Московский государственный технический университет им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

Домашнее Заданиепо дисциплине

«Методы машинного обучения»

Выполнил:

студент группы ИУ5И-23М

Цзян Юхуэй

Москва — 2024 г.

Задание

Домашнее задание по дисциплине направлено на анализ современных методов машинного обучения и их применение для решения практических задач. Домашнее задание включает три основных этапа:

- выбор задачи;

- теоретический этап;

- практический этап.

Этап выбора задачи предполагает анализ ресурса paperswithcode. Данный ресурс включает описание нескольких тысяч современных задач в области машинного обучения. Каждое описание задачи содержит ссылки на наиболее современные и актуальные научные статьи, предназначенные для решения задачи (список статей регулярно обновляется авторами ресурса). Каждое описание статьи содержит ссылку на репозиторий с открытым исходным кодом, реализующим представленные в статье эксперименты. На этапе выбора задачи обучающийся выбирает одну из задач машинного обучения, описание которой содержит ссылки на статьи и репозитории с исходным кодом. Теоретический этап включает проработку как минимум двух статей, относящихся к выбранной задаче. Результаты проработки обучающийся излагает в теоретической части отчета по домашнему заданию, которая может включать:

- описание общих подходов к решению задачи;

конкретные топологии нейронных сетей, нейросетевых ансамблей или других моделей машинного обучения, предназначенных для решения задачи;

- математическое описание, алгоритмы функционирования, особенности обучения используемых для решения задачи нейронных сетей, нейросетевых ансамблей или других моделей машинного обучения;

- описание наборов данных, используемых для обучения моделей;

- оценка качества решения задачи, описание метрик качества и их значений;

- предложения обучающегося по улучшению качества решения задачи. Практический этап включает повторение экспериментов авторов статей на основе представленных авторами репозиториев с исходным кодом и возможное улучшение обучающимися полученных результатов. Результаты проработки обучающийся излагает в практической части отчета по домашнему заданию, которая может включать:

- исходные коды программ, представленные авторами статей, результаты документирования программ обучающимися с использованием диаграмм UML, путем визуализации топологий нейронных сетей и другими способами;

- результаты выполнения программ, вычисление значений для описанных в статьях метрик качества, выводы обучающегося о воспроизводимости экспериментов авторов статей и соответствии практических экспериментов теоретическим материалам статей;

- предложения обучающегося по возможным улучшениям решения задачи, результаты практических экспериментов (исходные коды, документация) по возможному улучшению решения задачи.

Выбранная задача: «Категоризация текста»

Теоретический этап

1. Выбор задачи

Была выбрана задача категоризации текста. Категоризация текста - это фундаментальная задача обработки естественного языка, целью которой является отнесение входного текста к заранее определенным категориям, и широко используется в таких областях, как обнаружение спама, анализ настроения и классификация тем.

2. Исследуемые статьи

Статья 1: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Основные достижения:

Двунаправленный кодер-трансформер: Предлагается двунаправленный кодер-трансформер - инновация, которая позволяет модели изучать языковое представление с обеих сторон контекста. В отличие от традиционных однонаправленных моделей, двунаправленные модели могут лучше передавать контекстную информацию.

Неконтролируемые задачи предварительного обучения: Введены две неконтролируемые задачи предварительного обучения: Masked Language Model (MLM) и Next Sentence Prediction (NSP), где MLM заставляет модель понимать контекст, маскируя часть словарного запаса и позволяя модели предсказывать эти слова, а NSP заставляет модель понимать контекст путем NSP помогает

модели понять межпредметные связи, определяя, встречаются ли два предложения последовательно или нет.

Крупномасштабное предварительное обучение: предварительное обучение на массивных текстовых данных (например, Wikipedia и BooksCorpus) значительно повышает обобщающую способность модели.

Методы тонкой настройки: предложен метод тонкой настройки предварительно обученных моделей на конкретные задачи, который показывает хорошие результаты на различных задачах обработки естественного языка.

Методология:

Архитектура модели: BERT основана на архитектуре Transformer, которая использует несколько слоев кодирующих устройств Transformer для обработки входной текстовой последовательности.

Модель маскированного языка (MLM): случайным образом маскирует некоторые слова во входном тексте и позволяет модели предсказывать эти маскированные слова, что позволяет модели изучать контекстуальные отношения между словами.

Предсказание следующего предложения (Next Sentence Prediction, NSP): учитывая пару предложений, модель должна определить, является ли второе предложение преемником первого, что позволяет модели понять связь между предложениями.

Предварительное обучение и тонкая настройка: предварительное обучение проводится сначала на крупномасштабных немаркированных данных, а затем тонкая настройка проводится на данных, специфичных для конкретной задачи.

Оценка:

Производительность: BERT достигает самых высоких результатов в ряде задач обработки естественного языка, особенно в бенчмарках GLUE, системах вопросов и ответов SQuAD и анализе настроений.

Влияние: Предложение BERT значительно продвинуло область обработки естественного языка и послужило основой для многих последующих исследований. Его двунаправленный кодер и система предварительного обучения и точной настройки стали стандартной практикой.

Применение: BERT широко используется в различных реальных приложениях, таких как поисковые системы, диалоговые системы и классификация текстов.

Статья 2: RoBERTa: A Robustly Optimized BERT Pretraining Approach

Основные достижения:

Оптимизация предварительного обучения: RoBERTa улучшает процесс предварительного обучения BERT с помощью ряда стратегий оптимизации, включая увеличение объема данных для предварительного обучения, удаление задачи предсказания следующего предложения, увеличение количества шагов обучения и стратегии динамической маскировки.

Большие массивы данных: использование больших массивов данных для предварительного обучения позволяет модели изучать более богатые языковые представления.

Количество шагов обучения: значительное увеличение количества шагов предварительного обучения демонстрирует, что более длительное время обучения может значительно улучшить производительность модели.

Динамическое маскирование: маскирование слов динамически на каждом этапе обучения, а не статически, обеспечивает более разнообразную обучающую выборку.

Методология:

Предварительное обучение на крупномасштабных данных: RoBERTa предварительно обучена на 160 ГБ текстовых данных, включая BooksCorpus, английскую Википедию, CC-News, OpenWebText и Stories.

Удаление задачи "Предсказание следующего предложения": из BERT удалена задача "Предсказание следующего предложения" (NSP), которая, как

было показано, вносит незначительный вклад в эффект предварительного обучения.

Увеличение количества шагов обучения: Увеличение количества шагов предварительного обучения с 1 миллиона в BERT до 5 миллионов в RoBERTa позволяет модели более полно изучить языковое представление.

Динамическая маскировка: динамическая генерация входных образцов с маской на каждой итерации вместо их статической генерации на этапе предварительной обработки обеспечивает большее разнообразие данных.

Оценка:

Производительность: RoBERTa превосходит BERT на нескольких эталонных наборах данных (например, GLUE, SQuAD), демонстрируя важность стратегий предварительного обучения.

Влияние: Предложение RoBERTa подтверждает критическое влияние объема данных предварительного обучения и времени обучения на производительность модели, а также задает новое направление для последующих исследований.

Применение: RoBERTa широко используется в различных задачах обработки естественного языка, таких как категоризация текстов, системы вопросов и ответов, генерация текстов и т.д. Предложенная модель широко используется в области категоризации текстов, систем вопросов и ответов, генерации текстов и т.д.

3. Практический компонент

Модели BERT и RoBERTa были загружены, и эксперименты проводились на наборе данных для классификации SMS. Мы вычислили точность, прецизионность, отзыв и F1 score моделей на тестовом наборе. Практический код приведен ниже:

Установка библиотеки и загрузка наборов данных для обработки:

```
!pip install transformers[torch]
!pip install accelerate -U
import pandas as pd
import torch
from transformers import RobertaTokenizer, RobertaForSequenceClassification
from transformers import Trainer, TrainingArguments
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_recall_fscore_support
# 加载数据集
url = "https://raw.githubusercontent.com/justmarkham/pycon-2016-tutorial/master/data/sms.tsv"
df = pd.read_csv(url, sep='\t', header=None, names=['label', 'message'])
# 数据预处理
df['label'] = df['label'].map({'ham': 0, 'spam': 1})
X = df['message']
y = df['label']
# 划分训练集和测试集
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

BERT:

```
# 加载 BERT tokenizer 和模型
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained('bert-base-uncased')
# 数据集类
class SMSDataset(torch.utils.data.Dataset):
    def __init__(self, texts, labels, tokenizer, max_len):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_len = max_len
    def __len__(self):
        return len(self.texts)
    def __getitem__(self, index):
        text = self.texts.iloc[index]
        label = self.labels.iloc[index]
        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=self.max_len,
            return_token_type_ids=False,
```

```
            padding='max_length',
            truncation=True,
            return_attention_mask=True,
            return_tensors='pt',
        )
        return {
            'text': text,
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'label': torch.tensor(label, dtype=torch.long)
        }
# 创建数据集
train_dataset = SMSDataset(X_train, y_train, tokenizer, max_len=128)
test_dataset = SMSDataset(X_test, y_test, tokenizer, max_len=128)
# 定义训练参数
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=3,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=32,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
```

```
)

# 计算指标

def compute_metrics(pred):

    labels = pred.label_ids

    preds = pred.predictions.argmax(-1)

    precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='binary')

    acc = accuracy_score(labels, preds)

    return {

        'accuracy': acc,

        'precision': precision,

        'recall': recall,

        'f1': f1,

    }

# 创建 Trainer

trainer = Trainer(

    model=model,

    args=training_args,

    train_dataset=train_dataset,

    eval_dataset=test_dataset,

    compute_metrics=compute_metrics

)

# 训练模型

trainer.train()
```

```
# 评估模型
results = trainer.evaluate()
print(results)
```

RoBERTa：

```
# 加载 RoBERTa tokenizer 和模型
tokenizer = RobertaTokenizer.from_pretrained('roberta-base')
model = RobertaForSequenceClassification.from_pretrained('roberta-base')
# 数据集类
class SMSDataset(torch.utils.data.Dataset):
    def __init__(self, texts, labels, tokenizer, max_len):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_len = max_len
    def __len__(self):
        return len(self.texts)
    def __getitem__(self, index):
        text = self.texts.iloc[index]
        label = self.labels.iloc[index]
        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
```

```
                max_length=self.max_len,
                return_token_type_ids=False,
                padding='max_length',
                truncation=True,
                return_attention_mask=True,
                return_tensors='pt',
            )
            return {
                'text': text,
                'input_ids': encoding['input_ids'].flatten(),
                'attention_mask': encoding['attention_mask'].flatten(),
                'label': torch.tensor(label, dtype=torch.long)
            }
# 创建数据集
train_dataset = SMSDataset(X_train, y_train, tokenizer, max_len=128)
test_dataset = SMSDataset(X_test, y_test, tokenizer, max_len=128)
# 定义训练参数
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=3,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=32,
    warmup_steps=500,
```

```
        weight_decay=0.01,
        logging_dir='./logs',
    )
    # 计算指标
    def compute_metrics(pred):
        labels = pred.label_ids
        preds = pred.predictions.argmax(-1)
        precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='binary')
        acc = accuracy_score(labels, preds)
        return {
            'accuracy': acc,
            'precision': precision,
            'recall': recall,
            'f1': f1,
        }
    # 创建 Trainer
    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=test_dataset,
        compute_metrics=compute_metrics
    )
```

```
# 训练模型

trainer.train()

# 评估模型

results = trainer.evaluate()

print(results)
```

4. Экспериментальные результаты и анализ

	eval_accuracy	eval_precision	eval_recall	eval_f1
BERT	0.9711	0.9134	0.9510	0.9533
RoBERTa	0.9910	0.9484	0.9866	0.9671

RoBERTa лучше справляются с набором данных.

5. Список использованных источников

[1] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[2] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692.

[3] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. arXiv preprint arXiv:1906.08237.

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30, 5998-6008.

[5] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.