Milestone 3: Project Report
DS677 Deep Learning
Spring 2025

Daniel Ayer↑[5], Fahriye Gurel↑[6], Jaime Corea↑[7]
↑ : Indicates Equal Contribution

Abstract

This project aims to examine three different transformer-based architectures to evaluate their performance on a text classification task using the same dataset. First, we implement a custom base transformer model inspired by DistilBERT, constructed from scratch with token and positional embeddings, stacked self-attention layers, and a classification head. Next, we enhance this architecture by adding dropout regularization, aiming to reduce overfitting and improve generalization. Finally, we fine-tune a pretrained DistilBERT model from the Transformers library, leveraging transfer learning. By comparing the training and validation metrics across these three approaches, we assess how architectural modifications and transfer learning influence model performance.

The datasets that are used in this project consist of approximately 20,000 abstracts from randomized controlled trials (RCTs). Each sentence is labeled according to its rhetorical role in the abstract, using one of five predefined categories: *background*, *objective*, *method*, *result*, or *conclusion*. And the numbers in the datasets are replaced by the @ sign.

We were able to train de novo DistilBERT models both with and without dropout which achieved similar performance as a fine tuned DistilBERT model.

1. Introduction

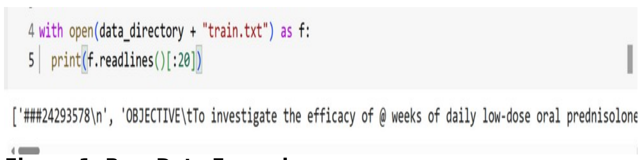
Bidirectional Encoder Representations from Transformers (BERT) is a neural network architecture which produces models which can represent text as numeric vectors. It was developed by Google researchers in 2019.[1] The trained models can perform Next Sentence Prediction, *Masked Token Prediction*, and *Sentence Sentiment Classification*. DistilBERT was devloped in 2020 by Victor Sanh, Lysandre Debut, Julien Chaumond, Thomas Wolf.[2] It utilized a teacher and student model pair where the student model would approximate the output of the teacher model with fewer parameters at a low cost to performance. "It retains 97% of BERT's performance while being 40% smaller and 60% faster making it highly efficient for NLP tasks such as text classification, sentiment analysis and question-answering."[3]

Our DistilBERT inspired model uses 6 Transformer blocks to create a lightweight BERT model without the need for the Teacher/Student model dynamic. BERT uses 12 transformer blocks.

2. The Dataset

In this section we describe the dataset used in training these models. The same dataset was used for each model type.

The dataset comes from 20,000 medical record abstracts[4] and is labeled by unique identifier for the abstract, and class labels for each line. Classes: OBJECTIVE, METHODS, RESULTS, CONCLUSIONS, BACKGROUND.



```
with open(data_directory + "train.txt") as f:
  print(f.readlines()[:20])
```

```
['###24293578\n', 'OBJECTIVE\tTo investigate the efficacy of @ weeks of daily low-dose oral prednisolone
```

Figure 1. Raw Data Example

2.1 Data Description and Visualization

Figure 2 visualizes the distribution of different sections in a dataset labeled with five scientific article components: OBJECTIVE, METHODS, RESULTS, CONCLUSIONS, and BACKGROUND. The largest proportion belongs to the OBJECTIVE class, comprising 33% of the data, followed closely by METHODS at 32.2%. The RESULTS section makes up 15.1%, while CONCLUSIONS and BACKGROUND constitute smaller shares of 12.1% and 7.69%, respectively.

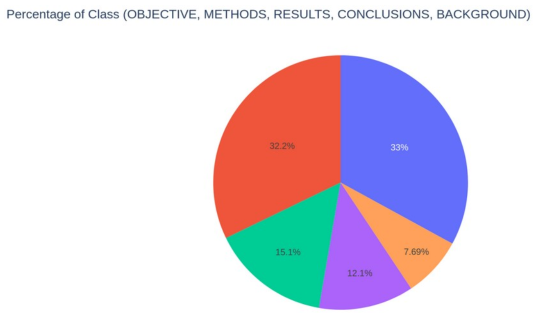


Figure 2. Class Labels Percent Occurance

Figure 3 illustrates the distribution of the dataset used for training, validation, and testing. The dataset is divided into three subsets: 74.9% is allocated to the training set (180,040 samples), which forms the majority and is used to train the model. The validation set comprises 12.6% of the data and is used to tune model hyperparameters and monitor overfitting during training. Lastly, the test set accounts for 12.5%, reserved for evaluating the model's performance on unseen data.

1 https://arxiv.org/abs/1810.04805
2 https://arxiv.org/abs/1910.01108
3 https://www.geeksforgeeks.org/distilbert-in-natural-language-processing/
4 https://arxiv.org/abs/1710.06071
5 da632@njit.edu
6 fg245@njit.edu
7 jdc53@njit.edu

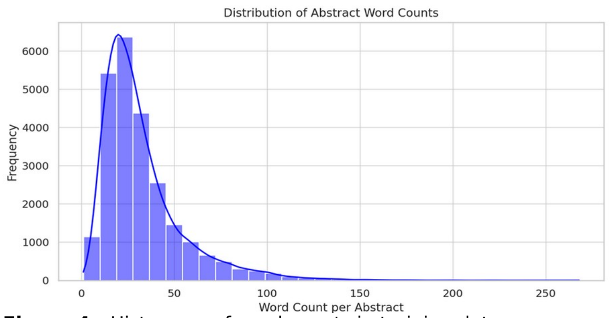


Figure 4. Histogram of word counts in training data.

Figure 4 displays the distribution of abstract word counts within a dataset. The x-axis represents the number of words per abstract, while the y-axis shows the frequency of abstracts with that word count. The distribution is right-skewed, with the majority of abstracts containing between 10 and 60 words. **Figure 5** demonstrates the frequency of the top 20 most frequent words (omitting common articles) in the data.

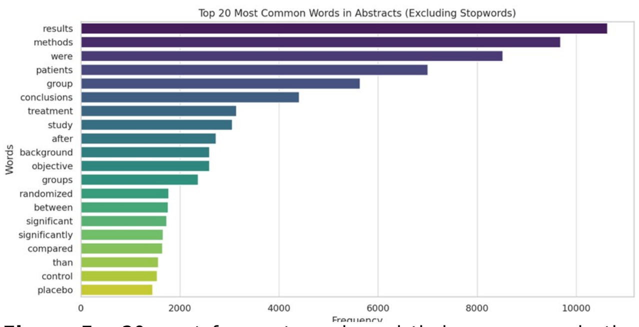
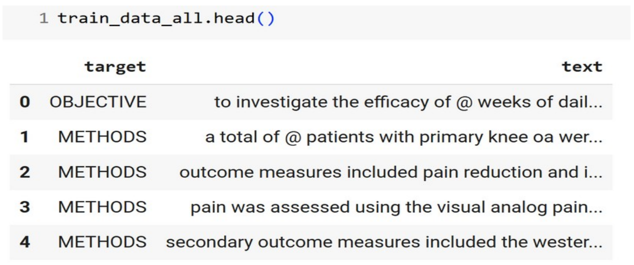


Figure 5. 20 most frequent words and their occurrence in the training data.

2.2 Data Preprocessing

To prepare the raw abstract data for processing, we created a *preprocess_data* function. This function reads a structured text file containing labeled research abstracts and extracts the relevant information in a clean, structured format. It processes the file line by line, identifying new abstracts with lines starting in "###" and recognizing the end of each abstract using blank lines. For each abstract, it splits the content into individual lines, separates the labels (targets) and corresponding text using tab delimiters, and standardizes the text by converting it to lowercase. The processed data is stored as a list of dictionaries, where each dictionary contains a target and text key. This preprocessing step ensures the data is formatted appropriately for training our models for sentence classification. **Figure 6** shows a typical result.



```
train_data_all.head()
```

	target	text
0	OBJECTIVE	to investigate the efficacy of @ weeks of dail...
1	METHODS	a total of @ patients with primary knee oa wer...
2	METHODS	outcome measures included pain reduction and i...
3	METHODS	pain was assessed using the visual analog pain...
4	METHODS	secondary outcome measures included the wester...

Figure 6. Processed data in dataframe.

3. Data Sculpting

3.1 Label Encoder Tokenization & Padding

We created the TextDataset class to efficiently prepare and serve textual data to the DistilBert-based models within a PyTorch training pipeline. This custom dataset takes in a DataFrame containing text samples and their corresponding labels, along with a tokenizer and an optional maximum sequence length (defaulted to 128). To tokenize the input text, we use the DistilBertTokenizerFast from the pre-trained "distilbert-base-uncased" model.

This tokenizer converts raw text into token IDs suitable for DistilBERT while also generating the corresponding attention masks.

Within the *__getitem__* method in the TextDataset class, each text sample is tokenized with truncation and padding applied to the specified length, and the labels are encoded into a numerical format using a label encoder. Each returned sample includes input_ids, an attention_mask, and the encoded label, all formatted as PyTorch tensors. This setup is essential for integrating with a PyTorch DataLoader to enable efficient batching and model training.

3.2 The Data Loaders

After defining the TextDataset class, we instantiate dataset objects for the training, validation, and test splits. These datasets handle preprocessing and tokenization of the text samples.. Next, we wrap each dataset in a PyTorch DataLoader, which enables efficient batch loading and optional shuffling. The data loader is configured with a *batch_size of 16* and *shuffle=True* to ensure the model sees the training data in a different order each epoch, promoting better generalization. The *val_loader* and *test_loader* use a larger batch size of *32* for evaluation efficiency and do not shuffle the data, preserving the original order. These data loaders are essential for feeding batches of tokenized inputs and labels to the model during training and evaluation.

4. The Train and Evaluate Functions

We designed the train function to handle the training process of a PyTorch-based deep learning model over multiple epochs. It accepts the model, data loaders for training and validation sets, a loss function, an optimizer, the computing device (CPU or GPU), and the number of training epochs. The function maintains a history dictionary to track and store the loss and accuracy for both the training and validation sets after each epoch. During the training mode, the function iteratively processes batches of input data. The model generates predictions, computes the loss, performs backpropagation, and updates the model weights using the optimizer. Predictions and true labels are collected to compute training accuracy.

The evaluate function is designed to assess the performance of a trained model on a given dataset without updating its parameters. It sets the model to evaluation mode using model.eval() to ensure that layers like dropout or batch normalization behave accordingly, and the function iterates over the provided dataloader. For each batch, the model generates output predictions. The loss between the predictions and the true labels is computed and stored.

5. Experiments

5.1 The Base Model

The base model is a transformer-based architecture designed for sequence classification tasks. It consists of several key components:

1. **Embeddings:** The model uses two types of embeddings:
   Token Embeddings: These embeddings are learned representations for each token in the input vocabulary. The embedding layer is initialized with a vocabulary size (vocab_size) and a hidden dimension (hidden_size).
   Positional Embeddings: These embeddings encode the position of each token within the input sequence, helping the model understand the order of tokens. The size of this embedding layer is determined by the maximum sequence length (max_position_embeddings) and the hidden dimension.

2. **Transformer Blocks:** The model employs a stack of transformer layers (TransformerBlock), which is responsible for learning contextualized representations of input tokens. Each transformer block consists of attention mechanisms (multi-head self-attention) that enable the model to attend to different parts of the input sequence simultaneously, layer normalization that stabilizes and accelerates the training process, and feed-forward layers, which are applied iteratively over the input data. The feedforward component consists of two fully connected layers with a GELU activation function in between.

3. **Classifier:** After the input sequence passes through the transformer blocks, the model extracts the representation of the first token (typically used as the [CLS] token in transformer models) and passes it through a linear layer (nn.Linear) to output the class probabilities. The number of output units in this layer corresponds to the number of target classes (num_classes).

This architecture is designed to process sequences efficiently by using token and positional embeddings, followed by a stack of transformer layers that capture contextual information from the input. The final classifier produces a prediction for sequence-level classification tasks.

After the architecture is designed, the model is initialized as an instance of the custom DistilBERT class, which is initialized with five output classes for classification. The loss function chosen is CrossEntropyLoss, which is suitable for multi-class classification problems as it measures the performance of a classification model whose output is a probability distribution across classes. The optimizer used is AdamW, an improved version of Adam that decouples weight decay from the gradient update, which often leads to better generalization. The learning rate is set to 5e-5, a common starting point for fine-tuning transformer-based models. This setup forms the foundation for training and optimizing the model effectively.

The training and validation output indicates that the model is learning effectively and generalizing reasonably well. Over the five epochs, the training loss consistently decreases from 0.6170 to 0.3027, and the training accuracy improves steadily from 76.61% to 88.71%, demonstrating that the model is fitting the training data increasingly well.

For the validation set, the loss starts at 0.5031 and fluctuates slightly, ending at 0.5159, while the validation accuracy improves from 80.99% to a peak of 82.50% before slightly dipping. This pattern suggests that while the model is learning, it may be starting to slightly overfit after the third or fourth epoch, as indicated by the increasing validation loss.

Overall, the model demonstrates strong performance with a clear upward trend in training metrics and generally stable validation metrics, suggesting a successful training process with potential for further fine-tuning or regularization to improve generalization.

5.2 The Updated Model With Dropout

In this section, the model with dropout builds upon the base by adding regularization through dropout to enhance generalization and reduce overfitting. The model comprises token and positional embedding layers, followed by a stack of transformer blocks like the transformer block that is used in the base model. Each block contains a multi-head self-attention mechanism and a position-wise feedforward network, both followed by residual connections and layer normalization. A dropout layer is added before the final classifier, which randomly turns off some parts of the [CLS] token's representation during training (with a 30% chance). This helps the model avoid relying too much on certain features and makes it more general and reliable. After that, a final linear layer takes the [CLS] token and predicts one of the output classes (by default, 5), making the model suitable for text classification.

After training the model with the same dataset, the results show strong and consistent performance over five training epochs. Training accuracy steadily improved from 75.9% to 87.4%, while the training loss decreased significantly from 0.6372 to 0.3429, indicating effective learning. Validation accuracy remained relatively stable, starting at 81.3% and slightly improving to 82.6% by the fifth epoch. Although the validation loss fluctuated slightly, it remained within a narrow range around 0.49–0.51. These results suggest that the model is learning well without severe overfitting, and the addition of dropout likely helped maintain generalization on the validation set.

5.3 The Fine-Tuning Model

In this section, it is taken a pretrained model is used that has already learned general language representations, and this process leverages transfer learning, where the model retains its understanding of language structure while learning to perform a new task such as sentiment analysis and text classification. During fine-tuning, the model is trained on the same dataset.

In the results, from the first epoch, it starts strong with a validation accuracy of 85.9% and continues improving, peaking at 86.7% in the second epoch. Training accuracy steadily climbs from 84.5% to 92.5% by the fifth epoch, showing effective learning of the dataset. Although validation loss begins to increase slightly after epoch 2—from 0.3666 to 0.4402—validation accuracy remains consistently high, indicating only mild overfitting and strong generalization overall.

5.4. Masked Language Modeling (MLM) Fine-Tuning

As part of the project, we decided to try something different by using Masked Language Modeling (MLM) with DistilBERT. The idea was to see how the model would perform if, instead of classifying whole sentences, we trained it to predict missing words within sentences from the abstracts.

We used our dataset but removed the labels, so we just had plain text. Then We set up the model to randomly mask some words and train it to fill them in. We used the DistilBertForMaskedLM model for this, along with the data collator that handles the random masking part. We trained it for 3 epochs and saw the loss go down from about 2.07 to 1.73, which shows it was learning.

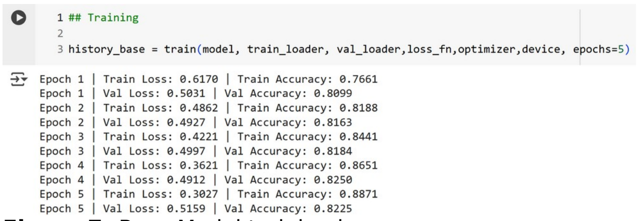
6. Results

6.1 The Base Model

The training and validation output indicates that the model is learning effectively and generalizing reasonably well. Over the five epochs, the training loss consistently decreases from 0.6170 to 0.3027, and the training accuracy improves steadily from 76.61% to 88.71%, demonstrating that the model is fitting the training data increasingly well.

For the validation set, the loss starts at 0.5031 and fluctuates slightly, ending at 0.5159, while the validation accuracy improves from 80.99% to a peak of 82.50% before slightly dipping. This pattern suggests that while the model is learning, it may be starting to slightly overfit after the third or fourth epoch, as indicated by the increasing validation loss.

Overall, the model demonstrates strong performance with a clear upward trend in training metrics and generally stable validation metrics, suggesting a successful training process with potential for further fine-tuning or regularization to improve generalization.



```
## Training

history_base = train(model, train_loader, val_loader,loss_fn,optimizer,device, epochs=5)
```

```
Epoch 1 | Train Loss: 0.6170 | Train Accuracy: 0.7661
Epoch 1 | Val Loss: 0.5031 | Val Accuracy: 0.8099
Epoch 2 | Train Loss: 0.4862 | Train Accuracy: 0.8188
Epoch 2 | Val Loss: 0.4927 | Val Accuracy: 0.8163
Epoch 3 | Train Loss: 0.4221 | Train Accuracy: 0.8441
Epoch 3 | Val Loss: 0.4997 | Val Accuracy: 0.8184
Epoch 4 | Train Loss: 0.3621 | Train Accuracy: 0.8651
Epoch 4 | Val Loss: 0.4912 | Val Accuracy: 0.8250
Epoch 5 | Train Loss: 0.3027 | Train Accuracy: 0.8871
Epoch 5 | Val Loss: 0.5159 | Val Accuracy: 0.8225
```

Figure 7: Base Model training log.

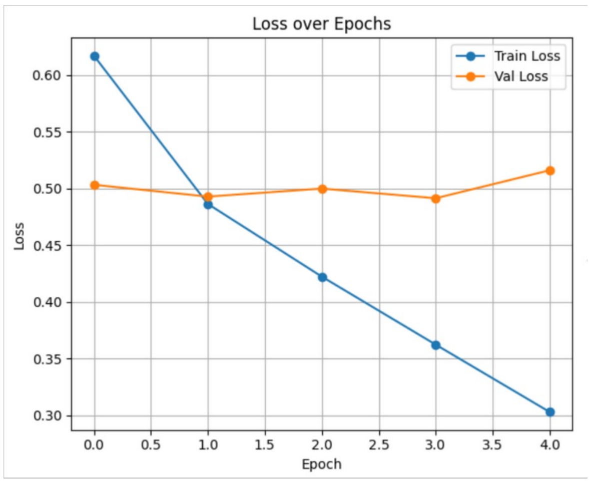


Figure 8: Base Mode loss plot.

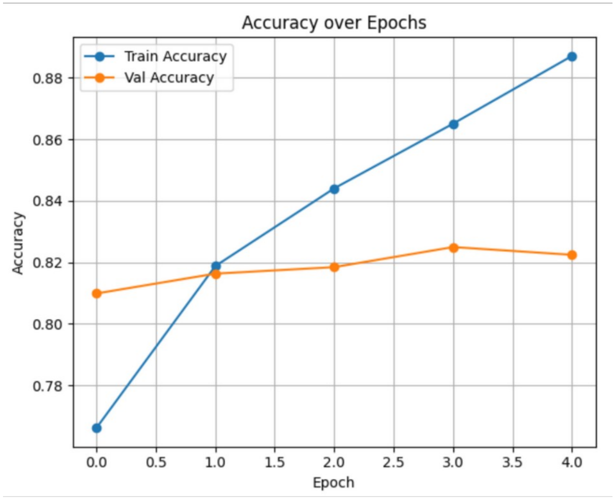
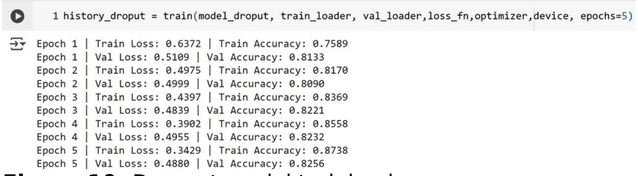


Figure 9: Base Model accuracy plot.

6.2 The Updated Model With Dropout

After training the model with the same dataset, the results show strong and consistent performance over five training epochs. Training accuracy steadily improved from 75.9% to 87.4%, while the training loss decreased significantly from 0.6372 to 0.3429, indicating effective learning. Validation accuracy remained relatively stable, starting at 81.3% and slightly improving to 82.6% by the fifth epoch. Although the validation loss fluctuated slightly, it remained within a narrow range around 0.49–0.51. These results suggest that the model is learning well without severe overfitting, and the addition of dropout likely helped maintain generalization on the validation set.



```
history_dropout = train(model_droput, train_loader, val_loader,loss_fn,optimizer,device, epochs=5)
```

```
Epoch 1 | Train Loss: 0.6372 | Train Accuracy: 0.7589
Epoch 1 | Val Loss: 0.5109 | Val Accuracy: 0.8133
Epoch 2 | Train Loss: 0.4975 | Train Accuracy: 0.8170
Epoch 2 | Val Loss: 0.4999 | Val Accuracy: 0.8090
Epoch 3 | Train Loss: 0.4397 | Train Accuracy: 0.8369
Epoch 3 | Val Loss: 0.4839 | Val Accuracy: 0.8221
Epoch 4 | Train Loss: 0.3902 | Train Accuracy: 0.8558
Epoch 4 | Val Loss: 0.4955 | Val Accuracy: 0.8232
Epoch 5 | Train Loss: 0.3429 | Train Accuracy: 0.8738
Epoch 5 | Val Loss: 0.4880 | Val Accuracy: 0.8256
```

Figure 10: Dropout model training log.

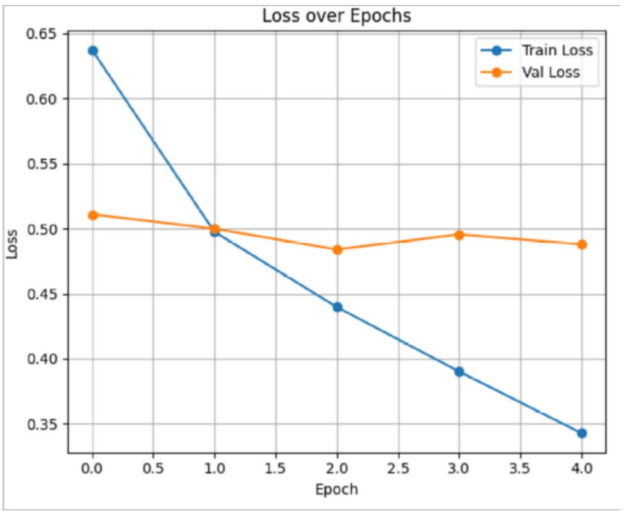


Figure 11: Dropout model loss plot.

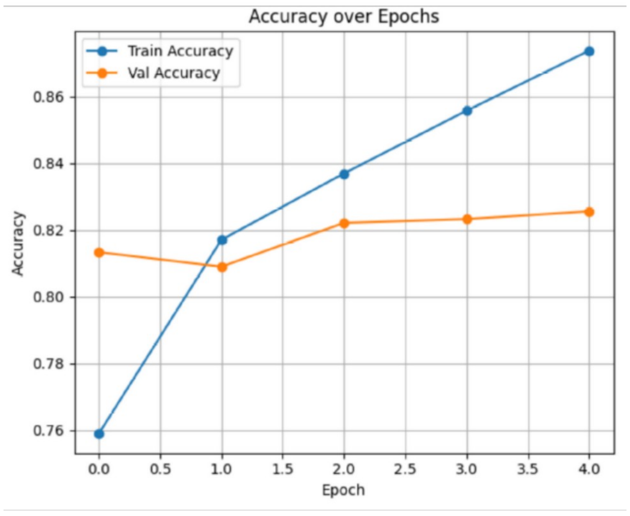
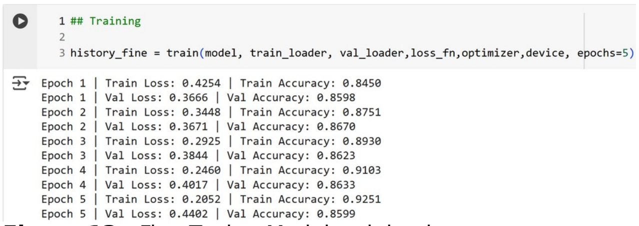


Figure 12: Dropout Model accuracy plot.

6.3 The Fine-Tuning Model

In the results, from the first epoch, it starts strong with a validation accuracy of 85.9% and continues improving, peaking at 86.7% in the second epoch. Training accuracy steadily climbs from 84.5% to 92.5% by the fifth epoch, showing effective learning of the dataset. Although validation loss begins to increase slightly after epoch 2—from 0.3666 to 0.4402—validation accuracy remains consistently high, indicating only mild overfitting and strong generalization overall.



```
## Training

history_fine = train(model, train_loader, val_loader,loss_fn,optimizer,device, epochs=5)
```

```
Epoch 1 | Train Loss: 0.4254 | Train Accuracy: 0.8450
Epoch 1 | Val Loss: 0.3666 | Val Accuracy: 0.8598
Epoch 2 | Train Loss: 0.3448 | Train Accuracy: 0.8751
Epoch 2 | Val Loss: 0.3671 | Val Accuracy: 0.8670
Epoch 3 | Train Loss: 0.2925 | Train Accuracy: 0.8930
Epoch 3 | Val Loss: 0.3844 | Val Accuracy: 0.8623
Epoch 4 | Train Loss: 0.2460 | Train Accuracy: 0.9103
Epoch 4 | Val Loss: 0.4017 | Val Accuracy: 0.8633
Epoch 5 | Train Loss: 0.2052 | Train Accuracy: 0.9251
Epoch 5 | Val Loss: 0.4402 | Val Accuracy: 0.8599
```

Figure 13: Fine Tuning Model training log.

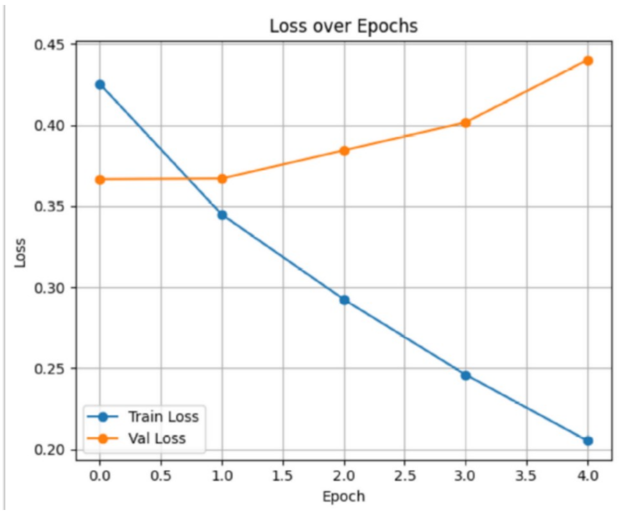


Figure 14: Fine Tuning Model loss plot.

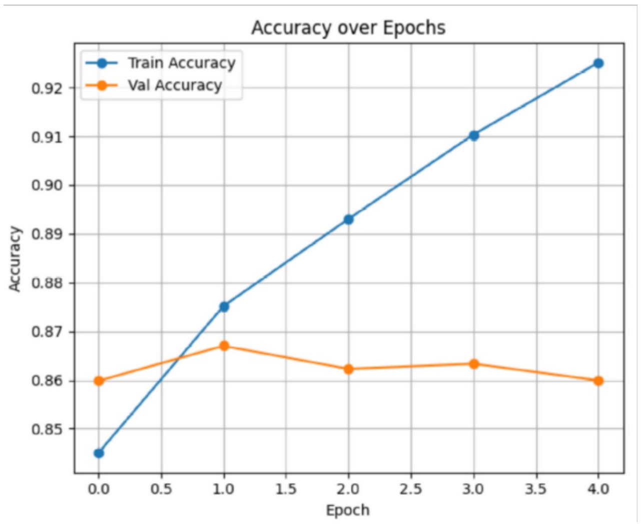
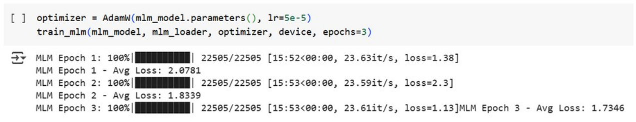


Figure 15: Fine Tuning accuracy plot.

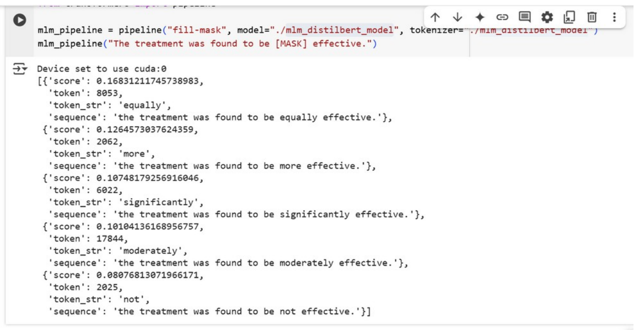
6.4 Masked Language Modeling (MLM) Fine Tuning



```
optimizer = AdamW(mlm_model.parameters(), lr=5e-5)
train_mlm(mlm_model, mlm_loader, optimizer, device, epochs=3)
```

```
MLM Epoch 1: 100%|██████████| 22505/22505 [15:52<00:00, 23.63it/s, loss=1.38]
MLM Epoch 1 - Avg Loss: 2.0781
MLM Epoch 2: 100%|██████████| 22505/22505 [15:53<00:00, 23.59it/s, loss=2.3]
MLM Epoch 2 - Avg Loss: 1.8339
MLM Epoch 3: 100%|██████████| 22505/22505 [15:53<00:00, 23.61it/s, loss=1.13]MLM Epoch 3 - Avg Loss: 1.7346
```

Figure 16: MLM Fine Tuning Training Log.



```
mlm_pipeline = pipeline("fill-mask", model="./mlm_distilbert_model", tokenizer="./mlm_distilbert_model")
mlm_pipeline("The treatment was found to be [MASK] effective.")
```

```
Device set to use cuda:0
[{'score': 0.16831211745738983,
  'token': 8053,
  'token_str': 'equally',
  'sequence': 'the treatment was found to be equally effective.'},
 {'score': 0.1264573037624359,
  'token': 2062,
  'token_str': 'more',
  'sequence': 'the treatment was found to be more effective.'},
 {'score': 0.10748179256916046,
  'token': 6022,
  'token_str': 'significantly',
  'sequence': 'the treatment was found to be significantly effective.'},
 {'score': 0.10104136168956757,
  'token': 17844,
  'token_str': 'moderately',
  'sequence': 'the treatment was found to be moderately effective.'},
 {'score': 0.08076813071966171,
  'token': 2025,
  'token_str': 'not',
  'sequence': 'the treatment was found to be not effective.'}]
```

Figure 16: MLM Fine Tuning Performance Measure

7. Comparison

Comparing the base model and the dropout model, both show strong training performance over five epochs, with steady improvements in accuracy and reductions in training loss. The base model achieved slightly higher final training accuracy (88.7%) compared to the dropout model (87.4%) and lower training loss (0.3027 vs. 0.3429), indicating that the base model learned the training data more aggressively. However, the dropout model demonstrated better generalization. Its validation accuracy increased more consistently from 81.3% to 82.6%, whereas the base model's validation accuracy peaked slightly earlier and ended at 82.3%. More notably, the dropout model maintained a relatively stable validation loss across epochs, while the base model's validation loss slightly increased toward the end (from 0.4912 to 0.5159), suggesting the beginning of overfitting.

7.1 Comparison Task Data

For Sequence Classification the *article_example.json* object encodes four abstracts ready to be classified. Each model was used to classify each article in the comparison dataset. The results were used to showcase the performance differences between the models.

https://pubmed.ncbi.nlm.nih.gov/22244707/ is the abstract selected to use for comparison. This abstract has 14 lines, the most of all 4.

Section Classification	Base Model	Dropout	Fine Tuned DistilBERT
BACKGROUND	Baclofen, a GABA(B) receptor agonist, represents a promising pharmacotherapy for alcohol dependence (AD). Previously, we performed a randomized clinical trial (RCT), which demonstrated the safety and efficacy of baclofen in patients affected by AD and cirrhosis. albumin, INR) in alcohol-dependent HCV-infected cirrhotic patients. Baclofen promotes alcohol abstinence in alcohol dependent cirrhotic patients with hepatitis C virus (HCV) infection.	Hepatitis C virus (HCV) and alcoholic liver disease (ALD), either alone or in combination, count for more than two thirds of all liver diseases in the Western world. Baclofen, a GABA(B) receptor agonist, represents a promising pharmacotherapy for alcohol dependence (AD). Previously, we performed a randomized clinical trial (RCT), which demonstrated the safety and efficacy of baclofen in patients affected by AD and cirrhosis. Among the 84 subjects randomized in the main trial, 24 alcohol-dependent cirrhotic patients had a HCV infection; 12 received baclofen 10mg t.i.d. albumin, INR) in alcohol-dependent HCV-infected cirrhotic patients. Baclofen promotes alcohol abstinence in alcohol dependent cirrhotic patients with hepatitis C virus (HCV) infection.	Hepatitis C virus (HCV) and alcoholic liver disease (ALD), either alone or in combination, count for more than two thirds of all liver diseases in the Western world. Baclofen, a GABA(B) receptor agonist, represents a promising pharmacotherapy for alcohol dependence (AD). Previously, we performed a randomized clinical trial (RCT), which demonstrated the safety and efficacy of baclofen in patients affected by AD and cirrhosis.
CONCLUSIONS	There is no safe level of drinking in HCV-infected patients and the most effective goal for these patients is total abstinence. Among the 84 subjects randomized in the main trial, 24 alcohol-dependent cirrhotic patients had a HCV infection; 12 received baclofen 10mg t.i.d. Baclofen may represent a clinically relevant alcohol pharmacotherapy for these patients..	There is no safe level of drinking in HCV-infected patients and the most effective goal for these patients is total abstinence. In conclusion, baclofen was safe and significantly more effective than placebo in promoting alcohol abstinence, and improving some Liver Function Tests (LFTs) (i.e. Baclofen may represent a clinically relevant alcohol pharmacotherapy for these patients..	There is no safe level of drinking in HCV-infected patients and the most effective goal for these patients is total abstinence. Baclofen may represent a clinically relevant alcohol pharmacotherapy for these patients.. Baclofen promotes alcohol abstinence in alcohol dependent cirrhotic patients with hepatitis C virus (HCV) infection.
METHODS	and 12 received placebo for 12-weeks.	Any patient with HCV infection was selected for this analysis. and 12 received	

		placebo for 12-weeks.	
OBJECTIVE	The goal of this post-hoc analysis was to explore baclofen's effect in a subgroup of alcohol-dependent HCV-infected cirrhotic patients.	The goal of this post-hoc analysis was to explore baclofen's effect in a subgroup of alcohol-dependent HCV-infected cirrhotic patients.	The goal of this post-hoc analysis was to explore baclofen's effect in a subgroup of alcohol-dependent HCV-infected cirrhotic patients.
RESULTS	Hepatitis C virus (HCV) and alcoholic liver disease (ALD), either alone or in combination, count for more than two thirds of all liver diseases in the Western world. Any patient with HCV infection was selected for this analysis. With respect to the placebo group (3/12, 25.0%), a significantly higher number of patients who achieved and maintained total alcohol abstinence was found in the baclofen group (10/12, 83.3%; p=0.0123). Furthermore, in the baclofen group, compared to placebo, there was a significantly higher increase in albumin values from baseline (p=0.0132) and a trend toward a significant reduction in INR levels from baseline (p=0.0716). In conclusion, baclofen was safe and significantly more effective than placebo in promoting alcohol abstinence, and improving some Liver Function Tests (LFTs) (i.e.	With respect to the placebo group (3/12, 25.0%), a significantly higher number of patients who achieved and maintained total alcohol abstinence was found in the baclofen group (10/12, 83.3%; p=0.0123). Furthermore, in the baclofen group, compared to placebo, there was a significantly higher increase in albumin values from baseline (p=0.0132) and a trend toward a significant reduction in INR levels from baseline (p=0.0716).	Any patient with HCV infection was selected for this analysis. Among the 84 subjects randomized in the main trial, 24 alcohol-dependent cirrhotic patients had a HCV infection; 12 received baclofen 10mg t.i.d. and 12 received placebo for 12-weeks. With respect to the placebo group (3/12, 25.0%), a significantly higher number of patients who achieved and maintained total alcohol abstinence was found in the baclofen group (10/12, 83.3%; p=0.0123). Furthermore, in the baclofen group, compared to placebo, there was a significantly higher increase in albumin values from baseline (p=0.0132) and a trend toward a significant reduction in INR levels from baseline (p=0.0716). In conclusion, baclofen was safe and significantly more effective than placebo in promoting alcohol abstinence, and improving some Liver Function Tests (LFTs) (i.e. albumin, INR) in alcohol-dependent HCV-infected cirrhotic patients.

7.2 Comparison Observations

We can see that the models do perform differently. While some errors are noticed (sentence fragmentation and omission) the models have some interesting differences in classification. All models classified the same line to **OBJECTIVE**. Overlap was also seen in **CONCLUSIONS** and **RESULTS**.

Of note the line: “Hepatitis C virus (HCV) and alcoholic liver..” was classified as belonging to the **RESULTS** by the *Base Model*, whereas the *Dropout Model* and *Fine Tuned DistilBERT Model* both classified it as **BACKGROUND**.

The models achieved accuracies of *BaseModel*: 82%, *Dropout*: 83%, and *Fine Tuned DistilBERT*: 86%. These slight differences in accuracy (+/- 4 pts) do appear to have an effect on performance.

8. Conclusions

We have showcased the ability of DistilBERT basaed models to classify sentences based on their context in the document. We also showcased DistilBERT based models’ ability to predict masked or missing words in a document also based on the context of the masked or missing word.

Our own DistilBERT models, both with and without Dropout, were able to accomplish similar results as the fine tuned Hugging Face DistilBERT model. Our models were trained on a much smaller dataset, and have a much simpler architecture. This demonstrates the power of domain specific training, and the fine tuning capabilities of the DistilBERT model.

9. Recommendations

Some refining of the training and application of the models would improve performance. Fragmentation errors occurred due to a lack of sepcific new line terminators in the *article_example.json* object. Adding this specificity would bring the tast data more in line with the training data which did have unique newline terminators.

References

Franck Dernoncourt, Ji Young Lee (2017) PubMed 200k RCT: a Dataset for Sequential Sentence Classification in Medical Abstracts. https://arxiv.org/abs/1710.06071

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/abs/1810.04805

Victor Sanh, Lysandre Debut, Julien Chaumond, Thomas Wolf (2020) DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. https://arxiv.org/abs/1910.01108