A Natural Language Question Answering System on Mathematics

Amy Wei
8/12/2021

Problem Statement

- Parse, understand, and answer mathematical natural language questions, such as:
  - “is (0.0,0.0) (0.0,1.0) (1.0,1.0) an acute triangle”
  - “are (0.0,0.0) (0.0,1.0) and (1.0,0.0) (1.0,1.0) perpendicular”
  - “is (-0.5,-0.5) inside the triangle (-1.0,0.0) (0.0,0.0) (0.0,-1.0)”

Technical Approach

- Prolog (logic-based, declarative)

1. Implement algorithm predicates
2. Define grammar components and sentences
3. Process user input, call corresponding predicates

Technical Approach: Algorithm Predicates

Ex: triangle(X1,Y1,X2,Y2,X3,Y3)

- Solve on paper
- Translate into code
  - Rigorous testing
  - Consider all possible corner cases

Technical Approach: Algorithm Predicates

To be a triangle:

- Must contain 3 distinct points
- Cannot form a straight line

Translate into code:

- Must satisfy our “threepoints” predicate
- Must not satisfy our “straightline” predicate

Technical Approach: Algorithm Predicates

```
threepoints(point(X1,Y1),point(X2,Y2),point(X3,Y3)):-
   Z1 is abs(X2-X1)+abs(Y2-Y1), Z1>0,
   Z2 is abs(X3-X2)+abs(Y3-Y2), Z2>0,
   Z3 is abs(X3-X1)+abs(Y3-Y1), Z3>0.

straightline(point(X,Y1),point(X,Y2),point(X,Y3)):-
   threepoints(point(X,Y1),point(X,Y2),point(X,Y3)),!.
straightline(point(X1,Y1),point(X2,Y2),point(X3,Y3)):-
   threepoints(point(X1,Y1),point(X2,Y2),point(X3,Y3)),
   (X1-X2)*(X2-X3)*(X3-X1)=\=0,
   A is (Y1-Y2)/(X1-X2),
   B is (Y2-Y3)/(X2-X3),
   C is (Y3-Y1)/(X3-X1),
   A=B, B=C.
```

Technical Approach: Grammar

- “is (0.0,0.0) (1.75,0.0) a vertical line”

is	verb
(0.0,0.0)	point
(1.75,0.0)	point
a	determiner
vertical	adjective (VH)
line	noun (L)

Technical Approach: Grammar

Vocabulary:

```
prep(P)-->[P],{member(P,[inside,between])}.
conj(C)-->[C],{member(C,[and])}.
```

```
v(V)-->[V],{member(V,[is,are])}.

adjOE(A)-->[A],{member(A,[odd,even])}.
adjPN(A)-->[A],{member(A,[positive,negative])}.
adjVH(A)-->[A],{member(A,[vertical,horizontal])}.
adjARO(A)-->[A],{member(A,[acute,obtuse,right])}.
adjPP(A)-->[A],{member(A,[parallel,perpendicular])}.

det(D)-->[D],{member(D,[the,a,an])}.

nN(Noun)-->[Noun],{member(Noun,[number,numbers])}.
nL(Noun)-->[Noun],{member(Noun,[line])}.
nT(Noun)-->[Noun],{member(Noun,[triangle])}.
```

“is 8 horizontal”
(Invalid input)

Technical Approach: Grammar

Numbers:

```
digit(X)-->[X], {member(X,[0,1,2,3,4,5,6,7,8,9])}.

/* recursion: an integer can be another integer plus a digit */
/* test with:  phrase(integer(X),[1,2,3]),!. */
integer([])-->[].
integer([X|Y])-->digit(X),integer(Y).

/* negative integer: */
sign-->['-'].
neginteger(Y)-->sign,integer(Y).

/* float point number: (can be an integer) */
dot(X)-->[X],{member(X,['.'])}.
floatnumber(X1,Y,X2)-->integer(X1),dot(Y),integer(X2).
floatnumber(X1,Y,X2)-->sign,integer(X),dot(Y),integer(X2), {append(['-'], X, X1)}.
```

Technical Approach: Grammar

Points:

```
/* point: */
leftparen-->['('].
rightparen-->[')'].
comma-->[','].
point(X1,A1,B1,X2,A2,B2)-->leftparen,floatnumber(X1,A1,B1),comma,
    floatnumber(X2,A2,B2),rightparen.
```

Technical Approach: Grammar

- Individual components make up sentences

```
/* s0: is 123 odd/even? */
s([s0,V,N,A])-->v(V),integer(N),adjOE(A).
```

```
/* s1: is 12 an even number? */
s([s1,V,N,D,A,Noun])-->v(V),integer(N),det(D),adjOE(A),nN(Noun).
```

Technical Approach: Processing User Input

- is 12 an even number

After Processing:

- [is, 1, 2, an, even, number]

Call Predicate:

- even(12)

```
s1(V,N,D,A,Noun):-
   atomics_to_string(N,N1),atom_number(N1,N2),
   S=[A,N2],X=..S,call(X),write('Yes'),nl,!;
   (write('No'),nl).
```

Technical Approach: Processing User Input

- is (0.0,0.0) (0.0,0.0) a vertical line

After Processing:

- [is, ‘(‘, 0, ‘.’, 0, ‘,’, 0, ‘.’, 0,‘)’, ‘(‘, 0, ‘.’, 0, ‘,’, 0, ‘.’, 0,‘)’, a, vertical, line]

Call Predicate:

- vertical(0.0, 0.0, 0.0, 0.0)

```
S=[A,NaNumber,NbNumber,NcNumber,NdNumber],
XX=..S,call(XX),write('Yes'),nl,!;
(write('No'),nl).
```

Results

- is 1 odd
- is 1 even
- is 1 an even number
- is 1 an odd number
- is 1 positive
- is (0.0,0.0) (0.0,0.1) vertical
- is (0.0,0.0) (0.0,0.1) horizontal

```
?- ask.
|: is 12 even
Yes
|: is 123 even
No
|: is 12 odd
No
|: is 123 odd
Yes
```

Results

- is (0.0,0.0) (0.0,0.1) a vertical line
- is (0.0,0.0) (0.0,0.1) a horizontal line
- is (0.0,0.0) (1.0,0.0) (0.0,1.0) a triangle
- is (0.0,0.0) (1.0,0.0) (0.0,1.0) an acute triangle
- is (0.0,0.0) (1.0,0.0) (0.0,1.0) a right triangle
- is (0.0,0.0) (1.0,0.0) (0.0,1.0) an obtuse triangle

Results

- is (0.0,0.0) inside the triangle (0.0,0.0) (1.0,0.0) (0.0,0.1)
- are (0.0,0.0) (0.0,1.0) and (1.0,0.0) (1.0,1.0) parallel
- are (0.0,0.0) (0.0,1.0) and (1.0,0.0) (1.0,1.0) perpendicular
- odd numbers between 0 and 100
- even numbers between 0 and 100
- calculate sin(pi)*log(10)

Summary

- Algorithm predicates (corner cases, testing)
- Grammar (personalized structure)
- Process user input, call corresponding predicates

Further Steps

- is (0.0,0.0) inside the rectangle (0.0,0.0) (0.0,0.0) (0.0,0.0) (0.0,0.0)
- is (0.0) (0.0) (0.0) a straight line
- is (0.0,0.0) (0.0,1.0) (1.0,1.0) (1.0,0.0) a square
- prime numbers between 0 and 1000

References and Acknowledgements

- Advisor Dr. Wei Wei
- *Learn Prolog Now!* by Patrick Blackburn, Johan Bos and Kristina Striegnitz
- *Logic Programming with Prolog Second Edition* by Max Bramer
- *Compilers - Principles, Techniques, & Tools* by Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman
- Point in polygon algorithm: https://en.wikipedia.org/wiki/Point_in_polygon