Handwritten Text Recognition

Heet Dave, 241110028, heetmd24@iitk.ac.in
Junth Basnet, 241110090, junthb24@iitk.ac.in
Pavan Kumar Moka, 241110043, pavankumar24@iik.ac.in
Suyamoon Pathak, 241110091, suyamoonp24@iitk.ac.in
Ujjaval Patel, 241110047, pujjaval24@iitk.ac.in

CS685: Data Mining

Abstract

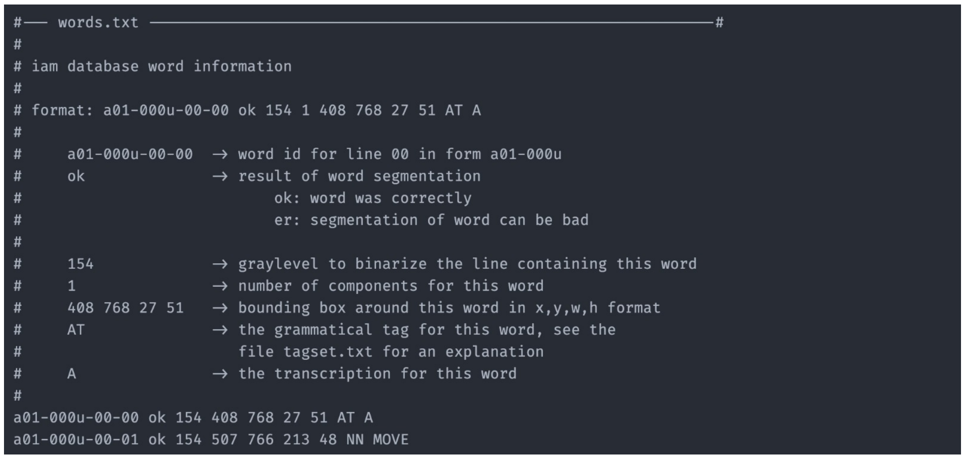
Handwritten text recognition is important for converting handwritten notes, forms, and historical documents into digital text. This project uses the IAM Words Dataset, a collection of handwritten words, to train a machine learning model called Convolutional Recurrent Neural Network (CRNN). The model processes images of handwritten words and predicts the text. We achieved 87.32% accuracy for recognizing individual characters and 64.92% accuracy for recognizing entire words. The project includes steps like preparing the data, preprocessing images, and using data augmentation to improve performance. While the character recognition is good, word recognition accuracy needs improvement because even small mistakes count as wrong predictions. In the future, we will test advanced techniques, like better data transformation and new model designs, to make the system more accurate and useful for real-world applications. This project shows how computers can learn to understand handwriting and help with digitizing handwritten documents.

1 Motivation of the Problem

Handwritten text is still used in many fields, such as historical records, healthcare, education, and banking. Manually converting handwritten documents to digital text is slow, expensive, and prone to errors, especially for large datasets. Automating this process with a Handwritten Text Recognition (HTR) system can save time, reduce mistakes, and make handwritten content easy to search and analyze. The main challenge is recognizing diverse handwriting styles and improving accuracy for different use cases. Solving this problem can help in digitizing records, preserving historical documents, and automating tasks like data entry and classification.

2 Data Used

For this project, we utilized the IAM Handwriting Word Database, a widely recognized benchmark dataset for handwritten text recognition. The dataset provides high-quality handwritten samples collected from diverse writers, ensuring variability in handwriting styles, penmanship, and spacing. Specifically, the IAM Word Dataset was employed, which consists of individual word images cropped from handwritten text lines.



```
#--- words.txt ---------------------------------------------------------#
#
# iam database word information
#
# format: a01-000u-00-00 ok 154 1 408 768 27 51 AT A
#
#     a01-000u-00-00  -> word id for line 00 in form a01-000u
#     ok              -> result of word segmentation
#                            ok: word was correctly
#                            er: segmentation of word can be bad
#
#     154             -> graylevel to binarize the line containing this word
#     1               -> number of components for this word
#     408 768 27 51   -> bounding box around this word in x,y,w,h format
#     AT              -> the grammatical tag for this word, see the
#                        file tagset.txt for an explanation
#     A               -> the transcription for this word
#
a01-000u-00-00 ok 154 408 768 27 51 AT A
a01-000u-00-01 ok 154 507 766 213 48 NN MOVE
```

Figure 1: IAM Words Dataset Metadata

3 Methodology

The methodology for handwritten text detection involves data preparation, preprocessing, model training, and evaluation. The detailed steps are outlined as follows:

3.1 Dataset Preparation

The IAM Handwriting Dataset is utilized for this project. The dataset is organized by:

1. Downloading and extracting the dataset using a compressed archive containing images and annotations.
2. Splitting the dataset into training (90% - 86810 images), validation (5%- 4823 images), and testing (5%- 4823 images) sets.

3. Filtering out erroneous data entries and constructing file paths to corresponding images using metadata in the `words.txt` file.

3.2 Image Preprocessing

The preprocessing pipeline includes:

- Grayscale images are loaded from a given directory using the `get_img_files` function, which searches for supported image formats such as `.png`, `.jpg`, `.jpeg`, and `.bmp`. The image is then resized using the `prepare_img` function, where the height is adjusted to a fixed value (in this case, 1000 pixels), preserving the aspect ratio.

- The function `detect` uses a combination of morphological operations, including Gaussian blurring and thresholding, to detect potential regions containing text. Adjustable parameters such as `kernel_size`, `sigma`, `theta`, and `min_area` allow for controlling the sensitivity of the detection.

- Once the text regions are detected, the words are segmented using bounding boxes. These bounding boxes are represented by their coordinates (`x`, `y`, `w`, `h`), which denote the position and dimensions of each word within the image. Each segmented word is saved as an individual image in the `test_images` directory. The function `sort_multiline` is used to ensure that words are sorted correctly in a multiline context, preserving the reading order.

- The segmented words are saved as images named according to their line and word index (e.g., `line0word1.jpg`). These image names are also saved to a text file for reference. This allows for easier retrieval and further processing in the next stage.

3.3 Data Augmentation

To improve model robustness, we have used augmentation techniques- flipping and resizing.

3.4 Training and Validation Process

The training process employs a TensorFlow-based Convolutional Recurrent Neural Network (CRNN) architecture:

- The dataset is split into three parts: training (90% i.e. 86810 images), validation (5% i.e. 4823 images), and testing (5% i.e. 4823 images). This split ensures that the model is trained on a large portion of the data while maintaining separate sets for hyperparameter tuning and evaluation.

- The function `get_image_paths_and_labels` extracts the paths to the segmented images and their corresponding labels from the `words.txt` file. The labels are cleaned to extract only the target word, which is then used for training.
- Each image is preprocessed before being passed to the model. The function `distortion_free_resize` resizes the image while preserving the aspect ratio, padding as necessary to ensure uniform dimensions (128x32 pixels). The pixel values are normalized to lie in the range [0, 1].
- Labels are converted into numerical sequences using the `StringLookup` layer, which maps each character to a unique integer. Labels are padded to a maximum length of 21 characters using a padding token of value 99.

4 Model

The model architecture for handwritten text recognition consists of the following components:

4.1 Feature Extraction

The input grayscale images are passed through a Convolutional Neural Network (CNN) backbone, which extracts spatial features. The CNN layers include:

- 2 convolutional layers, each with 64 filters of size 3 x 3 and ReLU activations.
- 2 Max-pooling layers of size 2 x 2 and stride 2, each after convolution layers to reduce spatial dimensions while retaining critical features. Hence downsampled feature maps are 4x smaller the number of filters in the last layer is 64.

4.2 Sequence Modeling

The extracted spatial features are reshaped into sequences, sent to a dense layer with dropout of 0.2, and then fed into 2 Bidirectional Long Short-Term Memory (BiLSTM) layers. The first BiLSTM layer has 128 units, and the second one has 64 units respectively. Each layer has a dropout of 0.25. These layers capture temporal dependencies in handwriting styles and ensure bidirectional context awareness.

4.3 Output Layer

The model uses a dense layer with softmax activation to predict probabilities for each character in the vocabulary. The sequence of probabilities is decoded using Connectionist Temporal Classification (CTC) decoding for aligning predictions with ground truth.

4.4 Evaluation Metrics

The Edit Distance metric is used to measure the average number of insertions, deletions, or substitutions required to convert predictions into ground truth labels. This metric provides insight into the accuracy of character sequence predictions. At the end of 50 epochs, the mean edit distance was 7.28.

4.5 Optimizer

The model is trained for 50 epochs using the Adam optimizer with a learning rate scheduler for dynamic adjustment. Batch size was selected to be 64.

5 Results

We got an overall 64.92% accuracy on the words dataset, with precision 0.72, recall 0.65, and F1 score of 0.63. But, it is not a fair measure because, even 1 letter difference would make the prediction a misclassification. So, we tried to do character level accuracy, and we found it out to be 87.32%.

6 Conclusions and Future Directions

6.1 Conclusions

This project successfully implemented a Convolutional Recurrent Neural Network (CRNN) for handwritten word recognition using the IAM Words Dataset. Key achievements include:

- Achieved character-level accuracy of 87.32%, demonstrating robustness in recognizing characters even with handwriting variability.
- Developed an effective preprocessing pipeline that combines morphological operations and segmentation for accurate text detection and preparation.
- Leveraged augmentation techniques to improve model generalizability across diverse handwriting styles.

Despite these accomplishments, the word-level accuracy of 64.92% suggests room for improvement, particularly in handling minor variations that result in misclassifications.

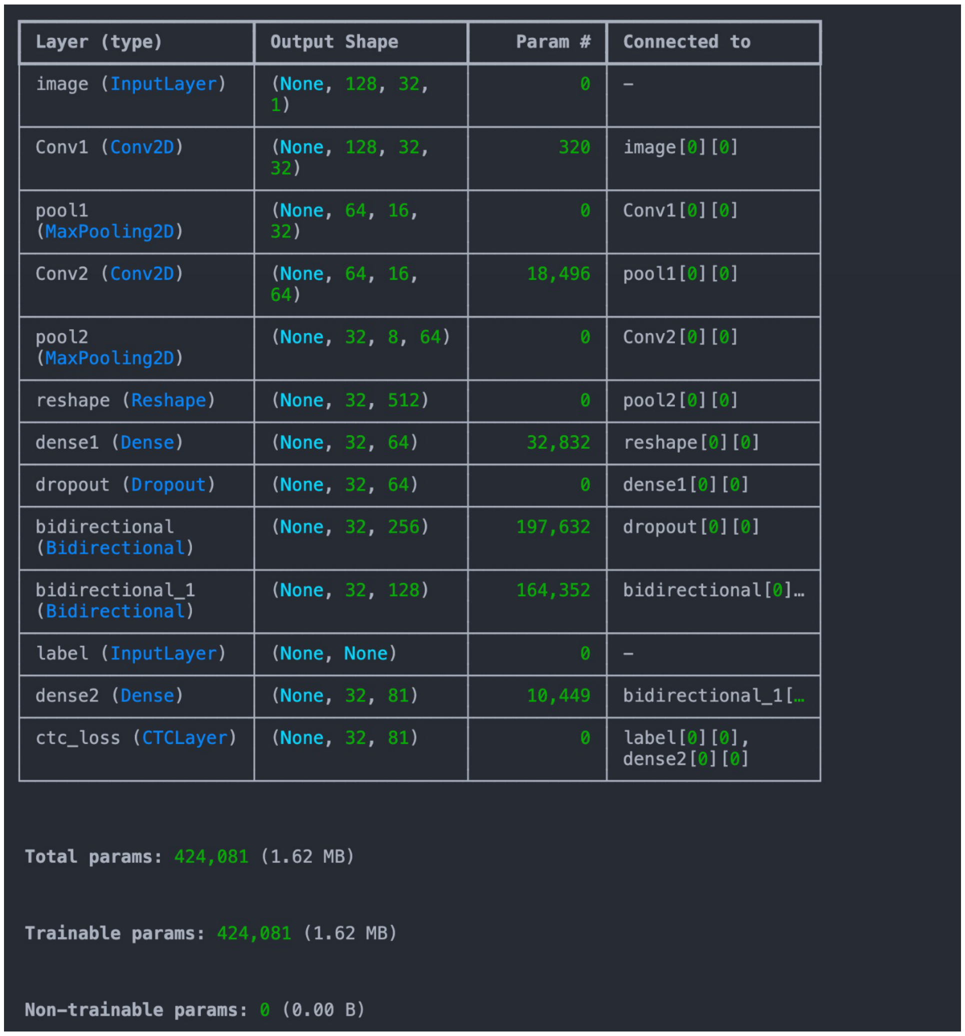
6.2 Future Directions

- Explore advanced augmentation techniques, such as elastic transformations, to simulate more diverse handwriting styles.
- Incorporate a larger dataset or synthetic data generation to enhance the model's exposure to a broader variety of handwriting patterns.
- Investigate transformer-based architectures for better sequence modeling and improved context understanding.
- Integrate transfer learning from pre-trained models to expedite training and improve accuracy.

7 Team Contributions

The project was a collaborative effort, with each team member contributing equally across various phases:

1. **Data Acquisition and Preprocessing:** Pavan Kumar and Ujjwal Patel collaborated to prepare the IAM dataset, and design preprocessing pipelines.
2. **Word Segmentation:** Heet Dave worked on the implementation of word segmentation techniques.
3. **Model Development:** Junth Basnet and Suyamoon Pathak participated in designing and training the CRNN architecture, ensuring optimal performance through hyperparameter tuning.
4. **Evaluation:** The evaluation metrics and analysis of results were carried out collectively, including character-level and word-level accuracy assessments.
5. **Documentation and Presentation:** All team members contributed equally to documenting the methodology, results, and findings, and preparing the final report.



```
┏━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)        ┃ Output Shape       ┃     Param # ┃ Connected to       ┃
┡━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━┩
│ image (InputLayer)  │ (None, 128, 32,    │           0 │ -                  │
│                     │ 1)                 │             │                    │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ Conv1 (Conv2D)      │ (None, 128, 32,    │         320 │ image[0][0]        │
│                     │ 32)                │             │                    │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ pool1               │ (None, 64, 16,     │           0 │ Conv1[0][0]        │
│ (MaxPooling2D)      │ 32)                │             │                    │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ Conv2 (Conv2D)      │ (None, 64, 16,     │      18,496 │ pool1[0][0]        │
│                     │ 64)                │             │                    │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ pool2               │ (None, 32, 8, 64)  │           0 │ Conv2[0][0]        │
│ (MaxPooling2D)      │                    │             │                    │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ reshape (Reshape)   │ (None, 32, 512)    │           0 │ pool2[0][0]        │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ dense1 (Dense)      │ (None, 32, 64)     │      32,832 │ reshape[0][0]      │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ dropout (Dropout)   │ (None, 32, 64)     │           0 │ dense1[0][0]       │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ bidirectional       │ (None, 32, 256)    │     197,632 │ dropout[0][0]      │
│ (Bidirectional)     │                    │             │                    │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ bidirectional_1     │ (None, 32, 128)    │     164,352 │ bidirectional[0]…  │
│ (Bidirectional)     │                    │             │                    │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ label (InputLayer)  │ (None, None)       │           0 │ -                  │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ dense2 (Dense)      │ (None, 32, 81)     │      10,449 │ bidirectional_1[…  │
├─────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ ctc_loss (CTCLayer) │ (None, 32, 81)     │           0 │ label[0][0],       │
│                     │                    │             │ dense2[0][0]       │
└─────────────────────┴────────────────────┴─────────────┴────────────────────┘

 Total params: 424,081 (1.62 MB)

 Trainable params: 424,081 (1.62 MB)

 Non-trainable params: 0 (0.00 B)
```

Figure 2: Model Summary

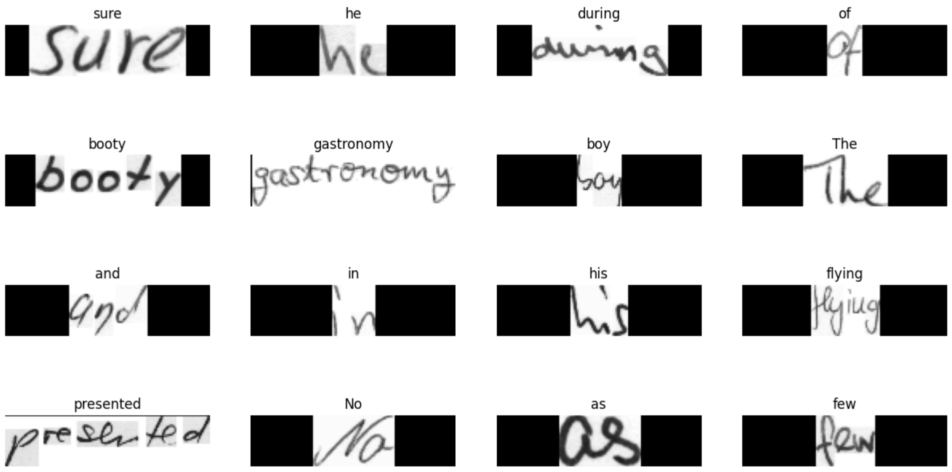


Figure 3: Some test predictions

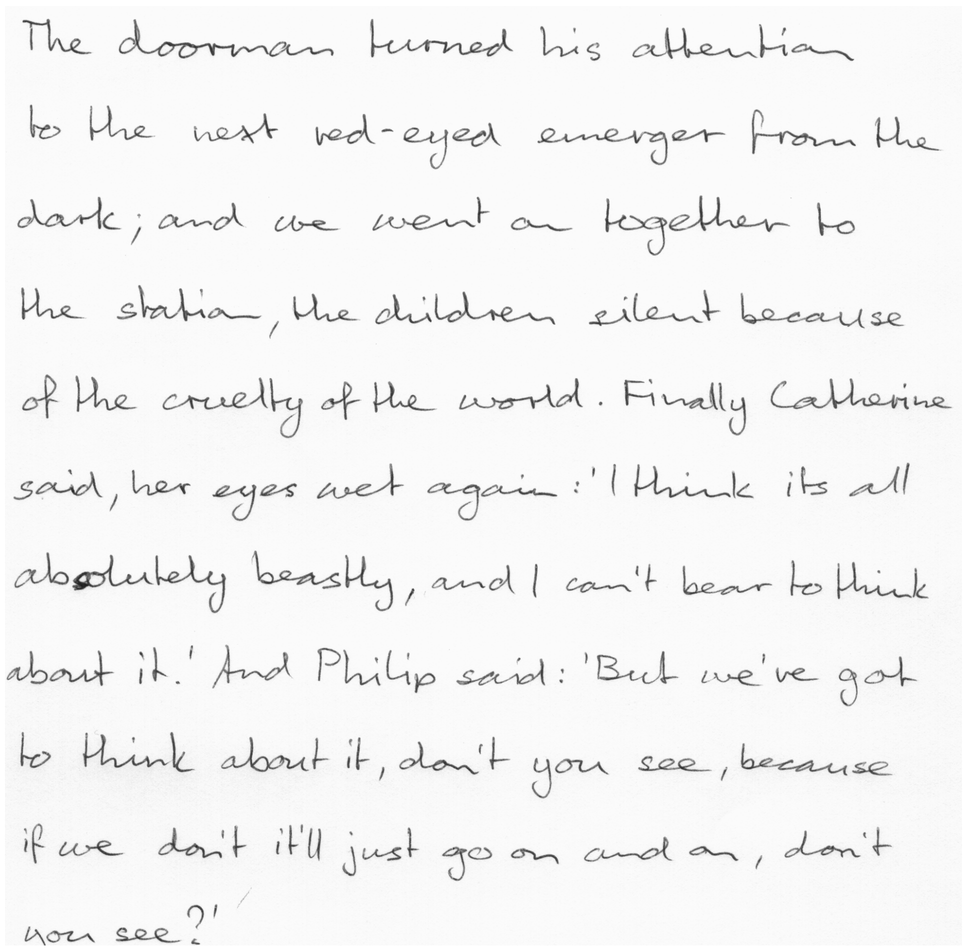

The doorman turned his attention
to the next red-eyed emerger from the
dark; and we went on together to
the station, the children silent because
of the cruelty of the world. Finally Catherine
said, her eyes wet again: 'I think its all
absolutely beastly, and I can't bear to think
about it.' And Philip said: 'But we've got
to think about it, don't you see, because
if we don't it'll just go on and on, don't
you see?'

Figure 4: Input Image for inference

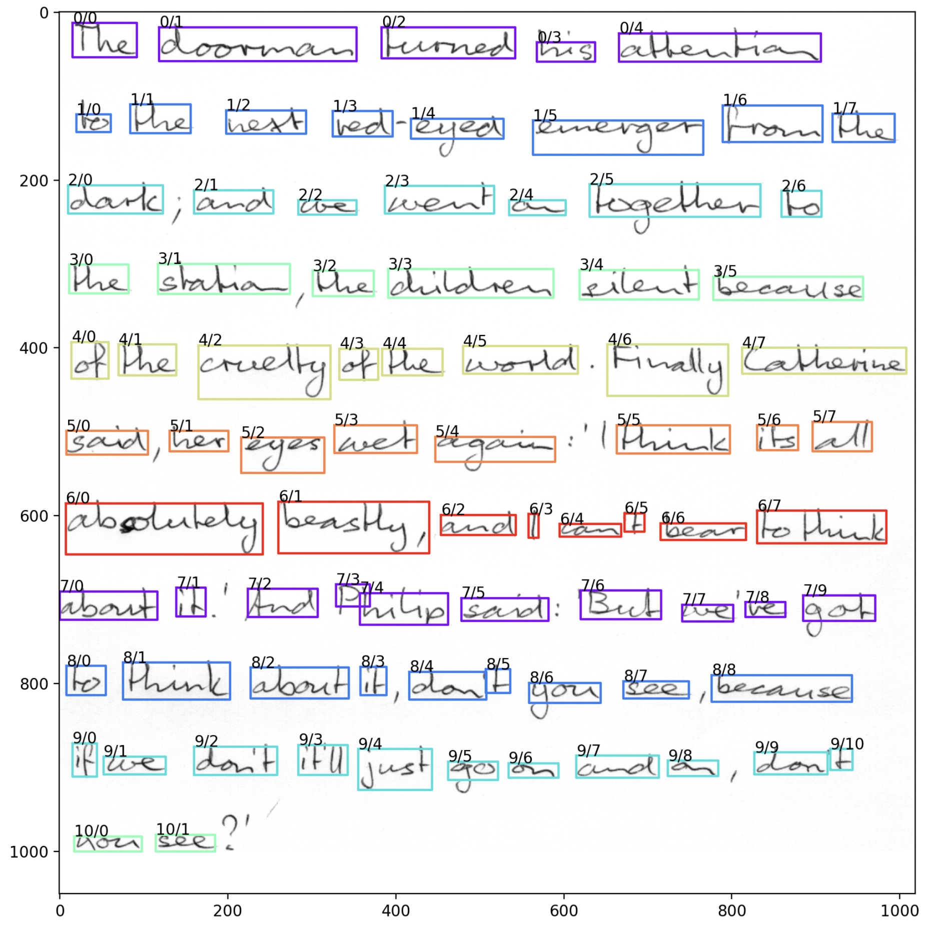
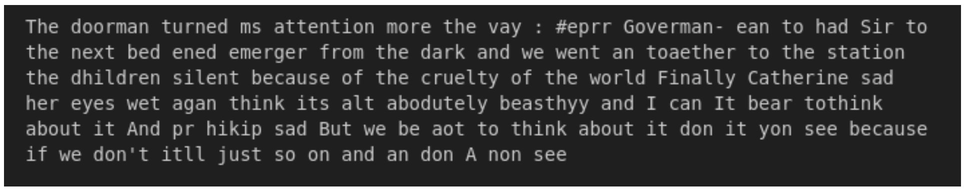


Figure 5: Segmented words



```
The doorman turned ms attention more the vay : #eprr Goverman- ean to had Sir to
the next bed ened emerger from the dark and we went an toaether to the station
the dhildren silent because of the cruelty of the world Finally Catherine sad
her eyes wet agan think its alt abodutely beasthyy and I can It bear tothink
about it And pr hikip sad But we be aot to think about it don it yon see because
if we don't itll just so on and an don A non see
```

Figure 6: Output for inference