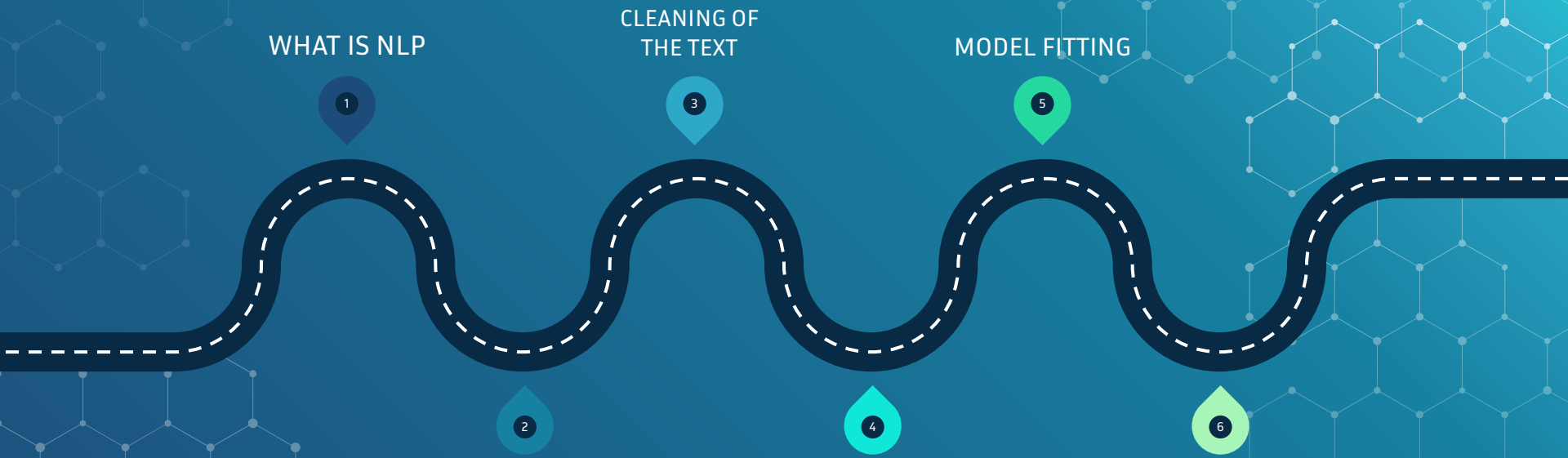
NATURAL LANGUAGE PROCESSING
+
CODE REVIEW

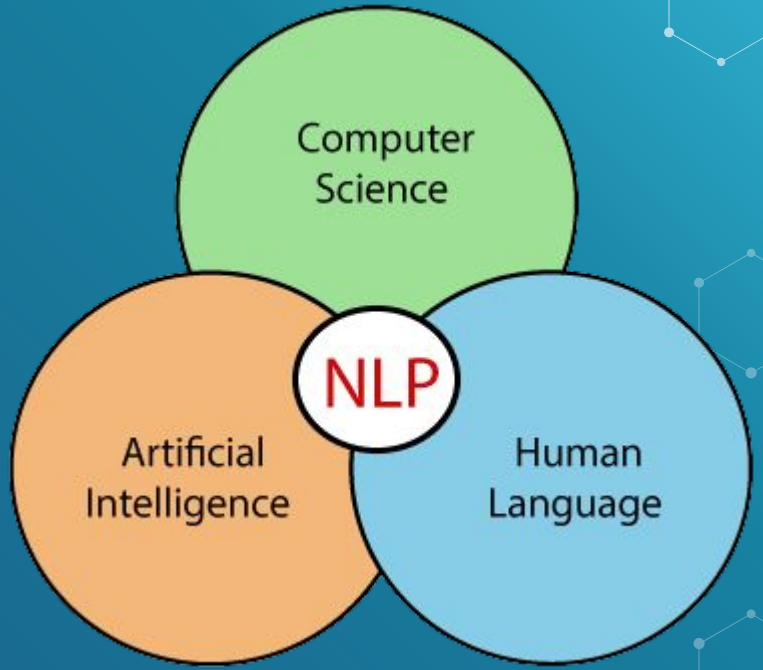
IMT2020039 - Anshul Jindal
IMT2020535 - Shreeya Venneti

ROADMAP


WHAT IS NLP
1
EXPLORATORY DATA ANALYSIS
2
CLEANING OF THE TEXT
3
PREPROCESSING
4
MODEL FITTING
5
CONCLUSION & THANK YOU
6

What is NLP?

- Ability given to computer to be able to process Human Language.
- Sub-domain of ML.



NLP CONCEPTS WE RESEARCHED

- SENTIMENT ANALYSIS
- TOKENIZATION
- NLP TEXT NORMALIZATION TECHNIQUES
  - STEMMING
  - LEMMATIZATION
- NLP TEXT MODELING TECHNIQUES
  - BAG OF WORDS
  - TF-IDF

1 EXPLORATORY DATA ANALYSIS 🧐

Let's analyse data and get insights

BASIC EDA

- **PIE - CHART**
  - Showing category distribution.
  - **INFERENCE** - Can expect a similar distribution in test data.
- **CORRELATION MATRIX**
  - Checking relation (if any) between label columns.
  - **INFERENCE** - We check if columns can be linked for training our model.

WORD BASED ANALYSIS

- Extract a data frame consisting of words, their lengths and their frequency of occurrence.
- Check if extremely long words are real english words or not.
- **INFERENCE** - Words having length greater than 40 or 50 (if any) and if they mostly don't make sense, then they can be removed.

POLARITY

- Polarity refers to **strength of an opinion**.
- Polarity values lie in the range [-1,1] with [-1,0) implying negative sentiment, 0 implying neutral and (0,1] implying positive sentiment.
- **INFERENCE:**
  - Most commenters (about 80,000) have a neutral sentiment polarity or no sentiment.
  - The rest of the commenters about 9,000 have made polar comments, which is the subject of our analysis.
  - Thus we can expect similar results upon fitting our classification model.

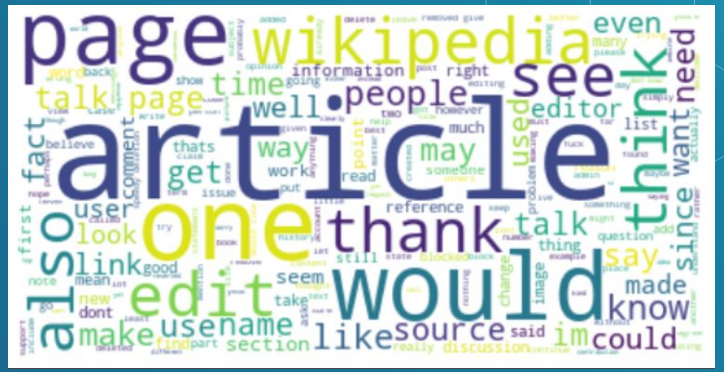
SUBJECTIVITY

- Subjectivity in text indicates presence of the writer's own feelings, experiences or emotions. This is in contrast to objective text which is mostly factual and not opinion based.
- Indicates extent or degree of involvement of the person in the object. Lies in the range [-1,1] and thus takes both positive and negative values.
- **INFERENCE:**
  - Comments - highly biased towards subjectivity, i.e every comment is subjective and is purely based on the commenter's opinion & experience and is not objective/fact-based. Also indicates presence of strong emotions/feelings of the commenter.

WORD CLOUD

- Visual Representation of the most frequently occurring words in the dataset.
- Larger the size of word in cloud **=>** Higher the frequency

INFERENCE - We can manually remove the frequently occurring words which are not helpful in sentiment analysis. This will make preprocessing and training faster.



Example of a Word Cloud

CATEGORY WISE WORD CLOUDS

- Observed Word Clouds i.e most occurring words for each category.
- Manually classified the sentences in which these words were occurring with probability of 0.99 irrespective of model output.

RESULTS - Accuracy Increased but not by considerable amount.

2

CLEANING OF THE TEXT

Let’s remove all the dirt.

ONE - SHOT CLEANING

- Removing occurrences which contain some pattern using Regular Expressions.
- Stuff which doesn't contribute towards sentiment analysis is removed.
- This involves:
  - Removal of URLs from the text.
  - Remove all punctuation marks and special characters except !, ? and words which contain * in them.
  - Lowering the text.
  - Removing all the numeric data.

MANUAL CLEANING AND EXPANDING

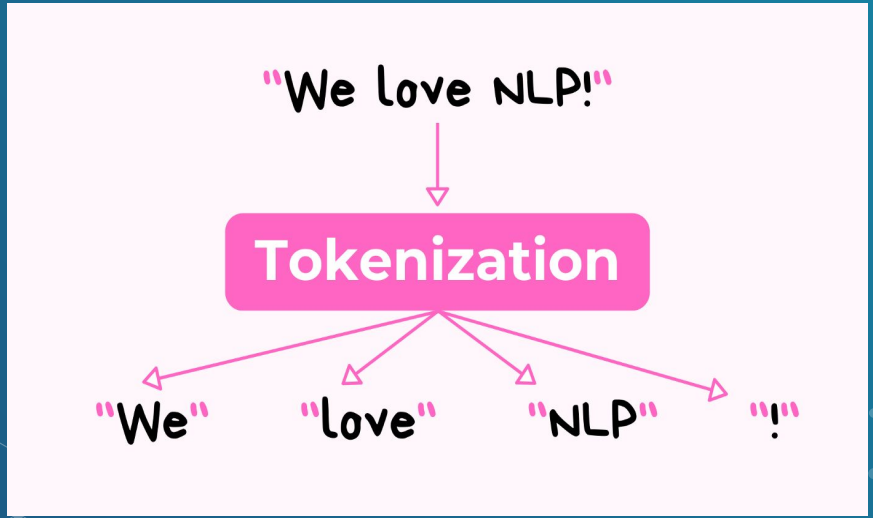
- Manual conversion of emojis to the texts which depict their true meanings..
  - Example: :) has been substituted by word “smiling”.
- Expanding includes substituting certain word formats/shortcuts to match to their real sentences.
  - For example, “ ‘ll ” should be written as “I will”

3 PREPROCESSING

Let's start the Engine - nltk is the key

TOKENIZATION

- Splitting a text into a list of words or sentences.



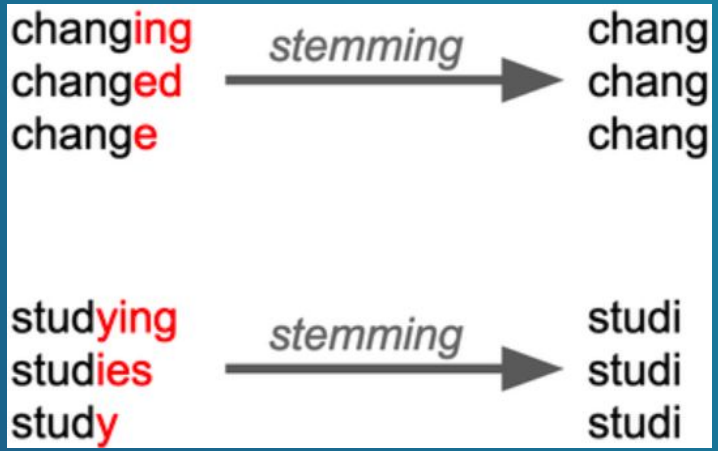
TYPES OF TOKENIZERS

Word Tokenizer	**Sentence Split** *"Sample sentence here"*
Whitespace-based tokenization	["It's", 'true,', 'Ms.', 'Martha', 'Jones!', '#Truth']
Punctuation-based tokenization	['It', "'", 's', 'true', ',', 'Ms', '.', 'Martha', 'Jones', '!', '#', 'Truth']
Default/Treebank Tokenizer	['It', "'s", 'true', ',', 'Ms.', 'Martha', 'Jones', '!', '#', 'Truth']
Tweet Tokenizer	["It's", 'true', ',', 'Ms', '.', 'Martha', 'Jones', '!', '#Truth']
MWE Tokenizer	['It', "'s", 'true', ',', 'Ms.', 'Martha_Jones', '!', '#', 'Truth']

OBSERVATION - For Sentiment Analysis, Tweet Tokenizer works better than others because words with # or @ as prefix are of much importance and have to dealt separately. In our case, Tweet Tokenizer and Whitespace Tokenizers gave almost same accuracy.

STEMMING

- Process of reducing a word to it's word stem or root.



- **IMPORTANCE** - To reduce redundancy. These words have same influence in sentiment analysis.

LEMMATIZATION

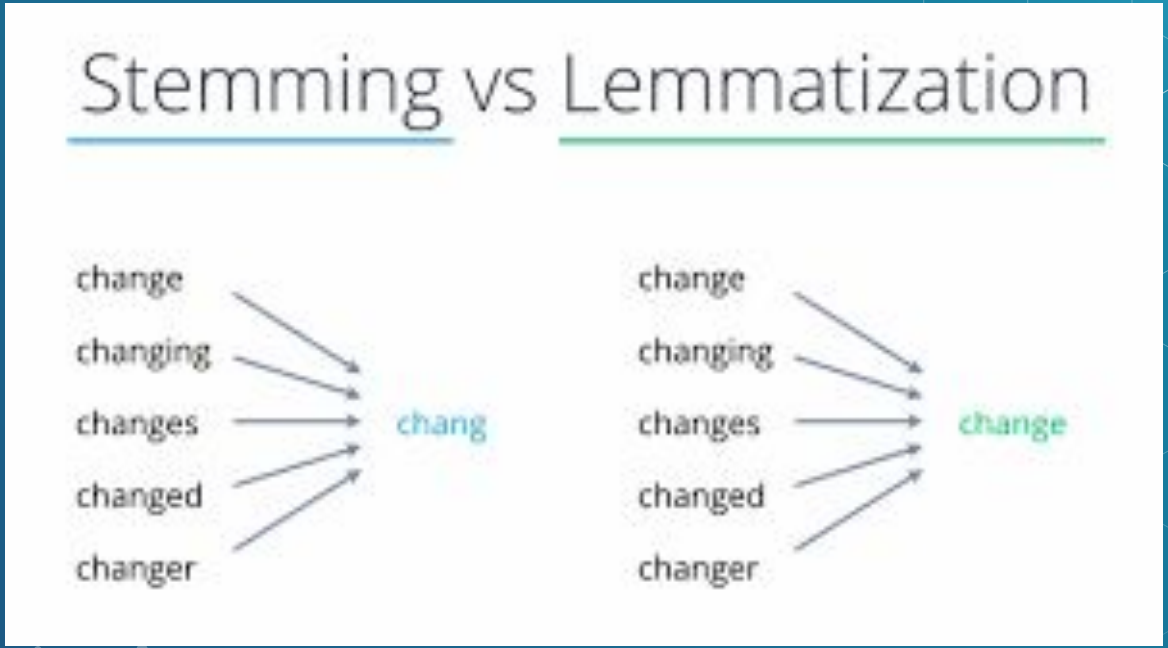
- Reduction of word to its root word while retaining their meaning.

- **NOTE:** Only one of either Stemming or Lemmatization is preferred and not both depending on the requirements.

STEMMING V/S LEMMATIZATION

Stemming	Lemmatization
The word “better” could be over-stemmed to “bet”, “bett” or “better” is retained.	Here, the algo understands that “better” is derived from “good” and thus it is lemmatized to “good”.
Lower accuracy, since word is being chopped off abruptly.	Higher accuracy, however is computationally expensive.
Used in applications where performance is main focus	Used in applications where understanding meaning is crucial

STEMMING V/S LEMMATIZATION (cont..)



NOW NEXT WHAT...?? 🤔

- Till now we have done:
  - EDA
  - Cleaning of text
  - Manipulation of the words in the text.
- But still, we are not in stage to fit and train any ML model. For training, we need features. But for now, we only have bunch of text. 😕
- So, let's bring features into the picture. 😉

BAG OF WORDS

- An algorithm to convert text into fixed-length vectors.
- Every sentence has a vector corresponding to it.
- The vector's elements are the frequency of words of the sentence.
- Let's understand the concept better with an example.

EXAMPLE and LIMITATION of BOW.

- Consider 3 sentences: He is a good boy; She is a good girl; Boy and girl are good. The vector table for the same is:

	good	boy	girl
Sentence1	1	1	0
Sentence 2	1	0	1
Sentence 3	1	1	1

Can you see the features here? 🤔

- **LIMITATIONS:** Every word is given equal weightage. But in sentiment analysis, some words (like good) are more imp than others.

TF - IDF VECTORIZATION

- An algorithm to convert text into fixed-length vectors with some weighted information.

- Term Frequency (TF) = $\frac{\text{No of rep of word in sentence}}{\text{No of words in sentence}}$

- Inverse Document Freq (IDF) = $\log\left(\frac{\text{No of sentences}}{\text{No. of sentences containing the word}}\right)$

TF - IDF VECTORIZATION

- We have done:
  - **Word Vectorization:**
    - Features will be made from words.
  - **Character Vectorization.**
    - Features will be made of characters instead of words.
- Later we have merged them horizontally to form combined feature matrix.

SAMPLING OF DATA

- Our Dataset is highly imbalanced.
- This bias in the training dataset can influence many machine learning algorithms, leading some to ignore the minority class entirely.
- Random **Over or Under Sampling** are some of the ways to handle Imbalanced Dataset available in **imbalanced-learn.**
- Let's have a look what we have implemented in our project.

OVER-SAMPLING

- **RandomOverSampler**
  - from imblearn.over_sampling import RandomOverSampler
  - Random oversampling involves randomly duplicating examples from the **minority class** and adding them to the training dataset.

- **SMOTE - Synthetic Minority Oversampling Technique**
  - from imblearn.over_sampling import SMOTE
  - Samples are generated using concept of **k nearest neighbours.**

UNDER-SAMPLING

- **RandomUnderSampler**
  - from imblearn.over_sampling import RandomUnderSampler
  - Random undersampling involves randomly selecting examples from the **majority class** to delete from the training dataset.

RESULTS - Only Over-Sampling - Increased for some models while decreased for other models.

Only Under-Sampling - Decreased the accuracy for every model.

CAN WE DO A MIX OF BOTH ?? 🤔🤔

ANSWER IS YES! 😱

- **PIPELINE**
  - from imblearn.pipeline import Pipeline
  - Results are achieved by combining both random oversampling and undersampling.

RESULTS - Using Pipeline had almost no effect on the accuracy.

4 MODEL FITTING

Let's GOOO !!!!!

LOGISTIC REGRESSION

- From sklearn.linear_model import LogisticRegression.
- TYPE I - UNBALANCED
  - Class labels are are assigned equal weight.
  - Got Accuracy of **0.98116** on Kaggle.
- TYPE II - BALANCED
  - Class labels will be assigned weights inversely proportional to their freq.
  - Got Accuracy of **0.98295** on Kaggle.

RANDOM FOREST CLASSIFIER

- From sklearn.ensemble import RandomForestClassifier.
- Parameters Controlled:
  - Max_features = 2500
  - Max_depth = 100 (max depth of tree)
  - Min_samples_split = 10 (then only split, otherwise don't split further)
  - Criterion = 'gini'
  - N_estimaters = 120 (max total number of trees)
- Got Accuracy of **0.96953** on Kaggle.

RIDGE CLASSIFIER

- From sklearn.linear_model import Ridge
- Parameters Controlled:
  - alpha = 30 (Slope)
  - Max_iter = 1000 (max number of iterations)
- Got Accuracy of **0.98314** on Kaggle.

PERCEPTRON

- from sklearn.linear_model import Perceptron
- Parameters Controlled:
  - Alpha = 0.1 (Learning rate)
  - Penalty = “l2” (L2 loss - Squared Loss)
  - Max_iter = 150 (Maximum Iterations)
- Got Accuracy of **0.58991** on Kaggle.

SVM - with RBF Kernel

- from sklearn.svm import SVC
- Parameters Controlled:
  - Kernel = RBF
  - Probability = True (We want probabilities for our predictions)
  - Max_iiter = 1000 (Max Iterations)
  - Class_weight = 'balanced'
- Ran for 8+ hrs. Got Accuracy of **0.94357** on Kaggle.

GRADIENT BOOSTING CLASSIFIER

- from sklearn.ensemble import GradientBoostingClassifier
- Parameters Controlled:
  - n_estimators = 100 (The number of boosting stages
  - Max_depth = 50 (Depth of every stage)
- Ran for 6+ hrs and didn't classify even single label. My laptop was like 🥵.

ADABOOST CLASSIFIER

- from sklearn.ensemble import AdaBoostClassifier
- Parameters Controlled:
  - n_estimators = 100 (The number of boosting stage)s
  - learning_rate = 0.1 (Learning rate of the model)
- Got Accuracy of **0.54134** on Kaggle.
- Seeing this, other models were like:

XG-BOOST CLASSIFIER

- import xgboost as xgb
- Advance version of Gradient Boosting and faster..
- Parameters Controlled:
  - All defaut
- Got Accuracy of **0.96703** on Kaggle.

Decision Tree Classifier

- from sklearn.tree import DecisionTreeClassifier
- Guessed it wasn't gonna be good as we already implemented Random Forest.
- Parameters Controlled:
  - Max_depth = 1000 (Max depth of the tree)
  - Class_weight = 'balanced'
- Got Accuracy of **0.73080** on Kaggle.

Voting Classifier - 💪💪

- from sklearn.ensemble import VotingClassifier
- Worked Best playing us on 4th Spot on Leaderboard.
- Models Embedded:
  - Logistic Regression
  - Random Forest
  - SGD Classifier
  - XG Boost
- Got Accuracy of **0.98576** on Kaggle.

COMPARISON OF MODELS

MODELS	ACCURACY
Logistic Regression	0.98295
Random Forest	0.96953
Ridge Classifier	0.98314
AdaBoost Classifier	0.54314

MODELS	ACCURACY
XG-Boost Classifier	0.96703
Perceptron	0.58991
SVM with RBF Kernel	0.94357
Decision Tree Classifier	0.73080
VOTING CLASSIFIER	**0.98576**

4

BEFORE WE END

Some Logistics !!

DIVISION OF WORK

- Anshul Jindal 🙋
  - Analysis and fitting of different models. Optimizing the models and finding the best fit.
  - This PPT.
- Shreeya Venneti 👧
  - NLP Preprocessing techniques, tokenization and lemmatization. Exploratory Data Analysis involving word-based analysis and sentiment analysis, etc.
  - Report of the Project.

LEARNING OUTCOMES

- NLP was completely new for us. Getting to know about this domain of ML was exciting.
- Got extensive knowledge of sklearn and other such libraries. Got better understanding of models.
- Got to know and explore Kaggle.
- Team work & coordination. Also there was much observed competitive spirit for this project.

CHALLENGES FACED ❗❗

- Lot of time - Required constant monitoring.
  - **Overcoming** - Ran half-half part of classifications on both of our laptops and later merged them.
- Resource limitation - KERNEL CRASH 😡😡.
  - **Overcoming** - Tried running some models with GPU. Shifted to Kaggle from Colab.
- Redundant Running of Code - Preprocessing parts
  - **Overcoming** - Save the preprocessed data in pickle file.

FUTURE SCOPE 👀

- One can implement advance NN Algorithms and models such as LSTM, etc.
- If resources permit, one can remove the limitations put for models such as max_features and make models more stronger.
- At last: Keep trying until 100% 🙂

THANKS!

That Marks the end for Code Review !!

ANY QUESTIONS?