رضا رستمی

۳۹۹۱۶۳۴

Peas ربات فوتبالیست

معیارکارایی: برد بازی ، گل زدن بیشتر از تیم حریف ، بیرون نرفتن از چهارچوب بازی ، خطا نکردن روی بازیکن حریف ، سرعت جابجایی قابل قبول برای پشت سر گذاشتن بازیکن تیم حریف ، توانایی شناسایی و تشخصی عوامل محیطی و فیزیکی بازی به راحتی مانند (حریف ، بازیکن خودی ، دروازه ها ، توپ) ، استفاده نکردن دست بجز دروازه بان

محیط : زمین چمن (مصنوعی،طبیعی) ، فوتسال ، ساحلی (شن)

عملگرها : شوت ، چیپ ، سانتر کردن ،
پاس دادن ، گرفتن توپ توسط دست (دروازه بان) ، هد زدن ، تکل زدن

سنسور:، سنسور تشخیص توپ ، سرعت توپ و جهت آن ، سنسور تشخیص فاصله ، سنسور آنالیز بازیکنان حریف ، سنسور خطوط (برای بیرون نرفتن توپ از محیط زمین) ، سنسور تشخیص بازیکن حریف/خودی / دروازه ها ، سنسور شناسایی عوامل بیرونی محیطی / فیزیکی ، سنسور دوربین ۳۶۰ درجه ، سنسور اب و هوا (برای لیز نخوردن و تشخیص جهت باد قبل از شوت زدن)، سنسور افساید

با مسائل غیر قطعی چگونه رفتار میکنیم؟

راه حل مسائل غیر قطعی در هوش مصنوعی مرتبط با مدیریت و تصمیم گیری در شرایطی که دارای عدم قطعیت هستند می باشد. برای حل این گونه مسائل، می توان از رویکردها و تکنیک های زیر استفاده کرد:

رضا رستمی

۳۹۹۱۶۳۴

1. احتمالات و آمار: استفاده از مفاهیم احتمالات و آمار برای مدل سازی و پیش بینی وقوع رویدادها در شرایط عدم قطعیت.

2. مدل سازی بیزی: استفاده از مدل های بیزی برای نمایش علاقه مندی ها و توزیع های احتمالی در مسائل غیر قطعی.

3. تئوری تصمیم گیری: اعمال تکنیک های تصمیم گیری چون مدل های مارکوف تصمیم گیری (MDP) و فرآیندهای تصمیم گیری مارکوف (Markov Decision Processes) برای تعیین تصمیم های بهینه در شرایطی که دارای عدم قطعیت هستند.

4. اطلاعات فازی: استفاده از اطلاعات فازی برای مدل سازی عدم قطعیت و عدم دقت در داده ها و تصمیم گیری ها.

5. تکنیک های ترکیبی: ترکیب اطلاعات احتمالی و داده های مشاهده شده با دانش پیشین و تجربی به منظور بهبود تصمیم گیری در شرایط عدم قطعیت.

6. الگوریتم های بهینه سازی: استفاده از الگوریتم های بهینه سازی برای یافتن راه حل های بهینه در مسائل غیر قطعی.

7. تکنیک های تحلیل حساسیت: تجزیه و تحلیل حساسیت برای درک تأثیر پارامترها و عوامل مختلف بر نتایج تصمیم گیری در شرایط عدم قطعیت.

8. شبکه های عصبی: استفاده از شبکه های عصبی برای مدل سازی و پیش بینی در شرایط عدم قطعیت.

رضا رستمی

۳۹۹۱۶۳۴

ترکیبی از این رویکردها و تکنیک ها بسته به مسئله مورد نظر و میزان عدم قطعیت می تواند به راه حل های موثری در مسائل غیر قطعی در هوش مصنوعی منجر شود.

کد۸ وزیر رو پیدا کنید

```
def is_safe(board, row, col, n):
    # چک کردن آیا می توان وزیری را در سلول (row, col) قرار داد یا خیر

    # چک کردن ردیف افقی (سمت چپ)
    for i in range(col):
        if board[row][i] == 1:
            return False

    # چک کردن قطر بالا به چپ
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    # چک کردن قطر پایین به چپ
    for i, j in zip(range(row, n, 1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False
```

رضا رستمی

۳۹۹۱۶۳۴

```
return True


def solve_n_queens_util(board, col, n):
    # حالت پایه: اگر تمام وزیرها قرار گرفته باشند
    if col >= n:
        return True

    # برای هر سلول در ستون فعلی
    for i in range(n):
        # چک کردن آیا می توان وزیر را در این سلول قرار داد
        if is_safe(board, i, col, n):
            # قرار دادن وزیر در این سلول
            board[i][col] = 1

            # ادامه به جستجوی ستون بعدی
            if solve_n_queens_util(board, col + 1, n):
                return True

            # اگر قرار گرفتن وزیر در این سلول به حل مسئله منجر نشود، آن را از صفحه حذف می کنیم
            board[i][col] = 0
```

رضا رستمی

۳۹۹۱۶۳۴

```
    # اگر هیچ یک از سلول ها منجر به حل مسئله نشود
    return False

def solve_n_queens(n):
    # ایجاد صفحه شطرنج خالی
    board = [[0 for _ in range(n)] for _ in range(n)]

    # حل مسئله با فراخوانی اولیه از ستون اول
    if not solve_n_queens_util(board, 0, n):
        print("هیچ راه حلی وجود ندارد.")
        return False

    # نمایش جواب
    for i in range(n):
        for j in range(n):
            print(board[i][j], end=" ")
        print()
    return True

# تابع را فراخوانی می کنیم با n=8 برای حل مسئله 8 وزیر
solve_n_queens(8)
```

رضا رستمی

۳۹۹۱۶۳۴

به نظر شما پیچیدگی زمانی دستو پاگیر تره یا پیچیدگی حافظه؟

بستگی به نوع مسئله و الگوریتمی که برای حل آن استفاده می شود دارد که پیچیدگی حافظه یا زمانی کدامیک بیشتر است. در برخی موارد، مسائلی وجود دارند که پیچیدگی حافظه آن ها بسیار بالاست و نیاز به استفاده از منابع حافظه بالا دارند. برای مثال، الگوریتم هایی که برای پردازش تصویر و صدا استفاده می شوند، به دلیل بزرگی حجم داده های ورودی نیاز به استفاده از حافظه بالا دارند.

در مقابل، در بسیاری از مسائل، پیچیدگی زمانی بیشتر از پیچیدگی حافظه است. به عنوان مثال، الگوریتم هایی که برای مرتب سازی اعداد استفاده می شوند، نیاز به حافظه کمتری دارند ولی زمان بیشتری برای اجرای آن ها لازم است.

بنابراین، برای انتخاب بهترین الگوریتم برای حل یک مسئله، باید به دو پیچیدگی حافظه و زمانی توجه کرد و الگوریتمی را انتخاب کرد که برای آن مسئله پیچیدگی کمتری داشته باشد.