Structured and Unstructured Data

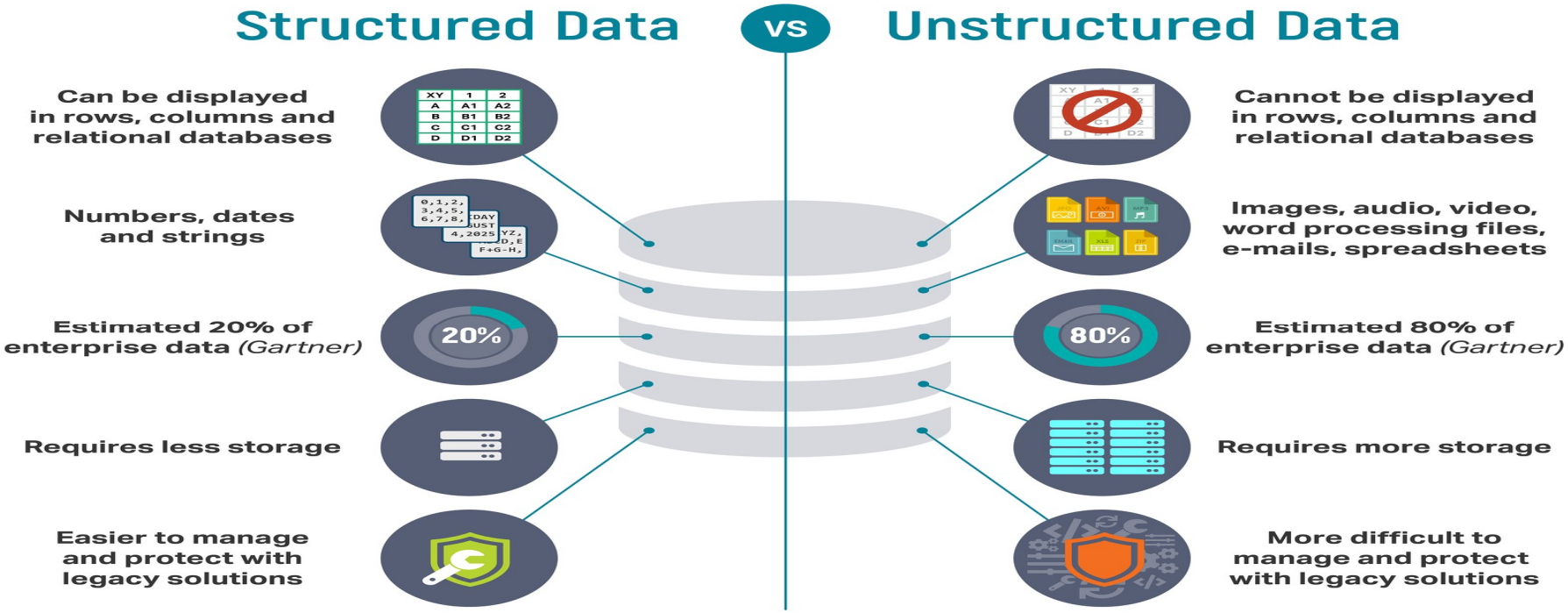
Dr. JASMEET SINGH
ASSISTANT PROFESSOR, CSED
TIET, PATIALA

Data

- Data is a unprocessed fact, value, text, sound or picture that is not being interpreted and analyzed.
- Data is the most important part of all Data Analytics, Machine Learning, Artificial Intelligence.
- Big Enterprises are spending lots of money just to gather as much certain data as possible.

In 2021, Facebook acquire WhatsApp by paying a huge price of $19 billion

Structured vs. Unstructured Data in ML



Structured Data in ML

- Structured data in Machine Learning is stored in the form of rows and columns.
- Each instance of structured data represents a feature/attribute.
- For instance a well known ML dataset, *Iris,* has five features about species namely Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, Species.

Types of Features

Quantitative (Numerical)

- Quantitative data being measured.
- Can be **Continuous** (infinite values- length, mass, weight) or **Discrete** (finite integer values- no. of workers absent)

Binary

- Which have two values (0/1, yes/no, true/false)
- Examples- Marital Status, Permanent Employee, etc.

Qualitative Nominal (Categorical)

- Categorical data where order of categories is arbitrary
- Example- account type (savings, current, fixed term, etc).

Types of Features Contd....

Qualitative Ordinal (Ranked)

- Categorical data where there is some logical ordering of categories
- Example: Size (S, M, L, XL, XXL, etc.), Likert Scale (Strongly Disagree, Disagree, Neutral, Agree, etc.)

Interval

- Has meaningful intervals between measurement.
- No true starting point (zero)
- Example- Temperature

Ratio

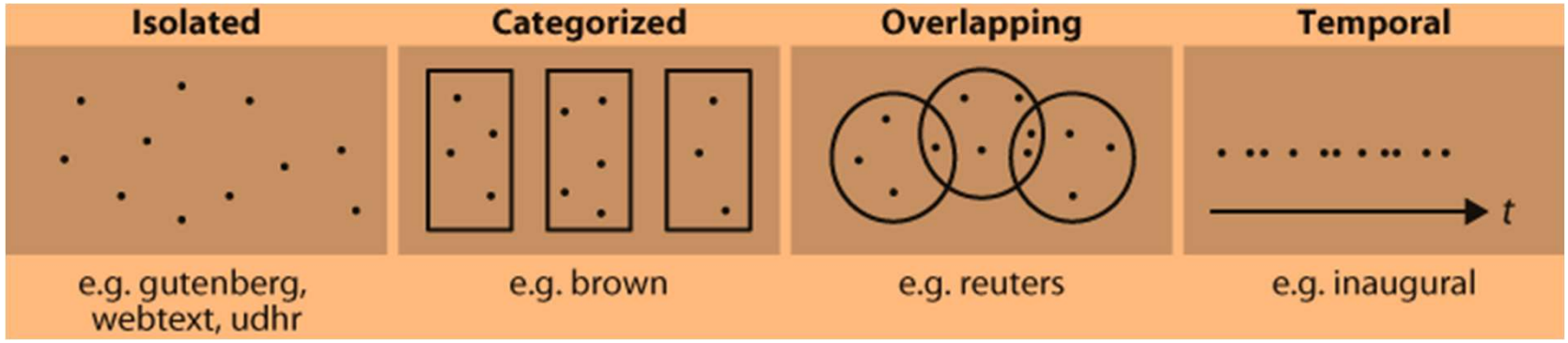
- Have highest level of measurement.
- Ratios between measurements and intervals are meaningful because there is true starting point (zero)
- Example: weight, Age, height

Unstructured Textual Data

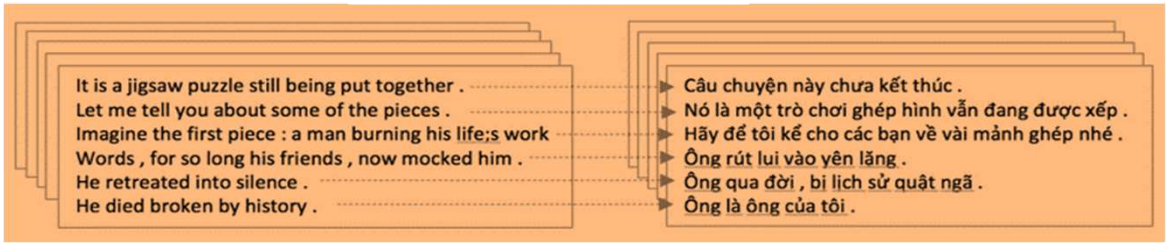
- Textual Machine Learning models typically uses large bodies of linguistic data, or **corpora.**

- A **computer corpus** is a large body of machine-readable texts.

Introduction to Textual Data Contd....



Parallel

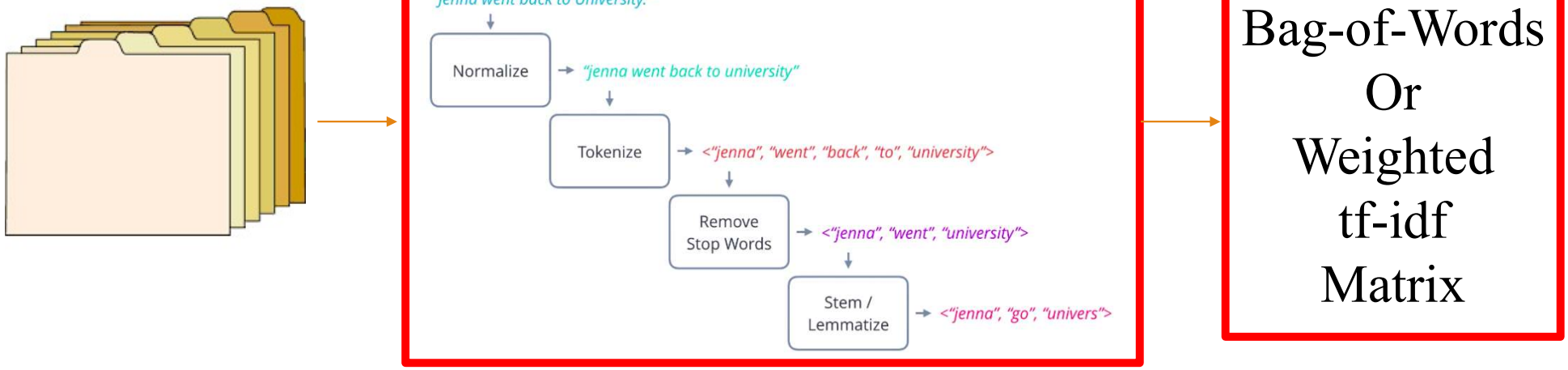


e.g. Glottopedia

Introduction to Textual Data Contd….

- **Isolated**: The simplest kind of corpus is a collection of isolated texts with no particular organization.
- **Categorical**: Some corpora are structured into categories, such as genre (Brown Corpus)
- **Overlapping**: some categorizations overlap, such as topic categories (Reuters Corpus)
- **Temporal**: corpora represent language use over time (Inaugural Address Corpus)
- **Parallel**: corpus of similar text but written in more than one language (he corpora are the translations of each other)

Corpus to Features



Pre-processing- Step 1: Normalization

- Normalization step involves removal of special symbols, numbers, urls, and converting the whole text into same case (i.e. either lower case or upper case).
- The advantages of normalization is that it reduces the dimensionality of feature set (as the words of the text will become features in the feature vector). So, a word in lower or upper case will represent the same feature.

Example:

- *Consider the following Input text:*

'Hi , My name is Rajeshwar Singh . My email id is rajesh@gmail.com. I am 34 years old . I am studying Conversational AI'

- *The output of the normalization phase on the given input text will be:*

hi my name is rajeshwar singh my email id is i am years old i am studying conversational ai (all numbers, urls, special symbols removed and text converted to lower case)

Pre-Processing- Step 2: Tokenization

- Tokenization involves splitting the text as set of sentences and each sentence as set of words.
- This step is required in order to perform next two steps of pre-processing (as in the next two steps certain operations are to be applied on the individual words).

Example:

The tokenized input text (output of step 1 in previous slide) will be tokenized as follows:

'hi', 'my', 'name', 'is', 'rajeshwar', 'singh', 'my', 'email', 'id', 'is', 'i', 'am', 'years', 'old', 'i', 'am', 'studying', 'conversational', 'ai'

Pre-Processing-
Step 3: Stop-word Removal

- Stop words are basically a set of commonly used words in any language.
- Stop words are high frequency words of the language that do not have any semantic importance.
- For example, words like of, but, any, I, me , she are some stop words of English language.
- Stop words are removed in the pre-processing phase as these words do not provide any information in the feature vector, and hence the dimensionality of the feature matrix is reduced.

- Example

The output of the stop-word removal phase on the tokenized text (output of step 2 in the previous step) is as follows:

'hi', 'name', 'rajeshwar', 'singh', 'email', 'id', 'years', 'old', 'studying', 'conversational', 'ai'

Pre-Processing
Step 4: Morphological Analysis

- In this pre-processing step, the morphological variant words must be mapped to their surface/base word form.
- For example, *player, plays, played, playing are morphological variants and are mapped to the base word play.*
- The purpose of morphological analysis is as follows:

1. It matches the variant word forms in the documents and the queries to the base word form, thereby solving the problem of vocabulary mismatch.
2. The conflation of the words having the same stem into single class and hence reduces the size of the vocabulary/feature vector.

- *Stemming and Lemmatization* methods are employed to handle these morphological variants in NLP.

Pre-Processing
Step 4: Morphological Analysis (Contd…)

- The difference between stemming and lemmatization are as follows:

Stemming	Lemmatization
1. The output product of stemming is 'stem'.	1. The output of lemmatization is 'lemma'.
2. The stem may be valid, fully understandable word (free stem) or invalid word which requires an affix to make a word (bound stem). For instance, 'perish' is a free stem and 'dur' is a bound stem.	2. Lemmas, on the other hand, are valid linguistic components and a dictionary form of a lexeme. Lexeme corresponds to a collection of all the word variant forms that have similar meaning and lemma is one particular variant used to represent the lexeme.
3. Stemmers thus handle both inflectional and derivational variations (inflectional variants –related in meaning; derivational variants- may not be related in meanings; For example- departs, departed , departing are inflectional variant where as departure is derivational variant)	3. Lemmatizers can only handle inflectional variations.

Pre-Processing
Step 4: Morphological Analysis (Contd…)

Stemming	Lemmatization
4. The algorithms / programs which perform stemming are called stemmers	4. The algorithms / programs which perform lemmatization are called lemmatizer.
5. Stemming is a simpler, easier and faster process that makes use of rules to determine the stem without considering the vocabulary, context of the word or part-of-speech.	5. Lemmatization is a comparatively complex procedure which first determines the part-of-speech and context of the word to return the lemma.
6. Stemmers can be developed in an unsupervised manner (using machine learning techniques) without the use of any linguistic resource or expert.	6. Currently, no method has been proposed in the literature for training a lemmatizer in an unsupervised manner.
7. Stemmers are used in applications where meanings are not important for example text classification, sentiment analysis, information retrieval, etc.	7. Lemmatizers are used in applications where meanings are important for example word sense disambiguation, machine translation, etc.

Pre-Processing
Step 4: Morphological Analysis (Contd…)

- While converting corpus to bag-of words (feature matrix), stemmer is used over lemmatizer
- As stemmer is fast and simple, although it may not return linguistically valid word (for example, truncated truncates, truncating are stemmed to *truncat*).
- Porter Stemmer, Lovin Stemmer, Lancaster Stemmer, Dawson Stemmer are number of stemmers proposed in literature. Although, Porter stemmer is the most popular stemmer.

- Example:

The output of the stemming phase on the output of Step 3 (stop-word removal) of the input text after application of Porter Stemmer is as follows:

'hi name rajeshwar singh email id year old studi convers ai'

Feature Vector

- After pre-processing the text (documents) of the corpus, we can construct feature matrix from the pre-processed corpus and hence convert the unstructured data to structured form.

- Following two types of feature matrix are broadly used for textual data:

1. Term-Document Matrix (TDM)/ Document-Term Matrix (DTM)/Bag-of-Words (BoW)
2. Term-Term Matrix (TTM)/Co-occurrence matrix

TDM/DTM/BoW

- A Term- Document Matrix (TDM) is a matrix whose rows represents the vocabulary (unique words) of the corpus and columns represent the documents of the corpus. Each ijth entry of the matrix represents frequency of occurrence of the ith word in the jth document.
- A Document-Term (DTM) is a transpose of TDM matrix wherein the rows represent the documents and columns represent the terms.
- It is called a "bag" of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

Variants of TDM

- Depending upon how the ijth entry of the TDM matrix is filled, there are different variants of TDM matrix:

1. Binary
2. Term Frequency
3. Term Frequency with length normalization
4. Term Frequency wit maximum normalization
5. Term Frequency with log normalization
6. Term Frequency-Inverse Document Frequency

1. Binary-Term Document Matrix

- In a binary TDM, each ijth entry of the matrix is either 1 or 0. It is 1 if the ith term is present in the jth document else it is 0.

$$tdm_{ij} = \begin{cases} 1 \; if \; i^{th} \; term \; is \; present \; in \; j^{th} \; doc \\ 0 \; otherwise \end{cases}$$

- For example, consider the corpus with following three documents:

D1: Data Science is an important field of science.

D2: The cars are driven on the roads

D3: The trucks are driven on the highways.

- The binary TDM for the given corpus is shown below (with no stop-word removal and stemming in the pre-processing phase)

	D1	D2	D3
an	1	0	0
are	0	1	1
cars	0	1	0
data	1	0	0
driven	0	1	1
field	1	0	0
highways	0	0	1
important	1	0	0
is	1	0	0
of	1	0	0
on	0	1	1
road	0	1	0
sceince	1	0	0
the	0	1	1
trucks	0	0	1

2. Term Frequency-Term Document Matrix

- In a term frequency TDM, each ijth entry of the matrix is either f_{ij} or 0. where f_{ij} represent frequency of occurrence of ith term in the jth document.

$$tdm_{ij} = \begin{cases} f_{ij} \; if \; i^{th} \; term \; is \; present \; in \; j^{th} \; doc \\ 0 \; otherwise \end{cases}$$

- For example, consider the corpus with following three documents:

D1: Data Science is an important field of science.

D2: The cars are driven on the roads

D3: The trucks are driven on the highways.

- The term frequency TDM for the given corpus is shown below (with no stop-word removal and stemming in the pre-processing phase).

	D1	D2	D3
an	1	0	0
are	0	1	1
cars	0	1	0
data	1	0	0
driven	0	1	1
field	1	0	0
highways	0	0	1
important	1	0	0
is	1	0	0
of	1	0	0
on	0	1	1
road	0	1	0
sceince	2	0	0
the	0	2	2
trucks	0	0	1

3. Term Frequency with length Normalization-Term Document Matrix

- In a term frequency with length normalization TDM, each ij^{th} entry of the term frequency matrix is normalized with the length of document i.e. total numbers of terms in the document

$$tdm_{ij} = \begin{cases} \dfrac{f_{ij}}{\sum_{t' \in j} f_{t'j}} & if\ i^{th}\ term\ is\ present\ in\ j^{th}\ doc \\ 0\ otherwise \end{cases}$$

- For example, consider the corpus with following three documents:

D1: Data Science is an important field of science.

D2: The cars are driven on the roads

D3: The trucks are driven on the highways.

- The term frequency with length normalization TDM for the given corpus is shown below (with no stop-word removal and stemming in the pre-processing phase).

	D1	D2	D3
an	1/8	0	0
are	0	1/7	1/7
cars	0	1/7	0
data	1/8	0	0
driven	0	1/7	1/7
field	1/8	0	0
highways	0	0	1/7
important	1/8	0	0
is	1/8	0	0
of	1/8	0	0
on	0	1/7	1/7
road	0	1/7	0
sceince	1/4	0	0
the	0	2/7	2/7
trucks	0	0	1/7

4. Term Frequency with Max Normalization-Term Document Matrix

- In a term frequency with max normalization TDM, each ij[th] entry of the term frequency matrix is normalized with the maximum frequency of the terms of the document.

$$tdm_{ij} = \begin{cases} \dfrac{f_{ij}}{max_{t' \in j}(f_{t'j})} & if\ i^{th}\ term\ is\ present\ in\ j^{th}\ doc \\ 0\ otherwise \end{cases}$$

- For example, consider the corpus with following three documents:

D1: Data Science is an important field of science.

D2: The cars are driven on the roads

D3: The trucks are driven on the highways.

- The term frequency with max normalization TDM for the given corpus is shown below (with no stop-word removal and stemming in the pre-processing phase).

	D1	D2	D3
an	1/2	0	0
are	0	1/2	1/2
cars	0	1/2	0
data	1/2	0	0
driven	0	1/2	1/2
field	1/2	0	0
highways	0	0	1/2
important	1/2	0	0
is	1/2	0	0
of	1/2	0	0
on	0	1/2	1/2
road	0	1/2	0
sceince	1	0	0
the	0	1	1
trucks	0	0	1/2

5. Term Frequency with Log Normalization-Term Document Matrix

- In a term frequency with log normalization TDM, each ij[th] entry of the term frequency matrix is normalized by taking log of $(1+ f_{ij})$

$$tdm_{ij} = \log(1 + f_{ij})$$

- For example, consider the corpus with following three documents:

D1: Data Science is an important field of science.

D2: The cars are driven on the roads

D3: The trucks are driven on the highways.

- The term frequency with log normalization TDM for the given corpus is shown below (with no stop-word removal and stemming in the pre-processing phase).

	D1	D2	D3
an	0.30103	0	0
are	0	0.30103	0.30103
cars	0	0.30103	0
data	0.30103	0	0
driven	0	0.30103	0.30103
field	0.30103	0	0
highways	0	0	0.30103
important	0.30103	0	0
is	0.30103	0	0
of	0.30103	0	0
on	0	0.30103	0.30103
road	0	0.30103	0
sceince	0.477121	0	0
the	0	0.477121	0.477121
trucks	0	0	0.30103

6. Term Frequency- Inverse Document Frequency TDM

- Term Frequency- Inverse Document Frequency (often referred as tf-idf) is one of the most important and commonly used variant of term document matrix.
- Each entry of tf-idf is computed as:

$$tf - idf_{ij} = tf_{ij} \times idf_i$$

- where tf_{ij} is the term frequency (computed using any of the first five variants) and is called a document level feature because for each term (feature) there is a different value in each document.
- and idf_i is the inverse document frequency of term i in the corpus and is a corpus level feature i.e. for each term (feature) there is only one value which is computed from corpus and not individual documents.

6. Term Frequency- Inverse Document Frequency TDM (Contd.....)

- Inverse Document Frequency (idf) of a term is computed as follows:

$$idf_i = log\left(\frac{|D|}{|D_i|}\right)$$

Where |D| represents total number of documents in the corpus and |Di| represents number of documents in which i^{th} term is present.

- Thus, idf of a term is zero if it is present in all the terms; or very low if it is present in majority of documents but it is high if it is present in lesser number of documents.

6. Term Frequency- Inverse Document Frequency TDM (Contd.....)

- To summarize the key intuition motivating TF-IDF is the importance of a term is inversely related to its frequency across documents.
- TF gives us information on how often a term appears in a document and IDF gives us information about the relative rarity of a term in the collection of documents. By multiplying these values together we can get our final TF-IDF value.
- The higher the TF-IDF score the more important or relevant the term is; as a term gets less relevant, its TF-IDF score will approach 0.

6. Term Frequency- Inverse Document Frequency TDM (Contd.....)

- Consider the corpus of three documents:

D1: Data Science is an important field of science. **D2**: The cars are driven on the roads **D3**: The trucks are driven on the highways

- The tf-idf matrix for the corpus is as follows (with no stop-word removal and stemming in the pre-processing phase). 3rd variant is used for term frequency.

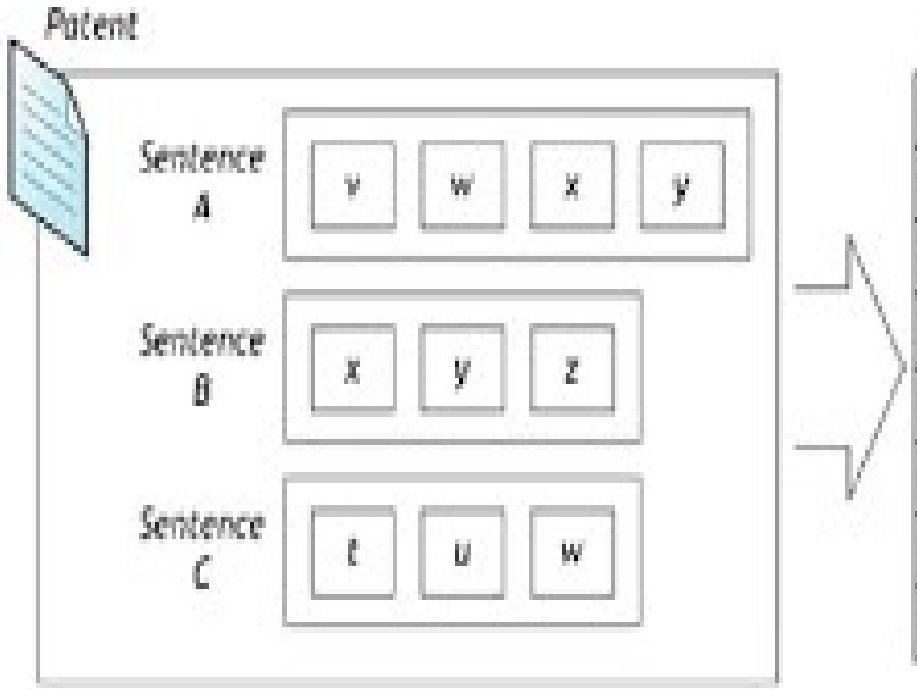
- Thus in the first doc, *data, science, important, field* are keywords. In the second doc, *cars, driven, road* are keywords. In the third doc, *trucks, driven, highways* are keywords (as rest all terms have 0 tf-idf score or are stop-words which will be eliminated in pre-processing phase).

	Term Frequnecy (TF)			IDF	Term Frequency- Inverse Document Freqeuncy (TF-IDF)		
	D1	D2	D3		D1	D2	D3
an	1/8	0	0	log(3/1)=0.4771	0.0596	0.0000	0.0000
are	0	1/7	1/7	log(3/2)=0.1761	0.0000	0.0252	0.0252
cars	0	1/7	0	log(3/1)=0.4771	0.0000	0.0682	0.0000
data	1/8	0	0	log(3/1)=0.4771	0.0596	0.0000	0.0000
driven	0	1/7	1/7	log(3/2)=0.1761	0.0000	0.0252	0.0252
field	1/8	0	0	log(3/1)=0.4771	0.0596	0.0000	0.0000
highways	0	0	1/7	log(3/1)=0.4771	0.0000	0.0000	0.0682
important	1/8	0	0	log(3/1)=0.4771	0.0596	0.0000	0.0000
is	1/8	0	0	log(3/1)=0.4771	0.0596	0.0000	0.0000
of	1/8	0	0	log(3/1)=0.4771	0.0596	0.0000	0.0000
on	0	1/7	1/7	log(3/2)=0.1761	0.0000	0.0252	0.0252
road	0	1/7	0	log(3/1)=0.4771	0.0000	0.0682	0.0000
sceince	1/4	0	0	log(3/1)=0.4771	0.1193	0.0000	0.0000
the	0	2/7	2/7	log(3/2)=0.1761	0.0000	0.0503	0.0503
trucks	0	0	1/7	log(3/1)=0.4771	0.0000	0.0000	0.0682

Term-Term Matrix

- Term-Term Matrix (TTM) is another method to construct a feature matrix from textual documents besides TDM.
- As the name suggests, the rows and columns of the TTM represents the vocabulary (unique terms) of the matrix. Each ij^{th} entry of the TTM matrix represents co-occurrence frequency i.e. number of documents in which i^{th} and j^{th} terms occurs. So, a TTM is called co-occurrence matrix.
- If the order of TDM is mXn (where m represents vocabulary size and n represents number of documents), then order of TTM is mXm.
- Unlike TDM, co-occurrence matrix also capture some sort of semantic/contextual information from the corpus; i.e., terms which co-occur quite frequently (whose co-occurrence frequency is high) are usually related to each other contextually.

Term-Term Matrix (Contd….)



Co-occurrence matrix

	v	w	x	y	z	t	u
v	0	1	1	1	0	0	0
w	1	0	1	1	0	1	1
x	1	1	0	2	1	0	0
y	1	1	2	0	1	0	0
z	0	0	1	1	0	0	0
t	0	1	0	0	0	0	1
u	0	1	0	0	0	1	0

Term-Term Matrix (Contd…)

How to construct Co-occurrence/ TTM?

- Co-occurrence matrix can be computed from binary TDM as follows:

1. Construct a binary-term document matrix (TDM) from a given corpus.
2. Multiply TDM with its transpose.
3. Set the diagonal elements to 0.

Term-Term Matrix (Example)

- Consider the corpus of three documents:

D1: Data Science is an important field of science.

D2: The cars are driven on the roads

D3: The trucks are driven on the highways

The binary term matrix for the given corpus is

- **Step 1:** The binary TDM for the given corpus is shown below (with no stop-word removal and stemming in the pre-processing phase):

	D1	D2	D3
an	1	0	0
are	0	1	1
cars	0	1	0
data	1	0	0
driven	0	1	1
field	1	0	0
highways	0	0	1
important	1	0	0
is	1	0	0
of	1	0	0
on	0	1	1
road	0	1	0
sceince	1	0	0
the	0	1	1
trucks	0	0	1

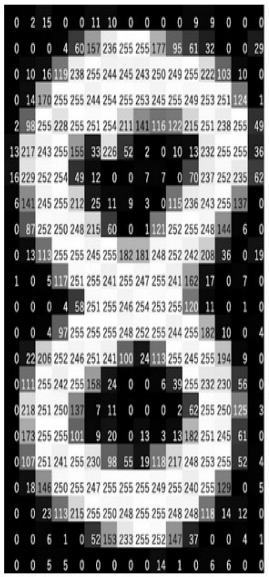
Term-Term Matrix (ExampleContd….)

- **Step 2 and Step3:** After multiplication of TDM matrix with its transpose and setting diagonal elements to 0, we obtain co-occurrence matrix.

	an	are	cars	data	driven	field	highways	important	is	of	on	road	sceince	the	trucks
an	0	0	0	1	0	1	0	1	1	1	0	0	1	0	0
are	0	0	1	0	2	0	1	0	0	0	2	1	0	2	1
cars	0	1	0	0	1	0	0	0	0	0	1	1	0	1	0
data	1	0	0	0	0	1	0	1	1	1	0	0	1	0	0
driven	0	2	1	0	0	0	1	0	0	0	2	1	0	2	1
field	1	0	0	1	0	0	0	1	1	1	0	0	1	0	0
highways	0	1	0	0	1	0	0	0	0	0	1	0	0	1	1
important	1	0	0	1	0	1	0	0	1	1	0	0	1	0	0
is	1	0	0	1	0	1	0	1	0	1	0	0	1	0	0
of	1	0	0	1	0	1	0	1	1	0	0	0	1	0	0
on	0	2	1	0	2	0	1	0	0	0	0	1	0	2	1
road	0	1	1	0	1	0	0	0	0	0	1	0	0	1	0
sceince	1	0	0	1	0	1	0	1	1	1	0	0	0	0	0
the	0	2	1	0	2	0	1	0	0	0	2	1	0	0	1
trucks	0	1	0	0	1	0	1	0	0	0	1	0	0	1	0

Unstructured Image Data

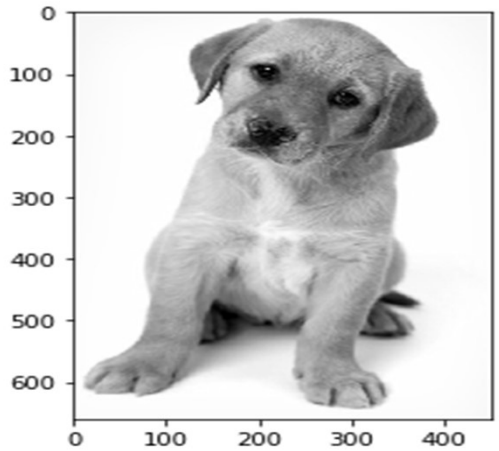
- Machines store images in the form of matrix of numbers.
- These numbers are the pixel intensities in the image.
- Smaller numbers (closer to zero) represent black, and larger numbers (closer to 255) denote white.
- The size of this matrix depends on the number of pixels we have in any given image (i.e. *(height x width)*).
- Hence total number of features are same as number of pixels.



```
  0   2  15   0   0  11  10   0   0   0   0   9   9   0   0   0
  0   0   0   4  60 157 236 255 255 177  95  61  32   0   0  29
  0  10  16 119 238 255 244 245 243 250 249 255 222 103  10   0
  0  14 170 255 255 244 254 255 253 245 255 249 253 251 124   1
  2  98 255 228 255 251 254 211 141 116 122 215 251 238 255  49
 13 217 243 255 155  33 226  52   2   0  10  13 232 255 255  36
 16 229 252 254  49  12   0   0   7   7   0  70 237 252 235  62
  6 141 245 255 212  25  11   9   3   0 115 236 243 255 137   0
  0  87 252 250 248 215  60   0   1 121 252 255 248 144   6   0
  0  13 113 255 255 245 255 182 181 248 252 242 208  36   0  19
  1   0   5 117 251 255 241 255 247 255 241 162  17   0   7   0
  0   0   0   4  58 251 255 246 254 253 255 120  11   0   1   0
  0   0   4  97 255 255 255 248 252 255 244 255 182  10   0   4
  0  22 206 252 246 251 241 100  24 113 255 245 255 194   9   0
  0 111 255 242 255 158  24   0   0   6  39 255 232 230  56   0
  0 218 251 250 137   7  11   0   0   0   2  62 255 250 125   3
  0 173 255 255 101   9  20   0  13   3  13 182 251 245  61   0
  0 107 251 241 255 230  98  55  19 118 217 248 253 255  52   4
  0  18 146 250 255 247 255 255 255 249 255 240 255 129   0   5
  0   0  23 113 215 255 250 248 255 255 248 248 118  14  12   0
  0   0   6   1   0  52 153 233 255 252 147  37   0   0   4   1
  0   0   5   5   0   0   0   0   0  14   1   0   6   6   0   0
```

Method #1: Grayscale Pixel Values as Features

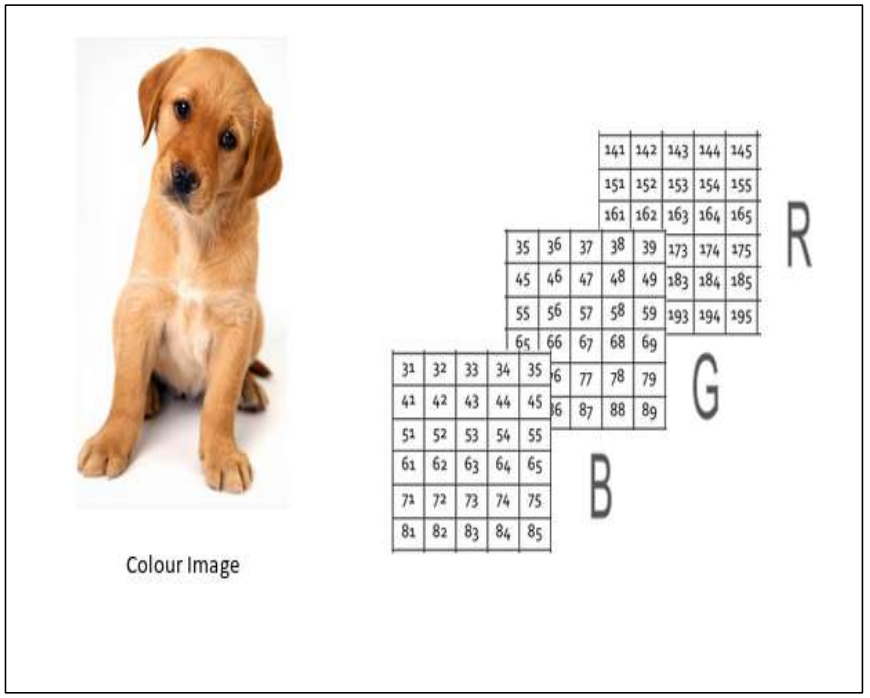
- *The simplest way to create features from an image is to use these raw pixel values as separate features.*
- The image shape shown in the figure is 650 x 450.
- Hence, the number of features should be 297,000.



(297000,)
array([0.96470588, 0.96470588, 0.96470588, ...,
0.96862745, 0.96470588, 0.96470588])

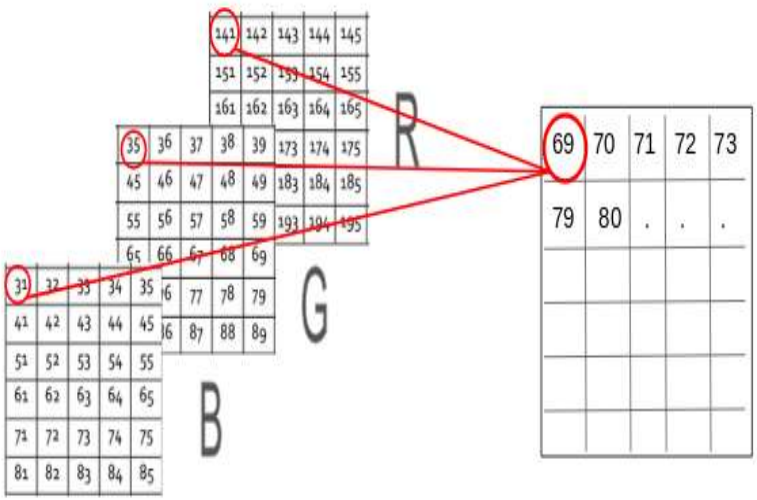
Method #2: Mean Pixel Value of Channels

- *A colored image is typically composed of multiple colors and almost all colors can be generated from three primary colors – red, green and blue.*
- Hence, in the case of a colored image, there are three Matrices (or channels) – Red, Green, and Blue.
- Each matrix has values between 0-255 representing the intensity of the color for that pixel.



Method #2: Mean Pixel Value of Channels

- This time, the image has a dimension (660, 450, 3), where 3 is the number of channels.
- We can go ahead and create the features as we did previously. The number of features, in this case, will be 660*450*3 = 891,000.
- *Instead of using the pixel values from the three channels separately, we can generate a new matrix that has the mean value of pixels from all three channels.*

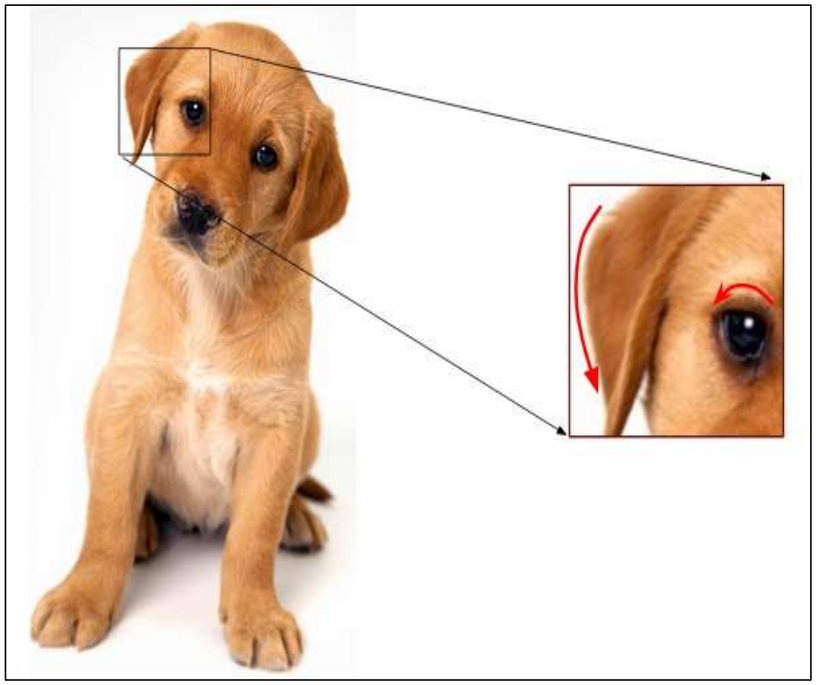


Method #3: Extracting Edge Features

- Consider that we are given the image and we need to identify the objects present in it.
- The features like shape, color or size could be important factors to decide.
- To make machine learn the shape, one idea is to extract edges as features.

Method #3: Extracting Edge Features

- Edge is basically where there is a sharp change in color.
- We could identify the edge because there is a change in color from white to brown (in the right image) and brown to black (in the left).
- And as we know, an image is represented in the form of numbers. So, we will look for pixels around which there is a drastic change in the pixel values.



Method #3: Extracting Edge Features

- To identify if a pixel is an edge or not, we will simply subtract the values on either side of the pixel.
- For this example, we have the highlighted value of 85. We will find the difference between the values 89 and 78. Since this difference is not very large, we can say that there is no edge around this pixel.
- Now consider the pixel 125 highlighted. Since the difference between the values on either side of this pixel is large, we can conclude that there is a significant transition at this pixel and hence it is an edge.
- There are various kernels that can be used to highlight the edges in an image.

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

Further Reading

1. Webpage: Understanding TF-IDF for Machine Learning: https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/

2. Book: Introduction to Information Retrieval

By: by Christopher D. Manning (Author), Prabhakar Raghavan (Author), Hinrich Schütze (Author)

3. Book: Information Retrieval: Implementing and Evaluating Search Engines (The MIT Press)

by Stefan Buttcher (Author), Charles L. A. Clarke (Author), Gordon V. Cormack (Author)