Innov8: The Eightfold Challenge

Data Collection and Preparation

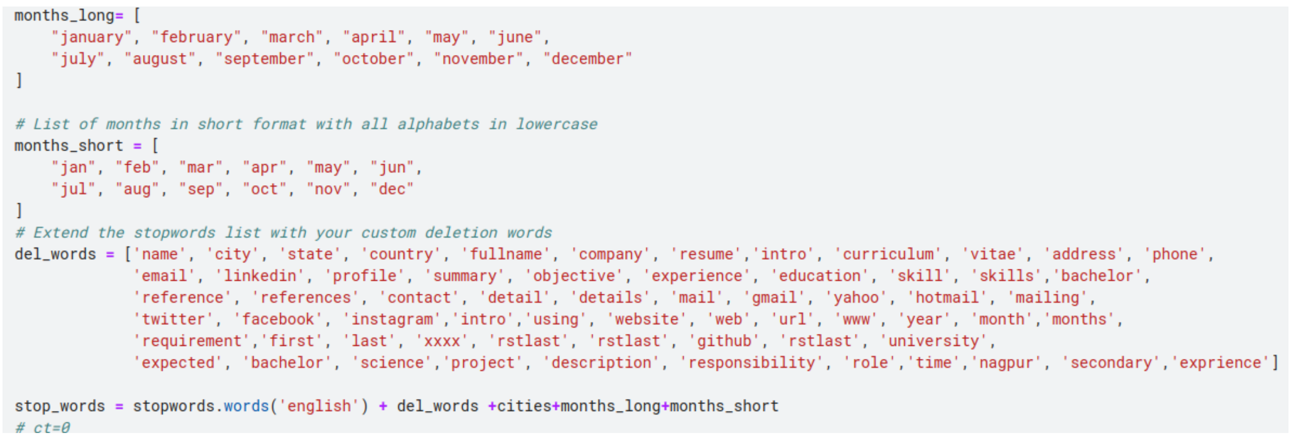
Despite initially receiving a dataset, I undertook the initiative to further augment this collection by curating additional resumes from various sources. This effort was aimed at enriching the dataset's diversity, ensuring a broader representation of industries, job roles, and candidate profiles.
https://github.com/florex/resume_corpus
https://www.kaggle.com/datasets/oo7kartik/resume-text-batch
https://www.kaggle.com/datasets/chingkuangkam/resume-text-classification-dataset

Geographical Data Collection

Expanded the dataset's contextual understanding by incorporating a comprehensive list of cities from India and the USA. This step aimed to enhance the model's capability to recognize and differentiate location-based data within resumes, further refining its categorization accuracy.

Exclusion List Creation

Developed a meticulously curated list of words to be ignored during the resume analysis process. This exclusion list helped in eliminating noise and focusing the model's attention on relevant information that directly contributes to the accuracy of resume categorization and relevance matching.



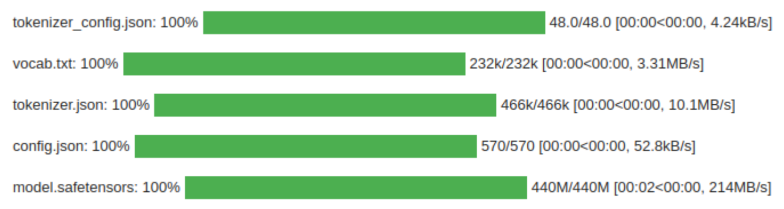
```
months_long= [
    "january", "february", "march", "april", "may", "june",
    "july", "august", "september", "october", "november", "december"
]

# List of months in short format with all alphabets in lowercase
months_short = [
    "jan", "feb", "mar", "apr", "may", "jun",
    "jul", "aug", "sep", "oct", "nov", "dec"
]
# Extend the stopwords list with your custom deletion words
del_words = ['name', 'city', 'state', 'country', 'fullname', 'company', 'resume','intro', 'curriculum', 'vitae', 'address', 'phone',
             'email', 'linkedin', 'profile', 'summary', 'objective', 'experience', 'education', 'skill', 'skills','bachelor',
             'reference', 'references', 'contact', 'detail', 'details', 'mail', 'gmail', 'yahoo', 'hotmail', 'mailing',
             'twitter', 'facebook', 'instagram','intro','using', 'website', 'web', 'url', 'www', 'year', 'month','months',
             'requirement','first', 'last', 'xxxx', 'rstlast', 'rstlast', 'github', 'rstlast', 'university',
             'expected', 'bachelor', 'science','project', 'description', 'responsibility', 'role','time','nagpur', 'secondary','exprience']

stop_words = stopwords.words('english') + del_words +cities+months_long+months_short
# ct=0
```

Preprocessing Techniques Applied: Implemented a comprehensive text preprocessing pipeline to cleanse and standardize resume data. This included the removal of URLs, social media mentions, hashtags, RTs, and cc tags to ensure the text was free from irrelevant or noisy data. Further, I eliminated all non-ASCII characters, punctuations, and numbers to focus on meaningful textual content.

Normalization and Noise Reduction: Applied lowercasing and whitespace normalization across the dataset to maintain consistency. Additionally, a custom filter was employed to exclude stop words and short tokens, retaining only significant words that contribute to understanding resume content effectively.

Model Selection: Opted for the state-of-the-art BERT (Bidirectional Encoder Representations from Transformers) model, specifically the bert-base-uncased variant, known for its effectiveness in understanding the context and nuances in text data.

Library Utilization: Leveraged the transformers library by Hugging Face to access pre-trained models and tokenizers, enabling the application of transfer learning techniques to adapt the model to the specialized task of resume categorization and matching.



```
tokenizer_config.json: 100% 48.0/48.0 [00:00<00:00, 4.24kB/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 3.31MB/s]
tokenizer.json: 100% 466k/466k [00:00<00:00, 10.1MB/s]
config.json: 100% 570/570 [00:00<00:00, 52.8kB/s]
model.safetensors: 100% 440M/440M [00:02<00:00, 214MB/s]
```

Custom Dataset Class: Developed a CustomDataset class inheriting from PyTorch's Dataset module. This class is tailored to tokenize and prepare the input data specifically for the BERT model, utilizing efficient batch processing to handle large-scale text data.

Tokenization and Encoding: Implemented the tokenizer within the initialization of the dataset, configuring it to handle text tokenization with truncation and padding to the maximum length, ensuring uniform input size for model training.

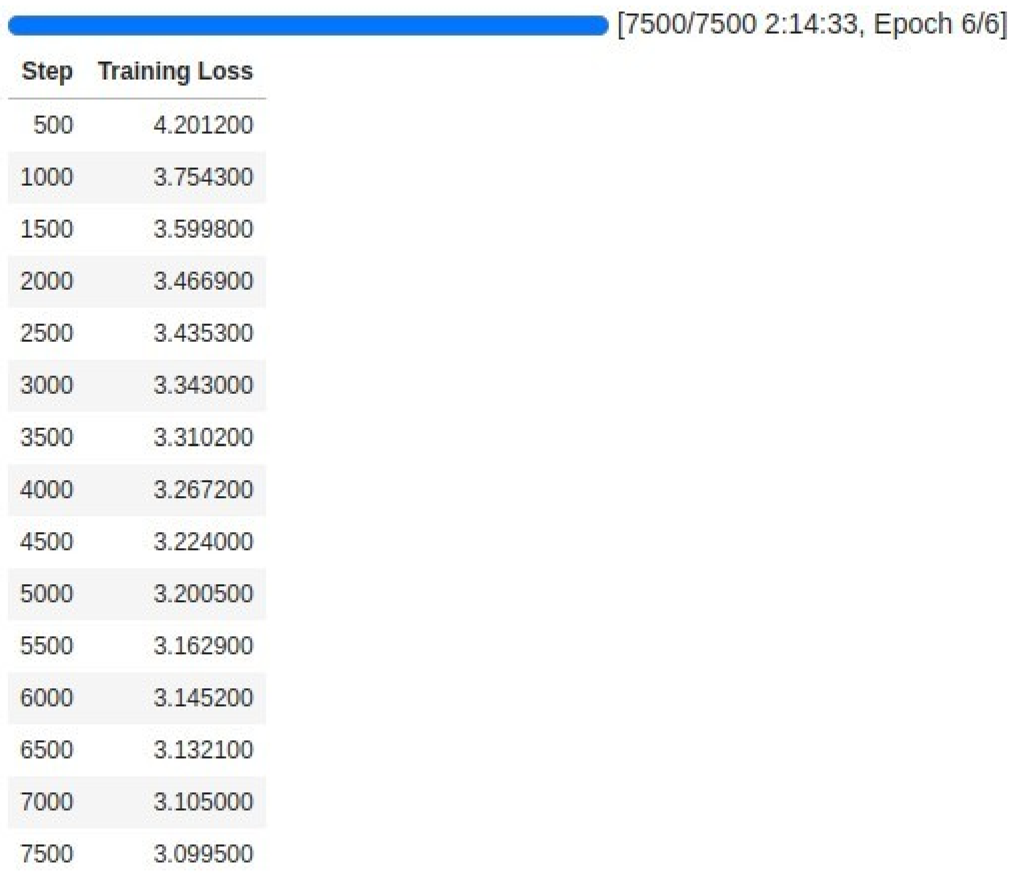
Dynamic Masking with Data Collator:
Employed DataCollatorForLanguageModeling with the tokenizer, enabling dynamic token masking with a 15% probability. This setup is essential for training the model using the Masked Language Modeling (MLM) objective, which is fundamental to BERT's training methodology.

MLM Probability Setting: The mlm_probability parameter was set to 0.15, meaning that there was a 15% chance of any given token being masked out in the training data. This probability rate was chosen in line with the original BERT paper's recommendations and is a standard practice in training transformers for MLM tasks. It ensures that a sufficient number of tokens are masked to challenge the model to learn contextual relationships between words without overwhelming it with too many masks.

MLM Training Objective: The data collator's primary role was to randomly mask tokens in the input data, which is essential for the MLM training objective. Under this paradigm, the model learns to predict the original identity of the masked tokens based on the surrounding context. This process not only helps in learning deep contextual representations but also allows the model to focus on understanding the entire context of the input sequence.

Model Fine-Tuning and Persistence:

With the custom dataset and data collator configured, I proceeded to define the training parameters and initiate the fine-tuning process for the BERT model.



[7500/7500 2:14:33, Epoch 6/6]

Step	Training Loss
500	4.201200
1000	3.754300
1500	3.599800
2000	3.466900
2500	3.435300
3000	3.343000
3500	3.310200
4000	3.267200
4500	3.224000
5000	3.200500
5500	3.162900
6000	3.145200
6500	3.132100
7000	3.105000
7500	3.099500

Model Prediction:

For each line of text, I performed a series of cleaning operations including the removal of any '.js' substrings, an artifact commonly observed in technical resumes. Following this, non-ASCII characters were filtered out to standardize the text for processing.

The cleaned lines were then split into individual sentences using period separators. Each sentence underwent further preprocessing, where stopwords and words with insufficient length were removed to focus on the significant content. These preprocessed sentences were then transformed into vector embeddings using the trained BERT model.

To ascertain the relevance of each sentence to the job category in question, I employed cosine similarity measures. This method allowed me to compare the sentence embeddings with a vector representative of the resume category. The similarity scores for each sentence were stored in a dictionary, associating text chunks with their respective relevancies.

I choose the top 20 best chunks per pages to highlight in the pdf.