中国科学院大学
University of Chinese Academy of Sciences

# 自然语言处理课程实验报告

## word 文档多音字与易错字的自动注音


小组成员 (排名不分先后):

曾楠馨, 江震南, 张志成, 左斌斌,

柏天佑, 孟鑫攀, 朱英健, 孟鑫攀,

郝锐, 黄少平, 龚子俊, 沙霖


2024 年 10 月 2 日

# 目录

# 1 摘要

# 2 研究背景

信息技术的迅猛发展引领人类社会迈向信息化与智能化新时代，数据呈指数级增长。此背景下，自然语言处理（NLP）作为人机沟通的纽带，其重要性与日俱增。从计算机科学分支跃升至发掘非结构化文本数据潜能、促信息处理高效化和智能化的核心技术，NLP 的发展伴随互联网、社交媒体、数字文献等产生的海量数据激增及计算能力建设的飞越，实现了前所未有的技术突破与应用拓宽。

汉语文本处理中，多音字与易错字构成重大障碍。手工筛查耗时且难以保证准确，尤其在大规模文档中，效率与精确性两难求解，愈发突显自动化处理工具的迫切需求。

与此同时，人工智能技术，特别是深度学习领域的飞速进步，如深度神经网络的广泛部署，极大地增强了机器理解和处理自然语言的能力。这些进展不仅仅优化了语音识别、机器翻译等领域，亦为解决自动标注汉字读音等细粒度任务铺平了道路，展现了技术在特定应用场景中的实用性与潜力。

# 3 研究动机

本文旨在利用自然语言处理技术，实现对 Word 文档中多音字与易错字的自动标注。通过构建多音字与易错字的标注模型，实现对文档中多音字与易错字的自动标注，提高文档处理效率，减少人工标注工作量。研究动机分为以下两个方面：

## 3.1 实际需求

自动注音系统的研发，其首要动机在于解决现实生活中的实际问题，提高人们处理文本的准确性和效率。在演讲准备、教材制作、语言教学、有声书录制等领域，自动为文档加上正确的注音能够显著降低错误发生率，确保发音准确无误，如演讲者念稿时避免误读多音字，确保传达的信息清晰无误。这不仅提升了交流的专业度，还增强了信息接收者的理解体验，使得语言文化传播更为流畅和准确。

此外，在教育、科学研究、新闻媒体、法律、医疗健康等领域，文档处理需求日益增长，对文档内容的准确性和专业性提出更高要求。然而，由于汉语的独特性质，如一字多音、同音异义现象的存在，使得中文文档处理面临特别的挑战。手动校对多音字和易错字费时费力，特别是在处理大规模文档时，效率和准确性难以保障。这种情况下，自动化文本处理工具的需求显得尤为迫切。

为了便于实际使用，我组成员还设计了用户交互界面，使得用户可以方便地上传文档，即使不会写任何程序，也可以轻松地使用我们的系统进行多音字的自动注音。

### 3.2 能力提升

通过这个项目，我们能够将课堂上学到的自然语言处理理论知识与实际问题解决结合起来，加深了对 NLP 技术本质的理解，如词法分析、语义理解、机器学习模型等概念不再是抽象的文字，而是变成了具体实现代码和技术选型的决策过程。通过处理多音字和易错字的注音任务，我们能够更好地掌握汉语语言特性与语法规则，从而加深对汉语语言学的理解。这个过程需要我们不断思考如何在复杂的语言环境中精确识别多音字的读音，并通过各种算法策略进行纠错。这不仅提升了我们的理论知识，还增强了在实际应用场景中灵活运用所学知识的能力。

在项目的实践中，我们有机会接触到多种 NLP 技术，例如分词、词性标注、语音合成等。我们使用 python 和 pytorch 编写程序，git 进行代码协作与管理，提升了团队成员之间的协作能力，也让我们在编程技能和工程实践上得到了很大的提升。每个阶段的方案讨论和调整，都是对我们逻辑思维、问题分析和解决能力的综合考验。

此外，这个作业让我们更深入地体会到了科研的严谨性与创新性。为了实现高精度的注音功能，我们需要广泛查阅文献，分析现有的多音字处理方案，并训练了自己的模型。在这个过程中，我们不仅学会了如何全面理解一个研究问题，还培养了在面对复杂问题时，能够从多角度提出创新性解决方案的能力。为使工具易于使用且效果直观，在 ui 界面时我们不得不站在用户的角度思考，提升了我们的产品设计思维。

## 4 技术路线

### 4.1 使用 pypinyin 库

python-pinyin 是一个用于将汉字转换为拼音的 python 库, 其开源地址为https://github.com/mozillazg/python-pinyin。它支持多种拼音风格，包括声调和无声调。我们可以使用它来实现对文档中多音字的自动标注。

```
>>> from pypinyin import pinyin, lazy_pinyin, Style
>>> pinyin('中心')  # or pinyin(['中心']), 参数值为列表时表示输入的是已分词后的数据
[['zhōng'], ['xīn']]
>>> pinyin('中心', style=Style.TONE2, heteronym=True)
[['zho1ng', 'zho4ng'], ['xi1n']]
>>> lazy_pinyin('中心')  # 不考虑多音字的情况
['zhong', 'xin']
>>> lazy_pinyin('战略', v_to_u=True)  # 不使用 v 表示 ü
['zhan', 'lüe']
```

Code Listing 1: python-pinyin 的使用示例

**Algorithm 1:** 使用 pypinyin 库实现对 word 文档中多音字的自动标注

**Input:** 原 word 文档, 多音字字库 `polyphone_data`

**Output:** 注音后的 word 文档

```
读取 Word 文档内容，获取文本段落 paragraphs;
foreach paragraph ∈ paragraphs do
    foreach character ∈ paragraph do
        if character ∈ polyphone_data then
            添加多音字及其注音到输出段落;
        else
            直接添加字符到输出段落;
输出注音后的 word 文档;
```

以下是部分核心代码展示:

```python
def add_pinyin_to_polyphone_words(paragraphs, polyphone_data):
    polyphonic_chars = {item['char'] for item in polyphone_data}
    rare_chars = {item['char'] for item in rare_char_data}
    output_paragraphs = []

    for paragraph in paragraphs:
        pinyin_paragraph = pinyin(paragraph, style=Style.TONE)
        output_paragraph = ""
        char_index = 0
        for char in paragraph:
            if char in polyphonic_chars :
                output_paragraph += char + '(' + pinyin_paragraph[
                    char_index][0] + ')'
            elif char in rare_chars:
                output_paragraph += char + '[' + get_pinyin(char)[0] + ']'
            else:
                output_paragraph += char
            char_index += 1
        output_paragraphs.append(output_paragraph)
    return output_paragraphs
```

Code Listing 2: pypinyin 自动注音的部分核心代码展示

### 4.2 方法二

### 4.3 UI 设计

# 5 成果展示

# 6 小组分工

| 工作内容 | 成员名称 |
| --- | --- |
| 主逻辑代码实现 | 张志成，江震南 |
| 代码优化调整 | 左斌斌，鑫攀 |
| UI 代码实现 | 曾楠馨 |
| 报告撰写 | 朱英健，黄少平 |
| Pre | |
| 清洗和收集数据 | |