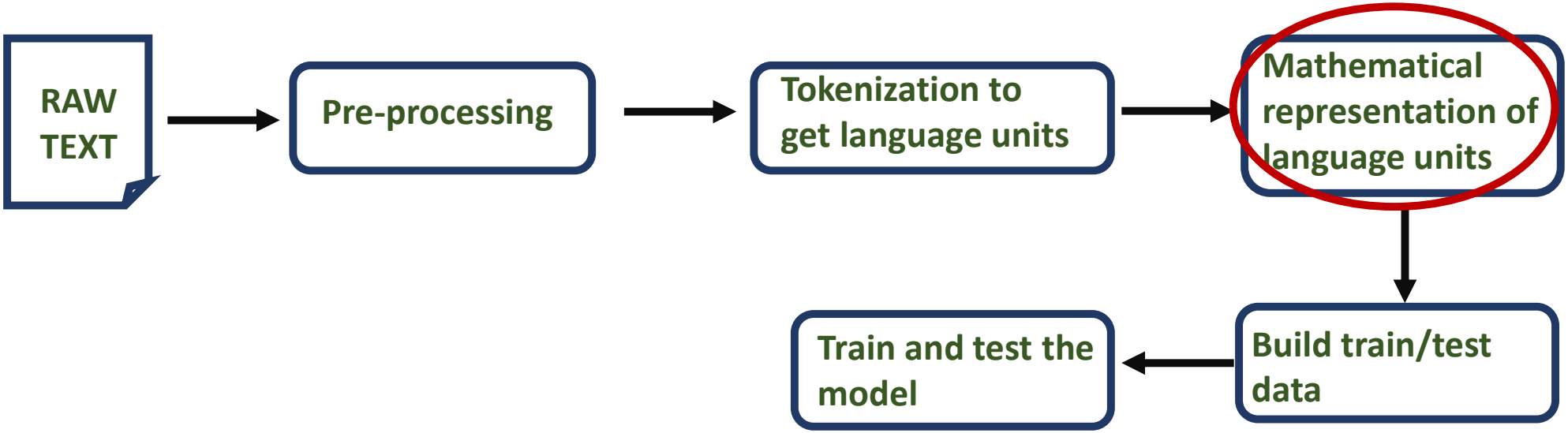


- The chunks can be words, phrases, characters, etc. Their form depends on the type of problem you are trying to solve

NLP Toolkits

- Natural Language ToolKit (NLTK): It is one of the leading platform for building Python programs to work with human language data
- Re: Regular Expression (Re) library is mostly used for text cleaning, for e.g., it lets you check if a particular string matches a given regular expression
- String: Provides additional tools to manipulate strings
- spaCy: Similar to NLTK but has more advanced features. Outperforms NLTK in handling large text data
- Gensim: Free open-source library based on Cython used for unsupervised topic modelling, document indexing, and other NLP functionalities

NLP Pipeline



Two main steps

- Most important denominator across all NLP tasks:
  - How do we represent the text as input to our models into numerical format
  - Which models are used for processing the numerical data to achieve a desired goal or task

- Representation learning is a set of techniques that learn a feature, where the raw data input is transformed to a representation that can be effectively exploited in machine learning tasks.

Feature Engineering in NLP

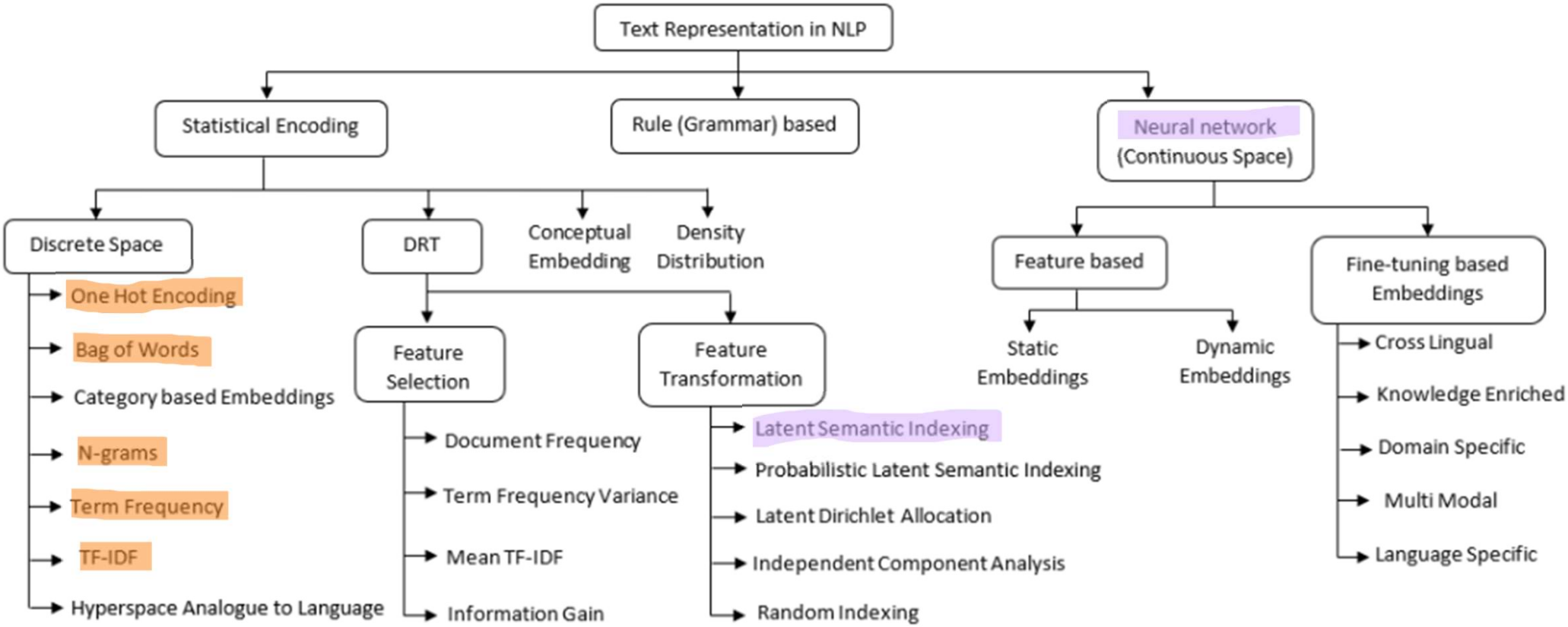


Image Source: https://www.researchgate.net/publication/369966528_A_Survey_of_Text_Representation_and_Embedding_Techniques_in_NLP

Statistical Encoding

- Words are represented using vectors of numbers, and the corpus is represented as a collection of such vectors forming a matrix
- Linear algebra operations can be used on the vector representations to manipulate the vectors and compute the distances and similarities
- The focus in statistical embedding is given to the frequency of the words used in the corpus

One Hot Embedding (OHE)

- OHE maps each word to a unique ID, making it a natural representation to start with
- Sentence: “He informed that he would leave in 20 minutes”
- Vocabulary of unique sorted words = [‘20’, ‘he’, ‘in’, ‘informed’, ‘leave’, ‘minutes’, ‘that’, ‘would’]

The words in the given sentence

	20	he	in	informed	leave	minutes	that	would
He	0	1	0	0	0	0	0	0
informed	0	0	0	1	0	0	0	0
that	0	0	0	0	0	0	1	0
he	0	1	0	0	0	0	0	0
would	0	0	0	0	0	0	0	1
leave	0	0	0	0	1	0	0	0
in	0	0	1	0	0	0	0	0
20	1	0	0	0	0	0	0	0
minutes	0	0	0	0	0	1	0	0

OHE

- It retains the ordering of words in the sentence and does not lose any information from the original text. The original text can be reconstructed from the matrix
- It ignores the context, and therefore fails to capture the relationships and meaning of the words
  - Each word is an independent unit vector, D('cat', 'refrigerator') = D('cat', 'dog')
- The cosine similarity between OHE vectors does not convey any meaningful information, as it is always a zero
- The representations are sparse using more storage space and in-turn making the models costlier to train
  - Cannot scale to large or infinite vocabularies and are computationally expensive
  - Cannot handle unseen words in the test set

Bag of Words

- It is a more concise representation with each row representing a sentence and each column representing a unique word from vocabulary

- # of rows is equal to the number of sentences, and # of columns is equal to the vocabulary size of the corpus

- Sentence 1: “The show was long and slow”
- Sentence 2: “The show was long but was captivating”
- Sentence 3: “The show was okay”

	The	show	was	long	and	slow	but	captivating	okay
Sentence 1	1	1	1	1	1	1	0	0	0
Sentence 2	1	1	2	1	0	0	1	1	0
Sentence 3	1	1	1	0	0	0	0	0	1

Bag of Words

- The matrix's row size is significantly smaller compared to OHE, and requires less time for model training
- It enables finding similarity scores among sentences or between documents, $A.B = \sum_{i=1}^{n} A_i B_i$
  - Based on exact matching, document similarity is correct only if exact words are used, do not consider synonyms

× BoW ignores the word ordering originally preserved in OHE

× Different sentences with different semantics can have the same BoW representation
  - BoW usage is limited to spam filters and sentiment analysis applications

N-gram model

- Because word order is lost, the sentence meaning is weakened.

Sentence: The cat sat on the dog
word id s: 1 14 5 3 1 12
BoW featurization:
Vector 2, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1

Sentence: The dog sat on the cat
word id's: 1 12 5 3 1 14
BoW featurization:
Vector 2, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1

- Word order is important, especially the order of nearby words.
- N-grams capture this, by modeling tuples of consecutive words.

N-grams

Sentence: The cat sat on the dog

2-grams: the-cat, cat-sat, sat-on, on-the, the-dog

Sentence: The dog sat on the cat

2-grams: the-dog, dog-sat, sat-on, on-the, the-cat

Notice how even these short n-grams "make sense" as linguistic units.

Sentence: The cat sat on the dog

3-grams: the-cat-sat, cat-sat-on, sat-on-the, on-the-dog

Which capture still more of the meaning:

Sentence: The dog sat on the cat

3-grams: the-dog-sat, dog-sat-on, sat-on-the, on-the-cat

- Remember: "the white house" will generally have very different influences from the sum of influences of "the", "white", "house".

N-grams size

- N-grams are used in document classification, clustering, and sentiment analysis, capturing important group of words (representing topic or concept) that occur frequently across the documents, classifying the documents based on the occurrence of N-grams in them
- Frequent N-grams such as ‘of the’, ‘so that’ etc., occurs too frequently but don’t possess the ability to discriminate or classify the documents
- The N-gram used to train the model might not be in the same sequence as the test data
- N-grams pose some challenges in feature set size. If the original vocabulary size is $|V|$, the number of 2-grams is $|V|^2$, while for 3-grams it is $|V|^3$

Term Frequency Embedding

- BoW considered the frequency of words, but did not consider the importance of a word relative to the other words in the document

Documents	“Politics”	Doc-Length	Normalized TF of “Politics”
Doc-A	50	150	50/150 = 0.33
Doc-B	1000	1000000	1000/1000000 = 0.001

- In Normalized Term Frequency the importance of a word relative to other words in the document is considered
- The cosine similarity of two documents helps to find whether the two documents are similar to each other
- This approach fails to consider the semantic aspects of words, such as synonyms, antonyms, analogies, etc as words are considered independent while computing the frequencies

TF-IDF Embedding

- The importance of a word relative to other words in the document is captured by NTF, but the uniqueness of the word relative to other words in the corpus is not demonstrated
- Erroneous results obtained in similarity, as common words ('and', 'of', 'the', 'that') prevalent across the corpus will lead to high similarity scores
- The similarity score should be penalized provided the words which are not unique to the documents are considered – Inverse Document Frequency, $\boldsymbol{IDF_i = \log \frac{N}{df_i}}$
- NTF → Increases the similarity between vectors if similar words are shared in similar proportions
- IDF → Reduces the similarity between vectors provided similar words are not unique to these documents

TF-IDF Embedding

- TF-IDF score assigns a numeric value to the importance of the word in a document, given its usage across the entire corpus

$$w_{i,j} = tf_{i,j}\, idf_i = tf_{i,j}\left(\log \frac{N}{df_i}\right)$$

	house	apartment	repair	maintenance	labor-cost	safety	commute	school	grocery	env
doc-A	125	210	100	200	32	0	0	0	0	0
doc-B	150	235	0	0	0	71	55	81	32	64
doc-C	175	261	150	350	21	0	0	0	0	0
doc-D	140	284	0	0	0	98	44	76	28	53
IDF	1	1	2	2	2	2	2	2	2	2

Take Home Message

- The NLP techniques discussed till now, treat words as atomic symbols. Every 2 words are therefore equally apart
- They do not have any notion of either syntactic or semantic similarity between parts of language
- This is one of the chief reasons for the poor/mediocre performance of NLP based models

But this has changed dramatically in last few years