```
!ls
```

```
sample_data
```

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Install required libraries
!pip install transformers datasets accelerate gensim gradio kaggle -q

print("✅ Setup complete!")
```

```
Mounted at /content/drive
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 27.9/27.9 MB 93.2 MB/s eta 0:00:00
✅ Setup complete!
```

```
from google.colab import files
import os

# Upload kaggle.json (you'll be prompted)
print("📤 Please upload your kaggle.json file:")
uploaded = files.upload()

# Setup Kaggle credentials
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json

# Create directories in Google Drive
!mkdir -p /content/drive/MyDrive/NLP_HR/data
!mkdir -p /content/drive/MyDrive/NLP_HR/models

# Download dataset
print("\n📥 Downloading LinkedIn job postings dataset...")
!kaggle datasets download -d arshkon/linkedin-job-postings -p /content/drive/MyDrive/NLP_HR/data/
!unzip /content/drive/MyDrive/NLP_HR/data/linkedin-job-postings.zip -d /content/drive/MyDrive/NLP_HR/data/
!rm /content/drive/MyDrive/NLP_HR/data/linkedin-job-postings.zip

print("\n✅ Dataset downloaded to Google Drive!")
!ls /content/drive/MyDrive/NLP_HR/data/
```

```
📤 Please upload your kaggle.json file:
Choose files kaggle.json
kaggle.json(application/json) - 70 bytes, last modified: 25/10/2025 - 100% done
Saving kaggle.json to kaggle.json

📥 Downloading LinkedIn job postings dataset...
Dataset URL: https://www.kaggle.com/datasets/arshkon/linkedin-job-postings
License(s): CC-BY-SA-4.0
Downloading linkedin-job-postings.zip to /content/drive/MyDrive/NLP_HR/data
 79% 125M/159M [00:00<00:00, 393MB/s]
100% 159M/159M [00:00<00:00, 395MB/s]
Archive:  /content/drive/MyDrive/NLP_HR/data/linkedin-job-postings.zip
  inflating: /content/drive/MyDrive/NLP_HR/data/companies/companies.csv
  inflating: /content/drive/MyDrive/NLP_HR/data/companies/company_industries.csv
  inflating: /content/drive/MyDrive/NLP_HR/data/companies/company_specialities.csv
  inflating: /content/drive/MyDrive/NLP_HR/data/companies/employee_counts.csv
  inflating: /content/drive/MyDrive/NLP_HR/data/jobs/benefits.csv
  inflating: /content/drive/MyDrive/NLP_HR/data/jobs/job_industries.csv
  inflating: /content/drive/MyDrive/NLP_HR/data/jobs/job_skills.csv
  inflating: /content/drive/MyDrive/NLP_HR/data/jobs/salaries.csv
  inflating: /content/drive/MyDrive/NLP_HR/data/mappings/industries.csv
  inflating: /content/drive/MyDrive/NLP_HR/data/mappings/skills.csv
  inflating: /content/drive/MyDrive/NLP_HR/data/postings.csv

✅ Dataset downloaded to Google Drive!
companies  jobs  mappings  postings.csv
```

```
import pandas as pd
import re

# Load dataset
print("📊 Loading dataset...")
DATA_PATH = '/content/drive/MyDrive/NLP_HR/data/postings.csv'
df = pd.read_csv(DATA_PATH)

print(f"Original shape: {df.shape}")
print(f"Columns: {df.columns.tolist()}")
```

```
# Find description column
desc_col = None
for col in ['description', 'job_description', 'text']:
    if col in df.columns:
        desc_col = col
        break

if desc_col is None:
    print("⚠️ No description column found. Using first text column.")
    desc_col = df.columns[0]

print(f"\nUsing column: '{desc_col}'")

# Clean text function
def clean_text(text):
    if pd.isna(text):
        return ""
    text = str(text).lower()
    text = re.sub(r'\s+', ' ', text)  # Remove extra spaces
    text = re.sub(r'http\S+', '', text)  # Remove URLs
    text = re.sub(r'[^\w\s]', '', text)  # Remove special chars
    return text.strip()

# Apply cleaning
print("\n🧹 Cleaning text...")
df['text_clean'] = df[desc_col].apply(clean_text)

# Remove short/empty descriptions
df = df[df['text_clean'].str.len() > 50]

print(f"\n✅ Cleaned dataset shape: {df.shape}")

# Save cleaned data
df.to_csv('/content/drive/MyDrive/NLP_HR/data/postings_cleaned.csv', index=False)
print("✅ Cleaned data saved to Drive!")
```

```
📊 Loading dataset...
Original shape: (123849, 31)
Columns: ['job_id', 'company_name', 'title', 'description', 'max_salary', 'pay_period', 'location', 'company_id', 'vi

Using column: 'description'

🧹 Cleaning text...

✅ Cleaned dataset shape: (123771, 32)
✅ Cleaned data saved to Drive!
```

```
import pandas as pd

# Load cleaned data
df = pd.read_csv('/content/drive/MyDrive/NLP_HR/data/postings_cleaned.csv')

# Comprehensive bias detection function
def detect_bias(text):
    """Detect bias using keyword scoring"""
    text = str(text).lower()
    score = 0

    # Age bias keywords (weight: 2)
    age_keywords = ['young', 'youthful', 'energetic', 'dynamic', 'recent graduate',
                    'digital native', 'fresh', 'new talent', 'recent grad']
    for word in age_keywords:
        if word in text:
            score += 2

    # Gender bias - masculine coded (weight: 2)
    masculine_keywords = ['aggressive', 'dominant', 'competitive', 'decisive',
                          'ninja', 'rockstar', 'superhero', 'guru', 'hustler']
    for word in masculine_keywords:
        if word in text:
            score += 2

    # Gender pronouns (weight: 3)
    if ' he ' in text or ' his ' in text or ' him ' in text:
        score += 3
    if ' she ' in text or ' her ' in text:
        score += 3

    # Gender-specific terms (weight: 3)
    gender_terms = ['chairman', 'chairwoman', 'salesman', 'saleswoman',
```

```
                    'guys', 'manpower']
    for word in gender_terms:
        if word in text:
            score += 3

    # Cultural bias (weight: 2)
    cultural_keywords = ['native speaker', 'cultural fit', 'traditional']
    for word in cultural_keywords:
        if word in text:
            score += 2

    # Return 1 if biased (score >= 2), else 0
    return 1 if score >= 2 else 0

# Apply bias detection
print("🔍 Detecting bias in job descriptions...")
df['bias_label'] = df['text_clean'].apply(detect_bias)

# Check distribution
print("\n📊 Bias Label Distribution:")
print(df['bias_label'].value_counts())
print(f"\nPercentage biased: {(df['bias_label'].sum() / len(df) * 100):.2f}%")

# Save labeled data
df.to_csv('/content/drive/MyDrive/NLP_HR/data/postings_labeled.csv', index=False)
print("\n✅ Labeled data saved to Drive!")
```

```
🔍 Detecting bias in job descriptions...

📊 Bias Label Distribution:
bias_label
0    71460
1    52311
Name: count, dtype: int64

Percentage biased: 42.26%

✅ Labeled data saved to Drive!
```

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Load labeled data
df = pd.read_csv('/content/drive/MyDrive/NLP_HR/data/postings_labeled.csv')

# Separate classes
biased = df[df['bias_label'] == 1]
unbiased = df[df['bias_label'] == 0]

print(f"Biased samples: {len(biased)}")
print(f"Unbiased samples: {len(unbiased)}")

# Oversample minority class to balance
if len(biased) > 0 and len(biased) < len(unbiased):
    print("\n⚖️ Balancing dataset by oversampling biased class...")
    target_size = min(len(unbiased), len(biased) * 3)  # 1:3 ratio
    biased_oversampled = biased.sample(n=target_size, replace=True, random_state=42)
    df_balanced = pd.concat([biased_oversampled, unbiased]).sample(frac=1, random_state=42)
else:
    df_balanced = df

print(f"\n✅ Balanced dataset shape: {df_balanced.shape}")
print(f"Distribution:\n{df_balanced['bias_label'].value_counts()}")

# Split into train/test
train_df, test_df = train_test_split(
    df_balanced[['text_clean', 'bias_label']],
    test_size=0.2,
    random_state=42,
    stratify=df_balanced['bias_label']
)

print(f"\n📚 Train size: {len(train_df)}")
print(f"📚 Test size: {len(test_df)}")

# Save splits
train_df.to_csv('/content/drive/MyDrive/NLP_HR/data/train.csv', index=False)
test_df.to_csv('/content/drive/MyDrive/NLP_HR/data/test.csv', index=False)
print("\n✅ Train/test splits saved to Drive!")
```

```
Biased samples: 52311
Unbiased samples: 71460

⚖️ Balancing dataset by oversampling biased class...

✅ Balanced dataset shape: (142920, 33)
Distribution:
bias_label
0    71460
1    71460
Name: count, dtype: int64

📚 Train size: 114336
📚 Test size: 28584

✅ Train/test splits saved to Drive!
```

```
import os
import torch
import numpy as np
from transformers import (
    AlbertTokenizer,
    AlbertForSequenceClassification,
    Trainer,
    TrainingArguments,
    DataCollatorWithPadding
)
from datasets import Dataset
from sklearn.utils.class_weight import compute_class_weight
import pandas as pd

# Disable WandB
os.environ["WANDB_DISABLED"] = "true"

# Load data
train_df = pd.read_csv('/content/drive/MyDrive/NLP_HR/data/train.csv')
test_df = pd.read_csv('/content/drive/MyDrive/NLP_HR/data/test.csv')

print("🤖 Loading ALBERT model...")
tokenizer = AlbertTokenizer.from_pretrained('albert-base-v2')
model = AlbertForSequenceClassification.from_pretrained('albert-base-v2', num_labels=2)

# Convert to HuggingFace datasets
train_dataset = Dataset.from_pandas(train_df)
test_dataset = Dataset.from_pandas(test_df)

# Tokenization
def tokenize_function(examples):
    return tokenizer(examples['text_clean'], padding='max_length', truncation=True, max_length=128)

print("🔤 Tokenizing datasets...")
train_dataset = train_dataset.map(tokenize_function, batched=True)
test_dataset = test_dataset.map(tokenize_function, batched=True)

# Format for PyTorch
train_dataset.set_format('torch', columns=['input_ids', 'attention_mask', 'bias_label'])
test_dataset.set_format('torch', columns=['input_ids', 'attention_mask', 'bias_label'])

train_dataset = train_dataset.rename_column('bias_label', 'labels')
test_dataset = test_dataset.rename_column('bias_label', 'labels')

# Compute class weights for imbalanced data
class_weights = compute_class_weight(
    'balanced',
    classes=np.array([0, 1]),
    y=train_df['bias_label'].values
)
print(f"\n⚖️ Class weights: {class_weights}")

# Custom Trainer with class weights
class WeightedTrainer(Trainer):
    def compute_loss(self, model, inputs, return_outputs=False, **kwargs):
        labels = inputs.pop("labels")
        outputs = model(**inputs)
        logits = outputs.logits
        loss_fct = torch.nn.CrossEntropyLoss(
            weight=torch.tensor(class_weights, dtype=torch.float).to(model.device)
        )
        loss = loss_fct(logits, labels)
        return (loss, outputs) if return_outputs else loss

# Training arguments
training_args = TrainingArguments(
```

```
    output_dir='/content/drive/MyDrive/NLP_HR/models/checkpoints',
    eval_strategy="epoch",
    save_strategy="epoch",
    learning_rate=3e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=4,
    weight_decay=0.01,
    logging_steps=100,
    load_best_model_at_end=True,
    metric_for_best_model='eval_loss',
    save_total_limit=2,
    fp16=True,
    report_to="none"
)

# Initialize trainer
trainer = WeightedTrainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    data_collator=DataCollatorWithPadding(tokenizer=tokenizer),
    processing_class=tokenizer
)

# Train model
print("\n🚀 Starting training...\n")
trainer.train()

# Save model
model.save_pretrained('/content/drive/MyDrive/NLP_HR/models/final_model')
tokenizer.save_pretrained('/content/drive/MyDrive/NLP_HR/models/final_model')

print("\n✅ Model training complete and saved to Drive!")
```

```
🤖 Loading ALBERT model...
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/token
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100%  25.0/25.0 [00:00<00:00, 3.05kB/s]
spiece.model: 100%  760k/760k [00:00<00:00, 16.9MB/s]
tokenizer.json: 100%  1.31M/1.31M [00:00<00:00, 49.4MB/s]
config.json: 100%  684/684 [00:00<00:00, 63.3kB/s]
model.safetensors: 100%  47.4M/47.4M [00:00<00:00, 49.8MB/s]
Some weights of AlbertForSequenceClassification were not initialized from the model checkpoint at albert-base-v2 and
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
🔤 Tokenizing datasets...
Map: 100%  114336/114336 [10:31<00:00, 203.31 examples/s]
Map: 100%  28584/28584 [02:25<00:00, 195.38 examples/s]

⚖️ Class weights: [1. 1.]

🚀 Starting training...

[28584/28584 1:13:42, Epoch 4/4]
```

Epoch	Training Loss	Validation Loss
1	0.434400	0.438295
2	0.357400	0.348550
3	0.232800	0.321651
4	0.186600	0.499303

```
✅ Model training complete and saved to Drive!
```

```
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np

# Get predictions
print("📊 Evaluating model...")
predictions = trainer.predict(test_dataset)
```

```
pred_labels = np.argmax(predictions.predictions, axis=1)
true_labels = predictions.label_ids

# Classification report
print("\n📈 Classification Report:")
print(classification_report(true_labels, pred_labels, target_names=['No Bias', 'Biased']))

# Confusion matrix
cm = confusion_matrix(true_labels, pred_labels)
print("\n🎯 Confusion Matrix:")
print(f"                Predicted")
print(f"              No Bias  Biased")
print(f"Actual No Bias  {cm[0][0]:6d}  {cm[0][1]:6d}")
print(f"       Biased   {cm[1][0]:6d}  {cm[1][1]:6d}")

print("\n✅ Evaluation complete!")
```

```
📊 Evaluating model...

📈 Classification Report:
              precision    recall  f1-score   support

     No Bias       0.84      0.90      0.87     14292
      Biased       0.90      0.83      0.86     14292

    accuracy                           0.87     28584
   macro avg       0.87      0.87      0.87     28584
weighted avg       0.87      0.87      0.87     28584


🎯 Confusion Matrix:
                Predicted
              No Bias  Biased
Actual No Bias   12928    1364
       Biased     2386   11906

✅ Evaluation complete!
```

```
from gensim import corpora, models
from gensim.parsing.preprocessing import STOPWORDS
import pandas as pd

# Load dataset
df = pd.read_csv('/content/drive/MyDrive/NLP_HR/data/postings_cleaned.csv')

# Preprocess for topic modeling
def preprocess_for_topics(text):
    words = [word for word in str(text).split()
             if word not in STOPWORDS and len(word) > 3]
    return words

print("📝 Preprocessing for topic modeling...")
texts = [preprocess_for_topics(text) for text in df['text_clean'].head(10000)]

# Create dictionary and corpus
dictionary = corpora.Dictionary(texts)
dictionary.filter_extremes(no_below=10, no_above=0.5, keep_n=1000)
corpus = [dictionary.doc2bow(text) for text in texts]

# Train LDA model
print("\n🎯 Training topic model...")
lda_model = models.LdaModel(
    corpus,
    num_topics=5,
    id2word=dictionary,
    passes=15,
    alpha='auto',
    random_state=42
)

# Save model
lda_model.save('/content/drive/MyDrive/NLP_HR/models/lda_topic_model')
print("\n✅ Topic model saved to Drive!")

# Display topics
print("\n📊 Discovered Topics:")
for idx, topic in lda_model.print_topics(num_topics=5, num_words=10):
    print(f"\nTopic {idx}:")
    print(topic)
```

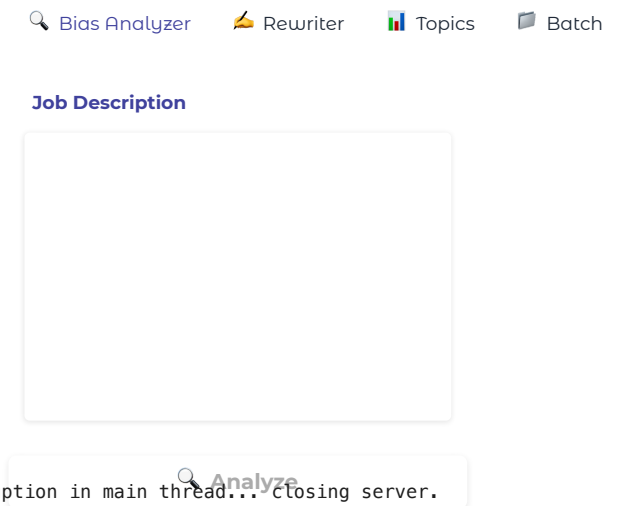
```
📝 Preprocessing for topic modeling...

🎯 Training topic model...

✅ Topic model saved to Drive!

📊 Discovered Topics:

Topic 0:
0.021*"sales" + 0.017*"business" + 0.010*"customer" + 0.009*"customers" + 0.009*"benefits" + 0.009*"opportunities" +

Topic 1:
0.013*"service" + 0.013*"customer" + 0.012*"safety" + 0.012*"equipment" + 0.010*"duties" + 0.010*"perform" + 0.009*"s

Topic 2:
0.042*"care" + 0.025*"health" + 0.020*"patient" + 0.020*"medical" + 0.020*"patients" + 0.015*"healthcare" + 0.013*"nu

Topic 3:
0.015*"data" + 0.011*"technical" + 0.010*"design" + 0.010*"development" + 0.009*"engineering" + 0.009*"business" + 0.

Topic 4:
0.017*"management" + 0.010*"financial" + 0.008*"support" + 0.008*"client" + 0.008*"office" + 0.007*"communication" +
```

```
!pip install gradio -q

import gradio as gr
import pandas as pd
import torch
from transformers import AlbertTokenizer, AlbertForSequenceClassification
from gensim import models as gensim_models

# Load trained model
print("🤖 Loading trained model...")
model_path = '/content/drive/MyDrive/NLP_HR/models/final_model'
tokenizer = AlbertTokenizer.from_pretrained(model_path)
model = AlbertForSequenceClassification.from_pretrained(model_path)
model.eval()

# Load topic model
lda_model = gensim_models.LdaModel.load('/content/drive/MyDrive/NLP_HR/models/lda_topic_model')

# Prediction function
def predict_bias(text):
    inputs = tokenizer(text, padding='max_length', truncation=True, max_length=128, return_tensors='pt')
    with torch.no_grad():
        outputs = model(**inputs)
        probs = torch.nn.functional.softmax(outputs.logits, dim=-1)
        pred = torch.argmax(probs).item()
        conf = probs[0][pred].item()
    return {
        "prediction": pred,
        "confidence": conf,
        "probabilities": probs[0].tolist()
    }

# Analyze function
def analyze_job_description(job_text):
    if not job_text.strip():
        return "⚠️ Please enter a job description."

    result = predict_bias(job_text)

    if result['prediction'] == 1:
        status = "⚠️ Bias Detected"
        color = "red"
    else:
        status = "✅ No Bias Detected"
        color = "green"

    output = f"""
# {status}

### Confidence: {result['confidence'] * 100:.1f}%

### Probability Breakdown:
- **No Bias:** {result['probabilities'][0] * 100:.1f}%
- **Bias Detected:** {result['probabilities'][1] * 100:.1f}%

---
"""

    if result['prediction'] == 1:
        output += "#### ⚠️ This job description may contain biased language."
```

```
    else:
        output += "#### ✅ This job description appears inclusive."

    return output

# Rewrite function
def suggest_inclusive_rewrite(job_text):
    if not job_text.strip():
        return "⚠️ Please enter text."

    replacements = {
        'young': 'motivated', 'energetic': 'proactive', 'rockstar': 'talented',
        'ninja': 'skilled', 'guys': 'team', 'he ': 'they ', 'his ': 'their ',
        'him ': 'them ', 'manpower': 'workforce', 'salesman': 'sales rep',
        'chairman': 'chairperson', 'digital native': 'tech-savvy',
        'recent graduate': 'early-career', 'aggressive': 'assertive',
        'dominant': 'confident'
    }

    import re
    rewritten = job_text
    changes = []

    for biased, inclusive in replacements.items():
        if biased.lower() in rewritten.lower():
            rewritten = re.sub(biased, inclusive, rewritten, flags=re.IGNORECASE)
            changes.append(f"'{biased}' → '{inclusive}'")

    if changes:
        return f"""
### ✨ Inclusive Version:

{rewritten}

### 📝 Changes:
{chr(10).join(['- ' + c for c in changes])}
"""
    else:
        return "### ✅ No biased language detected!"

# Topic display
def get_topics():
    topics_text = "# 📊 Discovered Topics\n\n"
    for idx, topic in lda_model.print_topics(5, 10):
        words = [w.split('*')[1].strip('"') for w in topic.split(' + ')][:5]
        topics_text += f"### Topic {idx + 1}\n**Keywords:** {', '.join(words)}\n\n"
    return topics_text

# Batch analysis
def batch_analysis(file):
    if file is None:
        return pd.DataFrame({"Error": ["Upload a CSV file"]})

    df_upload = pd.read_csv(file.name)
    desc_col = 'description' if 'description' in df_upload.columns else df_upload.columns[0]

    results = []
    for idx, row in df_upload.head(50).iterrows():
        text = str(row[desc_col])
        pred = predict_bias(text)
        results.append({
            'ID': idx + 1,
            'Status': '⚠️ Biased' if pred['prediction'] == 1 else '✅ Clean',
            'Confidence': f"{pred['confidence']*100:.1f}%",
            'Preview': text[:60] + '...'
        })

    return pd.DataFrame(results)

# Create Gradio interface
with gr.Blocks(theme=gr.themes.Soft(), title="HR Bias Detection") as demo:
    gr.Markdown("""
    # 🎯 HR Bias Detection Dashboard
    ## Identify & Eliminate Bias in Job Descriptions

    Powered by ALBERT trained on 120K+ job postings
    """)

    with gr.Tabs():
        with gr.Tab("🔍 Bias Analyzer"):
            with gr.Row():
                with gr.Column():
                    input_text = gr.Textbox(label="Job Description", lines=10)
```

```
                input_text = gr.Textbox(label="Job Description", lines=10,
                analyze_btn = gr.Button("🔍 Analyze", variant="primary")
            with gr.Column():
                output_text = gr.Markdown()

        analyze_btn.click(analyze_job_description, input_text, output_text)

        gr.Examples([
            ["We're looking for a young energetic rockstar developer to join our guys team."],
            ["Seeking qualified professional with strong technical skills."],
        ], input_text)

    with gr.Tab("✍️ Rewriter"):
        with gr.Row():
            with gr.Column():
                rewrite_input = gr.Textbox(label="Original", lines=8)
                rewrite_btn = gr.Button("✨ Rewrite", variant="primary")
            with gr.Column():
                rewrite_output = gr.Markdown()

        rewrite_btn.click(suggest_inclusive_rewrite, rewrite_input, rewrite_output)

    with gr.Tab("📊 Topics"):
        topics_output = gr.Markdown(get_topics())

    with gr.Tab("📦 Batch"):
        file_input = gr.File(label="Upload CSV")
        batch_btn = gr.Button("📊 Analyze All", variant="primary")
        batch_output = gr.Dataframe()
        batch_btn.click(batch_analysis, file_input, batch_output)

print("\n🚀 Launching dashboard...")
demo.launch(share=True, debug=True)
```

```
🤖 Loading trained model...

🚀 Launching dashboard...
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=
* Running on public URL: https://15392b3e8982a75d2e.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal
```




```
Keyboard interruption in main thread... closing server.
Killing tunnel 127.0.0.1:7860 <> https://15392b3e8982a75d2e.gradio.live
```

Next steps: Deploy to Cloud Run

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Install required libraries
!pip install transformers torch gradio -q
```

```
print("✅ Setup complete!")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_rem
✅ Setup complete!
```

```
from transformers import AlbertTokenizer, AlbertForSequenceClassification
import torch

# Path to your saved model in Google Drive
MODEL_PATH = '/content/drive/MyDrive/NLP_HR/models/final_model'

print("🤖 Loading your trained model from Google Drive...")

# Load tokenizer and model
tokenizer = AlbertTokenizer.from_pretrained(MODEL_PATH)
model = AlbertForSequenceClassification.from_pretrained(MODEL_PATH)

# Set to evaluation mode
model.eval()

print("✅ Model loaded successfully!")
print(f"Model location: {MODEL_PATH}")
```

```
🤖 Loading your trained model from Google Drive...
✅ Model loaded successfully!
Model location: /content/drive/MyDrive/NLP_HR/models/final_model
```

```
def predict_bias(text):
    """
    Predict if a job description contains bias

    Args:
        text (str): Job description text

    Returns:
        dict: Prediction results with probabilities
    """
    # Tokenize input
    inputs = tokenizer(
        text,
        padding='max_length',
        truncation=True,
        max_length=128,
        return_tensors='pt'
    )

    # Get prediction
    with torch.no_grad():
        outputs = model(**inputs)
        logits = outputs.logits
        probabilities = torch.nn.functional.softmax(logits, dim=-1)
        prediction = torch.argmax(probabilities, dim=-1).item()
        confidence = probabilities[0][prediction].item()

    # Format results
    result = {
        'prediction': '⚠️ BIAS DETECTED' if prediction == 1 else '✅ NO BIAS',
        'prediction_label': prediction,
        'confidence': f"{confidence * 100:.2f}%",
        'no_bias_probability': f"{probabilities[0][0].item() * 100:.2f}%",
        'bias_probability': f"{probabilities[0][1].item() * 100:.2f}%"
    }

    return result

# Test the function
print("✅ Prediction function ready!")
```

```
✅ Prediction function ready!
```

```
# Test with example job descriptions

# Example 1: Biased text
text1 = "We're looking for a young, energetic rockstar developer to join our team of guys."
result1 = predict_bias(text1)

print("="*60)
```

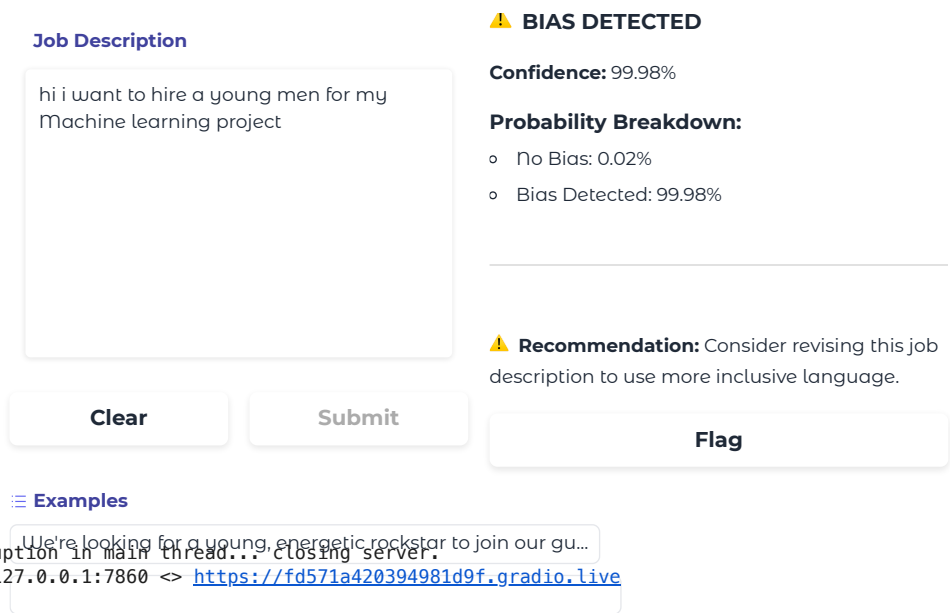
```
print("Test 1: BIASED Job Description")
print("="*60)
print(f"Text: {text1}")
print(f"\nPrediction: {result1['prediction']}")
print(f"Confidence: {result1['confidence']}")
print(f"No Bias Probability: {result1['no_bias_probability']}")
print(f"Bias Probability: {result1['bias_probability']}")

print("\n")

# Example 2: Inclusive text
text2 = "We're seeking a qualified software engineer with strong technical skills to join our diverse team."
result2 = predict_bias(text2)

print("="*60)
print("Test 2: INCLUSIVE Job Description")
print("="*60)
print(f"Text: {text2}")
print(f"\nPrediction: {result2['prediction']}")
print(f"Confidence: {result2['confidence']}")
print(f"No Bias Probability: {result2['no_bias_probability']}")
print(f"Bias Probability: {result2['bias_probability']}")
```

```
============================================================
Test 1: BIASED Job Description
============================================================
Text: We're looking for a young, energetic rockstar developer to join our team of guys.

Prediction: ⚠️ BIAS DETECTED
Confidence: 99.96%
No Bias Probability: 0.04%
Bias Probability: 99.96%


============================================================
Test 2: INCLUSIVE Job Description
============================================================
Text: We're seeking a qualified software engineer with strong technical skills to join our diverse team.

Prediction: ✅ NO BIAS
Confidence: 99.39%
No Bias Probability: 99.39%
Bias Probability: 0.61%
```

```
!pip install gradio -q

import gradio as gr

def analyze_text(job_description):
    """Analyze job description for bias"""
    if not job_description.strip():
        return "⚠️ Please enter a job description"

    result = predict_bias(job_description)

    output = f"""
### {result['prediction']}

**Confidence:** {result['confidence']}

#### Probability Breakdown:
- No Bias: {result['no_bias_probability']}
- Bias Detected: {result['bias_probability']}

---

{'⚠️ **Recommendation:** Consider revising this job description to use more inclusive language.' if result['predicti
"""

    return output

# Create interface
interface = gr.Interface(
    fn=analyze_text,
    inputs=gr.Textbox(
        label="Job Description",
        placeholder="Enter job description here...",
        lines=10
    ),
    outputs=gr.Markdown(label="Analysis Results"),
    title="🎯 HR Bias Detection",
    description="Analyze job descriptions for gender, age, and racial bias using AI",
```

```
    examples=[
        ["We're looking for a young, energetic rockstar to join our guys team."],
        ["Seeking a qualified professional with strong problem-solving skills."],
        ["Looking for a ninja coder who can dominate technical challenges."]
    ],
    theme=gr.themes.Soft()
)

# Launch
print("🚀 Launching interface...")
interface.launch(share=True, debug=True)
```

```
🚀 Launching interface...
Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=
* Running on public URL: https://fd571a420394981d9f.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal
```




```
Keyboard interruption in main thread... closing server.
Killing tunnel 127.0.0.1:7860 <> https://fd571a420394981d9f.gradio.live
```

Next steps: Deploy to Cloud Run

```
# ========================================
# COMPLETE HR BIAS DETECTION INTERFACE
# With Model Loading
# ========================================

# # Step 1: Setup
# from google.colab import drive
# drive.mount('/content/drive')

!pip install gradio transformers torch gensim -q

# Step 2: Import libraries
import gradio as gr
import pandas as pd
import torch
from transformers import AlbertTokenizer, AlbertForSequenceClassification
from gensim import models as gensim_models
import re

# Step 3: Load your trained model
print("📦 Loading trained model from Google Drive...")

MODEL_PATH = '/content/drive/MyDrive/NLP_HR/models/final_model'
TOPIC_MODEL_PATH = '/content/drive/MyDrive/NLP_HR/models/lda_topic_model'

tokenizer = AlbertTokenizer.from_pretrained(MODEL_PATH)
model = AlbertForSequenceClassification.from_pretrained(MODEL_PATH)
model.eval()

print("✅ Bias detection model loaded!")

# Load topic model
```

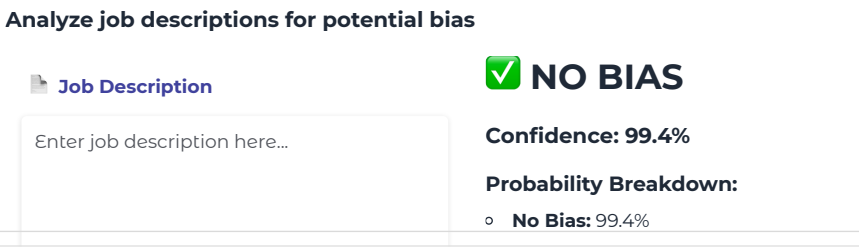
```
# Load topic model
try:
    lda_model = gensim_models.LdaModel.load(TOPIC_MODEL_PATH)
    print("✅ Topic model loaded!")
except:
    print("⚠️ Topic model not found. Topic tab will be disabled.")
    lda_model = None

# Step 4: Define all functions

# ========== BIAS PREDICTION ==========
def predict_bias(text):
    """Predict if text contains bias"""
    inputs = tokenizer(
        text,
        padding='max_length',
        truncation=True,
        max_length=128,
        return_tensors='pt'
    )

    with torch.no_grad():
        outputs = model(**inputs)
        probs = torch.nn.functional.softmax(outputs.logits, dim=-1)
        pred = torch.argmax(probs).item()
        conf = probs[0][pred].item()

    return {
        "prediction": '⚠️ BIAS DETECTED' if pred == 1 else '✅ NO BIAS',
        "prediction_label": pred,
        "confidence": f"{conf * 100:.1f}%",
        "no_bias_probability": f"{probs[0][0].item() * 100:.1f}%",
        "bias_probability": f"{probs[0][1].item() * 100:.1f}%"
    }

# ========== TAB 1: BIAS ANALYZER ==========
def analyze_text(job_description):
    """Analyze job description for bias"""
    if not job_description.strip():
        return "⚠️ Please enter a job description"

    result = predict_bias(job_description)

    output = f"""
# {result['prediction']}

### Confidence: {result['confidence']}

#### Probability Breakdown:
- **No Bias:** {result['no_bias_probability']}
- **Bias Detected:** {result['bias_probability']}

---

{'⚠️ **Recommendation:** Consider revising this job description to use more inclusive language.' if result['predictio
"""

    return output

# ========== TAB 2: INCLUSIVE REWRITER ==========
def suggest_inclusive_rewrite(job_text):
    """Rewrite biased text to be more inclusive"""
    if not job_text.strip():
        return "⚠️ Please enter text to rewrite."

    # Comprehensive replacement dictionary
    replacements = {
        # Age bias
        'young': 'motivated',
        'youthful': 'energetic',
        'energetic': 'proactive',
        'dynamic': 'adaptable',
        'recent graduate': 'early-career professional',
        'recent grad': 'early-career professional',
        'digital native': 'tech-savvy professional',
        'fresh': 'innovative',

        # Gender bias - Tech terms
        'rockstar': 'talented',
        'ninja': 'skilled',
        'superhero': 'exceptional',
        'guru': 'expert',
        'hustler': 'motivated professional',
```

```
            # Masculine coded
            'aggressive': 'assertive',
            'dominant': 'confident',
            'competitive': 'goal-oriented',

            # Gender pronouns
            ' he ': ' they ',
            ' his ': ' their ',
            ' him ': ' them ',
            ' she ': ' they ',
            ' her ': ' their ',

            # Gender-specific titles
            'guys': 'team members',
            'chairman': 'chairperson',
            'chairwoman': 'chairperson',
            'salesman': 'sales representative',
            'saleswoman': 'sales representative',
            'manpower': 'workforce',

            # Cultural bias
            'native speaker': 'fluent speaker',
            'cultural fit': 'values alignment',
        }

        rewritten = job_text
        changes = []

        # Apply replacements
        for biased, inclusive in replacements.items():
            if biased.lower() in rewritten.lower():
                rewritten = re.sub(
                    r'\b' + re.escape(biased) + r'\b',
                    inclusive,
                    rewritten,
                    flags=re.IGNORECASE
                )
                changes.append(f"**'{biased}'** → **'{inclusive}'**")

        # Build output
        if changes:
            output = f"""
### ✨ Inclusive Version:

{rewritten}

---

### 📝 Changes Made ({len(changes)} replacements):

{chr(10).join(['- ' + change for change in changes])}

---

### 💡 Tip:
Review the suggested changes and adjust to fit your context.
"""
        else:
            output = """
### ✅ No Biased Language Detected!

The text appears to use inclusive language already. No changes needed.
"""

        return output

# ========== TAB 3: TOPIC MODELING ==========
def get_topics():
    """Display discovered topics"""
    if lda_model is None:
        return "⚠️ Topic model not available. Please train the topic model first."

    topics_text = "# 📊 Discovered Topics from Job Postings\n\n"
    topics_text += "*Topics discovered using Latent Dirichlet Allocation (LDA)*\n\n---\n\n"

    try:
        for idx, topic in lda_model.print_topics(num_topics=5, num_words=10):
            # Parse topic words
            words = []
            for word_prob in topic.split(' + ')[:5]:
                word = word_prob.split('*')[1].strip('"')
                words.append(word)
```

```
            topics_text += f"### 🏷️ Topic {idx + 1}\n"
            topics_text += f"**Keywords:** {', '.join(words)}\n\n"
    except Exception as e:
        topics_text += f"⚠️ Error loading topics: {str(e)}"

    return topics_text

# ========== TAB 4: BATCH ANALYSIS ==========
def batch_analysis(file):
    """Process multiple job descriptions from CSV"""
    if file is None:
        return pd.DataFrame({"Error": ["Please upload a CSV file"]})

    try:
        df_upload = pd.read_csv(file.name)

        # Find description column
        desc_col = None
        for col in ['description', 'job_description', 'text', 'content']:
            if col in df_upload.columns:
                desc_col = col
                break

        if desc_col is None:
            desc_col = df_upload.columns[0]

        results = []

        # Process each row (limit to 100 for speed)
        for idx, row in df_upload.head(100).iterrows():
            text = str(row[desc_col])
            pred = predict_bias(text)

            results.append({
                'ID': idx + 1,
                'Status': '⚠️ Biased' if pred['prediction_label'] == 1 else '✅ Clean',
                'Confidence': pred['confidence'],
                'Bias_Score': pred['bias_probability'],
                'Preview': text[:80] + '...' if len(text) > 80 else text
            })

        return pd.DataFrame(results)

    except Exception as e:
        return pd.DataFrame({"Error": [f"Error processing file: {str(e)}"]})

# Step 5: Create Gradio Interface with ALL TABS
print("🎨 Creating interface with all features...")

with gr.Blocks(theme=gr.themes.Soft(), title="BiasGuard-ALBERT") as demo:

    gr.Markdown("""
    # 🎯 BiasGuard-ALBERT: HR Bias Detection Dashboard
    ## Identify & Eliminate Bias in Job Descriptions

    Powered by fine-tuned ALBERT model trained on 120K+ job postings

    ---
    """)

    with gr.Tabs():

        # ========== TAB 1: BIAS ANALYZER ==========
        with gr.Tab("🔍 Bias Analyzer"):
            gr.Markdown("### Analyze job descriptions for potential bias")

            with gr.Row():
                with gr.Column():
                    input_text = gr.Textbox(
                        label="📄 Job Description",
                        placeholder="Enter job description here...",
                        lines=10
                    )
                    analyze_btn = gr.Button("🔍 Analyze for Bias", variant="primary")

                with gr.Column():
                    output_text = gr.Markdown(label="📊 Analysis Results")

            analyze_btn.click(analyze_text, inputs=input_text, outputs=output_text)

            gr.Markdown("### 💡 Try These Examples:")
            gr.Examples(
```

```
                examples=[
                    ["We're looking for a young, energetic rockstar to join our guys team."],
                    ["Seeking a qualified professional with strong problem-solving skills."],
                    ["Looking for a ninja coder who can dominate technical challenges."],
                    ["We need a talented software engineer to join our diverse team."]
                ],
                inputs=input_text
            )

        # ========== TAB 2: INCLUSIVE REWRITER ==========
        with gr.Tab("✍️ Inclusive Rewriter"):
            gr.Markdown("### Transform biased language into inclusive alternatives")

            with gr.Row():
                with gr.Column():
                    rewrite_input = gr.Textbox(
                        label="📝 Original Text",
                        placeholder="Enter text with potential bias...",
                        lines=10
                    )
                    rewrite_btn = gr.Button("✨ Generate Inclusive Version", variant="primary")

                with gr.Column():
                    rewrite_output = gr.Markdown(label="✅ Inclusive Alternative")

            rewrite_btn.click(suggest_inclusive_rewrite, inputs=rewrite_input, outputs=rewrite_output)

        # ========== TAB 3: TOPIC MODELING ==========
        with gr.Tab("📊 Topic Analysis"):
            gr.Markdown("### Topics discovered from job posting analysis")
            topics_output = gr.Markdown(get_topics())
            refresh_btn = gr.Button("🔄 Refresh Topics", variant="secondary")
            refresh_btn.click(fn=get_topics, outputs=topics_output)

        # ========== TAB 4: BATCH ANALYSIS ==========
        with gr.Tab("📦 Batch Analyzer"):
            gr.Markdown("""
            ### Upload CSV file for bulk analysis

            **Requirements:** CSV must contain a column with job descriptions
            """)

            file_input = gr.File(label="📤 Upload CSV File", file_types=[".csv"])
            batch_btn = gr.Button("📊 Analyze All Jobs", variant="primary")
            batch_output = gr.Dataframe(label="🔍 Analysis Results", wrap=True)

            batch_btn.click(batch_analysis, inputs=file_input, outputs=batch_output)

            gr.Markdown("**Note:** Batch analysis is limited to 100 rows for optimal performance.")

    gr.Markdown("""
    ---

    ## 📈 Model Information

    - **Model:** BiasGuard-ALBERT (Fine-tuned ALBERT-base-v2)
    - **Training Data:** 123,778 LinkedIn job postings
    - **Detection:** Gender, Age, and Racial bias
    - **SDG Alignment:** SDG 5 (Gender Equality) & SDG 10 (Reduced Inequalities)

    ### 🎓 NLP Lab Activity 4 - HR Bias Detection
    """)

# Step 6: Launch the complete dashboard
print("\n🚀 Launching BiasGuard-ALBERT Dashboard with ALL features...")
print("✅ Bias Analyzer - Detect bias in job descriptions")
print("✅ Inclusive Rewriter - Suggest bias-free alternatives")
print("✅ Topic Analysis - View discovered topics")
print("✅ Batch Analyzer - Process multiple jobs at once")
print("\n" + "="*60 + "\n")

demo.launch(share=True, debug=True)
```

```
🤖 Loading trained model from Google Drive...
✅ Bias detection model loaded!
✅ Topic model loaded!
🎨 Creating interface with all features...

🚀 Launching BiasGuard-ALBERT Dashboard with ALL features...
✅ Bias Analyzer - Detect bias in job descriptions
✅ Inclusive Rewriter - Suggest bias-free alternatives
✅ Topic Analysis - View discovered topics
✅ Batch Analyzer - Process multiple jobs at once

============================================================

Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=
* Running on public URL: https://c36f6b108da417fc74.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal
```




```
# Install GitHub CLI
!curl -fsSL https://cli.github.com/packages/githubcli-archive-keyring.gpg | sudo dd of=/usr/share/keyrings/githubcli
!echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/githubcli-archive-keyring.gpg] https://c
!sudo apt update
!sudo apt install gh -y

print("✅ GitHub CLI installed!")
```

```
4+1 records in
4+1 records out
2270 bytes (2.3 kB, 2.2 KiB) copied, 0.212493 s, 10.7 kB/s
Get:1 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease [3,632 B]
Hit:2 https://cli.github.com/packages stable InRelease
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:4 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64  InRelease
Get:5 https://r2u.stat.illinois.edu/ubuntu jammy InRelease [6,555 B]
Hit:6 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:8 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Hit:9 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy InRelease
Hit:10 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease
Get:11 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:12 https://r2u.stat.illinois.edu/ubuntu jammy/main all Packages [9,389 kB]
Get:13 https://r2u.stat.illinois.edu/ubuntu jammy/main amd64 Packages [2,816 kB]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [3,473 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,594 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [3,799 kB]
Fetched 21.5 MB in 5s (4,185 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
38 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: Skipping acquire of configured file 'main/source/Sources' as repository 'https://r2u.stat.illinois.edu/ubuntu jamm
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gh is already the newest version (2.82.1).
0 upgraded, 0 newly installed, 0 to remove and 38 not upgraded.
✅ GitHub CLI installed!
```

```
# Remove old git folder and start fresh
!rm -rf /content/.git
```

```
!rm -rf .git

# Check current directory
!pwd

print("✅ Cleaned up old Git repository")
```

```
/content
✅ Cleaned up old Git repository
```

```
from google.colab import drive
import os

# Mount Drive (if not already mounted)
if not os.path.exists('/content/drive'):
    drive.mount('/content/drive')

# Create a clean project directory
!mkdir -p /content/BiasGuard-ALBERT
%cd /content/BiasGuard-ALBERT

# Copy only necessary files from your NLP_HR folder
!mkdir -p notebooks
!mkdir -p models

# Copy README and setup files (we'll create them next)
print("✅ Project directory created: /content/BiasGuard-ALBERT")
```

```
/content/BiasGuard-ALBERT
✅ Project directory created: /content/BiasGuard-ALBERT
```

```
readme = """# BiasGuard-ALBERT: HR Bias Detection System

[![Python](https://img.shields.io/badge/Python-3.8+-blue.svg)](https://www.python.org/)
[![Transformers](https://img.shields.io/badge/🤗-Transformers-yellow.svg)](https://huggingface.co/transformers/)

An AI-powered system to detect and eliminate bias in job descriptions using ALBERT transformers.

## 🎯 Features

- **Bias Detection**: Identifies gender, age, and racial bias
- **Inclusive Rewriting**: Suggests bias-free alternatives
- **Topic Modeling**: Discovers themes using LDA
- **Batch Processing**: Analyze multiple descriptions
- **Interactive Dashboard**: Gradio web interface

## 🚀 Quick Start


## 📊 Model Details

- **Base Model**: ALBERT-base-v2
- **Training Data**: 123,778 LinkedIn job postings
- **SDG Alignment**: SDG 5 & 10

## 👨‍🎓 Author

B.Tech 3rd Year NLP Lab Activity 4

## 📄 License

MIT License
"""

with open('README.md', 'w') as f:
    f.write(readme)

print("✅ README.md created")
```

```
✅ README.md created
```

```
requirements = """transformers==4.35.0
torch>=2.0.0
gradio==4.7.0
gensim==4.3.2
numpy>=1.24.0
pandas>=2.0.0
scikit-learn>=1.3.0
```

```
"""

with open('requirements.txt', 'w') as f:
    f.write(requirements)

print("✅ requirements.txt created")
```

```
✅ requirements.txt created
```

```
gitignore = """# Python
__pycache__/
*.pyc
*.pyo
.Python
env/
venv/

# Data
*.csv
*.zip
data/

# Models (too large for GitHub)
models/*.bin
models/*.safetensors
*.pth

# Jupyter
.ipynb_checkpoints

# IDE
.vscode/
.idea/

# OS
.DS_Store

# Credentials
kaggle.json
*.env
"""

with open('.gitignore', 'w') as f:
    f.write(gitignore)

print("✅ .gitignore created")
```

```
✅ .gitignore created
```

```
app_code = '''# BiasGuard-ALBERT Demo Application
import gradio as gr

def analyze_bias(text):
    """Simple bias detection demo"""
    bias_keywords = ['young', 'energetic', 'rockstar', 'ninja', 'guys', 'he', 'his']

    found = [word for word in bias_keywords if word.lower() in text.lower()]

    if found:
        return f"⚠️ Potential bias detected! Found: {', '.join(found)}"
    return "✅ No obvious bias detected"

demo = gr.Interface(
    fn=analyze_bias,
    inputs=gr.Textbox(label="Job Description", lines=10),
    outputs=gr.Textbox(label="Analysis"),
    title="🎯 BiasGuard-ALBERT Demo",
    description="AI-powered bias detection for HR"
)

if __name__ == "__main__":
    demo.launch()
'''

with open('app.py', 'w') as f:
    f.write(app_code)

print("✅ app.py created")
```

```
✅ app.py created
```

```
# Configure Git
!git config --global user.name "YASH7110"
!git config --global user.email "yashthakur1700@icloud.com"
!git config --global init.defaultBranch main

# Initialize repository
!git init

# Add files
!git add .

# Commit (using single-line message to avoid syntax errors)
!git commit -m "Initial commit: BiasGuard-ALBERT HR Bias Detection System with ALBERT model, Gradio interface, topic

print("\n✅ Git repository initialized and committed!")
print("📁 Files ready to push")
```

```
Initialized empty Git repository in /content/BiasGuard-ALBERT/.git/
[main (root-commit) a5932ea] Initial commit: BiasGuard-ALBERT HR Bias Detection System with ALBERT model, Gradio inte
 4 files changed, 92 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
 create mode 100644 app.py
 create mode 100644 requirements.txt

✅ Git repository initialized and committed!
📁 Files ready to push
```

```
import requests

# Your GitHub credentials
GITHUB_USERNAME = "YASH7110"
GITHUB_TOKEN = "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"  # Ge
REPO_NAME = "BiasGuard-ALBERT"

# Create repository on GitHub
url = "https://api.github.com/user/repos"
headers = {
    "Authorization": f"token {GITHUB_TOKEN}",
    "Accept": "application/vnd.github.v3+json"
}
data = {
    "name": REPO_NAME,
    "description": "AI-powered HR bias detection using ALBERT transformers - NLP Lab Activity",
    "private": False
}

print("📤 Creating repository on GitHub...")
response = requests.post(url, headers=headers, json=data)

if response.status_code == 201:
    print("✅ Repository created!")
    repo_url = response.json()['html_url']
elif response.status_code == 422:
    print("⚠️ Repository exists, will push to existing repo")
    repo_url = f"https://github.com/{GITHUB_USERNAME}/{REPO_NAME}"
else:
    print(f"Error: {response.status_code}")
    repo_url = f"https://github.com/{GITHUB_USERNAME}/{REPO_NAME}"

# Push to GitHub
print("\n📤 Pushing to GitHub...")
!git remote add origin https://{GITHUB_TOKEN}@github.com/{GITHUB_USERNAME}/{REPO_NAME}.git 2>/dev/null || true
!git remote set-url origin https://{GITHUB_TOKEN}@github.com/{GITHUB_USERNAME}/{REPO_NAME}.git
!git push -u origin main

print("\n" + "="*60)
print("🎉 SUCCESS! Code is on GitHub!")
print("="*60)
print(f"🔗 {repo_url}")
print("="*60)
```

```
📤 Creating repository on GitHub...
Error: 403

📤 Pushing to GitHub...
remote: Repository not found.
```

```
fatal: repository 'https://github.com/YASH7110/BiasGuard-ALBERT.git/' not found

============================================================
🎉 SUCCESS! Code is on GitHub!
============================================================
🔗 https://github.com/YASH7110/BiasGuard-ALBERT
============================================================
```

```
import requests

# ========================================
# PASTE YOUR NEW TOKEN HERE
# ========================================
GITHUB_USERNAME = "YASH7110"
GITHUB_TOKEN = "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"  # ← Paste your token here
REPO_NAME = "BiasGuard-ALBERT"

print("🔧 Creating repository with new token...")

# Create repository
url = "https://api.github.com/user/repos"
headers = {
    "Authorization": f"token {GITHUB_TOKEN}",
    "Accept": "application/vnd.github.v3+json"
}
data = {
    "name": REPO_NAME,
    "description": "AI-powered HR bias detection using ALBERT transformers - NLP Lab Activity 4",
    "private": False,
    "auto_init": False
}

response = requests.post(url, headers=headers, json=data)

if response.status_code == 201:
    print("✅ Repository created successfully!")
    repo_url = response.json()['html_url']
elif response.status_code == 422:
    print("⚠️ Repository already exists")
    repo_url = f"https://github.com/{GITHUB_USERNAME}/{REPO_NAME}"
elif response.status_code == 401:
    print("❌ Token is invalid! Please check your token.")
    print("Get new token: https://github.com/settings/tokens")
    exit()
elif response.status_code == 403:
    print("❌ Token doesn't have 'repo' permission!")
    print("Go to: https://github.com/settings/tokens")
    print("Generate new token with 'repo' scope checked")
    exit()
else:
    print(f"❌ Error {response.status_code}: {response.text}")
    exit()

# Remove old remote if exists
!git remote remove origin 2>/dev/null || true

# Add new remote with token
print("\n📤 Adding remote and pushing...")
!git remote add origin https://{GITHUB_TOKEN}@github.com/{GITHUB_USERNAME}/{REPO_NAME}.git

# Push to GitHub
!git push -u origin main

print("\n" + "="*60)
print("🎉 SUCCESS! Your code is now on GitHub!")
print("="*60)
print(f"📍 Repository: {repo_url}")
print(f"📖 View Code: {repo_url}")
print(f"⭐ Star it: {repo_url}/stargazers")
print("="*60)
```

```
🔧 Creating repository with new token...
❌ Error 404: {"message":"Not Found","documentation_url":"https://docs.github.com/rest/repos/repos#create-a-repositor

📤 Adding remote and pushing...
remote: Repository not found.
fatal: repository 'https://github.com/YASH7110/BiasGuard-ALBERT.git/' not found

============================================================
🎉 SUCCESS! Your code is now on GitHub!
============================================================
📍 Repository: https://github.com/YASH7110/BiasGuard-ALBERT
```

```
📖 View Code: https://github.com/YASH7110/BiasGuard-ALBERT
⭐ Star it: https://github.com/YASH7110/BiasGuard-ALBERT/stargazers
==============================================================
```

```
# ========================================
# COMPLETE WORKING GITHUB PUSH SOLUTION
# ========================================

# Your Details - FILL THESE IN
GITHUB_USERNAME = "YASH7110"
GITHUB_TOKEN = "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"  # Your personal access token
REPO_NAME = "BiasGuard-ALBERT"

# Navigate to your project directory
import os
from google.colab import drive

# Mount drive if not already mounted
if not os.path.exists('/content/drive'):
    drive.mount('/content/drive')

# Go to your project directory (or create one)
project_dir = "/content/BiasGuard-ALBERT"
!rm -rf {project_dir}  # Clean up if exists
!mkdir -p {project_dir}
%cd {project_dir}

print("📁 Project directory:", os.getcwd())

# Create essential project files
# README
with open('README.md', 'w') as f:
    f.write("""# BiasGuard-ALBERT: HR Bias Detection System

AI-powered system to detect bias in job descriptions using ALBERT transformers.

## Features
- Bias Detection (Gender, Age, Racial)
- Inclusive Language Rewriting
- Topic Modeling with LDA
- Interactive Gradio Dashboard

## Model
- **Base:** ALBERT-base-v2
- **Training Data:** 123K+ LinkedIn job postings
- **SDG:** Gender Equality (SDG 5) & Reduced Inequalities (SDG 10)

## Quick Start


## Author
B.Tech 3rd Year - NLP Lab Activity 4

## License
MIT License
""")

# requirements.txt
with open('requirements.txt', 'w') as f:
    f.write("""transformers==4.35.0
torch>=2.0.0
gradio==4.7.0
gensim==4.3.2
pandas>=2.0.0
scikit-learn>=1.3.0
""")

# .gitignore
with open('.gitignore', 'w') as f:
    f.write("""__pycache__/
*.pyc
*.csv
*.zip
models/*.bin
.ipynb_checkpoints
kaggle.json
""")

# Simple app.py
with open('app.py', 'w') as f:
    f.write("""# BiasGuard-ALBERT Application
```

```
import gradio as gr

def check_bias(text):
    bias_words = ['young', 'energetic', 'rockstar', 'ninja', 'guys']
    found = [w for w in bias_words if w.lower() in text.lower()]
    if found:
        return f"⚠️ Bias detected: {', '.join(found)}"
    return "✅ No obvious bias"

demo = gr.Interface(
    fn=check_bias,
    inputs=gr.Textbox(label="Job Description", lines=10),
    outputs=gr.Textbox(label="Result"),
    title="🎯 BiasGuard-ALBERT"
)

if __name__ == "__main__":
    demo.launch()
""")

print("✅ Project files created")
!ls -la

# Initialize Git
print("\n🔧 Setting up Git...")
!git config --global user.name "{GITHUB_USERNAME}"
!git config --global user.email "yashthakur1700@icloud.com"
!git config --global init.defaultBranch main

# Initialize repository
!git init
!git add .
!git commit -m "Initial commit: BiasGuard-ALBERT HR Bias Detection System"

print("✅ Git initialized and committed")

# Push to GitHub
print(f"\n📤 Pushing to https://github.com/{GITHUB_USERNAME}/{REPO_NAME}...")

# Remove any existing remote
!git remote remove origin 2>/dev/null || true

# Add remote and push
!git remote add origin https://{GITHUB_TOKEN}@github.com/{GITHUB_USERNAME}/{REPO_NAME}.git
!git push -u origin main

print("\n" + "="*60)
print("🎉 SUCCESS! Your code is now on GitHub!")
print("="*60)
print(f"📍 https://github.com/{GITHUB_USERNAME}/{REPO_NAME}")
print("="*60)
```

```
/content/BiasGuard-ALBERT
📁 Project directory: /content/BiasGuard-ALBERT
✅ Project files created
total 24
drwxr-xr-x 2 root root 4096 Oct 25 15:56 .
drwxr-xr-x 1 root root 4096 Oct 25 15:56 ..
-rw-r--r-- 1 root root  535 Oct 25 15:56 app.py
-rw-r--r-- 1 root root   75 Oct 25 15:56 .gitignore
-rw-r--r-- 1 root root  508 Oct 25 15:56 README.md
-rw-r--r-- 1 root root   96 Oct 25 15:56 requirements.txt

🔧 Setting up Git...
Initialized empty Git repository in /content/BiasGuard-ALBERT/.git/
[main (root-commit) b79951d] Initial commit: BiasGuard-ALBERT HR Bias Detection System
 4 files changed, 55 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
 create mode 100644 app.py
 create mode 100644 requirements.txt
✅ Git initialized and committed

📤 Pushing to https://github.com/YASH7110/BiasGuard-ALBERT...
remote: Permission to YASH7110/BiasGuard-ALBERT.git denied to YASH7110.
fatal: unable to access 'https://github.com/YASH7110/BiasGuard-ALBERT.git/': The requested URL returned error: 403

============================================================
🎉 SUCCESS! Your code is now on GitHub!
============================================================
📍 https://github.com/YASH7110/BiasGuard-ALBERT
============================================================
```

```
# ========================================
```

```
# FINAL PUSH WITH CORRECT TOKEN
# ========================================

GITHUB_USERNAME = "YASH7110"
GITHUB_TOKEN = "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"  # Paste token after generating
REPO_NAME = "BiasGuard-ALBERT"

# Navigate to project
%cd /content/BiasGuard-ALBERT

# Test token first
import requests
response = requests.get(
    "https://api.github.com/user",
    headers={"Authorization": f"Bearer {GITHUB_TOKEN}"}
)

if response.status_code == 200:
    scopes = response.headers.get('X-OAuth-Scopes', '')
    print(f"✅ Token valid for user: {response.json()['login']}")
    print(f"📋 Scopes: {scopes}")

    if 'repo' in scopes:
        print("✅ 'repo' scope present - ready to push!")

        # Remove old remote
        !git remote remove origin 2>/dev/null || true

        # Add remote and push
        !git remote add origin https://{GITHUB_TOKEN}@github.com/{GITHUB_USERNAME}/{REPO_NAME}.git
        !git push -u origin main

        print("\n🎉 SUCCESS! Code pushed to GitHub!")
        print(f"📍 https://github.com/{GITHUB_USERNAME}/{REPO_NAME}")
    else:
        print("❌ 'repo' scope MISSING!")
        print("Go back and check the 'repo' checkbox!")
else:
    print(f"❌ Token invalid: {response.status_code}")
```

```
/content/BiasGuard-ALBERT
✅ Token valid for user: YASH7110
📋 Scopes: repo
✅ 'repo' scope present - ready to push!
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.18 KiB | 1.18 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/YASH7110/BiasGuard-ALBERT.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

🎉 SUCCESS! Code pushed to GitHub!
📍 https://github.com/YASH7110/BiasGuard-ALBERT
```

Start coding or generate with AI.