NLP 第三次作业技术报告

基于部署的大语言模型完成文本分词任务

2025-12-25

尹超
中国科学院大学
University of Chinese Academy of Sciences
2025
2313AI
Template:https://github.com/hzkonor/bubble-template
GitHub 地址:https://github.com/CarterYin/NLP_UCAS_2025

目录

I. Chapter 1 Introduction

本次作业旨在探究大语言模型（LLM）在中文分词任务上的能力。利用北京大学标注的《人民日报》1998 年 1 月份的分词语料作为基准，借助外部网站部署的大语言模型 API（Qwen2.5 系列），完成以下任务：

1. 基准测试：测试大语言模型在随机抽取的 50 个《人民日报》句子上的平均分词性能。
2. 领域适应性：从网上爬取新闻、社交媒体（豆瓣）、科普文章（果壳）等不同类型的文本，测试模型在不同领域数据上的泛化能力。
3. 模型规模影响：对比分析不同参数量（7B, 14B, 32B, 72B）的模型对分词性能的影响。
4. 性能提升方法：提出并实验 Few-shot Prompting（少样本提示）方法，以提升模型的分词性能。

II. Chapter 2 Data Collection

本次作业使用的语料分为两部分：基准语料和多领域测试语料。

II.1. 2.1 基准语料 (Baseline)

使用《人民日报》1998 年 1 月份的分词语料（PKU Corpus）。该语料是中文分词任务的标准数据集，包含约 100 万词。我们从中随机抽取了 50 个句子作为测试集，用于评估模型的基准性能。

II.2. 2.2 多领域测试语料 (Multi-domain)

为了测试模型的泛化能力，我们构建了一个包含三个不同领域的测试集。数据的获取主要通过编写 Python 爬虫脚本（`q2/crawl_data.py`）实现。

II.2.1. 2.2.1 爬虫实现 (Crawler Implementation)

我们实现了一个 `NLPCleanScraper` 类，封装了针对不同网站的爬取和清洗逻辑。

1. 新闻语料 (News):
   - 来源: 中国新闻网 (ChinaNews) 的滚动新闻页面。
   - 逻辑: 构造按日期遍历的 URL（`scroll-news/{date}/news.shtml`），使用 `BeautifulSoup` 解析 HTML，提取 `.content_list` 下的新闻标题。
   - 特点: 文本规范，包含大量政治、经济术语。
2. 社交语料 (Social):
   - 来源: 豆瓣小组 (Douban Group) 的精选话题列表。
   - 逻辑: 遍历 `explore` 页面的分页链接，提取话题标题。为了应对反爬虫机制，设置了随机延时（`random.uniform`）和 User-Agent 伪装。
   - 特点: 口语化严重，包含网络流行语（如“安利”、“种草”）、表情符号和不规范标点。
3. 科普语料 (Kepu):
   - 来源: 果壳网 (Guokr) 的科学人栏目。
   - 逻辑: 采用广度优先搜索 (BFS) 策略。从栏目首页出发，提取页面中的文章链接加入队列，并解析文章页面的正文内容（`<p>` 标签）。
   - 特点: 篇幅较长，逻辑严密，包含大量专业名词（如生物学名、物理学术语）。

II.2.2. 2.2.2 数据清洗 (Data Cleaning)

所有爬取的原始文本均经过 `clean_text` 函数处理，以确保数据质量：

- 去除 HTML 实体: 如 ` ` 等。
- 字符过滤: 使用正则表达式 `[^\u4e00-\u9fa5a-zA-Z0-9,。！？、“”]`，仅保留汉字、数字、字母和常用中文标点，去除特殊符号和乱码。
- 去重与长度过滤: 剔除重复内容和长度小于 8 个字符的短句（通常为广告或导航链接）。

最终，我们从每个领域随机抽取了部分样本，并以 Jieba 分词结果作为参考标准（Reference）进行评估。

III. Chapter 3 Model Implementation

本次实验主要使用 Qwen2.5 系列模型，通过 SiliconFlow API 进行调用。

III.1. 3.1 模型选择

为了探究参数量对性能的影响，我们选择了以下四个版本的模型：

- Qwen/Qwen2.5-7B-Instruct：轻量级模型，推理速度快。
- Qwen/Qwen2.5-14B-Instruct：中等规模模型。
- Qwen/Qwen2.5-32B-Instruct：中大规模模型，平衡了性能与资源消耗。
- Qwen/Qwen2.5-72B-Instruct：大规模模型，具备更强的语义理解和指令遵循能力。

III.2. 3.2 提示词设计 (Prompt Engineering)

我们采用了两种提示策略：

1. Zero-shot (零样本)：直接要求模型对文本进行分词，不提供示例。
   - Prompt: “请对下面的中文句子进行分词，词与词之间用空格分隔：[Input Text]”
2. Few-shot (少样本)：在提示词中提供 3 个正确分词的示例，帮助模型理解分词规范（如颗粒度）。

III.3. 3.3 关键代码逻辑 (Key Implementation Logic)

为了确保实验的严谨性和可复现性，我们在代码实现中包含了以下关键模块：

1. 数据预处理 (Data Preprocessing): 针对 PKU 语料（GBK 编码），我们实现了 `load_pku_corpus` 函数。该函数利用正则表达式（Regex）清洗原始文本：
   - 去除词性标记（如 `/v`，`/n`）。
   - 去除复合词标记（如 `[ ]nt`）。

   最终提取出纯净的原始文本（Raw Text）和标准分词结果（Ground Truth）。
2. API 交互 (API Interaction): 通过 `get_llm_segmentation` 函数封装了大模型的调用逻辑。我们使用了 `openai` Python SDK 连接 SiliconFlow API。为了保证结果的确定性，我们将 `temperature` 参数设置为 `0.1`。该函数支持动态插入 Few-shot 示例，根据传入的 `few_shot_examples` 列表自动构建包含上下文的 Prompt。
3. 指标计算 (Metric Calculation): 在 `calculate_metrics` 函数中，我们采用了基于字符区间 (Character Intervals) 的匹配算法。
   - 将标准答案和模型预测结果分别转换为字符索引区间集合(例如:“北京”在句首转换为 `(0, 2)`)。

- 通过计算两个集合的交集（Intersection）来确定真正例（True Positives, TP）。
- 这种方法避免了直接比较字符串列表可能带来的对齐问题，确保了 Precision、Recall 和 F1 计算的准确性。

III.4. 3.4 评估指标

使用标准的 Precision (精确率)、Recall (召回率) 和 F1-Score 作为评估指标。

$$P = \frac{N_{\text{correct}}}{N_{\text{pred}}}, \quad R = \frac{N_{\text{correct}}}{N_{\text{gold}}}, \quad F1 = \frac{2 \times P \times R}{P + R}$$

其中，N_{correct} 为模型预测正确的词数，N_{pred} 为模型预测的总词数，N_{gold} 为标准答案的总词数。

IV. Chapter 4 Results and Discussion

IV.1. 4.1 基准测试结果 (Baseline)

在 PKU 语料上的测试结果如下：

Table 1 — PKU 语料基准测试结果

Precision	Recall	F1 Score
0.5971	0.7110	0.6491

分析：F1 分数（0.6491）低于预期。通过错误分析发现，主要原因是 ID 号的处理。PKU 语料中每句话前都有一个长 ID（如 `19980118-01-001-018`），标准答案将其视为一个 Token，而大模型倾向于将其拆分为多个数字和符号（如 `1998 01 18 ...`）。这种格式上的不一致严重拉低了 Precision。去除 ID 影响后，模型对正文的分词效果实际上较好。

IV.2. 4.2 领域适应性分析 (Domain Adaptation)

在不同领域语料上的测试结果如下（以 Jieba 分词为参考）：

Table 2 — 不同领域的性能对比

Domain	Precision	Recall	F1 Score
News (新闻)	0.7645	0.8186	0.7906
Social (社交)	0.7352	0.8087	0.7702
Kepu (科普)	0.4005	0.3276	0.3604

分析：

1. 新闻与社交领域：模型表现较好（F1 > 0.77）。在社交媒体文本中，模型能够正确识别网络用语（如“ins”、“安利”），显示出良好的泛化能力。
2. 科普领域：F1 分数显著偏低（0.3604）。这并非模型能力不足，而是分词颗粒度的差异。

- 日期与数字：Jieba 倾向于将日期切分得非常细（如 `2019 年 9 月`），而 LLM 倾向于将其合并（如 `2019年 9月`）。
- 专有名词：对于长实体（如 `UniversityofAlberta`），LLM 往往保持整体，而 Jieba 可能因未登录词而切分错误或过度切分。
- 由于科普文章篇幅长，这种颗粒度差异在全文中累积，导致指标大幅下降。

IV.3. 4.3 模型参数量的影响 (Model Size)

对比不同参数量的 Qwen2.5 模型在同一数据集上的表现：

Table 3 — 不同参数量模型的性能对比

Model	Precision	Recall	F1 Score
7B-Instruct	0.6580	0.7823	0.7148
14B-Instruct	0.2510	0.2553	0.2532
32B-Instruct	0.8128	0.9054	0.8566
72B-Instruct	0.7126	0.8385	0.7705

分析：

1. 32B 模型表现最佳：F1 分数达到 0.8566，显著优于其他模型，表明该规模的模型在中文分词任务上达到了较好的平衡，具备优秀的语义理解和指令遵循能力。
2. 14B 模型表现异常：F1 分数仅为 0.2532，远低于 7B 模型。经检查，该模型在遵循“仅输出分词结果”的指令上存在问题，输出了多余的解释性文字或未能正确使用空格分隔，导致自动评估失败。
3. 72B 模型未达预期：虽然参数量最大，但其性能（F1=0.7705）反而低于 32B 模型。这说明单纯增加参数量并不总是能线性提升特定任务的性能，或者在 Zero-shot 设置下，超大模型对提示词的敏感度不同。
4. 7B 模型性价比高：作为轻量级模型，其 F1 分数（0.7148）尚可，且推理速度最快，适合资源受限的场景。

IV.4. 4.4 提示工程的效果 (Prompting)

对比 Zero-shot 和 Few-shot（3-shot）的性能：

Table 4 — Zero-shot 与 Few-shot 性能对比

Method	Precision	Recall	F1 Score
Zero-shot	0.6517	0.7622	0.7026
Few-shot	0.8799	0.8716	0.8757

分析： Few-shot 方法带来了巨大的性能提升（F1 从 0.70 提升至 0.87）。

- 规范对齐：示例帮助模型理解了期望的分词颗粒度（例如，将“北京大学”视为一个词还是两个词）。
- 格式规范：示例强化了输出格式的要求，减少了格式错误的发生。

这说明在分词这种定义灵活的任务中，提供示例是比单纯增加模型参数更高效的提升手段。

V. Chapter 5 Conclusion

本实验通过多维度评估，得出以下结论：

1. 大模型具备强大的分词能力：即使是 7B 的小模型，在 Zero-shot 下也能达到尚可的效果。
2. 领域泛化能力强：在非正式的社交媒体文本上表现出色，优于传统基于规则的方法。
3. 颗粒度对齐是关键：LLM 与传统分词标准（如 PKU、Jieba）的主要差距在于分词颗粒度（日期、实体）。
4. Few-shot 至关重要：通过少量示例进行 In-Context Learning，可以显著对齐颗粒度，大幅提升分词性能（F1 提升约 17%）。

未来工作可以探索利用思维链（CoT）让模型处理更复杂的歧义切分，或通过微调（Fine-tuning）进一步对齐特定领域的分词标准。

VI. Appendix: Code Listings

代码仓库地址：https://github.com/CarterYin/NLP_UCAS_2025

以下是本次实验的核心代码实现。

VI.1. A.1 核心工具库 (utils.py)

该文件封装了通用的功能，包括加载语料、调用 LLM API 以及计算评估指标。

```
import os
import re
import json
from openai import OpenAI

# 配置 API Key 和 Base URL
API_KEY = os.getenv("LLM_API_KEY", "your_api_key_here")
BASE_URL = os.getenv("LLM_BASE_URL", "https://api.siliconflow.cn/v1")
MODEL_NAME = os.getenv("LLM_MODEL", "Qwen/Qwen2.5-7B-Instruct")

client = OpenAI(api_key=API_KEY, base_url=BASE_URL)

def get_llm_segmentation(text, model=MODEL_NAME, few_shot_examples=None):
    """
    调用 LLM 进行分词。
    支持 Zero-shot 和 Few-shot 模式。
    """
    prompt = "请对以下中文句子进行分词，词与词之间用空格分隔。直接输出分词结果，不要包含任何解释或其他内容。\n"

    if few_shot_examples:
        prompt += "示例：\n"
        for ex_raw, ex_seg in few_shot_examples:
            prompt += f"输入：{ex_raw}\n输出：{ex_seg}\n"
```

```
            prompt += "\n"

        prompt += f"输入: {text}\n输出: "

        try:
            response = client.chat.completions.create(
                model=model,
                messages=[
                    {"role": "system", "content": "你是一个专业的中文分词工具。"},
                    {"role": "user", "content": prompt}
                ],
                temperature=0.1, # 低温度以保证确定性
            )
            return response.choices[0].message.content.strip()
        except Exception as e:
            print(f"API调用失败: {e}")
            return text

def calculate_metrics(ground_truth_list, prediction_list):
    """
    计算分词的 Precision, Recall, F1。
    使用区间匹配算法 (Interval Matching) 确保准确性。
    """
    tp = 0
    gold_total = 0
    pred_total = 0

    for gold_words, pred_words in zip(ground_truth_list, prediction_list):
        def get_intervals(words):
            intervals = set()
            start = 0
            for w in words:
                end = start + len(w)
                intervals.add((start, end))
                start = end
            return intervals

        gold_intervals = get_intervals(gold_words)
        pred_intervals = get_intervals(pred_words)

        tp += len(gold_intervals & pred_intervals)
        gold_total += len(gold_intervals)
        pred_total += len(pred_intervals)

    precision = tp / pred_total if pred_total > 0 else 0
    recall = tp / gold_total if gold_total > 0 else 0
    f1 = 2 * precision * recall / (precision + recall) if (precision + recall)
> 0 else 0

    return precision, recall, f1
```

```
def load_pku_corpus(filepath):
    """
    加载并清洗人民日报语料 (PKU Corpus)。
    去除词性标记和复合词括号。
    """
    data = []
    try:
        with open(filepath, 'r', encoding='gbk') as f:
            for line in f:
                line = line.strip()
                if not line: continue
                # 清洗逻辑：去除 [ ]nt 和 /v 等标记
                line = re.sub(r'\[|\][a-z]*', '', line)
                tokens = line.split()
                words = []
                for token in tokens:
                    if '/' in token:
                        word = token.split('/')[0]
                        if word: words.append(word)
                if words:
                    data.append(("".join(words), words))
    except Exception as e:
        print(f"Error loading corpus: {e}")
    return data
```

VI.2. A.2 基准测试脚本 (q1/eval_pku.py)

该脚本用于在 PKU 语料上进行 Zero-shot 基准测试。

```
import sys
import os
import random
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
from utils import load_pku_corpus, get_llm_segmentation, calculate_metrics

def main():
    corpus_path = os.path.join("..", "clean_data", "ChineseCorpus199801.txt")
    data = load_pku_corpus(corpus_path)

    # 随机抽取 50 条样本
    random.seed(42)
    samples = random.sample(data, 50)

    ground_truth = []
    predictions = []

    for raw_text, gold_words in samples:
        # 调用 LLM
        seg_result = get_llm_segmentation(raw_text)
        pred_words = seg_result.split()

        ground_truth.append(gold_words)
```

```
        predictions.append(pred_words)

    # 计算指标
    p, r, f1 = calculate_metrics(ground_truth, predictions)
    print(f"Precision: {p:.4f}, Recall: {r:.4f}, F1: {f1:.4f}")

if __name__ == "__main__":
    main()
```

VI.3. A.3 领域适应性实验脚本 (q2/eval_crawled.py)

该脚本用于评估模型在爬取的多领域语料上的表现，以 Jieba 分词作为参考。

```
import sys
import os
import random
import jieba
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
from utils import get_llm_segmentation, calculate_metrics

def load_crawled_data(filepath, limit=20):
    """读取爬取的清洗后数据"""
    lines = []
    if os.path.exists(filepath):
        with open(filepath, 'r', encoding='utf-8') as f:
            for line in f:
                line = line.strip()
                if len(line) > 10: lines.append(line)
    if len(lines) > limit:
        return random.sample(lines, limit)
    return lines

def main():
    base_dir = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
    data_dir = os.path.join(base_dir, "clean_data")

    files = {
        "News": "corpus_news_clean.txt",
        "Social": "corpus_social_clean.txt",
        "Kepu": "corpus_kepu_clean.txt"
    }

    for type_name, filename in files.items():
        filepath = os.path.join(data_dir, filename)
        samples = load_crawled_data(filepath)

        jieba_refs = []
        llm_preds = []

        for raw_text in samples:
            # Jieba 分词作为伪标签 (Pseudo-Gold)
            jieba_seg = list(jieba.cut(raw_text))
            jieba_refs.append(jieba_seg)
```

```
            # LLM 分词
            llm_res = get_llm_segmentation(raw_text)
            llm_preds.append(llm_res.split())

        p, r, f1 = calculate_metrics(jieba_refs, llm_preds)
        print(f"{type_name} - Agreement with Jieba: F1={f1:.4f}")

if __name__ == "__main__":
    main()
```

VI.4. A.4 数据爬取脚本 (q2/crawl_data.py)

该脚本实现了针对新闻、社交媒体和科普网站的爬虫。

```
import requests
from bs4 import BeautifulSoup
import os
import time
import random
import re
from datetime import datetime, timedelta
import collections

class NLPCleanScraper:
    def __init__(self, storage_path):
        self.storage_path = storage_path
        self.session = requests.Session()
        self.headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) ...'
        }
        self.session.headers.update(self.headers)
        self.history_set = set()

    def clean_text(self, text):
        """清洗文本：去除HTML实体和非中文字符"""
        text = re.sub(r'&[a-z]+;|&#\d+;', '', text)
        clean_pattern = re.compile(r'[^\u4e00-\u9fa5a-zA-Z0-9,。！？、“”]')
        text = clean_pattern.sub('', text)
        return text.strip()

    def crawl_news(self, total_target, start_date_str):
        """爬取中国新闻网滚动新闻"""
        current_date = datetime.strptime(start_date_str, "%Y%m%d")
        count = 0
        while count < total_target:
            date_path = current_date.strftime("%Y/%m%d")
            url = f"https://www.chinanews.com.cn/scroll-news/{date_path}/
news.shtml"
            try:
                res = self.session.get(url, timeout=10)
                res.encoding = 'utf-8'
                if res.status_code == 200:
```

```
                    soup = BeautifulSoup(res.text, 'html.parser')
                    links = soup.select('.content_list .dd_bt a')
                    for l in links:
                        cleaned = self.clean_text(l.get_text())
                        if len(cleaned) > 8:
                            self._save_line(cleaned, "corpus_news_clean.txt")
                            count += 1
            except Exception as e:
                print(f"Error: {e}")
            current_date -= timedelta(days=1)

    def crawl_social(self, total_target):
        """爬取豆瓣小组话题"""
        count = 0
        for i in range(0, 500):
            url = f"https://www.douban.com/group/explore?start={i*25}"
            # ... (省略部分请求逻辑，同上) ...

    def crawl_guokr(self, total_target):
        """爬取果壳网科普文章 (BFS)"""
        base_url = "https://www.guokr.com/scientific/"
        queue = collections.deque([base_url])
        # ... (省略 BFS 逻辑) ...
```

VI.5. A.5 模型对比脚本 (q3/compare_models.py)

该脚本用于对比不同参数量的模型性能。

```
import sys
import os
import random
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
from utils import load_pku_corpus, get_llm_segmentation, calculate_metrics

def main():
    models = [
        "Qwen/Qwen2.5-7B-Instruct",
        "Qwen/Qwen2.5-14B-Instruct",
        "Qwen/Qwen2.5-32B-Instruct",
        "Qwen/Qwen2.5-72B-Instruct",
    ]

    corpus_path = os.path.join("..", "clean_data", "ChineseCorpus199801.txt")
    data = load_pku_corpus(corpus_path)
    random.seed(42)
    samples = random.sample(data, 50)

    for model_name in models:
        print(f"Testing {model_name}...")
        ground_truth = []
        predictions = []

        for raw_text, gold_words in samples:
```

```
            seg_result = get_llm_segmentation(raw_text, model=model_name)
            ground_truth.append(gold_words)
            predictions.append(seg_result.split())

        p, r, f1 = calculate_metrics(ground_truth, predictions)
        print(f"{model_name}: F1={f1:.4f}")

if __name__ == "__main__":
    main()
```

VI.6. A.6 性能提升实验脚本 (q4/improve_segmentation.py)

该脚本实现了 Few-shot Prompting，通过在提示词中加入示例来提升性能。

```
import sys
import os
import random
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
from utils import load_pku_corpus, get_llm_segmentation, calculate_metrics

def main():
    corpus_path = os.path.join("..", "clean_data", "ChineseCorpus199801.txt")
    data = load_pku_corpus(corpus_path)
    random.seed(42)

    # 划分测试集和示例集
    test_samples = random.sample(data, 50)
    remaining = [d for d in data if d not in test_samples]
    few_shot_examples = random.sample(remaining, 3) # 随机选取3个示例

    # 格式化示例
    formatted_examples = [(raw, " ".join(words)) for raw, words in
few_shot_examples]

    # 运行 Few-shot 实验
    gt_few = []
    pred_few = []
    for raw, gold in test_samples:
        # 传入示例
        res = get_llm_segmentation(raw, few_shot_examples=formatted_examples)
        gt_few.append(gold)
        pred_few.append(res.split())

    p3, r3, f1_3 = calculate_metrics(gt_few, pred_few)
    print(f"Few-shot F1: {f1_3:.4f}")

if __name__ == "__main__":
    main()
```