HATEFUL MEME DETECTION FOR SOCIAL MEDIA APPLICATIONS

DARSHAN.G (171EC137)

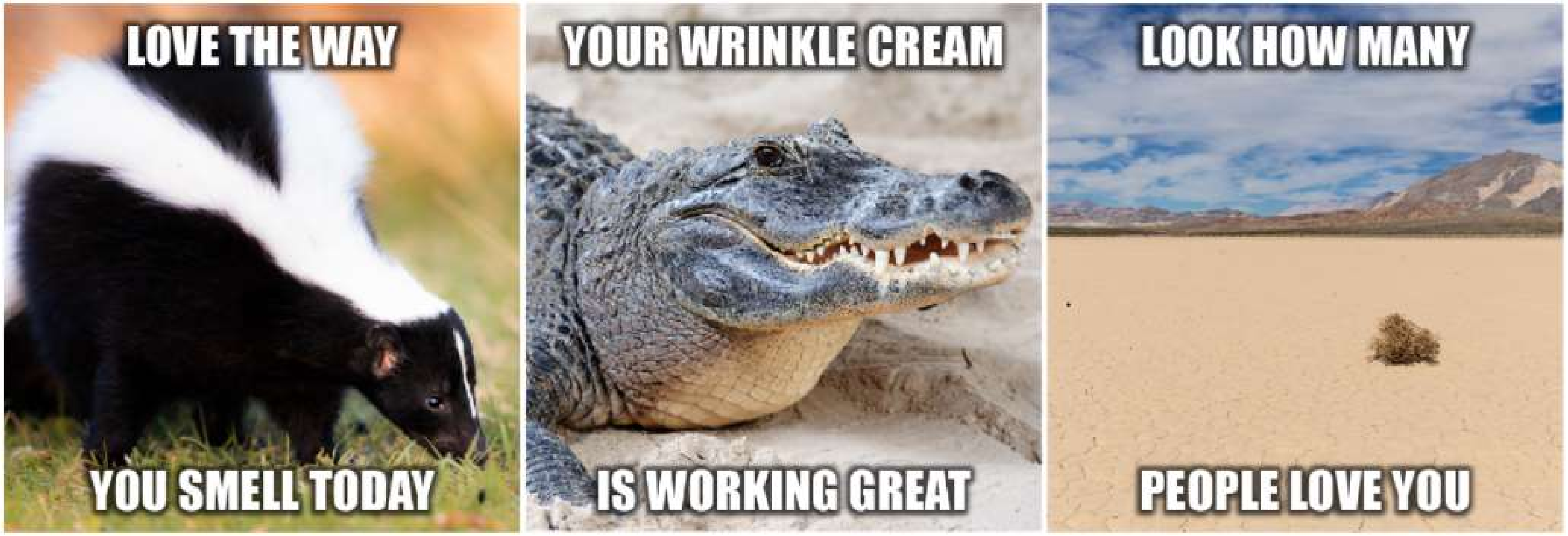
LIST OF MINI PROJECTS CARRIED OUT:

- CLASSIFICATION OF DOG AND CAT USING MACHINE LEARNING
- WILD PLANTS DETECTION USING DEEPLEARNING STUDIO
- ENHANCING THE IMAGE RESOLUTION OF BLURRED IMAGES USING DEEP LEARNING FOR SATELLITE SURVILLENCE
- AUDITORIUM CONTROL SYSTEM
- DIAMOND DETECTION
- AIR POLLUTION MONITERING SYSTEM

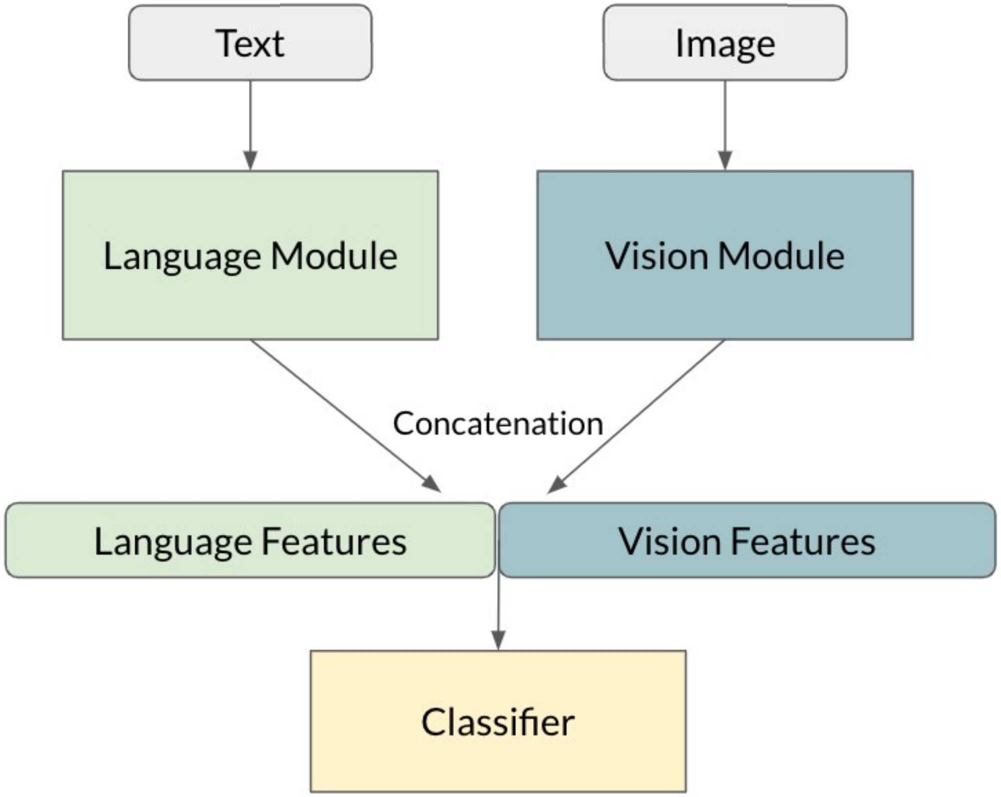
PROPOSED SYSTEM

- The challenges of harmful content affect the entire tech industry and society at large.
- The Hateful Memes data set consists of more than 10,000 newly created examples of multimodal content.
- The memes were selected in such a way that strictly unimodal classifiers would struggle to classify them correctly.
- The data is explored and loaded
- The trained LanguageAndVisionConcat is used as a classifier to predict that the meme is hateful or not.

HATEFUL MEME DATA



LANGUAGEANDVISIONCONCAT FLOWCHART


Text

Image

Language Module

Vision Module

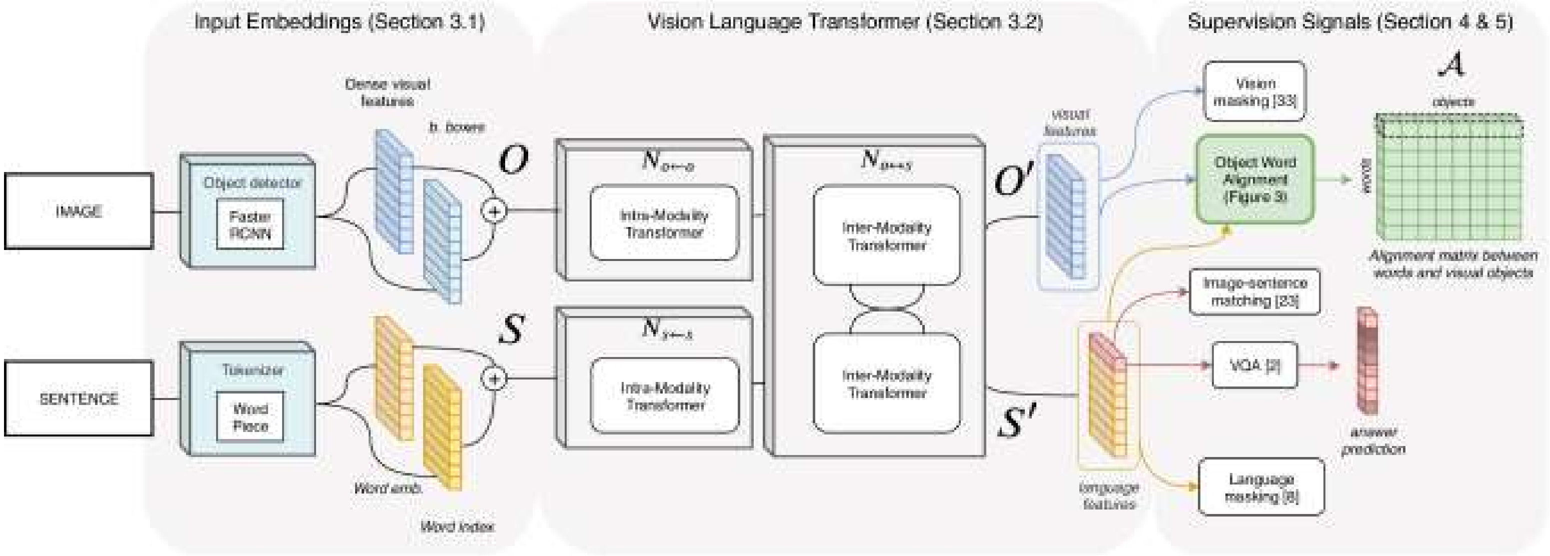
Concatenation

Language Features

Vision Features

Classifier

LANGUAGEANDVISIONCONCAT ARCHITECTURE


Input Embeddings (Section 3.1)
Vision Language Transformer (Section 3.2)
Supervision Signals (Section 4 & 5)
IMAGE
SENTENCE
Object detector
Faster RCNN
Tokenizer
Word Piece
Dense visual features
b. boxes
Word emb.
Word index
O
S
$N_{o\leftarrow o}$
$N_{s\leftarrow s}$
$N_{o\leftrightarrow s}$
Intra-Modality Transformer
Intra-Modality Transformer
Inter-Modality Transformer
Inter-Modality Transformer
O'
S'
visual features
language features
Vision masking [33]
Object Word Alignment (Figure 3)
Image-sentence matching [23]
VQA [2]
Language masking [8]
$\mathcal{A}$
objects
words
Alignment matrix between words and visual objects
answer prediction

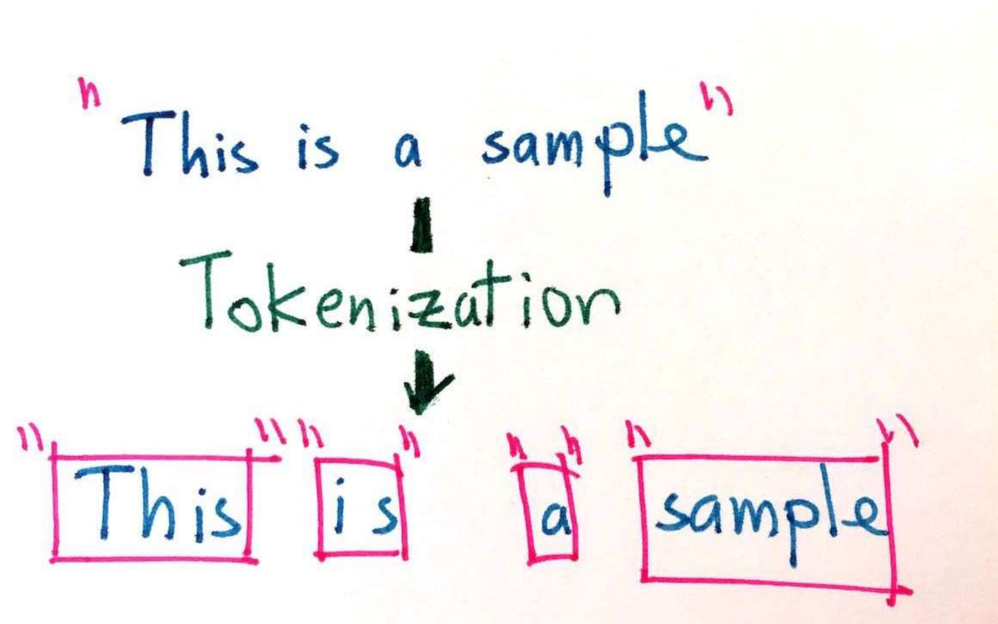
TOKENIZATION

- Tokenization is a common task in Natural Language Processing (NLP).
- It’s a fundamental step in both traditional NLP methods like Count Vectorizer and Advanced Deep Learning-based architectures.
- Tokenization is a way of separating a piece of text into smaller units called tokens.
- Here, tokens can be either words, characters, or subwords.
- Hence, tokenization can be broadly classified into 3 types – word, character, and subword (n-gram characters) tokenization.

WORD TOKENIZATION AND IT'S DRAWBACKS

- Word Tokenization is the most commonly used tokenization algorithm. It splits a piece of text into individual words based on a certain delimiter.
- One of the major issues with word tokens is dealing with **Out Of Vocabulary (OOV) words**.
- OOV words refer to the new words which are encountered at testing.
- These new words do not exist in the vocabulary.
- Hence, these methods fail in handling OOV words.

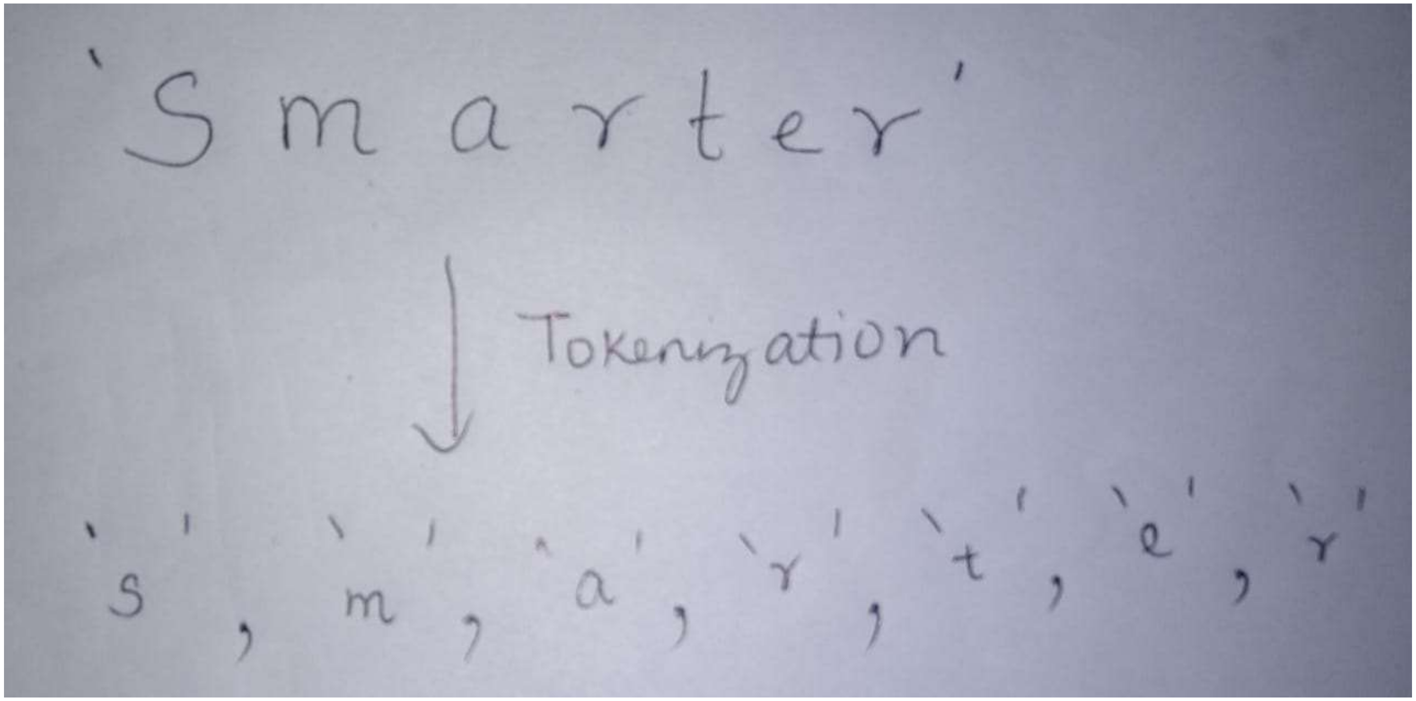
EXAMPLE FOR WORD TOKENIZATION



CHARACTER TOKENIZATION

- Character Tokenization splits apiece of text into a set of characters. It overcomes the drawbacks we saw above about Word Tokenization.
- Character Tokenizers handles OOV words coherently by preserving the information of the word. It breaks down the OOV word into characters and represents the word in terms of these characters
- It also limits the size of the vocabulary. Want to talk a guess on the size of the vocabulary? 26 since the vocabulary contains a unique set of characters

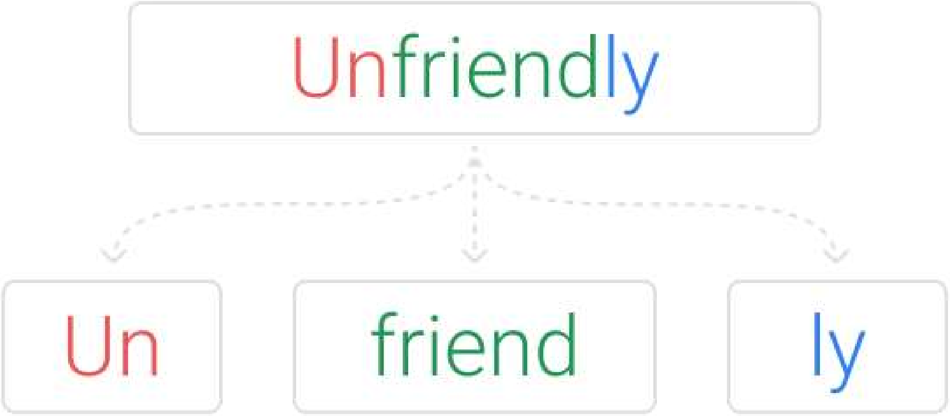
CHARACTER TOKENIZATION



DRAWBACKS OF CHARACTER TOKENIZATION AND HOW TO OVERCOME IT USING SUBWORK TOKENIZATION

- Character tokens solve the OOV problem but the length of the input and output sentences increases rapidly as we are representing a sentence as a sequence of characters.
- As a result, it becomes challenging to learn the relationship between the characters to form meaningful words.
- This brings us to another tokenization known as Subword Tokenization which is in between a Word and Character tokenization.
- Subword Tokenization splits the piece of text into subwords (or n-gram characters). For example, words like lower can be segmented as low-er, smartest as smart-est, and so on.

EXAMPLE OF SUBWORD TOKENIZATION



THANK
YOU!