天池SMP 2023 ChatGLM金融大模型挑战赛

馒头科技团队方案代码 B/C榜 TOP1

主要代码说明

- main.py
  - 包含全流程, 数据下载/pdf表格提取/表格生成/问题分类/问题关键词提取/SQL生成/问答结果生成/提交生成
- generate_answer_with_classify.py
  - generate_answer函数对每类问题生成问答结果

主要思路

问题分类

SQL生成

问答结果生成

Type1

核心处理代码

```
table_dict = {
    'A': ['basic_info'],
    'B': ['employee_info', 'dev_info'],
    'C': ['cbs_info', 'cscf_info', 'cis_info'],
    'G': ['basic_info', 'employee_info', 'dev_info', 'cbs_info', 'cscf_info', 'cis_info']
}
if len(company_abbrs) == 0:
    logger.warning('匹配到了类别{}, 但是不存在报表'.format(question_type))
else:
    # _, question_keywords = question_util.parse_question_keywords(model, ori_question, real_comp, years)
    logger.info('问题关键词: {}'.format(question_keywords))

```

```
    background = ''
    tot_matched_rows = []
    for year in years:
        pdf_table = load_tables_of_years(company, [year], pdf_tables, pdf_info)

        background += '已知{}(简称:{},证券代码:{}){}年的资料如下:\n'.format(company, abbr, code, year)
        matched_table_rows = []
        for keyword in question_keywords:
            matched_table_rows.extend(recall_pdf_tables(keyword, [year], pdf_table,
                min_match_number=3, valid_tables=table_dict[question_type]))

        if len(matched_table_rows) == 0:
            for table_row in pdf_table:
                if table_row[0] in table_dict[question_type]:
                    matched_table_rows.append(table_row)

        table_text = table_to_text(real_comp, ori_question, matched_table_rows, with_year=False)
        background += table_text
        background += '\n'

        tot_matched_rows.extend(matched_table_rows)

    tot_matched_rows = add_text_compare_in_table(tot_matched_rows)
    tot_text = table_to_text(real_comp, ori_question, tot_matched_rows, with_year=True)

    if '相同' in tot_text or '不相同且不同' in tot_text:
        answer = tot_text
    else:
        question_for_model = type1.get_prompt(ori_question, company, abbr, years).format(background, ori_question)
        logger.info('Prompt length {}'.format(len(question_for_model)))
        if len(question_for_model) > 5120:
            question_for_model = question_for_model[:5120]
        logger.info(question_for_model.replace('<', ''))
        answer = model(question_for_model)
```

主要处理步骤:

1. 对于Type1每类问题, 从该问题对应的报表中根据关键词召回可能对应的行

2. 对于行为字符串类型的, add_text_compare_in_table实现对比不同年份的字段内容是否相同
3. 对于问题问是否相同的, 直接返回召回的报表
4. 对于其他的, 通过prompt组合召回的报表来传给模型进行回答, 具体prompt函数见type1.get_prompt

Type2

核心处理代码

```
if type2.is_type2_growth_rate(ori_question):
        years_of_table = []
        for year in years:
            years_of_table.extend([year, str(int(year)-1)])
        pdf_table = load_tables_of_years(company, years_of_table, pdf_tables, pdf_info)
        pdf_table = add_growth_rate_in_table(pdf_table)
    elif type2.is_type2_formula(ori_question):
        pdf_table = load_tables_of_years(company, years, pdf_tables, pdf_info)
    else:
        logger.error('无法匹配, 该问题既不是增长率也不是公式计算')
        pdf_table = load_tables_of_years(company, years, pdf_tables, pdf_info)

    step_questions, step_keywords, variable_names, step_years, formula, question_formula = type2.get_step_questions(
        ori_question, ''.join(question_keywords), real_comp, years[0])
    step_answers = []
    variable_values = []
    if len(step_questions) > 0:
        for step_question, step_keyword, step_year in zip(step_questions, step_keywords, step_years):
            if len(step_keyword) == 0:
                logger.error('关键词为空')

            background = '已知{}{}年的资料如下:\n'.format(real_comp, step_year)
            # background += '----------------------------------------\n'

            matched_table_rows = recall_pdf_tables(step_keyword, [step_year], pdf_table,
                min_match_number=3, top_k=5)
            # print(matched_table_rows)
```

```
            if len(matched_table_rows) == 0:
                logger.warning('无法匹配keyword {}, 尝试不设置限制'.format(step_keyword))
                matched_table_rows = recall_pdf_tables(step_keyword, [step_year], pdf_table,
                min_match_number=2, top_k=None)
            if len(matched_table_rows) == 0:
                logger.error('仍然无法匹配keyword {}'.format(step_keyword))
                matched_table_rows = recall_pdf_tables(step_keyword, [step_year], pdf_table,
                min_match_number=0, top_k=10)

            table_text = table_to_text(real_comp, ori_question, matched_table_rows, with_year=False)
            if table_text != '':
                background += table_text

            question_for_model = prompt_util.get_prompt_single_question(ori_question, real_comp, step_year).format(background, step_question)
            logger.opt(colors=True).info('<cyan>{}</>'.format(question_for_model.replace('<', '')))
            step_answer = model(question_for_model)
            variable_value = type2.get_variable_value_from_answer(step_answer)
            if variable_value is not None:
                step_answers.append(step_answer)
                variable_values.append(variable_value)
            logger.opt(colors=True).info('<green>{}</><red>{}</>'.format(step_answer.replace('<', ''), variable_value))
    if len(step_questions) == len(variable_values):
        for name, value in zip(variable_names, variable_values):
            formula = formula.replace(name, value)
        result = None
        try:
            result = eval(formula)
        except:
            logger.error('Eval formula {} failed'.format(formula))
        if result is not None:
            answer = ''.join(step_answers)
            answer += question_formula
            answer += '得出结果{:.2f}({:.2f}%)'.format(result, result*100)
```

主要处理步骤:

1. 如果是计算增长率, 则需要召回前一年的数据
2. type2.get_step_questions将需要计算的内容转换为多个提问, 得到计算公式中的每个元素, 传给模型进行回答后, 提取数值作为公式的单元, 最后通过python计算该公式

    a. 例如对于增长率, 公式为(A-B)/B, A表示当年的数值, B表示上年的数值

    b. 通过prompt_util.get_prompt_single_question生成针对A和B的提问

    c. 提取回答中的数值type2.get_variable_value_from_answer

    d. eval(formula)得到计算的结果

Type3

核心代码

```
anoy_question, _ = question_util.parse_question_keywords(model, ori_question, real_comp, years)
logger.info('问题关键词: {}'.format(question_keywords))

background = '***************{}{}年年报***************\n'.format(
    real_comp, years[0])
matched_text = recall_annual_report_texts(model, anoy_question, ''.join(question_keywords),
    mactched_pdf_names[0], None)
for block_idx, text_block in enumerate(matched_text):
    background += '{}片段:{}{}\n'.format('-'*15, block_idx+1, '-'*15)
    background += text_block
    background += '\n'
question_for_model = prompt_util.prompt_question_tp31.format(
    background, ori_question, ''.join(question_keywords),
    ''.join(question_keywords), ''.join(question_keywords))

logger.info('Prompt length {}'.format(len(question_for_model)))
if len(question_for_model) > 5120:
    question_for_model = question_for_model[:5120]
logger.info(question_for_model.replace('<', ''))
answer = model(question_for_model)
```

主要处理步骤:

1. question_util.parse_question_keywords提取问题中的关键词
2. recall_annual_report_texts进行文本的召回

    a. 其中召回的主要代码如下, 这里按行读取pdf文件中的文本内容, 采用bm25算法, 分别对问题关键词和原问题进行召回, 然后合并召回的文本块, 最后按照和原问题的字符匹配长度取匹配

度最高的文本块:

```
text_pages = load_pdf_pages(key)
text_lines = list(itertools.chain(*[page.split('\n') for page in text_pages]))
text_lines = [line for line in text_lines if len(line) > 0]
if len(text_lines) == 0:
    return []
model = fastbm25(text_lines)
result_keywords = model.top_k_sentence(keywords, k=3)
result_question = model.top_k_sentence(anoy_question, k=3)
top_match_indexes = [t[1] for t in result_question + result_keywords]
block_line_indexes = merge_idx(top_match_indexes, len(text_lines), 0, 30)

text_blocks = ['\n'.join([text_lines[idx] for idx in line_indexes]) for line_indexes in block_line_indexes]
# text_blocks = [filter_header_footer(text_block) for text_block in text_blocks]
text_blocks = [re.sub(' {3,}', '\t', text_block) for text_block in text_blocks]

text_blocks = [(t, SequenceMatcher(None, anoy_question, t, autojunk=False).find_longest_match().size) for t in text_blocks]
for text_block, match_size in text_blocks:
    match = SequenceMatcher(None, anoy_question, text_block, autojunk=False).find_longest_match()
    print(anoy_question[match.a: match.a + match.size])
max_match_size = max([t[1] for t in text_blocks])
text_blocks = [t[0] for t in text_blocks if t[1] == max_match_size]

if sum([len(t) for t in text_blocks]) > 2000:
    max_avg_len = int(2000 / len(text_blocks))
    text_blocks = [t[:max_avg_len] for t in text_blocks]

text_blocks = [rewrite_text_block(t) for t in text_blocks]
text_blocks = ['```\n{}\n```'.format(t) for t in text_blocks]
```

3. prompt_util.prompt_question_tp31组合召回的文本形成prompt传给模型进行回答