Universidade do Porto
Faculdade de Engenharia
FEUP

Confiança em Fontes de Informação

Relatório Final

Agentes e Inteligência Artificial Distribuída

4º ano do Mestrado Integrado em Engenharia Informática e Computação

13 de Dezembro de 2014

Confiança em Fontes de Informação

Unidade Curricular: Agentes e Inteligência Artificial Distribuída

Curso: Mestrado Integrado em Engenharia Informática e Computação

Ano: 4º

Realizado por:

Francisco Miguel Amaro Maciel	201100692	ei11084@fe.up.pt
Hugo Miguel Ribeiro de Sousa	201100690	ei11083@fe.up.pt

8 de Novembro de 2014

Índice

Objetivo

O objetivo deste trabalho é a especificação e implementação de um sistema distribuído e descentralizado seguindo o paradigma de sistemas multiagente, no âmbito da inteligência artificial. No trabalho proposto especificamente, pretende-se também o estudo de modelos de confiança computacional e respetiva implementação.

O trabalho realizado pretende simular um jogo de perguntas e respostas, uma versão modificada do *"Quem quer ser Milionário?".*

Nesta versão, simule-se que os jogadores se encontram em salas separadas, não tendo conhecimento direto das ações dos outros jogadores. O apresentador vai, em cada ronda, caminhando de sala em sala e apresentando a questão ao jogador. O jogador tem de escolher um ajudante, de entre os disponíveis, para lhe ajudar na resposta. Os conhecimentos dos ajudantes variam de tema para tema, pelo que o jogador deve escolher aquele que ache mais apto para o ajudar na categoria da questão proposta. Para o ajudar a escolher o melhor ajudante, o jogador pode perguntar a opinião dos outros jogadores relativamente aos ajudantes para o tema pretendido. Assume-se que os jogadores são obrigados a dizer a verdade.

Um jogo é composto por 15 perguntas a cada jogador. Uma resposta certa vale 1 ponto, enquanto que uma resposta errada vale 0 pontos. Vence o jogador que no final do jogo tiver a maior pontuação.

Os jogadores serão automáticos, pelo que não haverá interação com o utilizador. Será possível executar um dado número de jogos consecutivos, sendo o objetivo principal testar e comparar os diferentes modelos de confiança computacional, que serão abordados posteriormente.

Especificação

É possível identificar 3 tipos de atores distintos no jogo: o apresentador, o jogador e o ajudante, representando cada um deles um agente distinto. Tal como seria de prever na realidade, a comunicação entre os agentes é síncrona. Por exemplo, o apresentador só passa para a próxima pergunta após receber a resposta do jogador. Segue-se a especificação dos agentes implementados.

PresenterAgent

Parâmetros:

- jogos - número de jogos a ser executado consecutivamente

Este agente trata de enviar as perguntas aos jogadores, receber as respetivas respostas e confirmar se esta está correta ou errada.

HelperAgent

Parâmetros:

- [categoria:experiência] – conjunto de pares categoria-experiência do agente, separados por espaço. A experiência deve ser um valor entre 0 e 100. Categorias não especificadas terão um valor de experiência 0.

Exemplo: HelperAgent("desporto:80, cultura:50, arte:75, cinema:25");

Este agente é responsável por ajudar os jogadores, tendo em conta a sua experiência na categoria da pergunta. Quanto maior a experiência, maior a probabilidade de saber a resposta correta de uma pergunta dessa categoria. Um valor de experiência 0 corresponde a errar todas as perguntas e uma experiência de valor 100 corresponde a acertar todas as perguntas. A experiência de um HelperAgent não se altera ao longo do tempo.

FIREPlayerAgent

Parâmetros:

- [variável:valor] – conjunto de variáveis e respetivos valores. O conjunto de valores modificáveis são:
  - "IT" – *threshold* a partir do qual se dá confiança máxima à interação direta com o ajudante. Valor por defeito: 5.
  - "W" – *threshold* do número de jogadores mínimo que respondem para que se atribua a confiança máxima à opinião destes. Deve ser um valor no intervalo [0, nJogadores – 1] para alcançar a confiança máxima. Valor por defeito: 2 (testes efetuados com 3 jogadores).
  - "NC" – *threshold* do número de interações diretas que tem que haver com um ajudante até que deixe de ser considerado um *newcomer*. Enquanto um ajudante é *newcomer*, é dada a oportunidade a esse ajudante, mesmo que tenha valor de confiança inferiores a outros. Valor por defeito: 5.

Este agente representa um jogador e implementa o modelo de confiança computacional FIRE. São considerados 2 componentes deste modelo: *interaction trust* e *witness reputation*. *Role-based trust* e *certified reputation* não foram implementados, por não se adequarem ao modelo de negócio.

SINALPHAPlayerAgent

Parâmetros:

- [variável:valor] – conjunto de variáveis e respetivos valores. O conjunto de valores modificáveis são:
  - "POS_L" – valor de compensação atribuído a uma resposta correta. Valor por defeito: 1.
  - "NEG_L" – valor de penalização atribuído a uma resposta errada. Valor por defeito: -1.5.
  - "OMEGA" – valor que dividido por π influencia a evolução da confiança. Valores maiores correspondem a uma evolução mais lenta da confiança. Valor por defeito: 40.

Este agente representa um jogador e implementa o modelo de confiança computacional SINALPHA. Este modelo apenas especifica a forma como evolui a confiança de um agente. A forma como são tratados os valores de confiança retornados pelos outros jogadores e a combinação com a interação direta não são especificados. Assim sendo, optou-se por atribuir um peso de 70% à confiança de interação direta e 30% à confiança proveniente dos outros jogadores. Isto deve-se ao facto de a confiança dos outros jogadores poder ser calculada por diferentes modelos, influenciando bastante o valor da nova confiança do SINALPHA, o que seria negativo para casos de teste.

BETAPlayerAgent

Este agente representa um jogador e implementa o modelo de confiança computacional BETA. Não foram implementadas as componentes *Discounting* e *Forgetting* por não se adequarem ao nosso sistema. Tal como no modelo "FIRE", estas comentes foram descartas pois assumem mudança no comportamento dos agentes a avaliar, tanto a nível de informações incorretas, como a nível de mudança de comportamento. Como os "HelperAgent" não mudam o seu nível de especialidade numa dada categoria não faz sentido contemplar regras que não se aplicam ao modelo de negócio.

Protocolo de Interação

Sendo o sistema síncrono, pode ser especificado por uma lista de mensagens trocadas entre os agentes, da seguinte forma:

1. O apresentador envia os dados da questão ao jogador respetivo.
2. O jogador pergunta aos outros jogadores o que sabem acerca de ajudantes relativamente à categoria da questão recebida.

3. Cada jogador, após receber a mensagem (2), responde com os valores de confiança calculados para os ajudantes com o qual já interagiu para a categoria referida.
4. O jogador, após calcular os valores de confiança da interação direta e dos dados recebidos na mensagem (3), seleciona o ajudante do qual pretende obter ajuda, enviando-lhe a questão e as respetivas opções.
5. O ajudante responde ao jogador com uma das opções.
6. O jogador responde ao apresentador com a resposta fornecida pelo ajudante na mensagem (5).
7. O apresentador informa o jogador se a resposta foi correta ou errada. O jogador atualiza os seus dados relativamente ao ajudante e categoria referidos.

No final do jogo, o apresentador trata também de enviar uma mensagem de performativa *INFORM* a todos os jogadores, informando que o jogo acabou.

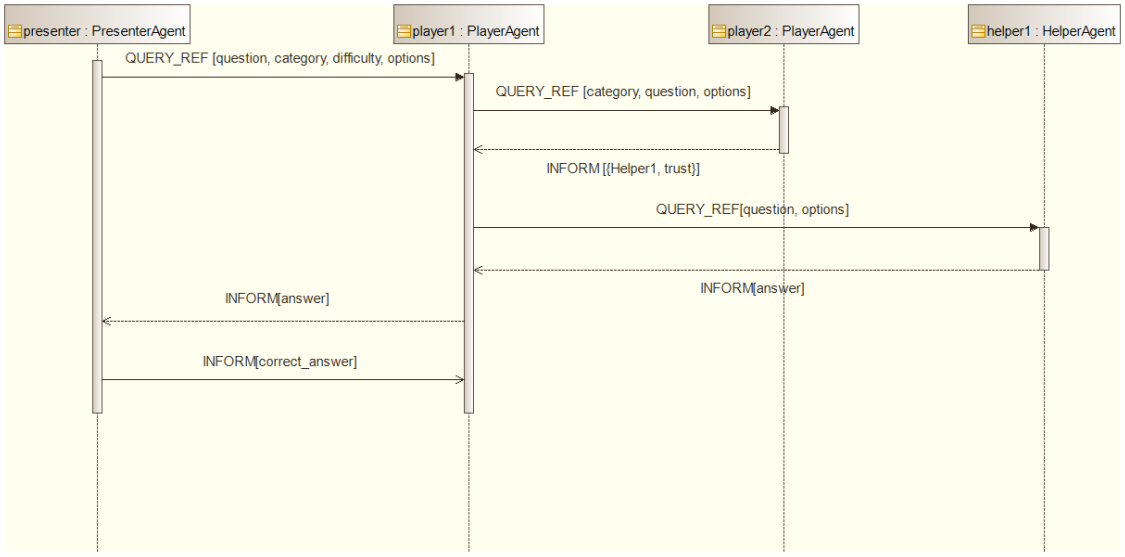


Figura 1 – Protocolo de interação entre os agentes numa ronda.

Desenvolvimento

O trabalho foi desenvolvido em *Java*, usando Eclipse como IDE em máquina com Windows 8.1. Foi usada a biblioteca *json-simple* que facilita a utilização do formato standard universal JSON, visto ser o formato usado para o envio de mensagens com múltiplos campos no seu conteúdo.

O sistema foi desenvolvido com recurso à *framework* JADE (Java Agent DEvelopment Framework). Esta *framework* simplifica a implementação de sistemas

multiagente, seguindo os standards especificados pela FIPA. O JADE permite que o sistema seja distribuído, podendo os agentes ser executados em diferentes máquinas.

As questões utilizadas no jogo são lidas, no início da execução, a partir de um ficheiro "questions.txt". Este ficheiro contém 891 questões diferentes. O apresentador trata de perguntar uma questão aleatória deste conjunto a cada jogador.

O sistema implementa uma arquitetura de agentes comunicativos. O desenrolar do jogo deve-se à correta colaboração entre os diversos agentes. As mensagens são distinguidas através da sua performativa, que não é necessariamente diferente em todos os tipos de mensagens. Visto o sistema ser síncrono, sabe-se que mensagens são esperadas num dado momento por um agente, podendo assim ser identificada a ação a ser executada por este.

Foi usado um agente genérico como superclasse dos outros agentes para reunir métodos comuns e isolar mais facilmente a comunicação dos detalhes relevantes dos modelos. Usando esta implementação, a lógica e as estruturas necessárias por trás de cada modelo são mantidas nas subclasses de *GenericPlayerAgent*.

O sistema é composto pelos diversos agente e respetivos comportamentos. O apresentador trata de iniciar e finalizar a conversação.

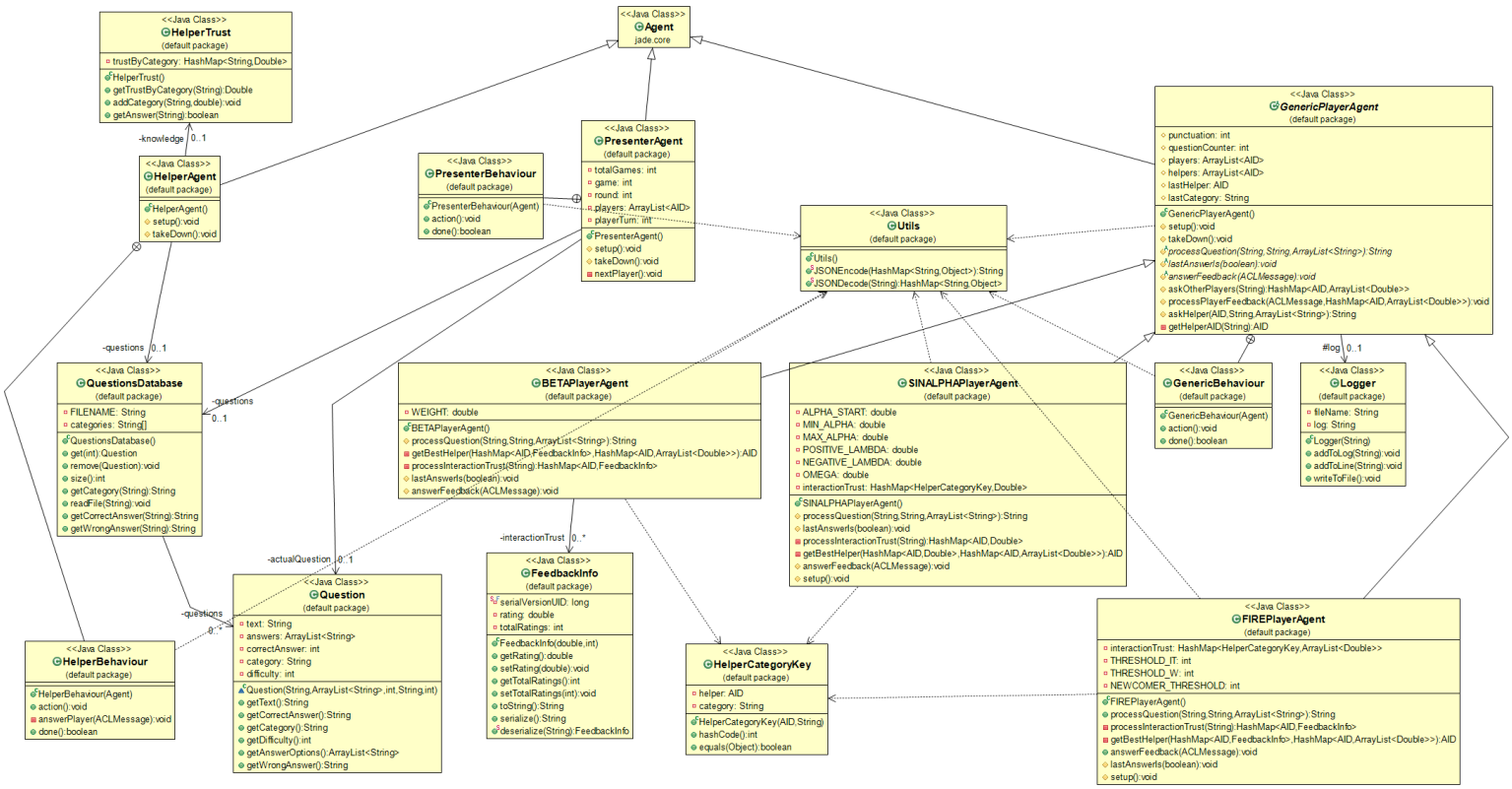


Figura 2 – Diagrama de classes do sistema.

Relativamente aos modelos de confiança computacional, é importante referir alguns aspetos da sua implementação. Optou-se, de um modo geral, por dar maior relevância à confiança obtida por interação direta, relativamente à confiança obtida por terceiros. Isto deve-se ao facto de todos os jogadores comunicarem entre si, independentemente do modelo de confiança computacional implementado. Sendo o objetivo do trabalho testar e comparar os diferentes modelos, dar demasiada relevância à informação vinda dos outros jogadores poderia influenciar os resultados pretendidos.

FIRE

A importância dada à informação proveniente de terceiros é maior, quanto maior o número de agentes que respondem. Por exemplo, se estão 4 jogadores em jogo, e apenas 1 responde com informação útil acerca de um dado par ajudante-categoria, espera-se que seja dada pouca relevância a esta informação. Este valor pode, no entanto, ser configurável (variável "W" passada como parâmetro). Da mesma forma, a relevância dada à informação adquirida anteriormente depende do número de avaliações obtidas anteriormente. Por exemplo, quando um ajudante acerta à primeira pergunta não lhe é dado a confiança de 1.0 mas sim um valor inferior configurável através do parâmetro "IT" que representa ao fim de quantas avaliações a confiança pode atingir o seu valor real sem ser reduzida.

Apesar de não estar especificado no modelo, foi também implementado uma solução para um problema recorrente no estudo dos modelos de confiança computacional: os *newcomers*. Se um ajudante começar inicialmente a acertar num dado tema, mesmo não sendo especialista, o jogador vai continuar a perguntar a esse mesmo ajudante, podendo no entanto haver jogadores com mais experiência a quem não foram dadas oportunidades de ajudar. Assim sendo, enquanto um jogador não tiver ajudado um dado número de vezes, é considerado um *newcomer* e será adicionado à lista de possíveis ajudantes a serem escolhidos nessa ronda. Mais uma vez, esta variável é configurável (variável "NC" passada como parâmetro).

SINALPHA

Este modelo apenas especifica a forma como a confiança evolui ao longo do tempo, através de uma função semelhante à sigmoide, mas menos acentuada nos extremos. No entanto, não é tratada a forma como os dados são enviados e tratados. Assim sendo, optou-se por fazer atribuir um peso de 70% para a interação direta e 30% para a confiança vinda de fontes externas. O valor final da confiança proveniente dos outros jogadores resulta da média dos valores individuais de confiança recebidos. Esta implementação não é prevista pelo modelo uma vez que este apenas considera a confiança própria e não a obtida através de outros jogadores.

A progressão da confiança evolui ao longo do tempo dependendo de três variáveis configuráveis. O valor "OMEGA" está relacionado com a velocidade de

progressão da função, pois o valor de ALPHA (valor usado para calcular o rating posteriormente) é incrementado segundo π /"OMEGA". Assim sendo, se "OMEGA" tiver o valor de 1.0, apenas um feedback positivo transforma o *rating* de 0.0 -> 1.0. Por defeito, este valor está definido com 40 para adquirir maior precisão na escolha dos agentes ajudantes, pois este modelo não considera *newcomers*. Assim sendo, valores "OMEGA" reduzidos causam variações elevadas nos ratings que não permitem obter estabilidade nos resultados e a escolha do melhor ajudante. Para além disso, a forma como o "OMEGA" é adicionado pode não ser linear, ou seja usando os valores de "L" (respetivamente "POS_L" e "NEG_L" para o caso da soma ou subtração de "OMEGA") pode ser controlada a penalização ou a valorização dos resultados. Por exemplo, por defeito, o valor de "NEG_L" é de -1.5 e de "POS_L" é de 1.0, o que penaliza as falhas. Desta forma, ao errar uma pergunta o modelo calcula o novo "ALPHA" como = "ALPHA" – 1.5 * "OMEGA".

BETA

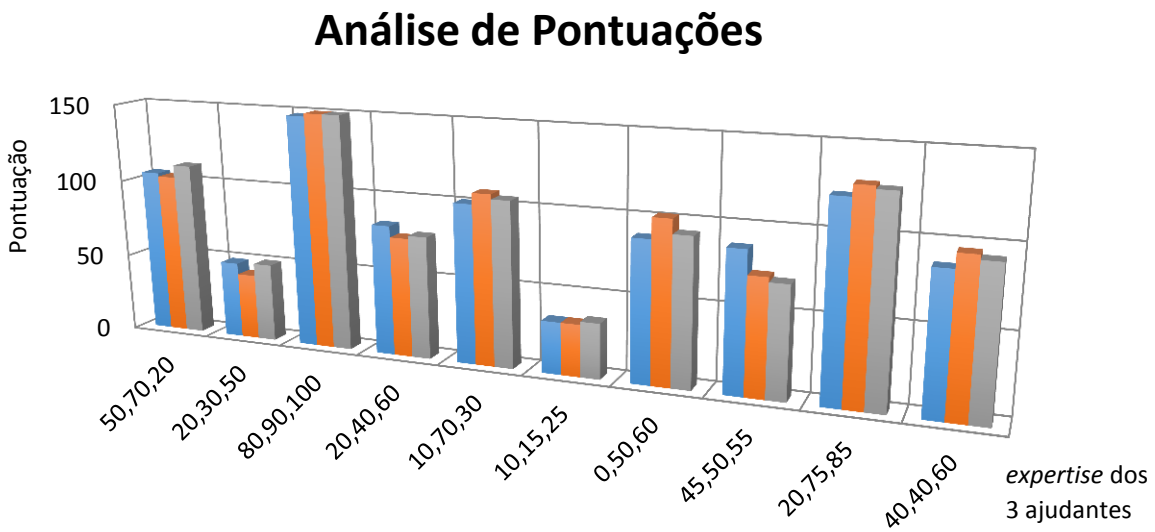
A principal dificuldade na implementação deste modelo foi a combinação do feedback dos outros jogadores com a confiança da interação direta. O modelo sugere que o agente envie aos outros o número de respostas corretas e o número de respostas erradas. No entanto, para a comunicação ser uniforme entre todos os agentes, apenas é enviado um valor único de confiança. Assim sendo, é obtido um valor *r* e *s*, correspondente ao número de respostas certas e erradas respetivamente, a partir do valor de confiança enviado, através das fórmulas $r = \frac{w(1+v)}{2}$ e $s = \frac{w(1-v)}{2}$, sendo *w* um valor fixo entre 0 e 1 (valor por defeito: 1) e *v* o valor de confiança enviado. Estes valores são multiplicados pelo número de interações diretas que o jogador teve, caso contrário o valor de interação seria insignificante, devido à variável *w*. Ou seja, assume-se que os outros jogadores tiveram tantas interações como o próprio, de forma a que o resultado final seja uma combinação de 50/50 entre interação direta e confiança resultante dos outros jogadores.

Experiências

Todas as experiências são executadas tendo em conta os valores por defeito dos jogadores. Para além disso, todas as experiências consideram apenas perguntas de uma só categoria para tornar os resultados mais transparentes. Introduzir mais categorias seria apenas introduzir paralelismo. Ou seja, utilizar várias categorias na mesma experiência corresponderia a realizar várias análises num só jogo, tendo menos perguntas para tirar conclusões.

Experiência 1

Para analisar o desempenho geral dos 3 modelos implementados, numa primeira análise, foi feita a comparação da pontuação de cada um destes. Esta pontuação foi analisada tendo em conta 150 rondas, ou seja sendo 150 a pontuação máxima possível. Todas as perguntas pertencem à mesma categoria e, em cada cenário, existem 3 ajudantes, com diferentes *expertises* para essa mesma categoria. Assim sendo, pretende-se também observar o desempenho dos modelos na escolha do melhor ajudante. Cada conjunto de 3 colunas corresponde a 150 rondas, estando representado no eixo horizontal os 3 valores de *expertise* de cada ajudante, correspondendo 0 a errar todas as perguntas e 100 a acertar todas.



	50,70,20	20,30,50	80,90,100	20,40,60	10,70,30	10,15,25	0,50,60	45,50,55	20,75,85	40,40,60
FIRE	105	49	148	83	101	33	89	88	122	87
SINALPHA	103	42	150	76	108	33	102	73	129	96
BETA	111	50	150	78	105	35	93	70	127	93

Gráfico 1 – Comparação de pontuações dos diferentes modelos, tendo em conta diferentes valores de *expertise* dos ajudantes.

Através desta experiência, pode verificar-se que os 3 modelos obtêm, em geral, um resultado bastante similar. Os modelos parecem funcionar como expetável, reforçando que as três abordagens apenas se distinguem em contextos bastante particulares.

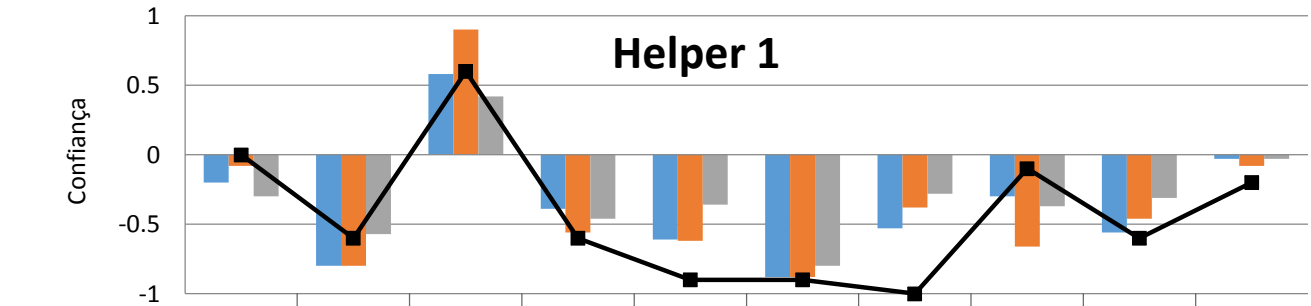
É importante realçar que no 3º teste o modelo FIRE obteve uma pontuação ligeiramente inferior, apesar de os três ajudantes terem uma grande probabilidade de acertar e de os outros modelos terem obtido pontuações máximas. Esta diferença deve-se ao facto do modelo implementar o sistema dos *newcomers,* revelando-se neste caso uma desvantagem. Com o valor por defeito em "NC" de 5, o jogador que implementa o modelo FIRE pergunta sempre pelo menos 5 vezes a cada ajudante, que o prejudica no caso do melhor ajudante ser evidente. Por exemplo, no 7º teste existe um ajudante que falha a todas a perguntas, o que leva o modelo a confirmar 5 vezes se este ajudante é realmente uma má escolha, e a falhar todas elas.

No entanto, a vantagem desta implementação é evidente no teste seguinte (8º teste). Neste caso, todos os ajudantes são muito equivalentes sendo 1 deles ligeiramente. Neste caso, a análise extra providenciada pela implementação dos *newcomers* permite ao modelo FIRE escolher posteriormente às 15 perguntas únicas (5 a cada um) o melhor agente, enquanto que os outros modelos se fixam no primeiro que começar a acertar mais vezes. Com esta vantagem, o modelo FIRE obtém uma pontuação superior.

Experiência 2

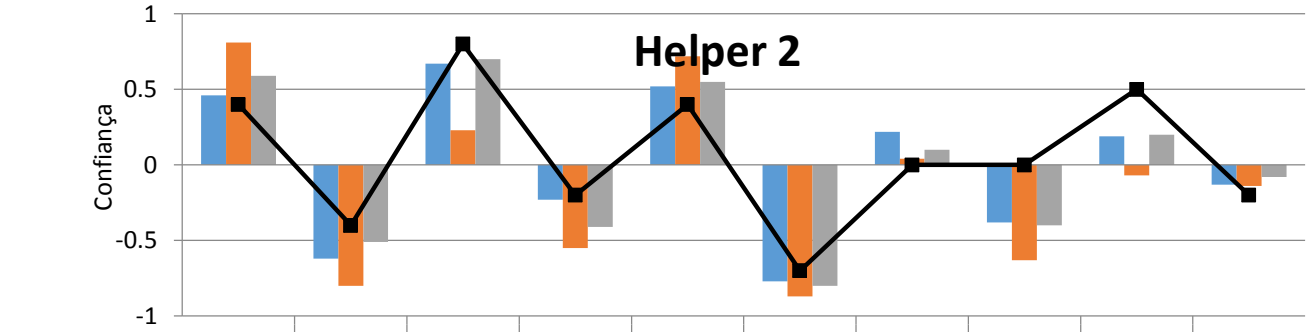
Para além das pontuações, é importante analisar a confiança final de cada modelo. Com isto, podemos verificar se o modelo escolhe realmente o ajudante mais favorável e com que variação para os restantes ajudantes. É, também, comparado com o valor que seria expetável da confiança para esse ajudante.

Os testes executados são os mesmos da Experiência 1, ou seja, 150 rondas com perguntas de apenas 1 categoria com 3 ajudantes de *expertises* variados semelhantes à experiência anterior. Os valores da linha "Expected" correspondem a uma tradução direta entre a probabilidade de acertar de um ajudante ([0,100]) para um rating [-1,1]. Assim sendo um *helper* com probabilidade de acertar de 50% terá um *rating* "Expected" de 0.0, se for 100% terá 1.0, ou por exemplo 60% corresponde a 0.2. Cada gráfico corresponde assim ao rating final para cada *helper*. Ou seja, a coluna 1 de cada gráfico corresponde aos dados recolhidos no mesmo teste, pelos diferentes modelos, para os diferentes *helpers*. Por exemplo, a 1ª coluna do Gráfico 2 mostra a confiança final dos 3 jogadores para o *Helper* 1. Os gráficos 3 e 4 mostram os mesmos resultados, mas relativamente ao *Helper* 2 e *Helper* 3, respetivamente.



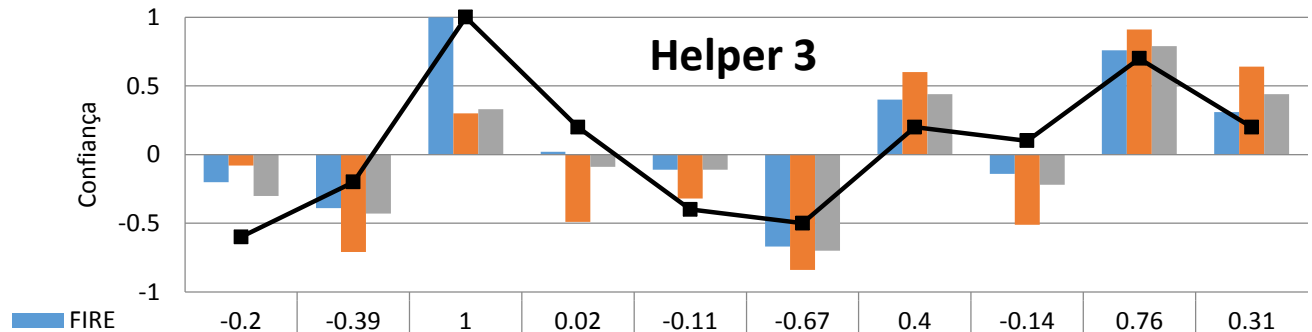
FIRE	-0.2	-0.8	0.58	-0.39	-0.61	-0.88	-0.53	-0.3	-0.56	-0.03
SINALPHA	-0.08	-0.8	0.9	-0.56	-0.62	-0.88	-0.38	-0.66	-0.46	-0.08
BETA2	-0.3	-0.57	0.42	-0.46	-0.36	-0.8	-0.28	-0.37	-0.31	-0.03
Expected	0	-0.6	0.6	-0.6	-0.9	-0.9	-1	-0.1	-0.6	-0.2

Gráfico 2 - comparação da confiança final obtida por cada modelo, com a confiança expetável, relativamente ao ajudante 1.



FIRE	0.46	-0.62	0.67	-0.23	0.52	-0.77	0.22	-0.38	0.19	-0.13
SINALPHA	0.81	-0.8	0.23	-0.55	0.72	-0.87	0.04	-0.63	-0.07	-0.14
BETA	0.59	-0.51	0.7	-0.41	0.55	-0.8	0.1	-0.4	0.2	-0.08
Expected	0.4	-0.4	0.8	-0.2	0.4	-0.7	0	0	0.5	-0.2

Gráfico 3 - comparação da confiança final obtida por cada modelo, com a confiança expetável, relativamente ao ajudante 2.



FIRE	-0.2	-0.39	1	0.02	-0.11	-0.67	0.4	-0.14	0.76	0.31
SINALPHA	-0.08	-0.71	0.3	-0.49	-0.32	-0.84	0.6	-0.51	0.91	0.64
BETA	-0.3	-0.43	0.33	-0.09	-0.11	-0.7	0.44	-0.22	0.79	0.44
Expected	-0.6	-0.2	1	0.2	-0.4	-0.5	0.2	0.1	0.7	0.2

Gráfico 4 – comparação da confiança final obtida por cada modelo, com a confiança expetável, relativamente ao ajudante 3.

Podem verificar-se algumas diferenças significantes entre a confiança final obtida e a expetável. Analisando, teste a teste:

1. Confiança no *helper* 3 demasiado acima ou abaixo do expetável em todos os modelos. Isto deve-se ao facto de, desde cedo, os modelos optarem pelo *helper* 2, não dando oportunidade a este, excluindo o FIRE.
2. O modelo Sinalpha com valores bastante pessimistas, devido à forte penalização do modelo face ao erro. Os restantes modelos apresentam-se com valores relativamente próximos do desejado.
3. Verifica-se que apenas o modelo FIRE foi capaz de atribuir a confiança correta ao *helper* com 100% de expertise. É o modelo que se mostra mais coerente neste tipo de teste em que os *helpers* têm valores próximos de *expertise*, devido à implementação de *newcomers*, dando a oportunidade a todos de mostrar o seu potencial. O modelo Sinalpha acabou por escolher o *helper* mais fraco e o BETA o intermédio, devido às suas interações iniciais com estes. Este jogo representa bem o problema dos *newcomers* no âmbito dos modelos de confiança computacional.
4. Mais uma vez, o Sinalpha a penalizar bastante quando os *expertises* não são muito elevados. Os restantes modelos apresentam-se com valores satisfatórios.
5. Jogo de suposta fácil distinção do melhor *helper*. Todos os modelos se comportaram da forma desejada, sendo o *helper* 2 o que finaliza com maior confiança em todos eles.
6. O modelo Sinalpha a mostrar-se mais penalizador para baixos valores de *expertise*, relativamente aos outros modelos. Valores relativamente próximos do expetável.
7. Tendo o *helper* 1 com 0% de expertise, nota-se que os jogadores o evitaram, sendo o seu valor de confiança longe do realmente expetável, sendo um sinal de perceção rápida do fraco valor deste. Todos os modelos terminaram com uma confiança mais elevada para o melhor ajudante.
8. Novamente o modelo Sinalpha a mostrar-se mais pessimista para baixos valores de *expertise*. Mesmo nos restantes modelos, os valores de confiança apresentam-se abaixo do expetável, talvez por um fator de sorte/azar.
9. Valores aproximados do expetável, exceto para o *helper* 2, a quem se verifica que desde cedo não foi dada a oportunidade de ajudar, ficando o seu valor de confiança aquém do desejado.
10. Todos os modelos escolhem de forma acertada, com valores satisfatório de confiança.

De uma forma geral, os modelos obtêm uma confiança final relativamente próxima ao desejado. As principais conclusões que se tiram desta experiência são:

- Modelo Sinalpha mostra-se bastante pessimista para valores baixos de expertise.
- Modelos, por vezes, acabam por não dar oportunidade a jogadores que são melhores.

- Modelos BETA e FIRE obtém confianças finais relativamente próximas. O FIRE tem a vantagem de implementar *newcomers*, tentando evitar o problema evidenciado na alínea anterior. Pode revelar-se uma desvantagem no caso de haver ajudantes muito fracos, entre ajudantes muito bons.

Experiência 3

Para além da confiança final obtida, é importante analisar de que forma foi a sua evolução e como a interação entre os diferentes jogadores afeta a mesma.

No primeiro teste, foram executadas 15 perguntas, de apenas 1 categoria, tendo 1 ajudante com *expertise* de 100% para essa mesma categoria. Os dois gráficos apresentados correspondem aos valores atribuídos pelos jogadores ao ajudante considerando apenas a sua própria avaliação (IT - *Interaction Trust*) ou considerando as informações dadas pelos outros (WR – *Witness Reputation*)

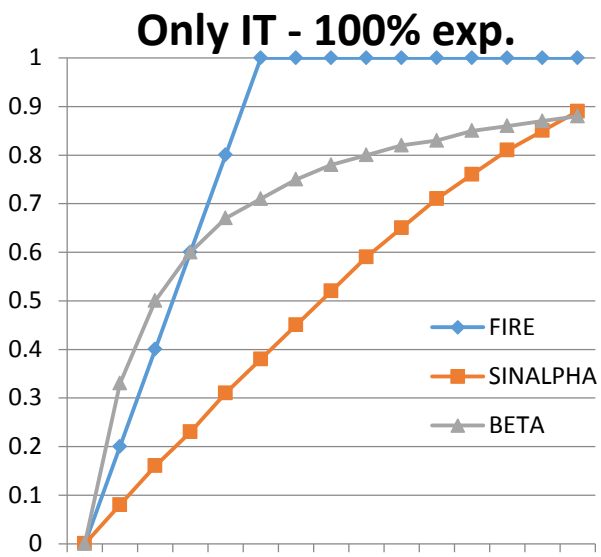


Gráfico 5 – evolução da confiança para os diferentes modelos apenas com interação direta (*interaction trust*) e *expertise* 100%.

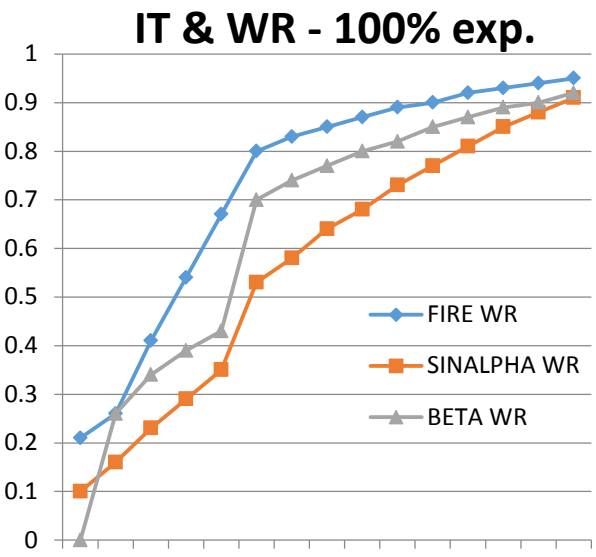


Gráfico 6 – evolução da confiança para os diferentes modelos com interação direta (*interaction trust*) e informação indireta (*witness reputation*) e *expertise* 100%.

Pode verificar-se neste teste que a evolução da confiança dos modelos, sabendo que o ajudante escolhido acerta sempre na resposta, varia bastante entre os diferentes modelos. As principais conclusões que se podem tirar são:

- O modelo FIRE adquire mais rapidamente a confiança, ao contrário dos outros 2 modelos, que têm um ganho mais conservativo.
- De acordo com os valores por defeito dos *newcomes*, o modelo FIRE atinge o seu máximo de valor de confiança após a 5ª interação, havendo aí um ligeiro pico quando existe combinação das confianças. Isto é evidente na progressão apenas com *Interaction Trust*. Se assim não fosse, no gráfico que considera apenas IT, o modelo FIRE teria um declive ainda mais acentuado passando da confiança 0 para 1.0 e mantendo a mesma todas as perguntas seguintes.
- A curva obtida pelo modelo SINALPHA pouco se assemelha à da função sigmoide pois o valor de "OMEGA" usado causa um crescimento pouco acentuado, não sendo tão notória a curvatura.
- A combinação das confianças faz com que o modelos FIRE e BETA evoluam mais lentamente e, pelo contrário, o SINALPHA tenha um crescimento mais acelerado. Isto deve-se ao facto de o SINALPHA crescer mais lentamente, influenciando os outros modelos.

No segundo teste, foram executadas 105 perguntas, de apenas 1 categoria, tendo 1 ajudante com *expertise* de 75% para essa mesma categoria.

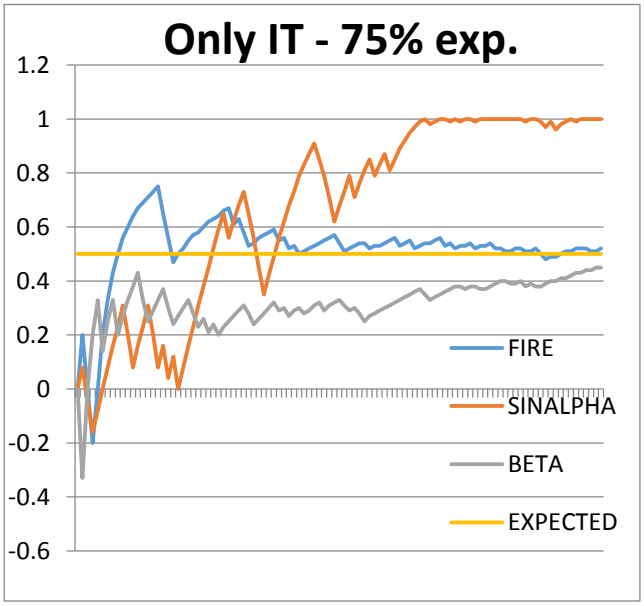


Gráfico 7 – evolução da confiança para os diferentes modelos apenas com interação direta (*interaction trust*) e expertise 75%.

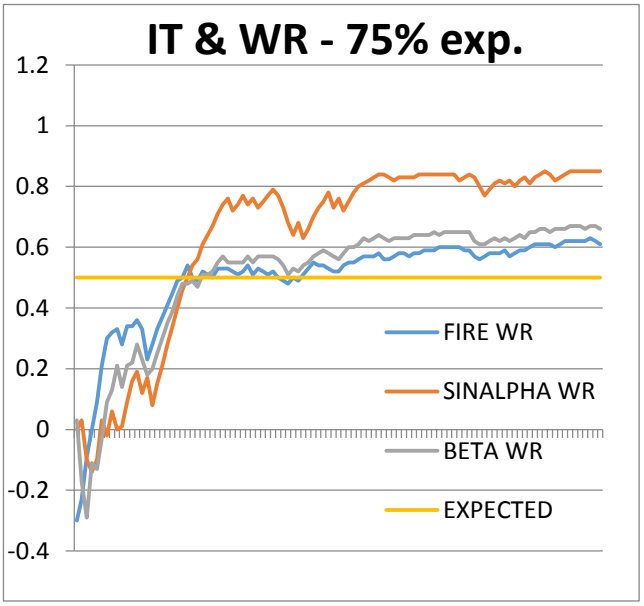


Gráfico 8 – evolução da confiança para os diferentes modelos com interação direta (*interaction trust*) e informação indireta (*witness reputation*) e *expertise* 75%.

Neste teste, verifica-se que o Sinalpha é o que mais se afasta da confiança expetável. Este modelo funciona, basicamente, por estados, sendo que uma resposta correta leva-o a um estado superior e, uma resposta errada leva-o a um estado inferior, até aos devidos limites. Mesmo com a penalização a ser 1.5x superior à compensação, o fator de 75% de expertise acaba por influenciar mais o resultado, chegando o modelo a atingir a confiança máxima no ajudante, devido a evolução da curva segundo a função sigmoide. Neste caso, o Sinalpha mostra-se ineficiente, devido ao facto de funcionar por estados e não calcular a confiança através das interações passadas, como é o caso do FIRE e o BETA. Estes últimos, por sua vez, mostram que os resultados estão bastante próximas do valor de confiança final expetável, mostrando-se eficientes.

No caso em que é incluída a *Witness Reputation*, ou seja os valores de confiança são trocados entre si, é de notar que em todos os modelos a confiança evolui mais suavemente, sem tantas mudanças abruptas (resultado das respostas erradas). Assim, é uma vantagem para todos os modelos, até atingirem a estabilidade, cooperarem entre si para tirar melhores conclusões. No entanto, posteriormente os modelos FIRE e BETA dão valores mais elevados à confiança do que o expectável, influenciados pelo modelo SINALPHA que é mais otimista.

Conclusões

O sistema multiagente implementado veio a tornar-se útil e relevante no trabalho, essencialmente pelo facto de ser distribuído. A comunicação entre os agentes também é facilitada, essencialmente, pela plataforma usada. Desta forma, é possível analisar e testar os diferentes modelos de confiança excluindo detalhes como comunicação e localização, permitindo implementar os modelos num ambiente abstrato.

Na generalidade dos casos, os modelos implementados acabam por conseguir escolher o ajudante mais apropriado ou um dos melhores, em casos de pequena diferença de *expertise* entre os ajudantes. Dos testes levados a cabo, os modelos FIRE e BETA parecem assemelhar-se mais à realidade, com valores de confiança mais coerentes. O Sinalpha parece atribuir valores de confiança extremos quando os valores de *expertise* são muito baixos ou muito elevados.

No entanto, esta interpretação toma em conta que o valor da confiança deve tender para o valor da expertise, o que pode ser um fator que valoriza alguns modelos segundo os outros.

Suponhamos novamente o 2º teste da experiência 3. Refletindo sobre a avaliação dos modelos, poderia ser precipitadamente concluído que o modelo SINALPHA é mais ineficiente. Isto pois, por outro ponto de vista, o modelo SINALPHA chegou a confiança máxima visto este ajudante ter realmente uma boa probabilidade

de acertar. Ao atribuir a confiança de 1.0 o modelo SINALPHA chegou a uma boa conclusão, pois este ajudante é fiável. Assim, embora o FIRE e o SINALPHA se mantenham pelo valor expectável, esta diferença é derivada da distinta forma como os dois modelos funcionam relativamente ao SINALHPA.

Suponhamos dois ajudantes, de probabilidade de acertar respetivamente de 60% e 75%. O modelo FIRE e BETA irão, ao fim de 150 perguntas atribuir valores de confiança de sensivelmente 0.2 e 0.5 respetivamente, tal como será expetável. No entanto, estes modelos fazem uma análise de todas as perguntas respondidas pelos ajudantes para concluírem o seu cálculo. O modelo SINALPHA funciona de forma diferente, guardando apenas o estado atual, evoluindo continuamente segundo a informação que vai recebendo, sem ter acesso a dados analisados anteriormente. Assim, este modelo irá tender para 1.0 em ambos os casos pois estes são ajudantes que, ao longo do tempo, vão provando ser competentes. A diferença entre a escolha entre o primeiro e o segundo não está no valor final da confiança mas sim naquele que lá chega mais rapidamente. Assim, a progressão do ajudante de 75% terá um declive mais acentuado, e será este o escolhido. Desta forma, embora os modelos funcionem de forma diferente, tomam a decisão correta. Isto é percetível pois embora a última experiência mostre confianças distintas, na primeira experiência todos os modelos obtêm aproximadamente a mesma pontuação no jogo. Outra vantagem para o SINALPHA pode ser, em ambientes de grandes quantidades de informação, a poupança de recursos, pelos fatores anteriormente referidos.

Melhoramentos

Seria possível estender ainda mais a aplicação, implementando e analisando mais a fundo outros modelos de confiança computacional. O sistema poderia, dinamicamente, gerar relatórios/gráficos mais apelativos e funcionais que o log implementado, o que facilitaria a análise dos modelos.

A fase inicial da aplicação, aquando da criação dos diferentes agentes deveria ser mais dinâmica. Da forma implementada, os agentes devem ser criados numa determinada ordem (ver Apêndice), devido a cada agente procurar os outros agentes com os quais vai interagir, aquando da sua criação, no método *setup*. Por exemplo, quando um jogador é criado, são imediatamente procurados todos os ajudantes que se encontram no sistema. Poderia ser criado no protocolo uma mensagem representativa do início do jogo, onde os agentes fariam a respetiva procura.

Uma outra perspetiva interessante seria executar um jogo com interação com o utilizador, em que este escolheria qual o modelo que pretendia implementar. A partir daí, o jogador ia tentando responder e poderia perguntar aos ajudantes quando achasse necessário. A aplicação poderia mais tarde ajudá-lo a escolher o melhor ajudante. No entanto, para que os modelos tenham valores de confiança coerentes são precisas muitas rondas, pelo que esta opção não seria muito benéfica para o jogador a curto prazo.

Recursos

Bibliografia

1. Urbano J, Rocha AP, Oliveira E (2009). Computing Confidence Values: Do Trust Dynamics Matter? The 14th Portuguese Conference on Artificial Intelligence, EPIA'2009, Aveiro, Portugal, October 2009
2. Urbano J, Rocha AP, Oliveira E. A Trust Aggregation Engine that Uses Contextual Information. *EUMAS 2009 7th European Workshop on Multi-Agent Systems, Ayia Napa, Cyprus*, 2009
3. Jøsang, A., Ismail, R.: "The Beta Reputation System", in Proceedings of the 15th Bled Electronic Commerce Conference, Sloven, 2002
4. T. Dong Huynh, Nicholas R. Jennings, Nigel R. Shadbolt. FIRE: An Integrated Trust and Reputation Model for Open Multi-Agent Systems. 16th European Conference on Artificial Intelligence, Valencia, Spain, 2004

Software

Jade - http://jade.tilab.com/

JSON-simple (Java Toolkit for JSON) - https://code.google.com/p/json-simple/

Contribuição dos elementos do grupo

Ambos os elementos contribuíram igualmente para o bom rumo deste trabalho. O desenvolvimento foi feito maioritariamente em pair-programming, analisando e discutindo os detalhes dos modelos antes da implementação.

Apêndice

Logs do programa

Para simplificar a análise dos dados, foi implementada a classe *Logger* que é chamada pelos diferentes agentes e regista informação a cada pergunta. No log, é registada a informação da última ocorrência. Para cada jogador, é criado um ficheiro de texto com o nome do agente. O ficheiro guarda todas as perguntas, sequencialmente, segundo um formato pré-definido. Um exemplo de uma das perguntas é o seguinte:

```
15 - desporto - Which term describes any evidence directly exposing a
criminal suspect?
Helper1: [ 0,20]
Helper2: [ 1,00]
Helper3: [ 0,38;  0,71]

[0/0] Helper1:  0,06
[5/9] Helper2:  0,25 <<<<<
[2/5] Helper3:  0,03

Wrong Answer
```

Como podemos ver, é escrito o número da pergunta, seguido da categoria e da pergunta em si. Nas primeiras três linhas estão os nomes dos ajudantes presentes em jogo, e a informação enviada pelos outros jogadores sobre estes ajudantes. Neste caso, para o Helper1 só um jogador tem informação sobre a confiança (0.20), enquanto que para o Helper3 dois outros jogadores responderam com confiança de 0.38 e 0.71. No caso de ninguém ter informações sobre um Helper estes valores são substidos pela string "-".

Nas três linhas seguintes é apresentada a confiança calculada por este agente para cada *helper*. Antes de cada ajudante, está presente a informação de quantas perguntas este *helper* acertou no total de perguntas respondidas a este jogador. Assim, o Helper3 respondeu a 9 perguntas acertando em 5 ([5/9]). Este foi também o *helper* escolhido pois contêm a indicação à frente da sua confiança (representado por "<<<<<"). É também dito que a resposta está incorreta pela *string* "Wrong Answer". Para além disso os jogadores FIRE incluem uma string "[NC]" indicando que o *helper* é *newcomer* e pode ser escolhido (não está incluído neste exemplo por se tratar dum jogador SINALPHA).

Execução do programa

Os agentes devem ser criados na seguinte ordem:

1. *Helpers*
2. *Players*
3. *Presenter*

Para executar a aplicação, deve-se criar o agente *MainSetup*:

```
jade.Boot-gui -local-port 1234 -agents m:MainSetup
```

Este agente é responsável por criar 3 jogadores, 3 ajudantes e o apresentador. Caso se pretendam alterar os parâmetros passados a cada agente, é também possível alterar o ficheiro MainSetup.java.

No caso de se pretender correr 1 ou vários agentes em diferentes máquinas: numa das máquinas, deve criar-se o agente *MainSetup*, comentando aqueles agentes que não se pretende criar nessa máquina. O programa espera por um *input* após a criação dos *Helpers* e dos *Players*, de forma a que possam ser adicionados noutra(s) máquina(s). Assim sendo, se se pretender adicionar um *Helper* noutra máquina, por

```
jade.Boot -host 192.168.1.76 -port 1234 -container –agents
      HelperX:HelperAgent("desporto:70, cultura:50")
```

exemplo, deve executar-se:

O ip usado neste exemplo é apenas representativo, devendo ser alterado pelo ip da máquina onde foi executado o *MainSetup*. Os parâmetros, tal como mostra o exemplo, devem ser separados por vírgula (“,”).