Homework1

November 14, 2020

1 Cognitive Modelling: Homework 1

Name: Vishal Sreenivasan
Student No.: S4196392

1.1 Importing files

[1]:
```
from model import Model
from dmchunk import Chunk
```

1.2 Initializing the model

[2]:
```
m = Model()
```

1.3 Adding numbers to declarative memory

[3]:
```
numbers = ["zero", "one",
           "two", "three",
           "four", "five",
           "six", "seven",
           "eight","nine",
           "ten","eleven",
           "twelve","thirteen",
           "fourteen", "fifteen"]

for i in range(0,len(numbers)-1):
    #Numbers are intialized using chunks
    fact = Chunk(name = "af" + numbers[i], slots ={"isa":"add-fact", "num1":␣
 ↪numbers[i], "num2" : numbers[i+1]})
    #Chunks are added to the declarative memory
    m.add_encounter(fact)
```

[4]:
```
print(m)
```

```
=== Model ===
Time: 0 s
Goal:None
```

```
DM:Chunk afzero
Slots: {'isa': 'add-fact', 'num1': 'zero', 'num2': 'one'}
Encounters: [0]
Fan: 0

Chunk add-fact
Slots: {}
Encounters: [0]
Fan: 15

Chunk zero
Slots: {}
Encounters: [0]
Fan: 1

Chunk one
Slots: {}
Encounters: [0]
Fan: 2

Chunk afone
Slots: {'isa': 'add-fact', 'num1': 'one', 'num2': 'two'}
Encounters: [0]
Fan: 0

Chunk two
Slots: {}
Encounters: [0]
Fan: 2

Chunk aftwo
Slots: {'isa': 'add-fact', 'num1': 'two', 'num2': 'three'}
Encounters: [0]
Fan: 0

Chunk three
Slots: {}
Encounters: [0]
Fan: 2

Chunk afthree
Slots: {'isa': 'add-fact', 'num1': 'three', 'num2': 'four'}
Encounters: [0]
Fan: 0

Chunk four
Slots: {}
Encounters: [0]
```

```
Fan: 2

Chunk affour
Slots: {'isa': 'add-fact', 'num1': 'four', 'num2': 'five'}
Encounters: [0]
Fan: 0

Chunk five
Slots: {}
Encounters: [0]
Fan: 2

Chunk affive
Slots: {'isa': 'add-fact', 'num1': 'five', 'num2': 'six'}
Encounters: [0]
Fan: 0

Chunk six
Slots: {}
Encounters: [0]
Fan: 2

Chunk afsix
Slots: {'isa': 'add-fact', 'num1': 'six', 'num2': 'seven'}
Encounters: [0]
Fan: 0

Chunk seven
Slots: {}
Encounters: [0]
Fan: 2

Chunk afseven
Slots: {'isa': 'add-fact', 'num1': 'seven', 'num2': 'eight'}
Encounters: [0]
Fan: 0

Chunk eight
Slots: {}
Encounters: [0]
Fan: 2

Chunk afeight
Slots: {'isa': 'add-fact', 'num1': 'eight', 'num2': 'nine'}
Encounters: [0]
Fan: 0

Chunk nine
```

```
Slots: {}
Encounters: [0]
Fan: 2

Chunk afnine
Slots: {'isa': 'add-fact', 'num1': 'nine', 'num2': 'ten'}
Encounters: [0]
Fan: 0

Chunk ten
Slots: {}
Encounters: [0]
Fan: 2

Chunk aften
Slots: {'isa': 'add-fact', 'num1': 'ten', 'num2': 'eleven'}
Encounters: [0]
Fan: 0

Chunk eleven
Slots: {}
Encounters: [0]
Fan: 2

Chunk afeleven
Slots: {'isa': 'add-fact', 'num1': 'eleven', 'num2': 'twelve'}
Encounters: [0]
Fan: 0

Chunk twelve
Slots: {}
Encounters: [0]
Fan: 2

Chunk aftwelve
Slots: {'isa': 'add-fact', 'num1': 'twelve', 'num2': 'thirteen'}
Encounters: [0]
Fan: 0

Chunk thirteen
Slots: {}
Encounters: [0]
Fan: 2

Chunk afthirteen
Slots: {'isa': 'add-fact', 'num1': 'thirteen', 'num2': 'fourteen'}
Encounters: [0]
Fan: 0
```

```
Chunk fourteen
Slots: {}
Encounters: [0]
Fan: 2

Chunk affourteen
Slots: {'isa': 'add-fact', 'num1': 'fourteen', 'num2': 'fifteen'}
Encounters: [0]
Fan: 0

Chunk fifteen
Slots: {}
Encounters: [0]
Fan: 1
```

1.4 Add function

[5]:
```python
def add(num1, num2):
    g = Chunk(name = "goal",
              slots = {"isa": "add-goal", "start": num1, "end": num2, "counter":
 ↪ "zero"}) #Initialize goal with a counter buffer
    m.goal = g
    done = False
    while not done:
        if not "current" in g.slots:
            #Initialize current with the start value
            g.slots["current"] = g.slots["start"]
            print(g.slots["current"])

            #Request for current
            request = Chunk(name = "request", slots = {"isa": "add-fact",␣
 ↪"num1": g.slots["current"]})
            #Request for counter
            count_request = Chunk(name = "count_request", slots = {"isa":␣
 ↪"add-fact", "num1": g.slots["counter"]})

            m.time += 0.05

            #Retrieve the next number to the current number
            chunk, latency = m.retrieve(request)
            #Retrieve the next number to the counter number
            count_chunk, count_latency = m.retrieve(count_request)

            #Update current
            g.slots["current"] = chunk.slots["num2"]
            #Update counter
            g.slots["counter"] = count_chunk.slots["num2"]

            m.time += (latency + count_latency)

        elif g.slots["counter"] != g.slots["end"]: #Check if counter does not␣
↪exceed end value
            print(g.slots["current"])

            request = Chunk(name = "request", slots = {"isa": "add-fact",␣
↪"num1": g.slots["current"]})
            count_request = Chunk(name = "count_request", slots = {"isa":␣
↪"add-fact", "num1": g.slots["counter"]})

            m.time += 0.05

            chunk, latency = m.retrieve(request)
            count_chunk, count_latency = m.retrieve(count_request)

            g.slots["current"] = chunk.slots["num2"]
            g.slots["counter"] = count_chunk.slots["num2"]

            m.time += (latency + count_latency)
        else:
            print(g.slots["current"])
            done = True
```

1.5 Result

[6]:
```
add("two", "ten")
```

```
two
three
four
five
six
seven
eight
nine
ten
eleven
twelve
```