گزارش تمرین پنجم پردازش زبان‌های طبیعی

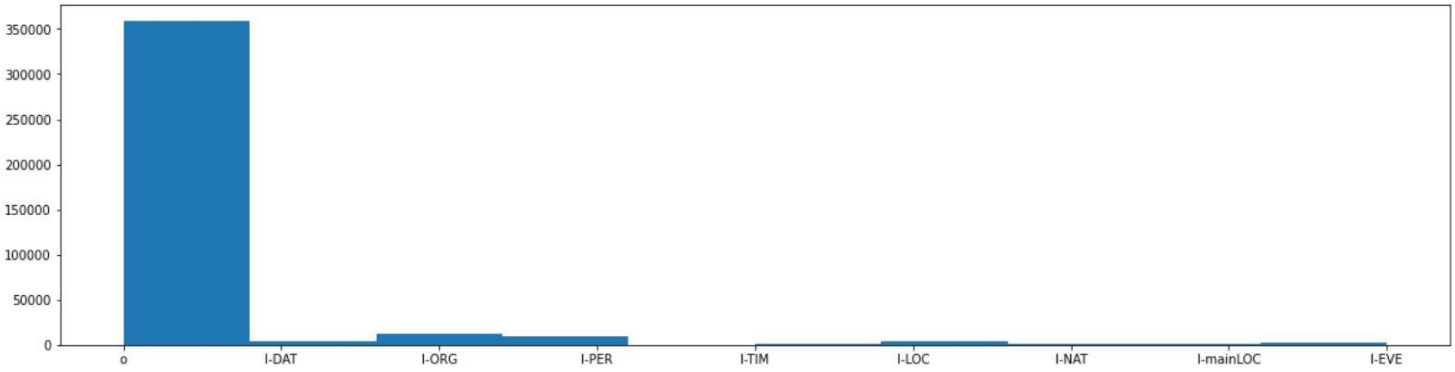
فاز اول – آماده‌سازی داده‌ها برای آموزش:

ابتدا از فایل نوتپدی که در اختیار ما قرار داده شد، داده هارا به شکل string وارد یک فایل Json و سپس CSV کرده(parsing) به طوری‌که از لحاظ ظاهری دقیقا شبیه به یک دیتافریم با دو ستون که ستون اول لیستی از توکن‌ها و ستون دوم لیستی از برچسب NER آن‌ها باشد. اما توجه شود که در حقیقت عناصر این دیتافریم لیست نبوده و string ای شبیه لیست هستند.

لذا به وسیله تابع replace() و split() و strip() این داده هارا تبدیل به دیتافریمی شامل لیست‌ها کرده تا برای پیش‌پردازش آماده باشند:

```
def clean_alt_list(list_):
    list_ = list_.replace('[','').replace(']','').split(',')
    return [item.replace("'","").strip() for item in list_]
```

سپس برای بررسی بالانس بودن برچسب‌ها، توزیع تکرار آن‌هارا بررسی کردیم که نتیجه زیر حاصل شد و مشخص شد داده‌ها به شدت unbalanced هستند:



سپس با استفاده از کتابخانه Datasets مختص HuggingFace داده‌هارا تبدیل به فرمت مخصوص برای آموزش ترنسفورمرها کردیم.

فاز دوم – آماده‌سازی برای تغییر Head ترنسفورمر از پیش آموزش داده شده برای Retraining آن روی برچسب‌های جدید:

برای اینکه قرار است در آینده Head ترنسفورمر را حذف کرده و Head جدیدی جایگزین آن کنیم (برای Retrain کردن ترنسفورمر ParsBERT-NER-Uncased) لذا دو دیکشنری id2label و label2id که مختص index کردن برچسب‌ها و بالعکس هستند را از روی برچسب‌ها بدست می‌آوریم:

```
labels = labels.tolist()
id2label = {}
label2id = {}
for i, item in enumerate(labels):
    id2label[i] = item
    label2id[item] = i
id2label
```

سپس ترنسفورمر را لود کرده و id2label و label2id و num_labels_ و num_labels را تغییر می‌دهیم (در عمل config آن را تغییر دادیم تا روی برچسب‌های جدید آموزش داده شود):

```
model = AutoModelForTokenClassification.from_pretrained(model_name, num_labels=len(id2label))
model.config.id2label = id2label
model.config.label2id = label2id
model.config._num_labels = len(id2label)
model.config.num_labels = len(label2id)
model
```

فاز سوم – توکنایز کردن داده‌ها به وسیله توکنایزر مخصوص ترنسفورمر انتخابی:

در این مرحله توکنایزر ترنسفورمر پارس‌برت را لود کرده و به وسیله تابع tokenize_and_align_labels که خودمان کدش را زدیم، و به وسیله تابع ()map کتابخانه Datasets، در batch های ۲۵۶ تایی، داده‌هارا توکنایز کرده و به ایندکس عددی تبدیل کرده و توکن های شروع و پایان را نیز به جملات اضافه می‌کنیم. حال داده‌ها آماده برای آموزش دوباره ترنسفورمر (fine-tuning) هستند.

```
batch_size = 256
task = "ner"
model_name = 'HooshvareLab/bert-fa-zwnj-base'
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
Downloading tokenizer_config.json: 100% 292/292 [00:00<00:00, 8.11kB/s]
Downloading config.json: 100% 565/565 [00:00<00:00, 11.3kB/s]
Downloading vocab.txt: 100% 416k/416k [00:00<00:00, 4.57MB/s]
Downloading tokenizer.json: 100% 1.06M/1.06M [00:00<00:00, 2.88MB/s]
Downloading special_tokens_map.json: 100% 134/134 [00:00<00:00, 3.95kB/s]
```

```
def tokenize_and_align_labels(examples):
    label_all_tokens = True
    tokenized_inputs = tokenizer(list(examples["token"]), padding="max_length",max_length= 128, truncation=True, is_split_into_words=True)

    labels = []
    for i, label in enumerate(examples['labels']):
        word_ids = tokenized_inputs.word_ids(batch_index=i)
        previous_word_idx = None
        label_ids = []
        for word_idx in word_ids:
            if word_idx is None:
                label_ids.append(-100)
            elif label[word_idx] == '0':
                label_ids.append(0)
            elif word_idx != previous_word_idx:
                label_ids.append(label2id[label[word_idx]])
            else:
                label_ids.append(label2id[label[word_idx]] if label_all_tokens else -100)
            previous_word_idx = word_idx
        labels.append(label_ids)

    tokenized_inputs["labels"] = labels
    return tokenized_inputs
```

```
dataset_test_tokenized=dataset_test.map(tokenize_and_align_labels,batched=True)
dataset_val_tokenized = dataset_val.map(tokenize_and_align_labels,batched=True)
dataset_train_tokenized = dataset_train.map(tokenize_and_align_labels,batched=True)
```

```
100% 1/1 [00:00<00:00, 4.23ba/s]
100% 1/1 [00:00<00:00, 4.51ba/s]
100% 5/5 [00:03<00:00, 1.24ba/s]
```

فاز چهارم – آموزش مجدد دسته‌بند ترنسفورمر:

ابتدا به وسیله ماژول TrainingArguments از کتابخانه Transformers، هایپرپارامترهای آموزش را تعیین کرده و سپس DataCollator و Metric ارزیابی را مشخص کرده(متریک ارزیابی همان Seqeval می‌باشد) و سپس یک تابع ارزیابی custom نیز پیاده‌سازی کردیم. به دلیل مشکل بالانس نبودن داده‌ها مجبور شدیم که Trainer اصلی HuggingFace را کنار گذاشته و یک Trainer جدید پیاده‌سازی کنیم که تابع خطای آموزش مدل را بتوانیم به Weighted Loss تغییر دهیم.

```
args = TrainingArguments(
    f"test-{task}",
    evaluation_strategy = "epoch",
    learning_rate=1e-4,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=10,
    weight_decay=1e-5,
)
```

```
data_collator = DataCollatorForTokenClassification(tokenizer)
metric = load_metric("seqeval")
```

```
def compute_metrics(p):
    predictions, labels = p
    predictions = np.argmax(predictions, axis=2)

    true_predictions = [[labels[p] for (p, l) in zip(prediction, label) if l != -100] for prediction, label in zip(predictions, labels)]
    true_labels = [[labels[l] for (p, l) in zip(prediction, label) if l != -100] for prediction, label in zip(predictions, labels)]

    results = metric.compute(predictions=true_predictions, references=true_labels)
    return {"precision": results["overall_precision"], "recall": results["overall_recall"], "f1": results["overall_f1"], "accuracy": results["overall_accuracy"]}
```

```
from torch import nn
class CustomTrainer(Trainer):
    def compute_loss(self, model, inputs, return_outputs=False):
        labels = inputs.get("labels")
        # forward pass
        outputs = model(**inputs)
        logits = outputs.get("logits")
        # compute custom loss (suppose one has 3 labels with different weights)
        loss_fct = nn.CrossEntropyLoss(weight=torch.tensor(list(label_counts.values()), device="cuda"))
        loss = loss_fct(logits.view(-1, self.model.config.num_labels), labels.view(-1))
        return (loss, outputs) if return_outputs else loss
```

```
trainer = CustomTrainer(
    model,
    args,
    train_dataset=dataset_train_tokenized,
    eval_dataset= dataset_val_tokenized,
    data_collator=data_collator,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)
```

```
trainer.train()
```

سپس مدل را آموزش مجدد داده و نتایج زیر حاصل شد:

Epoch	Training Loss	Validation Loss	Precision	Recall	F1	Accuracy
1	No log	1.093925	0.110603	0.244934	0.152392	0.784032
2	No log	1.263101	0.121355	0.275328	0.168459	0.767483
3	No log	1.810781	0.200980	0.366508	0.259603	0.834250
4	0.382000	2.027682	0.215737	0.398689	0.279975	0.845092
5	0.382000	2.644485	0.237842	0.399285	0.298109	0.858840
6	0.382000	2.507572	0.266449	0.436830	0.331000	0.870720
7	0.382000	3.115739	0.285379	0.435042	0.344665	0.882548
8	0.047100	3.272181	0.298211	0.446961	0.357739	0.885246
9	0.047100	3.476624	0.299209	0.428486	0.352365	0.889915
10	0.047100	3.589389	0.298925	0.414184	0.347240	0.892924

```
{'epoch': 10.0,
 'eval_accuracy': 0.9014743263853584,
 'eval_f1': 0.35183650925158794,
 'eval_loss': 3.7699403762817383,
 'eval_precision': 0.3155027241208519,
 'eval_recall': 0.3976279650436954,
 'eval_runtime': 13.8675,
 'eval_samples_per_second': 16.225,
 'eval_steps_per_second': 0.577}
```


سپس مدل را برای تست روی جمله تصادفی ذخیره کرده و مدل را روی HuggingFace آپلود کردیم. مدل و توکنایزر آن در آدرس زیر موجود هستند و می‌توانید به صورت آنلاین جمله دلخواه خود را به مدل داده و نتیجه را مشاهده کنید:

https://huggingface.co/pourmand1376/NER_Farsi

اعضای گروه: پویا خانی – امیر پورمند – مهدی آخی