Campus Live Project

ABES - TIMES INTERNET LTD.

# **OVERVIEW**

This project propels news classification to optimize advertising approaches and deepen user insights. It has two main goals: first, it gathers important details from news like people, groups, and themes to create precise audience groups and targeted ads. At the same time, it studies how users explore both our internal pages (TIL) and external pages to build complete user profiles that enhance ads. By combining news organization, audience targeting, and user understanding, it connects advertisers and users, resulting in tailored content and effective campaigns. This effort brings together advanced news organization, audience focus, and user insight, connecting advertisers and users by understanding news and behavior. The outcome is personalized content that stands out in the midst of data and leads to impactful campaigns that truly matter.

# **GOALS**

1. Intent Classification: Mapping article/urls to one or more nodes of IAB tree
2. Important Keywords Extraction
3. NER (Named-Entity Recognition)

**TECHNOLOGY STACK**

The technology stack for this project encompass the following components and tools:

**1. Programming Language**:

- Python

**2. NLP Libraries**:

- NLTK (Natural Language Toolkit): For fundamental NLP tasks such as tokenization, lemmatization, and part-of-speech tagging.

- spaCy: Known for its speed and efficiency in NLP processing, including named entity recognition.

- Gensim: For topic modeling and document similarity analysis.

- Transformers (Hugging Face): Provides state-of-the-art pre-trained models like BERT, GPT-3, etc., for various NLP tasks like generate word embeddings, NER, etc.

**3. Machine Learning Frameworks:**

- Scikit-Learn: For machine learning tasks related to NLP like classification and clustering.

- TensorFlow or PyTorch: For building and training deep learning models for NLP tasks.

**4. Data Manipulation**:

- Pandas: For data preprocessing, manipulation, and analysis.

- NumPy: For numerical operations.

**5. Data Visualization**:

- Matplotlib & Seaborn: For creating visualizations to better understand and present NLP results.

**6. Text Preprocessing**:

- Regular Expressions: For text cleaning and pattern matching.

- Spacy or NLTK: For more advanced text preprocessing tasks like lemmatization.

**7. Web Scraping**: [Doc](https://docs.google.com/document/d/1qU_HLYzcmUr32SMRf2BhFkxmjX6ZZu3p6XTw7FjIkSE/edit)

- BeautifulSoup and Selenium: For extracting text data from websites.

**10. Version Control**:

- Git: To manage code changes and collaborate with a team.

# **SPECIFICATIONS**

## **Data Sources and Collection Methods**

1. **Public Datasets**:

We initiated our project by sourcing datasets from reputable repositories such as Kaggle, UCI, and the Hugging Face dataset collection. These datasets served as invaluable resources, providing us with a foundational comprehension of news articles and dataset structure. For your reference, we are including the datasets that we individually retrieved from websites similar to those mentioned above.

Datasets: [Datasets (TIL Live Project)](https://docs.google.com/document/d/1Sm5N9IjnrLKYmcCH2khK3a93IgQXxRDGdy4Ve4nnbPc/edit)

**Challenge 1**: We noticed that many websites had similar data in different categories like ‘Health and Medical Services’ & ‘Medical Health’, showing that the topics were somewhat the same.

**Solution**: We merged the category ‘Fine Art’ & ‘Pop Culture’ into ‘Arts and Culture’ ,and ‘Health and Medical Services’ & ‘Medical Health’ into ‘Health’.

**Challenge 2**: Some categories had less counts so we needed another data collection method to compensate for this.

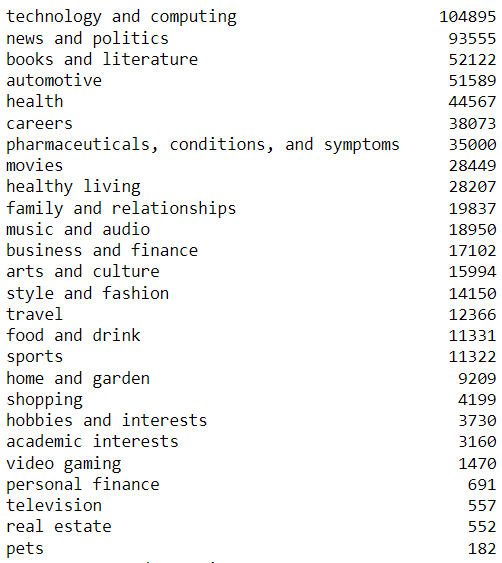


Fig 1. Data collected from public datasets

**Solution:** Web Scraping from websites and sitemaps.

1. **Web Scraping:**
2. **Sitemap -** Continuing our project’s progression, we experimented with data extraction from website sitemaps to obtain categorized website links.

**Challenge 3**: This approach sometimes retrieves irrelevant information, such as references to the PET exam when searching for pet-related content.

**Solution**: This highlighted the necessity for refinement of Web Scraping process

.

1. **Websites** - We transitioned towards the direct acquisition of information from reputable news and article sources such as '[Times of India](https://timesofindia.indiatimes.com/us)', '[Economic Times](https://m.economictimes.com/)', '[Games Radar](https://www.gamesradar.com/)', and '[ABC News](https://www.abc.net.au/news)'. This strategy enabled us to aggregate content from diverse and authoritative outlets, thereby giving our dataset a wide range of viewpoints and perspectives

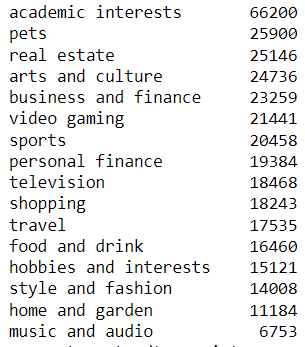


Fig 2. Data collected from web scraping

**Challenge 4**: Data for several categories remained incomplete as we encountered difficulties in obtaining data from the above mentioned web sources. Alternative web sources were explored; however, challenges were encountered with respect to Hindustan Times due to its complex structure, Business Standards due to accessibility issues, and CNN due to insufficient data for the desired category, among others.

**Solution**: To tackle this challenge, we used data augmentation techniques to reach the desired threshold.

**3.**  **Data Augmentation**:

Subsequently, data augmentation techniques were employed to achieve the desired threshold of 20,000 articles per category for those categories having insufficient article count. The methods of data augmentation included synonym replacement, back translation, text rotation, and random deletion.

**Challenge 5**: Following the data collection process, the resultant dataset had a mean word count of 180. The word count distribution within the gathered data exhibited considerable variability, ranging from 0 to 19,875 words. Approximately 60-62% of the dataset featured word counts below the calculated mean.

**Solution**: Data Addition and Data Restructuring:

Data Addition: Due to the large number of articles having less word counts, additional data was scraped to compensate for this variability.

Data Restructuring: To make the dataset more focused, the data was split into segments with a word count range from 100 to 500 words.This range was chosen to create a more homogeneous dataset while preserving an adequate sample size.

**Total Data Collected:**

****

Fig 3. Total data collected

## **Data Pre-processing**

The primary objective of this phase was to ensure that the data was of high quality, consistently structured, and lacking any unnecessary noise. This preprocessing consisted of several steps:

**1. Lowercasing**: To maintain a uniform appearance, the entirety of the text was transformed to lowercase using the `.lower()` function.

**2.Punctuation Removal**: Employing a custom function named `remove\_punct`, we stripped all punctuation marks from the text.

**3. URL and Emoji Removal**: We used `removeURLandEmoji` function to remove URLs and converted emojis into text-based representations which would be removed later as special characters.

**4.Email and Number Removal**: Using another customized function called `remove\_Emails\_and\_Numbers`, we erased email addresses and numeric values from the text data.

**5. Special Character Removal**: The `removeSpecialChar` function was purposefully designed to expunge non-alphabetic characters and replace specific characters such as '\n' and '\xa0'.

**6. HTML Element Removal**: Through an assessment by the `has\_html\_elements` function, HTML elements were identified and systematically removed.

Our goal with this preparation was to get valuable information for our news classification project.



Fig 4. Data collection after pre-processing

## **Word Embeddings**

We are taking a diverse approach to ensure that we find the optimal solution for our project. Each of our team members is working on a specific word embedding technique to explore which one aligns best with our objectives:

**Glove & DistilBERT:**

[Word Embeddings(Glove)-Aman Rai](https://docs.google.com/document/d/1nVEVR-INQtM_nmtvSXrtrtzcFX2EIpYN5PuJS664kT8/edit?usp=sharing) [Embeddings Using DistilBERT base model (uncased)](https://docs.google.com/document/d/1x3BiqAqPY0mm78JbzPYDltPB6TUQLlUXf2dQOU6D8o4/edit?usp=sharing)

**Model**

**DistilBERT** is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than bert-base-uncased, runs 60% faster.

**SPECIFICATIONS**

## **1.Model Selection**

I selected the '**distilbert-base-uncased** model from Hugging Face for embedding.

**2.Input Encoding and Model Inference**

It includes tokenization, padding, and encoding text as numerical tensors to feed into the model. It takes the encoded input and generates embeddings.

**3. Fine-tuning a DistilBERT model for a text classification task**

[FineTune\_DistilBERT](https://colab.research.google.com/drive/1fYwnFCMbVjQbCx-7-g_LSB4Fmnjdl44o#scrollTo=hPICQzSVJ1AM)

**Library Used:** transformers, memory\_profiler, pandas, numpy, tensorflow, DistilBertTokenizer, TFDistilBertForSequenceClassification, time

Now I have preprocessed text in “text” column and take all 26 categories in “target” column in the dataframe and take sample 2000 rows from each category (total 52000 rows) from the original DataFrame

**Data Splitting for Training and Evaluation:**

The dataset undergoes an 80-20 train-test split, ensuring class balance. The test set is further divided into 50% for validation and 50% for testing, resulting in an 80-10-10 distribution for model development and evaluation.

**Model Setup and Configuration**

**Process:**

The code begins by loading the DistilBERT tokenizer and a pre-trained model for sequence classification. The selected pre-trained model is 'distilbert-base-uncased,' configured for 26 output labels.

**Data Preparation and Model Training**

**Process:**

Tokenization is performed on the text data for training, validation, and test sets using the DistilBERT tokenizer. Labels are converted to TensorFlow tensors, and datasets are created for training, validation, and testing with specified batch sizes. The model is then set up with a sparse categorical crossentropy loss, Adam optimizer, and early stopping callback. The training process is executed, and memory usage and training time are monitored.

**Model Saving**

**Process:**

The trained DistilBERT model and tokenizer are saved to the specified directory ("/content/drive/MyDrive/DistilBERT\_FineTuned\_Model"). This enables the preservation of the model for future use and deployment.

**Model Loading and Evaluation on Test Set**

**Process:**

I load the previously saved DistilBERT model and tokenizer. Inference is performed on the test dataset, and the inference time is measured. The accuracy of the model is calculated, and a comprehensive classification report is generated to assess its performance.

| S. No. | Training Time | Memory Usage | Inference Time | Accuracy |
| --- | --- | --- | --- | --- |
| DistilBert  (Test Data) | 17737.15700817108 (sec) | 0.00685492578125 (MB) | 146 (sec) (82/82) | 81% |
| DistilBert  (TIL Test Data) |  |  | 67(sec)  54/54 |  |

**Inference on TIL Test Data**

**Process:**

The code reads an external test dataset, preprocesses the data, and tokenizes it using the loaded tokenizer. Inference is then conducted on the external test dataset, and the inference time is recorded. Predictions are decoded and mapped to target categories using a predefined label mapping. The predicted targets are added to the external test DataFrame for further analysis.

**For TIL Test Dataset (3400):**  [DistilBERT\_Text\_Classification\_Pipeline](https://colab.research.google.com/drive/1rWCkfwx3cOjvkxHt3v19s_gOtVL06sfM)

Inference Time: 290 (sec)