دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

پیشنهاد عملگرهای جهش در جاوا با استفاده از چت‌بات

Suggesting mutation operators in Java using a chatbot

پدیدآورندگان:

محمد امین کیانی

نسترن سرخوش

یزدان افرا

دانشجویان کارشناسی، دانشکده‌ی کامپیوتر، دانشگاه اصفهان، اصفهان،

استاد درس: جناب اقای دکتر شعرباف

نیمسال اول تحصیلی 04-1403

فهرست مطالب

مستندات

لینک ویدیو

https://github.com/M-Amin-Kiani/JavaParser-for-OOCode-with-AST

سیستم پیشنهاد عملگر

درباره Cohere

Cohere یک پلتفرم پردازش زبان طبیعی (NLP) است که مدل‌های مبتنی بر هوش مصنوعی را برای تولید متن، تحلیل داده‌های متنی، و بسیاری از کاربردهای دیگر ارائه می‌دهد و مدل‌های قدرتمندی مانند **command-xlarge-nightly** را ارائه می‌کند که برای تولید متن‌های پیچیده و دقیق طراحی شده‌اند.

کلید API چیست؟

کلید API یک کد منحصر به فرد است که برای احراز هویت کاربر در هنگام استفاده از سرویس‌های آنلاین مانند Cohere به کار می‌رود. این کلید تضمین می‌کند که تنها کاربران مجاز به سرویس دسترسی داشته باشند. در این پروژه، از کلید API برای ارتباط با مدل Cohere و دریافت پیشنهادات استفاده می‌شود.

اطلاعات درباره کلیدهای آزمایشی:(Trial keys)

1. **رایگان** :کلیدهای آزمایشی بدون هزینه هستند و برای آزمایش قابلیت‌های سرویس طراحی شده‌اند.
2. **محدودیت تعداد درخواست‌ها** :تعداد درخواست‌هایی که می‌توانید با این کلید ارسال کنید، محدود است. این محدودیت بستگی به سیاست‌های Cohere دارد (مثلاً تعداد مشخصی درخواست در روز یا ماه).
3. **عدم استفاده تجاری** :این کلیدها نمی‌توانند برای اهداف تجاری یا پروژه‌های واقعی استفاده شوند.
4. **تاریخ انقضا** :این کلیدها معمولاً محدود به مدت زمان مشخصی هستند (مثلاً چند هفته یا ماه). اگر تاریخ انقضای خاصی برای این کلید ذکر نشده باشد، باید مستندات Cohere یا تنظیمات حساب کاربری خود را بررسی کنید.

اگر محدودیت را پر کنید یا کلید منقضی شود:

- ممکن است سرویس متوقف شود و نیاز به ایجاد کلید جدید یا ارتقاء به **Production key** داشته باشید.

- برای استفاده تجاری و بدون محدودیت، باید یک کلید "Production" (کلید تولید) ایجاد کنید که نیاز به اشتراک یا پرداخت هزینه دارد.

کلیدهای آزمایشی Cohere محدودیت‌های مشخصی دارند. برای هر نقطه پایانی، محدودیت نرخ تماس در هر دقیقه به شرح زیر است:

- **Chat**: ۲۰ تماس در دقیقه
- **Cluster** : ۵ تماس در دقیقه
- **Embed** : ۵ تماس در دقیقه
- **EmbedJob**: ۵ تماس در دقیقه
- **Rerank** : ۱۰ تماس در دقیقه
- **Generate قدیمی** : ۵ تماس در دقیقه
- **Summarize قدیمی** : ۵ تماس در دقیقه

همچنین، کلیدهای آزمایشی محدودیت ماهانه ۱٬۰۰۰ تماس برای هر نقطه پایانی دارند .

Cohere

این کلیدها تا زمانی که به محدودیت‌های فوق نرسیده‌اید معتبر هستند و تاریخ انقضای مشخصی ندارند. با این حال، اگر نیاز به استفاده بیشتر یا در محیط تولید دارید، می‌توانید به کلید تولیدی ارتقاء دهید که محدودیت‌های کمتری دارد .

محدودیت‌های **نقطه تماس (endpoint limits)** به تعداد دفعاتی اشاره دارند که می‌توانید درخواست به یک سرویس خاص از API بفرستید، مثلاً برای تولید متن یا پردازش ورودی.

1. **تعداد درخواست در دقیقه:(Rate Limit per Minute)**
   - یعنی شما در هر **یک دقیقه** می‌توانید فقط تعداد مشخصی درخواست بفرستید.
   - مثلاً برای (**Generate قدیمی** (شما فقط می‌توانید ۵ درخواست در دقیقه بفرستید. اگر بیشتر بفرستید، درخواست‌های اضافی رد می‌شوند.
2. **تعداد درخواست در ماه:**
   - برای کلیدهای آزمایشی، حداکثر **۱۰۰۰ درخواست در ماه** می‌توانید به هر endpoint بفرستید.
   - یعنی اگر شما ۲۰ بار در روز از کلید خود استفاده کنید، می‌توانید حدود ۵۰ روز از کلید استفاده کنید (۱۰۰۰ درخواست تقسیم بر ۲۰ در روز).

هدف پروژه

هدف این پروژه تحلیل کد جاوا و پیشنهاد اپراتورهای میوتیشن (Mutation Operators) برای آزمایش کد است. اپراتورهای میوتیشن تغییرات کوچکی در کد منبع ایجاد می‌کنند تا کارایی تست‌های واحد بررسی شود.

توضیح کامل کد

نصب و وارد کردن کتابخانه‌ها

```
!pip install cohere
import cohere
import random
```

- **!pip install cohere** : این دستور برای نصب کتابخانه Cohere استفاده می‌شود که در صورت نصب نبودن، کتابخانه را به محیط اضافه می‌کند.
- **import cohere** : برای استفاده از قابلیت‌های API کتابخانه Cohere وارد می‌شود.
- **import random** : برای تولید انتخاب‌های تصادفی از اپراتورهای پیشنهادی استفاده می‌شود.

تنظیم کلیدAPI

```
co = cohere.Client('fE[redacted]')
```

- **cohere.Client** :با استفاده از کلیدAPI ، یک کلاینت برای برقراری ارتباط با مدل‌های Cohereایجاد می‌شود.
- 'fE...???L' :کلید API کاربر که برای احراز هویت در درخواست‌ها استفاده می‌شود.

تابعread_java_file

- این تابع برای خواندن محتوای فایل جاوا استفاده می‌شود.
- **file_path** : مسیر فایل ورودی.
- **try-except** : از وقوع خطای "فایل یافت نشد" جلوگیری می‌کند.
- **with open** : فایل جاوا را با حالت خواندن باز می‌کند.

```python
def read_java_file(file_path):
    try:
        with open(file_path, "r", encoding="utf-8") as file:
            return file.read()
    except FileNotFoundError:
        print("فایل یافت نشد!")
        return None
```

تابع suggest_mutation_operators

هدف :ارسال کد جاوا به مدل Cohere و دریافت پیشنهاد اپراتورهای میوتیشن.

```python
def suggest_mutation_operators(java_code):
    prompt = f"""
    You are an expert in Object Oriented Mutation Testing (Encap
    """
    try:
        response = co.generate(
            model='command-xlarge-nightly',
            prompt=prompt,
            max_tokens=500,
            temperature=0.5
        )
        return response.generations[0].text.strip()
    except Exception as e:
        return f"An error occurred while generating suggestions:
```

- **Prompt** : توضیحات و دستورالعمل‌ها برای مدل.
- **Model** : مشخص کردن مدل .**command-xlarge-nightly**

درباره مدل command-xlarge-nightly

- **command-xlarge** : این یکی از بزرگ‌ترین مدل‌های Cohere است که برای پردازش زبان طبیعی (NLP) طراحی شده است.
- **Nightly** : این نسخه هر شب به‌روزرسانی می‌شود تا عملکرد بهتری داشته باشد یا باگ‌های احتمالی رفع شوند.

- **بهترین انتخاب برای مدل:**
- **اگر دقت و کیفیت پاسخ مهم است،** استفاده از بزرگـترین مدل مثل command-xlarge-nightly مناسب است.
- **اگر سرعت و هزینه پردازش اهمیت دارد،** می‌توان از مدل‌های کوچـک‌تر مثل command-medium-nightly یا حتی command-small-nightly استفاده کرد. البته کیفیت پاسخ ممکن است کاهش یابد.

چگونه بهترین مدل را انتخاب کنیم؟

- **اگر تست‌های زیادی نداریم** و عملکرد بهتری می‌خواهیم: همان command-xlarge-nightly مناسب است.
- **اگر بودجه و سرعت مهم هستند** :می‌توانید مدل کوچـک‌تر مثل command-medium-nightly را تست کنید.

- **max_tokens** : تعیین حداکثر تعداد توکن‌های تولید شده.
- **Temperature** : تنظیم خلاقیت مدل. مقدار 0.5 تعادل بین دقت و خلاقیت را برقرار می‌کند.
  - پارامتر **temperature** در مدل‌های زبانی مانند Cohere یا OpenAI نشان‌دهنده سطح خلاقیت یا تصادفی بودن پاسخ است:
  - مقدار پایین‌تر مثل **0.0** یا **0.3** باعث می‌شود پاسخ‌ها قابل پیش‌بینی‌تر و قطعی‌تر باشند.
  - مقدار بالاتر مثل **0.7** یا **1.0** به مدل اجازه می‌دهد خلاق‌تر و متنوع‌تر عمل کند، اما احتمال تولید پاسخ‌های نامربوط یا غیرمفید نیز افزایش می‌یابد.
  - برای دریافت بهترین پیشنهادها در این سناریو:
  - **اگر پاسخ‌های دقیق و کاربردی بخواهیم** مثلاً فقط بهترین اپراتورها برای تست کد جاوا : **temperature = 0.3 یا 0.5** تنظیم می‌کنیم.
  - **اگر بخواهیم مدل تنوع بیشتری داشته باشد** و اپراتورهای مختلفی را پیشنهاد دهد : **temperature = 0.7** مناسب است.

- **response.generations[0].text.strip()** : پاسخ مدل را برمی‌گرداند.
- **try-except** : مدیریت خطاها در صورت بروز مشکل در ارتباط با.API

تابع extract_operators

```
def extract_operators(suggestions):
    valid_operators = ["AMC", "IHI", "IHD", ...]
    operators = []
    for op in valid_operators:
        if op in suggestions:
            operators.append(op)
    return operators
```

هدف: استخراج اپراتورهای معتبر از پاسخ مدل.

- **valid_operators**: لیست اپراتورهای معتبر میوتیشن.
- **for op in valid_operators**: بررسی وجود اپراتورها در پاسخ مدل.

تابع select_random_operators

```
def select_random_operators(operators, count=4):
    return random.sample(operators, min(count, len(operators)))
```

هدف: انتخاب تصادفی 4 اپراتور از لیست اپراتورهای معتبر.

- **random.sample** : انتخاب تصادفی از لیست.

تابع save_to_file

```
def save_to_file(output_path, operators):
    try:
        with open(output_path, "w", encoding="utf-8") as file:
            file.write(", ".join(operators))
        print(f"پیشنهادها با موفقیت ذخیره شد در: {output_path
    except Exception as e:
        print(f"خطا در ذخیره فایل: {e}")
```

هدف: ذخیره اپراتورهای پیشنهادی در یک فایل متنی.

- **with open** : فایل را باز کرده و داده‌ها را ذخیره می‌کند.

- **Join** : تبدیل لیست اپراتورها به رشته‌ای که با ", " جدا شده است.

تابع main

هدف :مدیریت کل فرآیند از دریافت فایل جاوا تا تولید و ذخیره پیشنهادات.

- **file_path** : مسیر فایل جاوا.
- **output_path** : مسیر فایل خروجی.
- **java_code** : محتوای کد جاوا خوانده شده از فایل.
- **Suggestions** : پیشنهادهای مدل برای اپراتورهای میوتیشن.
- **selected_operators** : اپراتورهای تصادفی انتخاب شده.
- **save_to_file** : ذخیره اپراتورها در فایل خروجی.

```
def main():
    print("لطفاً مسیر فایل جاوا را وارد کنید:")
    file_path = "/content/code.java"
    output_path = "mutation_suggestions.txt"

    java_code = read_java_file(file_path)
    if java_code:
        print("\nدر حال تحلیل کد...\n")
        suggestions = suggest_mutation_operators(java_code)
        if suggestions:
            print("پاسخ کامل مدل:")
            print(suggestions)
            operators = extract_operators(suggestions)
            if operators:
                selected_operators = select_random_operators(ope
                save_to_file(output_path, selected_operators)
                print("اپراتورهای انتخاب شده:", ", ".join(sele
            else:
                print("هیچ اپراتور مناسبی پیشنهاد نشد.")
        else:
            print("پاسخی از مدل دریافت نشد.")

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

خروجی نهایی

خروجی نهایی شامل لیستی از اپراتورهای پیشنهادی میوتیشن است که در فایلی به نام **mutation_suggestions.txt** ذخیره می‌شود. از این فایل به عنوان ورودی خودکار برای اجرای تست بر روی ورودی کاربر به جای تحلیل دستی و انالیز افراد خبره، استفاده می‌کنیم که مدل استفاده شده می‌تواند در مواردی نتایج دقیق‌تر و انالیز علمی‌تری نسبت به افراد مبتدی که تنها نیاز به تست دارند و نه اینکه در کد چه میگذرد، داشته باشد.

نتیجه‌گیری

این کد با استفاده از API Cohere فرآیند تحلیل کد جاوا و پیشنهاد اپراتورهای میوتیشن را به صورت خودکار انجام می‌دهد. هر مرحله از کد برای اطمینان از دقت و کارایی تست‌ها طراحی شده است . Cohere به عنوان یک ابزار NLP قوی، امکان پردازش زبان طبیعی را به شکل قابل اعتماد و دقیق فراهم می‌کند. استفاده از ورودی خودکار کار را برای افرادی که تنها قرار است کدی را تست کنند، بسیار مفید است زیرا ان‌ها از انالیز کد ناآشنا و حتی در مواردی پیچیده بی‌نیاز می‌کند و فرایند آزمون را تسریع می‌کند.