

Data Collection and Preprocessing Phase

Date	15 March 2024
Team ID	SWTID1728136330
Project Title	Fake News Analysis in Social Media Using NLP
Maximum Marks	6 Marks

Preprocessing Template

The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

Section	Description
Data Overview	Give an overview of the data, which you're going to use in your project.
Resizing	Resize images to a specified target size.
Normalization	Normalize pixel values to a specific range.
Data Augmentation	Apply augmentation techniques such as flipping, rotation, shifting, zooming, or shearing.
Denoising	Apply denoising filters to reduce noise in the images.
Edge Detection	Apply edge detection algorithms to highlight prominent edges in the images.

Color Space Conversion	Convert images from one color space to another.
Image Cropping	Crop images to focus on the regions containing objects of interest.
Batch Normalization	Apply batch normalization to the input of each layer in the neural network.
Data Preprocessing Code Screenshots	
Loading Data	<pre> import pandas as pd # Load data df = pd.read_csv('fake_news_data.csv') # Display basic data statistics print("Data Shape:", df.shape) print("Class Distribution:", df['label'].value_counts()) print("Sample Data:\n", df.head()) </pre>
Resizing	<pre> def resize_text(text, max_length=512): words = text.split() return ' '.join(words[:max_length]) if len(words) > max_length else text df['text'] = df['text'].apply(resize_text) </pre>
Normalization	<pre> import re def normalize_text(text): text = text.lower() # Convert to lowercase text = re.sub(r'\d+', '', text) # Remove numbers text = re.sub(r'^\w\s', '', text) # Remove punctuation return text df['text'] = df['text'].apply(normalize_text) </pre>

Data Augmentation	<pre> import random from nltk.corpus import wordnet # Synonym replacement augmentation def synonym_replacement(text, n=3): words = text.split() new_words = words[:] random_word_list = list(set(words)) random.shuffle(random_word_list) num_replaced = 0 for random_word in random_word_list: synonyms = wordnet.synsets(random_word) if synonyms: synonym = synonyms[0].lemmas()[0].name() new_words = [synonym if word == random_word else word for word in new_words] num_replaced += 1 if num_replaced >= n: break return ' '.join(new_words) df['text_aug'] = df['text'].apply(synonym_replacement) </pre>
Denoising	<pre> def remove_noise(text): text = re.sub(r'http\S+ www\S+ https\S+', '', text, flags=re.MULTILINE) # Remove URLs text = re.sub(r'@\w+ \#', '', text) # Remove mentions and hashtags text = re.sub(r'\s+', ' ', text) # Remove extra whitespaces return text df['text'] = df['text'].apply(remove_noise) </pre>
Edge Detection	<pre> from nltk.tokenize import word_tokenize df['tokens'] = df['text'].apply(word_tokenize) </pre>
Color Space Conversion	<pre> df['text'] = df['text'].str.encode('utf-8').str.decode('utf-8') </pre>
Image Cropping	<pre> # Extracting key sentences (first 3 sentences) for summarization or focused analysis import nltk nltk.download('punkt') from nltk.tokenize import sent_tokenize def crop_text(text, max_sentences=3): sentences = sent_tokenize(text) return ' '.join(sentences[:max_sentences]) df['text_cropped'] = df['text'].apply(crop_text) </pre>

Batch Normalization

```
from sklearn.preprocessing import StandardScaler
from sklearn.feature_extraction.text import TfidfVectorizer

# Example of TF-IDF feature extraction followed by normalization
vectorizer = TfidfVectorizer(max_features=1000)
X = vectorizer.fit_transform(df['text']).toarray()

# Batch normalization
scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)
```