**CS 171, Winter, 2021  
Artificial Intelligence  
Project, part 1**

Below is part 1 of the quarter-long chatbot project. It is due around the end of the third week of the quarter; precise date and time to be determined.

You must work on the project individually. You can talk with other people in the class about your and their general approaches, but do not copy any code or algorithm details.

We will supply some starter code in Python, which is a very simple weather chatbot, but you can do your own coding in another programming language. Java and C++ are fine. If you want to use another language, get approval first from Marzieh Ashrafiamiri, *mashrafi at uci dot edu*. Here are links to the three starter Python files (as .txt files):

* [Proj1.py](https://frost.ics.uci.edu/cs171W21/Proj1.py.txt)
* [CYKParse.py](https://frost.ics.uci.edu/cs171W21/CYKParse.py.txt)
* [Tree.py](https://frost.ics.uci.edu/cs171W21/Tree.py.txt)

Text-based chatbots typically have several components:

1. Code to convert the input string into a parse tree. (CYKParse, getSentenceParse)
2. Code to convert a parse tree into some convenient internal representation. (updateRequestInfo)
3. Code to retrieve information from a database, using the convenient internal representation. (getTemperature)
4. Code to format a reply to the user, based on the retrieved information. (reply)

You'll notice that in the starter code most of these components are implemented in a rudimentary way. As you proceed through the project, you'll need to design and code more robust and interesting replacements for the code supplied here.

Your assignment for part 1 of the project is to complete the following steps. You should get started soon, although it is likely that some details of the steps may change over the next week.

**Steps**

1. Become familiar with the supplied code, and translate it into another programming language if you choose. It would be good to run CYKParse as the main module, comment out and un-comment out various lines in the main() function, and carefully look at the output. Nothing to turn in for this step.
2. The CYK-Parse in the textbook (Fig. 23.5) and in the code require that the syntax be in Chomsky Normal Form. Sometimes this is inconvenient, and it would be helpful to also accept syntax rules of the form *X ⟶ Y [p]* — that is, with a single category on the right hand side. For instance, one of the rules in Fig. 23.4 has such a form. Let's call the modified CNF, relaxed to also permit single category right hand sides, "Chomsky 171 Form". Your assignment in this step is to modify the CYKParse implementation to accept C1F. Report: Describe your approach to this step, and an overview of the changes you made. Provide a listing of the code you modified.

要求：修改代码兼容Chomsky Normal Form和 C1F两种格式

报告：1、描述什么是Chomsky Normal Form

2、CNF（Chomsky Normal Form）和C1F的区别

3、C1F只有三个参数，CNF有四个，修改getGrammarSyntaxRules方法判断参数个数返回不同参数

def getGrammarSyntaxRules(grammar):

    rulelist = []

    for rule in grammar['syntax']:

        if(len(rule)==4):

            yield rule[0], rule[1], rule[2], rule[3]

        else:

            yield rule[0], rule[1], '', rule[2]

4、修改CYKParse方法，处理C1F格式文本

                printV('i:', i, 'j:', j, 'k:', k, '', X, '->', Y, '['+str(p)+']',

                        'PY =' ,getP(Y, i, j), p, '=', getP(Y, i, j) \* p)

                PYZ = getP(Y, i, j) \* p

                if PYZ > getP(X, i, k):

                    printV('     inserting from', i, '-', k, ' ', X, '->', T[Y+'/'+str(i)+'/'+str(j)],

                                'because', PYZ, '=', getP(Y, i, j), '\*', p, '>', getP(X, i, k), '=',

                                'getP(' + X + ',' + str(i) + ',' + str(k) + ')')

                    P[X + '/' + str(i) + '/' + str(k)] = PYZ

                    T[X + '/' + str(i) + '/' + str(k)] = Tree.Tree(X, T[Y+'/'+str(i)+'/'+str(j)], None)

1. Now that your code accepts C1F, take advantage of it. Modify the getGrammarWeather function to use C1F rules~~, with no change in functionality~~. This should make the new grammar somewhat more compact. Report: Describe your approach to this step, and an overview of the changes you made. Provide a listing of the revised getGrammarWeather function.

要求：修改getGrammarWeather测试对C1F的兼容性

报告：修改['S', 'NP', 'VP', 0.25]为['S', 'VP', 0.25],测试兼容C1F格式。

1. Here's a bold claim: "The probabilities associated with CNF and C1F lexicons, as seen in Fig. 23.3, are completely unnecessary and have no effect on the output of CYK-Parse." Is this statement true or false? Support your position, ideally by reporting on an experiment with your code. Report: Do you agree with the claim, or not? Back up your position, ideally by reporting on an experiment you ran.

修改lexicons中不同单词的权重(probabilities)没有得到不同的结果。

1. Improve and extend the weather-bot in several ways:
   1. Include the cities Tustin and Pasadena, and the time yesterday, in the system's capabilities. Report: Describe your approach to this step, and an overview of the changes you made. Provide a listing of the revised getGrammarWeather function.

要求：添加Tustin and Pasadena两个城市，和添加时间yesterday

报告：1、修改getGrammarWeather方法，在lexicon中添加

['Name', 'Tustin', 0.8],

['Name', 'Pasadena', 0.8],

['Adverb', 'yesterday', 0.3],

三个参数。

2、修改getTemperature方法，添加对应时间很地点

    elif location == 'Tustin':

        if time == 'now':

            return '58'

        elif time == 'today':

            return '58'

        elif time == 'tomorrow':

            return '60'

        elif time == 'yesterday':

            return '56'

        else:

            return 'unknown'

    elif location == 'Pasadena':

        if time == 'now':

            return '48'

        elif time == 'today':

            return '48'

        elif time == 'tomorrow':

            return '50'

        elif time == 'yesterday':

            return '46'

        else:

            return 'unknown'

* 1. Extend the grammar to understand and respond to questions of the sort "Will tomorrow be hotter than today in Irvine", "Will yesterday be hotter than tomorrow in Pasadena?" (Yes, the verb tense doesn't quite work in that one, but you can ignore that.) Report: Describe your approach to this step, and an overview of the changes you made. Provide listings of the code you changed or created.

要求：添加两个比较句的解析支持和对问句进行回答

报告：

* 1. 添加对比较句解析的支持：
  2. 在getGrammarWeather添加单词

['WQuestion', 'will', 0.5],

['Verb', 'be', 0.5],

['Verb', 'was', 0.5],

['Adjective', 'hotter', 0.5],

['Preposition', 'than', 1],

以支持两个问句

* 1. 在getGrammarWeather添加规则

['VP','Verb','Adjective',0.06],

['VP', 'Adverb', 'VP', 0.6 \* 0.15],

['NP','Adjective','AdverbPhrase',0.8],

['AdverbPhrase', 'Preposition', 'NP+AdverbPhrase', 0.2],

['AdverbPhrase', 'Preposition', 'AdverbPhrase', 0.2],

实现对问句的解析

* 1. 添加对比较句的处理:

2.1、获取两个比较的时间

        if leaf[0] == 'WQuestion' and leaf[1] == 'will':

            lookingForTime1 = True

        if lookingForTime1 and leaf[0] == 'Adverb':

            requestInfo['time1'] = leaf[1]

            lookingForTime1 = False

            continue

        if leaf[0] == 'Preposition' and leaf[1] == 'than':

            lookingForTime2 = True

        if lookingForTime2 and leaf[0] == 'Adverb':

            requestInfo['time2'] = leaf[1]

            lookingForTime2 = False

            continue

2.2获取比较的单词(hotter)

        if leaf[0] == 'Adjective':

            requestInfo['compare'] = leaf[1]

2.3根据时间和地点(原程序已经获取)，获取对应的温度，并且比较温度对用户做出回答

    if requestInfo['time1'] != '':

        if requestInfo['compare'] == 'hotter':

            temp1 = getTemperature(requestInfo['location'],requestInfo['time1'])

            temp2 = getTemperature(requestInfo['location'],requestInfo['time2'])

            salutation = ''

            if requestInfo['name'] != '':

                salutation = 'dear ' + requestInfo['name'] + ', '

            if temp1 > temp2 :

                salutation += requestInfo['time1'] + ' is hotter than ' + requestInfo['time2'] + ' in ' + requestInfo['location']

                print(salutation)

                return

            elif temp1 < temp2:

                salutation += requestInfo['time2'] + ' is hotter than ' + requestInfo['time1'] + ' in ' + requestInfo['location']

                print(salutation)

                return

            elif temp2 == temp1:

                salutation += requestInfo['time2'] + ' and ' + requestInfo['time1'] + ' have the same temperature in ' + requestInfo['location']

                print(salutation)

                return

            else:

                return

* 1. 添加测试到main方法
* T, P = CYKParse.CYKParse(['what', 'was', 'the', 'temperature', 'in', 'Pasadena', 'yesterday'], CYKParse.getGrammarWeatherC1F())
* sentenceTree = getSentenceParse(T)
* updateRequestInfo(sentenceTree)
* reply()
* T, P = CYKParse.CYKParse(['what', 'was', 'the', 'temperature', 'in', 'Tustin', 'yesterday'], CYKParse.getGrammarWeatherC1F())
* sentenceTree = getSentenceParse(T)
* updateRequestInfo(sentenceTree)
* reply()
* T, P = CYKParse.CYKParse(['will', 'yesterday', 'be', 'hotter', 'than', 'now', 'in', 'Irvine'], CYKParse.getGrammarWeatherC1F())
* sentenceTree = getSentenceParse(T)
* updateRequestInfo(sentenceTree)
* reply()
* T, P = CYKParse.CYKParse(['will', 'yesterday', 'be', 'hotter', 'than', 'tomorrow', 'in', 'Pasadena'], CYKParse.getGrammarWeatherC1F())
* sentenceTree = getSentenceParse(T)
* updateRequestInfo(sentenceTree)
* reply()