International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03305
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

DECODING REVIEW SENTIMENT ANALYSIS MODEL

(Mainly Using BERT)

Chehakpreet Kaur, UG Student, BCA, Maharaja Surajmal Institute, Delhi, India

Tashi Singh, UG Student, BCA, Maharaja Surajmal Institute, Delhi, India

Dr. Rimpy, Assistant Professor, Department of BCA, Maharaja Surajmal, Delhi, India

Abstract

Review Sentiment Analysis is performed so that, any business or organization can understand how their audience perceives their products or services. The study of their opinion provides them with extremely valuable information. The analysis helps in deciding where and on what aspects the organization should invest in. This paper reviews and looks over different types of methods of deep learning that can be used and expands on the BERT method. The BERT method is a deep learning model designed and used to improve the accuracy and efficiency of Natural Language Processing tasks.

Introduction

Sentiments are the different thoughts, emotions or attitudes caused by different circumstances or feelings. Every person has their own unique thoughts on any topic of discussion. For example, in context of review analysis, every customer has their own particular experience that shapes their sentiments towards various products/services. Something one person has a pleasant experience with, could be awful for another.

Sentiment Analysis is done to sort the experiences provided by different people. It is the process that is used to identify and categorize the emotions and thoughts of these people.

Different types of Sentimental Analysis :

Emotion Analysis : It is the type of analysis where in which different types of emotions- happiness, sadness, anger, etc. are classified.

Graded Sentiment Analysis : Used when the precision of the mood is important and should be further analysed. For example, using terms like- Very Happy, Very Sad, etc.

Aspect-based Sentiment Analysis : Focuses on the various aspects used to describe the different sentiments of the people.

Intent Analysis : The type where the central focus is on what the user intends to do. This allows better guidance of the people to their destinations.

Review Sentimental Analysis is mainly focused on organising different reviews in accordance to the sentiments of the customers. Many users leave reviews on sites like – Amazon, Flipkart, etc. about various products they bought and used.

These users make a lot of comments, it is almost impossible to process them manually. Sentiment analysis enables businesses to automatically classify large amounts of raw data.

With more data and information gathered through sentiment analysis, the organizations could develop an effective marketing strategy. The outcome from the strategies can be measured from the customers' positive or negative key messages.

Algorithms

We can sort Sentiment Analysis algorithms into three basic categories :

1. Lexicon based
2. Machine Learning based
3. Hybrid

Lexicon based :

First ever methods to be used for sentimental analysis. They can be approached in two ways – dictionary based and corpus based.

Dictionary based analysis is dependent on a dictionary with different terms. Unlike that, corpus-based analysis does not depend on a dictionary that is predefined but rather on statistical analysis of the data. Some techniques include K-NN, Hidden Markov models, etc.

Machine Learning based :

Machine Learning based methods are not dependent on any manually set rules but on different ML techniques. They can be categorized in two types – Traditional and Deep Learning models.

Traditional ML techniques consist of Naïve Bayes classifier, SVM's (Support Vector Machines), etc. Deep Learning techniques are based on artificial neural networks. These techniques provide better results than traditional methods.

Hybrid :

Hybrid methods fuse both Lexicon and ML based techniques together.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03305
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Visualization

Some different ways in which the data can be visualized (and examples) :

Word Cloud

Word Clouds break down the text of the data into words and count how frequently they appear. By this, we can understand what any dataset is about and what it's trying to portray.

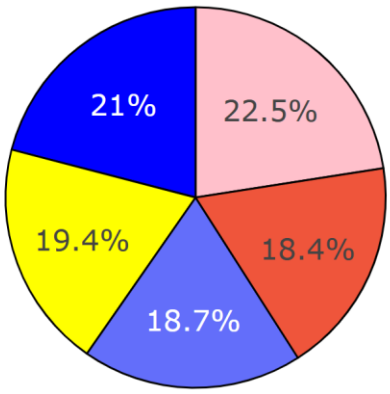
EXAMPLE :

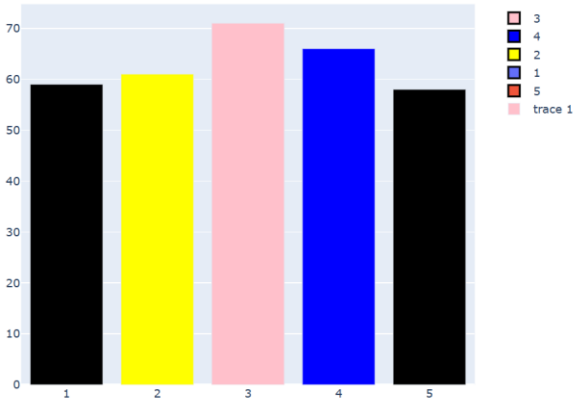
Target Count

Target Count tells the user about the number of targets the dataset has. This helps in deciding the type of model that is to be built and the methods that are to be used.

EXAMPLE :

International Scientific Journal of Engineering and Management (ISJEM)
Volume: 04 Issue: 05 | May – 2025
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI: 10.55041/ISJEM03305

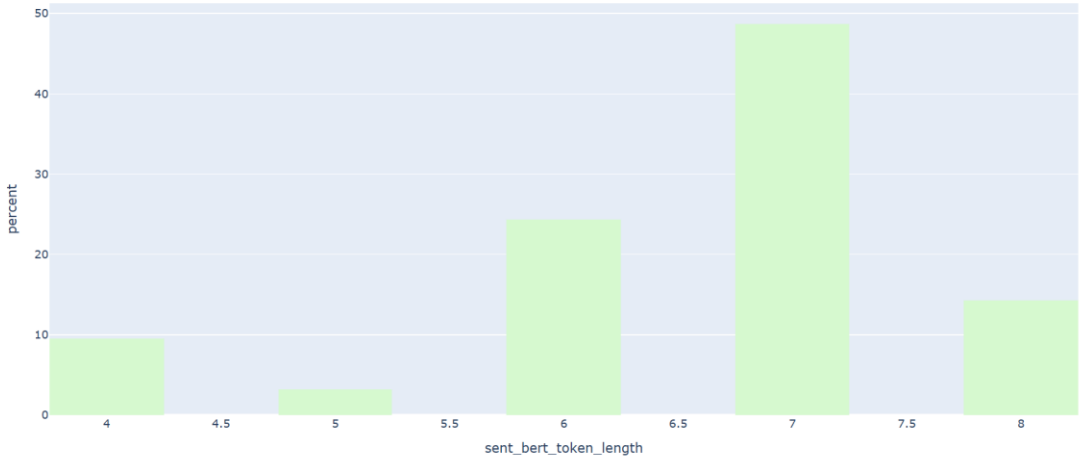




Token Counts with BERT tokenizer

This paper is mainly analysing the BERT method, so Token Counts here is very important input. Important value used in the model can be decided by taking a look at this.

EXAMPLE :



Transformer

Transformers are a type of neural network architecture that are used for NLP (Natural Language Processing) tasks particularly. It is a model that uses self-attention to process and transform the entire sentence. This mechanic is used to determine which words are most relevant to each other.

The structure is known as ‘encoder-decoder’. The encoder reads the text input and the decoder produces a prediction for the task.

ISSN: 2583-6129
DOI: 10.55041/ISJEM03305

BERT is a bi-directional transformer for pre-training over a lot of unlabelled textual data that can be used to fine-tune for specific machine learning tasks. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary.

Some other transformers out there :

RoBERTa. Introduced at Facebook, robustly optimized BERT approach RoBERTa, is a retraining of BERT with improved training methodology, 1000% more data and compute power. Larger batch-training sizes were also found to be more useful in the training procedure.

RoBERTa is basically a stronger, retrained version of BERT — but it's larger and needs more memory.

XLNet is a large bidirectional transformer that uses improved training methodology, larger data.

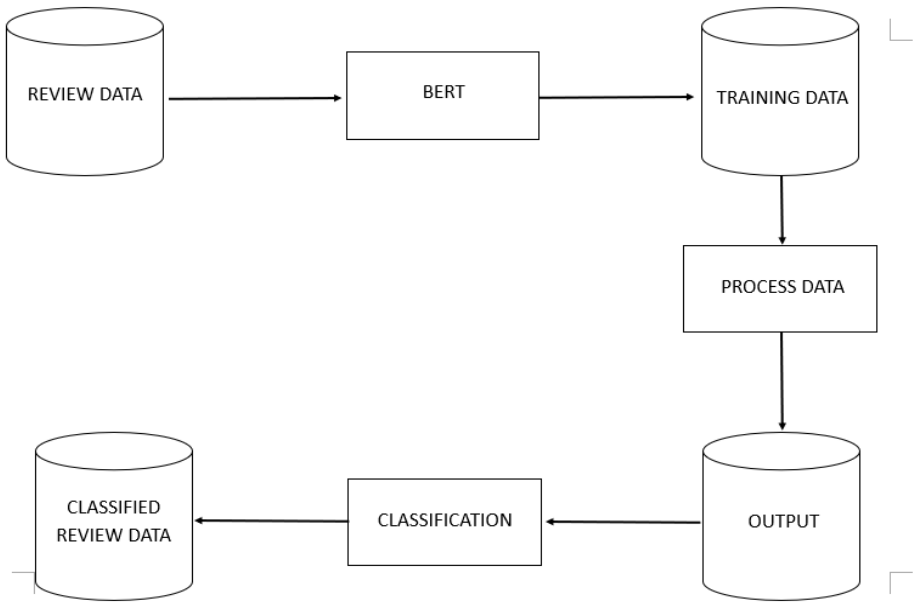
To improve the training, XLNet introduces permutation language modelling, where all tokens are predicted but in random order. This helps the model to learn bidirectional relationships and therefore better handles dependencies and relations between words.

XLNet is even more complex because it uses a special way of reading (permutation language modelling), so it's slower and heavier to train/fine-tune.

Methodology

Methodology represents the system and the procedures it takes to execute it. It gives a structured approach which tells us how the whole thing works. A framework is provided which carries us through the project lifecycle.

Architecture Design

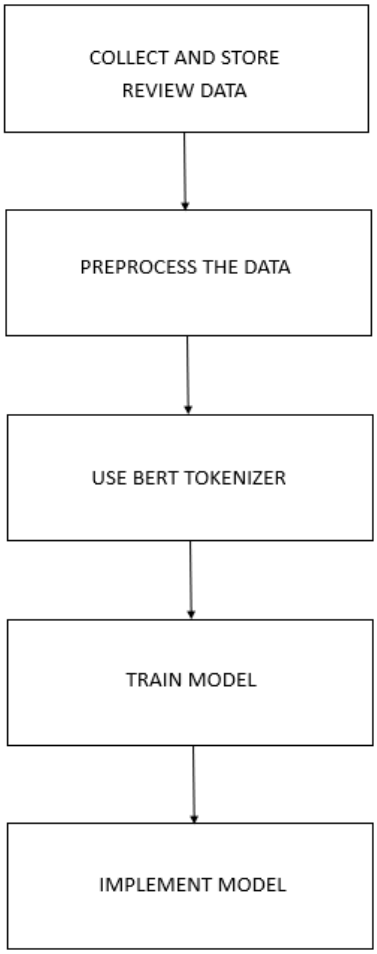


International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03305
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

This figure represents the architectural design. The architecture has many key components which helps in understanding the different layers of the system.

The Review Data is loaded in, pre-processed and then passed through the BERT layer. In the BERT layer, data is transformed and prepared for training. The data is trained and processed into the output. The output is then classified into different categories based on their review (and whether it was positive, neutral or negative)

Logical Design



The Logical Design of the data helps us in understanding the flow of the system. The data is loaded and stored, after it's pre-processed, it goes through the BERT tokenizer. The model is trained and implemented which sorts the data into different categories.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03305
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Process

Data Collection

The whole process begins with Data Collection.

	A	B
1	**review**	**rating**
2	Subpar materials, not durable.	1
3	Okayish product for the price.	3
4	Good value for the price.	3
5	A pleasant surprise, works great!	3
6	Good value for the price.	3
7	Exactly what I needed, highly recommend.	5
8	Subpar materials, not durable.	2
9	Not bad, could be better.	3
10	Five stars, absolutely love it!	4
11	Average quality, nothing special.	3
12	Fast shipping and excellent packaging.	5
13	Not what I expected, returned it.	1
14	Average quality, nothing special.	3
15	Good value for the price.	3

Dataset is generated and used as the basis of the whole model.

This data is pre-processed with methods like text cleaning, normalization and tokenization. These processes improve the data quality and efficiency of the model. After the data is collected, we can move to the next stage of the process.

Model Selection

Next step is the selection of the model.

These are contributing factors on which BERT is chosen :

- **Computational resources**: BERT is lightweight. Anyone can train BERT on a single medium-range GPU.
- **Bidirectional Understanding**: BERT reads text **in both directions** (left-to-right and right-to-left) at the same time.
- **Fine-Tuning is Easy**: BERT is designed to be easily fine-tuned on small datasets. You just add a small classification head and train for a few epochs.

Hence, we choose the BERT model for training.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03305
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Model Training

Finally, next is the model training.

The performance of the model is assessed through different metrics. After the accuracy of the model is calculated, the process is executed for a specific number of epochs. Epoch occurs when one complete pass through the entire training dataset is done by the model. After each epoch, the model improves and adjusts its parameters to minimize the error. It learns more from the data after every cycle.

```
Epoch 2/3
Training:   4%|  | 2/57 [00:23<10:35, 11.Training:   4%|  | 2/57 [00:36<10:35, 11.Training:   5%|  | 3/57
Training:  46%|▌ | 26/57 [05:26<07:03, 13.Training:  Training Loss: 0.009
Training:   4%|  | 2/57 [00:23<10:35, 11.Training:   4%|  | 2/57 [00:36<10:35, 11.Training:   5%|  | 3/57
Training:  46%|▌ | 26/57 [05:26<07:03, 13.Training:  Training Loss: 0.009
Training:  Training Loss: 0.009
Validation Loss: 0.001
Validation Loss: 0.001
Validation F1 Score: 1.000
Model saved to model_epoch_2.pt
Model saved to model_epoch_2.pt

Epoch 3/3
Epoch 3/3
Training Loss: 0.002
Training Loss: 0.002
Validation Loss: 0.001
Validation Loss: 0.001
Validation F1 Score: 1.000
Model saved to model_epoch_3.pt  & C:/Users/tashi/Desktop/major/env/Scripts/python.exe c:/Users/tashi/
```

Explaining the steps taken in our CODE

Various libraries imported in the code :

Random –

Used for generating random numbers and performing random selections.

NumPy (np) –

Used to support large, multi-dimensional arrays and provide with a collection of mathematical functions to operate with.

Pandas –

Used for data manipulation and analysis.

Torch –

Used to provide support for defining, training and running neural networks.

re –

Used to provide for regex (regular expressions) which allow pattern matching and string manipulation.

OS –

ISSN: 2583-6129
DOI: 10.55041/ISJEM03305

Used to interact with the operating system for things like path handling, file manipulation, etc.

Transformers –

Used to provide pre-trained models and tools for NLP (Natural Language Processing) tasks.

Scikit-learn –

Used to provide tools for classification, regression, clustering and model evaluation.

TQDM –

Used to track progress of long running operations. Adds progress bars to loops.

```
def set_seed(seed_val=42):
    random.seed(seed_val)
    np.random.seed(seed_val)
    torch.manual_seed(seed_val)
    torch.cuda.manual_seed_all(seed_val)
```

This is used in the beginning of the code to ensure the reproducibility of the model. It sets a seed value which ensures that the random processes produce the same results every time someone runs the code. It is crucial during both training and preprocessing phases.

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r"[^A-Za-z0-9.\s]", " ", text)
    return text

def label_encode(rating):
    mapping = {1: 0, 2: 0, 3: 1, 4: 2, 5: 2}
    return mapping.get(rating, -1)
```

Now, we move to the pre-processing. The clean_text function converts the text to lowercase and removes any characters that are not alphanumeric, periods or spaces.

The label_encode function transforms categorical labels into numerical values, since models typically make use of numeric labels. Here,

Ratings 1 and 2 are mapped to 0

International Scientific Journal of Engineering and Management (ISJEM)
Volume: 04 Issue: 05 | May – 2025
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI: 10.55041/ISJEM03305

Ratings 3 is mapped to 1

Ratings 4 and 5 are mapped to 2

```python
def create_optimizer_scheduler(model, dataloader, config):
    optimizer = AdamW(model.parameters(), lr=config.lr, eps=config.eps)
    scheduler = get_linear_schedule_with_warmup(
        optimizer,
        num_warmup_steps=0,
        num_training_steps=len(dataloader) * config.epochs
    )
    return optimizer, scheduler
```

The function is designed to create an optimizer and a learning rate scheduler for training a model. AdamW is a optimizer commonly used with transformer models because it performs well in training deep neural networks.

```python
def train_epoch(model, dataloader, optimizer, scheduler, device):
    model.train()
    total_loss = 0
    progress_bar = tqdm(dataloader, desc='Training', leave=False)
    for batch in progress_bar:
        batch = tuple(t.to(device) for t in batch)
        inputs = {'input_ids': batch[0], 'attention_mask': batch[1], 'labels': batch[2]}
        model.zero_grad()
        outputs = model(**inputs)
        loss = outputs.loss
        total_loss += loss.item()
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
        optimizer.step()
        scheduler.step()
        progress_bar.set_postfix({'loss': f'{loss.item():.3f}'})
    return total_loss / len(dataloader)
```

This function is used for training a single epoch of the model. It makes one full pass over the entire dataset.

International Scientific Journal of Engineering and Management (ISJEM)
Volume: 04 Issue: 05 | May – 2025
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI: 10.55041/ISJEM03305

```python
def evaluate_model(model, dataloader, device):
    model.eval()
    total_loss = 0
    predictions, true_labels = [], []
    with torch.no_grad():
        for batch in dataloader:
            batch = tuple(t.to(device) for t in batch)
            inputs = {'input_ids': batch[0], 'attention_mask': batch[1], 'labels': batch[2]}
            outputs = model(**inputs)
            loss, logits = outputs.loss, outputs.logits
            total_loss += loss.item()
            predictions.append(logits.detach().cpu().numpy())
            true_labels.append(inputs['labels'].cpu().numpy())
    avg_loss = total_loss / len(dataloader)
    predictions = np.concatenate(predictions, axis=0)
    true_labels = np.concatenate(true_labels, axis=0)
    return avg_loss, predictions, true_labels
```

This function is used to evaluate the performance of the model on a test dataset after training.

```python
def f1_score_weighted(preds, labels):
    preds_flat = np.argmax(preds, axis=1)
    labels_flat = labels.flatten()
    return f1_score(labels_flat, preds_flat, average='weighted')
```

This function computes the F1 score, which is a performance metric for classification problems.

F1 score combines precision and recall into a single metric. It is especially helpful when there is an uneven class distribution.

```python
class Config:
    lr = 2e-5
    eps = 1e-8
    epochs = 3
    batch_size = 8
    seed_val = 42
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model_name = 'distilbert-base-uncased'  # Lightweight fast model

config = Config()
```

The class is used to store various hyperparameters and settings needed for training the model.

The instance of Config is created and allows access to all configuration settings defined in the class.

International Scientific Journal of Engineering and Management (ISJEM)
Volume: 04 Issue: 05 | May – 2025
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI: 10.55041/ISJEM03305

```
df = pd.read_csv('reviews.csv')

if len(df) > 500:
    df = df.sample(n=500, random_state=42).reset_index(drop=True)

df['clean_review'] = df['review'].apply(clean_text)
df['label'] = df['rating'].apply(label_encode)
```

The dataset is loaded and filled into a pandas dataframe. If the dataset has more than 500 reviews, it selects them randomly for sampling.

The clean_text function is applied to each review in the column. The new text is stored in a new column – clean_review.

BEFORE		AFTER	
Review	rating	clean_review	label
"Great product!"	4	"great product"	2
"The product is okay."	3	"the product is okay"	1
"WORST PRODUCT EVER!!!"	2	"worst product ever"	0

This concludes pre-processing.

International Scientific Journal of Engineering and Management (ISJEM)
Volume: 04 Issue: 05 | May – 2025
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI: 10.55041/ISJEM03305

```
tokenizer = BertTokenizer.from_pretrained(config.model_name)

encoded = tokenizer.batch_encode_plus(
    df['clean_review'].values,
    add_special_tokens=True,
    return_attention_mask=True,
    padding='max_length',
    max_length=128,
    truncation=True,
    return_tensors='pt'
)

input_ids = encoded['input_ids']
attention_masks = encoded['attention_mask']
labels = torch.tensor(df['label'].values)
```

First, the BERT tokenizer is loaded. It converts raw text data into input tokens for the BERT model to understand.

The batch_encode_plus method takes input and returns tokenized input compatible with the model.

This portion of the code prepares the tokenized input, the attention mask and the labels that the model trains on.

```
train_inputs, val_inputs, train_masks, val_masks, train_labels, val_labels = train_test_split(
    input_ids, attention_masks, labels,
    test_size=0.1, random_state=config.seed_val, stratify=labels
)

train_data = TensorDataset(train_inputs, train_masks, train_labels)
val_data = TensorDataset(val_inputs, val_masks, val_labels)

train_dataloader = DataLoader(train_data, sampler=RandomSampler(train_data), batch_size=config.batch_size)
val_dataloader = DataLoader(val_data, sampler=SequentialSampler(val_data), batch_size=config.batch_size)

model = BertForSequenceClassification.from_pretrained(
    config.model_name,
    num_labels=3
)
model.to(config.device)
```

The dataset is split into training and validation sets. The TensorDataset is used to wrap the data so that it can be passed into a DataLoader.

DataLoader is used for training, sampling and validation. Pre-trained model is loaded for sequence classification tasks and then the model is moved to the specified device.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03305
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

```
def train(model, train_dataloader, val_dataloader, config):
    optimizer, scheduler = create_optimizer_scheduler(model, train_dataloader, config)
    for epoch in range(1, config.epochs + 1):
        print(f'\nEpoch {epoch}/{config.epochs}')
        train_loss = train_epoch(model, train_dataloader, optimizer, scheduler, config.device)
        print(f'Training Loss: {train_loss:.3f}')
        val_loss, predictions, true_labels = evaluate_model(model, val_dataloader, config.device)
        val_f1 = f1_score_weighted(predictions, true_labels)
        print(f'Validation Loss: {val_loss:.3f}')
        print(f'Validation F1 Score: {val_f1:.3f}')
        model_path = f'model_epoch_{epoch}.pt'
        torch.save(model.state_dict(), model_path)
        print(f'Model saved to {model_path}')

train(model, train_dataloader, val_dataloader, config)
```

The optimizer and learning rate scheduler are created for training the model. The model is trained for the specified number of epochs. Loss is calculated and the parameters are updated with each loop. Data is saved to a file at the end of each epoch.

The ending line is used to actually start the training process.

Future Enhancement

Regular updating of the model with different datasets will help in improvement. Making it future proof would make it a lot more effective. This will allow it to be relevant and increase its efficiency.

Future Enhancement should have the model be able to understand the reviews on a global scale. Multilingual analysis should be implemented so that users all over the globe can be integrated in the system. Organisations can reach customers outside of their scope and avoid language constraints.

Conclusion

In conclusion, the usage of BERT is an improvement for NLP related models. It helps the model to understand and analyse different linguistic sentiments.

The model has great accuracy in segregating different user generated reviews and classify them into the desired categories.

The customers play a pivotal role in shaping the decisions made by organisations. This model helps them in understanding their audience better and take steps toward financially smart directions.

ISSN: 2583-6129
DOI: 10.55041/ISJEM03305

References

[1] “Sentiment Analysis Based on Deep Learning: A Comparative Study” Nhan Cach Dang, María N. Moreno-García and Fernando De la Prieta

[2] “BERT: Pre-training of Deep Bidirectional Transformmers for Language Understanding” Jacob Devin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova

[3] “A Review On Sentiment Analysis Methodologies, Practices And Applications” Pooja Mehta, Dr. Sharnil Pandya

[4] “Deciphering Product Review Sentiments Using BERT and TensorFlow” D. Suraj, S. Dinesh, R. Balaji