ДОДАТОК А

Реалізація навчання моделі

```
        train_indices, val_indices = train_test_split(
        range(len(labels)), test_size=0.2, random_state=42)
        train_job_texts = [job_texts[i] for i in train_indices]
        train_resume_texts = [resume_texts[i] for i in
train_indices]
        train_labels = [labels[i] for i in train_indices]

         val_job_texts = [job_texts[i] for i in val_indices]
         val_resume_texts = [resume_texts[i] for i in
val_indices]
         val_labels = [labels[i] for i in val_indices]
         optimizer = torch.optim.AdamW(self.model.parameters(),
lr=learning_rate)
           criterion = nn.MSELoss()
           logger.info("Початок навчання...")
           best_val_loss = float('inf')
             patience = 3
             no_improvement = 0
             for epoch in range(epochs):
             self.model.train()
             total_loss = 0
             num_batches = 0
        progress_bar = tqdm(range(0, len(train_job_texts),
batch_size),
          desc=f"Epoch {epoch + 1}/{epochs}")
             for i in progress_bar:
              if i % 10 == 0:
              clear_gpu_memory()
           batch_job_texts = train_job_texts[i:i + batch_size]
           batch_resume_texts = train_resume_texts[i:i +
batch_size]
```

```
          batch_labels   =   torch.tensor(train_labels[i:i   +
batch_size]).float().to(self.device)
        optimizer.zero_grad()
        # Forward pass з mixed precision
        batch_features=             self.prepare_batch_features
(batch_job_texts, batch_resume_texts)
          with torch.cuda.amp.autocast():
            outputs = self.model(**batch_features)
            loss = criterion(outputs, batch_labels)
            # Backward pass з gradient scaling
            self.scaler.scale(loss).backward()
            self.scaler.step(optimizer)
            self.scaler.update()
            total_loss += loss.item()
            num_batches += 1
            progress_bar.set_postfix({'loss':
f'{loss.item():.4f}'})
        # Очищаємо проміжні тензори
        del outputs, loss, batch_features, batch_labels
           avg_loss = total_loss / num_batches
           logger.info(f"Epoch  {epoch  +  1},  Average  Loss:
{avg_loss:.4f}")
            self.model.eval()
            val_loss = 0
            val_batches = 0
          with torch.no_grad():
         for i in range(0, len(val_job_texts), batch_size):
         batch_job_texts = val_job_texts[i:i + batch_size]
         batch_resume_texts = val_resume_texts[i:i + batch_size]
         batch_labels    =    torch.tensor(val_labels[i:i    +
batch_size]).float().to(self.device)
         batch_features                                        =
self.prepare_batch_features(batch_job_texts,
batch_resume_texts)
       with torch.cuda.amp.autocast():
          outputs = self.model(**batch_features)
```

```
loss = criterion(outputs, batch_labels)
val_loss += loss.item()
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	74 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Кравченко І.В.			Інформаційна система обліку вакансій та кандидатів з персоналізованим AI-помічником з вбудованим NLP-модулем	Шифр групи	Код напр./спец.
Перевірив		Павленко Є.П.				ІТШІ-21-4	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Золотухін О.В.					