AI::CBR
Case-Based Reasoning
For Perl

Darko Obradovic
AI::CBR

Case-Based Reasoning for Perl

Darko Obradovic

Kaiserslautern.pm

http://kaiserslautern.pm.org
darko@kaiserslautern.pm.org

DFKI GmbH

http://www.dfki.uni-kl.de/~obradovic
darko.obradovic@dfki.de
Another Vietnam?

Ed Timms
Dallas Morning News
1995
Artificial Intelligence?

AI is the mimicking of human thought and cognitive processes to solve complex problems automatically.

(one definition amongst many)
Case-Based Reasoning
Foundations:
Roger Schank 1977/1983
1. new problem = new case
2. experience = case-base
3. remembering = retrieval
4. apply to new problem = reuse
5. observe & learn = revise
6. memorise = retain
1. new problem = new case
2. experience = case-base
3. remembering = retrieval
4. apply to new problem = reuse
5. observe & learn = revise
6. memorise = retain
AI::CBR
An example!
# the case
{
    age => 40,
    gender => 'male',
    job => 'programmer',
    symptoms => ['headache', 'cough'],
}

diagnosis = ?
It's all about similarity!
compare two objects A and B on attribute level with similarity functions:

\[ \text{sim}_1 = \text{sim}(\text{att}_A_1, \text{att}_B_1) \]
\[ \text{sim}_2 = \text{sim}(\text{att}_A_2, \text{att}_B_2) \]
\[ \vdots \]
\[ \text{sim}_n = \text{sim}(\text{att}_A_n, \text{att}_B_n) \]
compare two objects A and B on attribute level with similarity functions:

\[ sim_1 = sim(\text{att}_A_1, \text{att}_B_1) \]
\[ sim_2 = sim(\text{att}_A_2, \text{att}_B_2) \]
\[ \ldots \]
\[ sim_n = sim(\text{att}_A_n, \text{att}_B_n) \]

similarity values are between 0 and 1
- 0.0 is no similarity at all
- 1.0 is equality
the overall similarity between two objects A and B is the mean value of their attribute's similarities:

\[
\text{sim}_{A\_B} = \left( \frac{\text{sim}_1 + \text{sim}_2 + \ldots + \text{sim}_n}{n} \right)
\]
the overall similarity between two objects A and B is the mean value of their attribute's similarities:

\[ \text{sim}_A_B = \left( \frac{\text{sim}_1 + \text{sim}_2 + \ldots + \text{sim}_n}{n} \right) \]

similarity values are between 0 and 1
- 0.0 is no similarity at all
- 1.0 is equality
use CBR::AI::Sim qw(sim_eq sim_frac sim_set);

# for symbolic values
sim_eq('programmer', 'programmer'); # 1.0
sim_eq('programmer', 'manager');    # 0.0
use CBR::AI::Sim qw(sim_eq sim_frac sim_set);

# for symbolic values
sim_eq('programmer', 'programmer'); # 1.0
sim_eq('programmer', 'manager');    # 0.0

# for numbers
# fraction of the smaller wrt the greater
sim_frac(2, 4);    # 0.5
sim_frac(40, 30);  # 0.75
use CBR::AI::Sim qw(sim_eq sim_frac sim_set);

# for symbolic values
sim_eq('programmer', 'programmer');  # 1.0
sim_eq('programmer', 'manager');    # 0.0

# for numbers
# fraction of the smaller wrt the greater
sim_frac(2, 4);  # 0.5
sim_frac(40, 30); # 0.75

# for sets/lists of symbolic values
# |intersection elements| / |union elements|
sim_set([qw(a b c d)], ['a']);     # 0.25
sim_set([qw(a b c)], [qw(b c d)]); # 0.5
# the case
{
    age => 40,
    gender => 'male',
    job => 'programmer',
    symptoms => ['headache', 'cough'],
}

$diagnosis = ?
use AI::CBR::Case;
use AI::CBR::Sim qw(
    sim_eq sim_frac sim_set
);

my $new_case = AI::CBR::Case->new(
    age      => { sim => \&sim_frac },
    gender   => { sim => \&sim_eq   },
    job      => { sim => \&sim_eq   },
    symptoms => { sim => \&sim_set  },
);
set case values

$new_case->set_values(
    age => 40,
    gender => 'male',
    job => 'programmer',
    symptoms => ['headache','cough'],
);

# case definition and values
# could have been done in one step
use DBIx::Simple;
my $db = DBIx::Simple->new(...);

my @case_base = $db->query(
    SELECT age, gender, job, symptoms, diagnosis
    FROM patients
)->hashes();

foreach @case_base {
    $_->{symptoms} = [split ',', $_->{symptoms}]
}

retrieve most similar case

use AI::CBR::Retrieval;

my $r = AI::CBR::Retrieval->new($new_case, [@case_base]);
$r->compute_sims();

my $solution = $r->most_similar_candidate();
print $solution->{diagnosis};
# in case-base
{
  age       => 28,
  gender    => 'male',
  job       => 'programmer',
  symptoms  => ['headache'],
  diagnosis => 'hangover',
}
# for age:
sim_frac(40, 28) # 0.7
# for age:
sim_frac(40, 28) # 0.7
# for gender:
sim_eq('male', 'male') # 1.0
# for age:
sim_frac(40, 28) # 0.7
# for gender:
sim_eq('male', 'male') # 1.0
# for job:
sim_eq('programmer', 'programmer') # 1.0
# for age:
sim_frac(40, 28) # 0.7
# for gender:
sim_eq('male', 'male') # 1.0
# for job:
sim_eq('programmer', 'programmer') # 1.0
# for symptoms:
sim_set([['headache', 'cough'], ['headache']]) # 0.5
# for age:
sim_frac(40, 28) # 0.7
# for gender:
sim_eq('male', 'male') # 1.0
# for job:
sim_eq('programmer', 'programmer') # 1.0
# for symptoms:
sim_set(['headache', 'cough'], ['headache']) # 0.5

$sim = 3.2 / 4; # 0.8
# for age:
sim_frac(40, 28) # 0.7
# for gender:
sim_eq('male', 'male') # 1.0
# for job:
sim_eq('programmer', 'programmer') # 1.0
# for symptoms:
sim_set(['headache','cough'], ['headache']) # 0.5

$sim = 3.2 / 4; # 0.8

$diagnosis => 'hangover'!
What else can you do?
Similarity Functions

• define your own ones!
  • there are millions of different use-cases
  • flexibility with optional parameter
Similarity Functions

• define your own ones!
  • there are millions of different use-cases
  • flexibility with optional parameter
• code them in C
  • most efficient tuning
Retrieval

- multiple methods:
  - most_similar_candidate()
  - n_most_similar_candidates($n)
  - first_confirmed_candidate('diagnosis')
Retrieval

- multiple methods:
  - most_similar_candidate()
  - n_most_similar_candidates($n)
  - first_confirmed_candidate('diagnosis')
    1. Hangover
    2. Stress
    3. Whiplash Injury
    4. Stress
    5. ...
Retrieval

• multiple methods:
  • most_similar_candidate()
  • n_most_similar_candidates($n)
  • first_confirmed_candidate('diagnosis')
    1. Hangover
    2. Stress
    3. Whiplash Injury
    4. Stress
    5. …
Retrieval

- multiple methods:
  - `most_similar_candidate()`
  - `n_most_similar_candidates($n)`
  - `first_confirmed_candidate('diagnosis')`
    1. Hangover
    2. Stress
    3. Whiplash Injury
    4. Stress
    5. ...
- subclass & extend!
Intelligent Product Retrieval
Looking for older or classic cars? Use our 1982 and older search.
1. query = new case
2. products = case-base
3. search = retrieval
Compound Cases
Trip
Trip
Trip

0.9

0.9

0.9

0.9
Trip 0.9

0.9

0.1
use AI::CBR::Case::Compound;

my $case = AI::CBR::Case::Compound->new([
    $flight_spec,
    $hotel_spec,
    $destination_spec,
]);

$sim = nroot(3, $sim_flight * $sim_hotel * $sim_destination);
What did you do with it?
CYBORG

AI for a strategy game
observation → situations + decisions
situations + decisions

reuse
Conclusions

- observed 15-30 games per player
- thousands of comparisons per turn
  - ~400 turns per game

- system performance ok
- AI performance good

- for details, see homepage
The End

Thank you!

Questions?
matrix similarity function

my $os_matrix = {
    linux => { mac => 0.8, win => 0.2 },
    mac => { linux => 0.8, win => 0.5 },
    win => { linux => 0.2, mac => 0.5 },
};

$new_case = AI::CBR::Case->new(
    os => {
        sim => 
            \&sim_matrix,
        param => $os_matrix,
    },
    ...,
);
matrix similarity function

```perl
sub sim_matrix {
    my ($a, $b, $matrix) = @_; # unpack arguments
    return 1 if $a eq $b; # check if $a and $b are the same
    return $matrix->{$a}->{$b}; # return value from matrix
}
```