Desenvolvimento do projeto HUMANI como ferramenta didática para a aprendizagem de linguagens de programação

Mauro da Rocha Xavier Neto
NIEDUC - Fatec – Ourinhos
Av. Vitalino Marcusso, 1400 – Ourinhos – SP
+55(14) 33261486
mauroxavier@casadosnerds.com.br

Elaine Pasqualini,
Silvia H. de Oliveira Santos
NIEDUC - Fatec – Ourinhos
Av. Vitalino Marcusso, 1400 – Ourinhos - SP
+55(14) 33261486
elainepasqualini@hotmail.com

Viviane de F. Bartholo Potenza
NIEDUC - Fatec – Ourinhos
Av. Vitalino Marcusso, 1400 – Ourinhos – SP
+55(14) 33261486
vbartholo@gmail.com

ABSTRACT
This study aims to create a method of interpretation of human language to computer language and generating flowcharts, through the development of an educational software that uses artificial intelligence techniques to perform these transcripts in real time. This software can assist in the understanding of early programming courses where students have learning difficulties. It is possible for students to observe human language to be transformed in C and Python, as well as viewing the results graphically. The commands that the software understands are arithmetic, conditions, among others.

RESUMO
Este estudo visa a criação de um método de interpretação da linguagem humana para a linguagem computacional e geração de fluxogramas, por meio do desenvolvimento de um software educacional que utiliza técnicas de inteligência artificial para realizar estas transcrições em tempo real. Esse software pode auxiliar na compreensão de disciplinas iniciais de programação em que os alunos têm dificuldades de entendimento e aprendizagem. É possível aos alunos observarem a linguagem humana a ser transformada nas linguagens C e Python, bem como a visualização do resultado graficamente. Os comandos que o *software* entende são aritméticos, condições, entre outros.

Categories and Subject Descriptors
D.3.m – Miscellaneous: Application of artificial intelligence without canonical algorithms.

I.2.7 [**Natural Language Processing**]: Language parsing and understanding, machine translation, text analysis.

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods.

E.1 [Structure Data]: Arrays, lists, stacks and queues.

General Terms
Algorithms, Languages, Performance.

Keywords
Artificial Intelligence, Computer Language, Natural Language, Interpretation, Software.

1. RESUMO EXPANDIDO
Com a expansão constante do mercado de desenvolvimento de *softwares*, também é proporcional o número de pessoas que procuram formação nesta área. Porém, é preciso compreender determinadas regras e procedimentos para iniciar o desenvolvimento de um *software*, por menor que seja. Neste momento, o ser humano que nunca teve contato com algoritmos computacionais pode acabar se afastando da possibilidade de buscar formação na área caso sinta dificuldades na aprendizagem. Simas [1] aponta os dados do Instituto Nacional de Estudos Anísio Teixeira (Inep), que indicaram em 2012 um alto índice de evasão para os cursos de Processamento de Informação (36%) e Ciência de Computação (32%), cujo um dos motivos para a desistência é a dificuldade do curso.

A distância que envolve a compreensão da linguagem computacional por parte dos seres humanos pode diminuir se houver um *software* capaz de realizar esta interpretação em tempo real, de forma didática, clara e prática. Dessa necessidade surgiu a ideia do desenvolvimento do projeto *Human Machine Natural Interpreter* – HUMANI (Interpretador Natural Homem Máquina).

Neste artigo é apresentada a metodologia utilizada na concepção do projeto HUMANI, qual o nível de maturidade e resultado foi atingido na conclusão da primeira fase de desenvolvimento.

Inicialmente foi definido que o sistema deveria cumprir os seguintes requisitos:

- Interpretar textos em linguagem natural para linguagem computacional em tempo real utilizando um CPU com arquitetura x86 a partir de 2 Ghz com núcleo único;
- Oferecer opções para diminuir a intensidade de uso do CPU, permitindo a utilização do *software* em microcomputadores obsoletos, sem abrir mão da didática;
- Compreender a linguagem natural com restrições mínimas de sintaxe;
- Ser tão fácil de utilizar a ponto de que os usuários poderiam iniciar suas atividades sem ler qualquer manual;
- Gerar fluxogramas;
- Ser multiplataforma (Linux, Mac OS e Windows);
- Possuir um ambiente visual agradável e auto ajustável;
- Ser capaz de rodar em uma resolução mínima de 1024x600 pixels;

- Oferecer personalizações fáceis de serem aplicadas.

Mesmo se tratando de requisitos iniciais, percebeu-se a complexidade no desenvolvimento do projeto, e, portanto, foi considerado adequado dividir o desenvolvimento em fases.

Para a conclusão da primeira fase foi decidido que o sistema deveria ser capaz de efetuar a interpretação em tempo real da linguagem natural para as linguagens de programação C e Python, bem como a geração de fluxogramas. Para que fosse possível alcançar este objetivo, o *software* deveria ser otimizado de modo a torná-lo capaz de identificar e comparar na menor fração de tempo possível as palavras de cada parágrafo do texto escrito em linguagem natural, permitindo atingir um nível de interpretação e compreensão de frases que possibilite a transcrição para uma linguagem de programação em tempo real.

Foi verificado que seria essencial a utilização de algoritmos baseados parcialmente em inteligência artificial e Processamento de Linguagem Natural (PLN), algo que apesar de solucionar certas questões para a criação do sistema, em contrapartida poderia comprometer a performance final do *software*.

A preocupação com a performance do *software* era constante no estágio de *brainstorm* do projeto, pois foi constatado que a reprodução de uma inteligência artificial cognitiva que compreendesse frases envolvendo objetos, estados, verbos e questões ambíguas tornaria o código imenso, complexo e extremamente lento para ser executado por uma única máquina em tempo real.

Para acelerar o início do desenvolvimento, na primeira fase foi descartado o emprego de algoritmos canônicos de inteligência artificial e análises avançadas de semântica e pragmática. Foi utilizada uma abordagem mais simples, criando um algoritmo que simula um comportamento básico, sem compromissos na interpretação de frases complexas.

Segundo Norvig e Russel [2], para ser possível afirmar que um programa pensa como um ser humano, é preciso ter alguma forma de determinar como os seres humanos pensam, sendo necessário identificar os componentes reais da mente humana. Para isso existem duas maneiras: a introspecção, captando os próprios pensamentos simultaneamente ao seu desenvolvimento, ou por meio de experimentos psicológicos. Após a teoria da mente estar definida, deve ser possível expressá-la como um programa. Partindo deste conceito, para iniciar o desenvolvimento da IA (Inteligência Artificial), o processo de introspecção e pequenos experimentos com perguntas e ações se demonstraram úteis.

Com o objetivo de estabelecer um ponto de partida, foi aplicado testes comuns em 20 crianças de 1 a 12 anos de idade, na intenção de compreender e replicar via *software* parte do comportamento humano. Os testes consistiam de perguntas envolvendo objetos, em conjunto com ações relacionadas como trocas, empréstimos, condições e imposições, mas sem nenhum embasamento científico ou psicológico, sendo criados a fim de representar conceitos matemáticos básicos (atribuição, subtração, adição e divisão) representados pela forma que os participantes interpretavam as frases e ações.

Os testes foram aplicados sem a percepção dos participantes para que os resultados fossem os mais naturais possíveis. O ambiente escolhido foi aquele em que a criança se encontrava no momento, seja em seu bairro ou residência.

Mesmo com diversas respostas apresentando baixa coesão de sintaxe e vocabulário restrito, foi possível definir algumas características importantes no desenvolvimento inicial da IA, além da introdução de alguns verbos e palavras que podem referenciar certas ações na grande maioria das vezes:

- Compreensão de posse: haver, ter, possuir, etc.;
- Compreensão de adição: adquirir, ficar, investir, comprar, aceitar, pegar, achar, encontrar, ganhar, receber, obter, etc.;
- Compreensão de subtração: dar, perder, vender, ceder, doar, emprestar, devolver, oferecer, etc.;
- Compreensão de quantidade e comparações: igual, maior, menor, diferente, etc.;
- Compreensão de multiplicação, divisão e fracionamento: vezes, dobro, triplo, metade, terço, etc.;
- Compreensão de possibilidades: se, talvez, quando, caso contrário, etc.;
- Compreensão de estados: frio, gelado, calmo, irritado, aberto, fechado, etc.;
- Compreensão de perguntas: quantos, com quantas, mostre, exiba, etc.;
- Compreensão de aprendizagem: entender, aprender, compreender, etc.

Os verbos obtidos no teste foram inseridos diretamente no código fonte interno do *software* em vetores que representam separadamente cada tipo de ação e possibilidade, servindo de base inicial para as análises léxica, sintática e semântica.

Após esta etapa, foi utilizado o princípio de agente lógico para a interpretação da linguagem natural para linguagem computacional, e como o agente encarregado desta tarefa se tornou o principal núcleo do *software*, foi considerado importante tratá-lo como uma biblioteca distinta de código de programação, recebendo o nome de *Educational Linguistic Intelligence with Speculative Analysis* – ELISA (Inteligência linguística educacional com análise especulativa).

A IA ELISA é responsável por toda interpretação da linguagem natural para linguagem computacional do projeto HUMANI, e seu código foi separado de tal forma que é considerado uma biblioteca útil a ponto de permitir sua utilização em outros *softwares* futuramente.

Além dessa importância, a IA ELISA é quem define a performance de todo o projeto, pois é em seu bloco lógico de programação que estão contidos os processos heurísticos e atalhos para realizar a interpretação no menor tempo possível.

De acordo com os verbos inseridos no algoritmo da primeira fase do projeto, foi permitido definir quais das etapas da abordagem clássica da PLN seriam implementadas e em que nível seriam tratadas:

- Tokenização: apenas a separação baseada em espaços em branco entre palavras;
- Análise léxica: baseada nas palavras obtidas nos testes com as crianças;
- Análise sintática: não é baseada na sintaxe plena da frase em língua portuguesa. A análise parte da posição das palavras reconhecidas pelo dicionário léxico, e então o processamento ocorre nas proximidades desta região;
- Análise semântica: os objetos, pessoas e características são reconhecidos de acordo o resultado da análise sintática, que

por si só é limitada ao dicionário léxico. Como último recurso, conceitos heurísticos são utilizados para frases que não aparentam sentido ou caso ocorra alguma interpretação mal sucedida;

- Análise pragmática: não será implementada.

Como a linguagem natural pode fazer referências a fatos do mundo real, que seguem um padrão não-monotônico, decidiu-se portanto, aplicar este mesmo modelo de sistema na criação da IA. Segundo Rich [3] "ao projetar sistemas não-monotônicos, é importante assegurar que o sistema não gaste todo o tempo propagando mudanças", portanto, o índice de conferências para a propagação de mudanças foi limitado para até três passagens no texto integral, a fim de alcançar o mínimo de performance desejada.

Outro ponto chave para manter a performance foi a utilização de avaliações heurísticas. Whitby [4] explica a heurística como sendo um método empírico, uma suposição ou pista para a solução de um problema, e a resume como sendo uma maneira de colocar elementos do mundo real para ajudar na solução de um problema matemático. Graças às especulações sobre os resultados foi possível atingir uma melhor performance, e apesar de ocasionar uma maior margem de erros na interpretação, o ganho de velocidade ainda compensou esta deficiência.

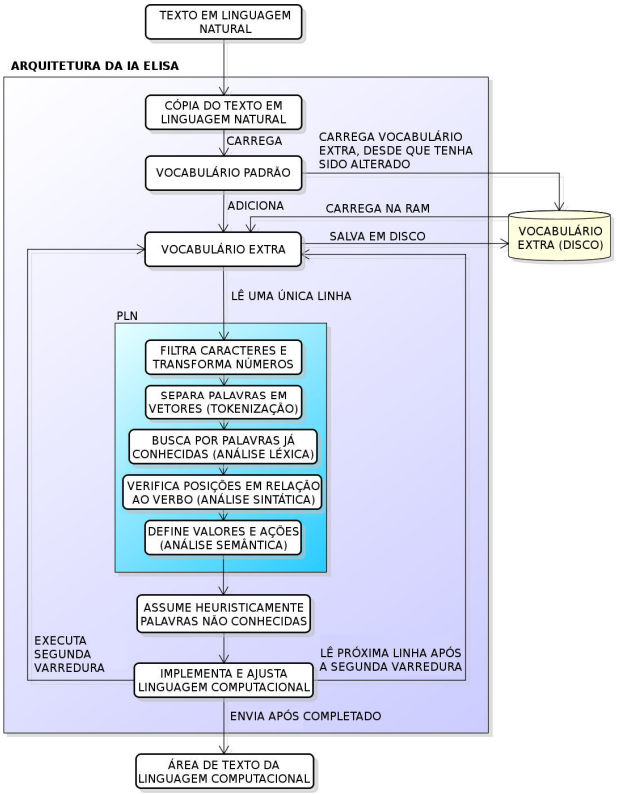

Figura 1. Representação da Arquitetura da IA ELISA

A Figura 1 mostra a arquitetura da IA ELISA. Tudo é processado em tempo real, seguindo as etapas abaixo:

- O texto é passado ao interpretador;
- É chamado o vocabulário principal, que já está presente na RAM desde a inicialização do *software;*
- É carregado o vocabulário extra do disco, desde que ele esteja diferente do vocabulário extra presente na RAM;
  - Uma única linha entra na etapa de interpretação;
  - A linha é filtrada: caracteres fora da tabela ASCII são apagados, acentos são removidos, todas as letras se tornam minúsculas, palavras que fazem referência a operações matemáticas são substituídas e os números escritos por extenso são convertidos para números arábicos;
  - São separadas e enumeradas cada palavra, número e símbolo da linha (tokenização);
  - É verificado na frase se há verbos que caracterizam ações específicas (análise léxica);
  - Ao encontrar um verbo conhecido, é examinado se ele realmente está relacionado com a ação pré-programada (análise sintática), caso contrário é efetuada uma nova pesquisa. Se nesta segunda varredura ainda não for encontrado palavras chaves ou ocorrências que façam sentido, a IA parte para a transcrição heurística e transforma palavras e símbolos conhecidos em composições matemáticas;
  - Se a definição do verbo e ação for encontrada com sucesso, o interpretador busca na frase por elementos não verbais já conhecidos e declarados em frases anteriores, como nomes e objetos, a fim de atribuir valores e ações (análise semântica);
  - Caso a definição do verbo e ação seja encontrada com sucesso, mas não existam elementos não verbais conhecidos, o interpretador faz suposições por meio de regras heurísticas para tentar encontrar pessoas e objetos, a fim de declarar as classes na linguagem de programação. Se for encontrado qualquer objeto, a IA remove o plural da palavra;
  - Se mesmo assim o contexto da frase ainda não retornar algo que faça algum sentido mínimo para o interpretador, ele ignora os erros e continua interpretando as linhas seguintes;
  - É criado uma linha em linguagem computacional, que pode ser parcial ou integral;
  - O interpretador faz uma nova leitura integral do texto a fim de tentar corrigir as frases e elementos que não foram interpretados adequadamente. O grau de eficiência desta vez é maior graças a utilização de um vocabulário aprimorado baseado na pesquisa do texto completo (análise léxica e sintática);
  - Após ser completada todas as operações, a última chamada de interpretação faz as modificações finais na linguagem computacional e devolve para a interface do usuário o resultado.

A Figura 2 mostra um exemplo de interpretação de uma linha com reconhecimento de uma nova pessoa e objeto.

Linha 1:	"Amanda tem cinco maçãs"	
Filtra frase:	amanda tem 5 macas	
Separação numérica:	1 = amanda	| 2 = tem
	3 = "5"	| 4 = macas
Verbo encontrado:	2 = tem	
Quem tem?	2 - 1 = 1 = amanda	
Quantos?	2 + 1 = 3 = "5"	
O quê?	3 + 1 = 4 = macas → s/ plural = maca	
Criar objeto em C?	Sim	

Construção em C:

```
amanda.maca = 5;
```

Figura 2: Exemplo de interpretação de uma linha

O exemplo da Figura 2 mostra onde as palavras sublinhadas representam um vocabulário já conhecido pela IA. Observe o passo a passo da interpretação, com pontos chave em negrito:

- O interpretador lê a frase "Amanda tem cinco maçãs";
- A frase é filtrada, removendo acentuações e deixando todas as letras minúsculas: "amanda tem cinco macas";
- Depois os números por extenso são convertidos para arábicos: "amanda tem 5 macas";
- As palavras são enumeradas e armazenadas em vetores (**tokenização**), por exemplo, a palavra "amanda" se torna o número 1, a palavra "tem" se torna o número 2, a palavra "5" se torna o número 3 e assim por diante. Observe que o número 5 presente na frase é tratado como palavra pelo *software* no ato da interpretação;
- É procurado em cada vetor se há verbos já conhecidos (**análise léxica**), neste caso, a palavra "tem" é marcada (número 2);
- Ao perceber o verbo que representa posse (ter), é verificado antes desta palavra a presença de alguém (**análise sintática**), para isso é efetuado a leitura de um número anterior ao vetor do verbo reconhecido, resultado no número 1, em que se encontra a palavra "amanda" (**análise semântica**);
- Após o verbo, existe grande possibilidade de existir a quantidade de objetos que estão em posse de "amanda", portanto é verificado o número posterior ao vetor da palavra "tem", sendo efetuado uma tentativa de converter esta palavra para um número real, e em caso positivo este numero é considerado o correto (**heurística**). Neste caso bem sucedido, foi obtido a palavra "5", que na linguagem computacional será convertido para número;
- Ao perceber que existe uma ligação com grandes chances de acerto por ter encontrado uma pessoa, um verbo e a quantidade de algo, o *software* assume a palavra seguinte como objeto. Neste caso, obtem-se a palavra "macas" (**análise semântica com heurística**);
- A palavra registrada como objeto é passada para o singular: "maca";
- O *software* marca a necessidade de criar uma classe na linguagem computacional após concluir a interpretação de
- Todo o texto, portanto, o programa criado terá o objeto "amanda", considerando a "maca" como atributo;
- Por fim é incluído a linha de interpretação em linguagem computacional, neste exemplo representando uma atribuição escrita em linguagem C: `amanda.maca = 5`.

Conforme a Figura 3, é perceptível que graças a interpretação da linha anterior (Figura 2), a IA executa a operação de forma mais eficiente, com o reconhecimento de 100% das palavras. Esta interpretação foi obtida graças aos resultados dos testes efetuados com as crianças, aplicando o conceito de posse de objetos e reproduzindo esta natureza na lógica da ELISA.

Linha 2:	"Amanda possui quantas maçãs?"	
Filtra frase:	amanda possui quantas macas	
Separação numérica:	1 = amanda	| 2 = possui
	3 = quantas	| 4 = macas
	5 = ?	
Verbo encontrado:	2 = possui	
Pergunta encontrada:	3 = quantas / 5 = ?	
Elementos encontrados:	amanda / maca	
Objeto encontrado:	`amanda.maca`	

Construção em C:

```
printf("Amanda possui %i macas",amanda.maca);
```

Figura 3: Interpretação com reconhecimento integral

A IA ELISA é capaz de perceber a diferença entre ter e adquirir algo. Por exemplo, na Figura 2 se a frase fosse "Amanda comprou cinco maçãs", o resultado em linguagem de programação seria: `amanda.maca = amanda.maca + 5;`

Além de posse, aquisição e subtração de objetos, a IA ELISA compreende algumas condições e comparações (Figura 4).

Linha 1: Ricardo não tem maçãs
Linha 2: João tem sete maçãs e deu duas para Ricardo
Linha 3: Se Ricardo tem mais de uma maçã, com quantas ele ficou?

Construção completa em C (sem inclusão de bibliotecas):

```
struct Sricardo                (1)
{
   int maca;                   (1)
};
struct Sjoao                   (2)
{
   int maca;                   (2)
};

struct Sricardo ricardo ;      (1)
struct Sjoao joao ;            (2)

int main ()
{
  ricardo.maca = 0 ;                    (1)
  joao.maca = 7 ;                       (2)
  joao.maca = joao.maca - 2 ;           (2)
  ricardo.maca = ricardo.maca + 2 ;     (2)
  if ( ricardo.maca > 1 )
(3)
  {
    printf ("ricardo ficou com %i
maca\n",ricardo.maca);
  }
}
```

Figura 4: Tratamento de condição

Existem também as chamadas condições aninhadas, em que uma condição está contida dentro da outra, o qual a ELISA é capaz de interpretar corretamente até cinco níveis de aninhamento, porém concluiu-se que a IA não deveria ir além disso para poupar performance e manter a identação do código fonte gerado.

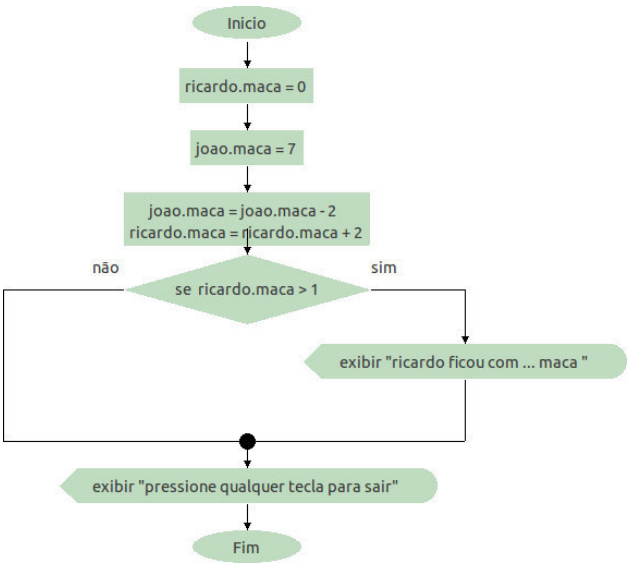

Figura 5: Fluxograma

Para oferecer ao usuário uma melhor compreensão sobre o funcionamento de cada etapa da linguagem de programação, optou-se pela geração de fluxogramas a fim de se obter uma representação gráfica da estrutura. Pode-se observar na Figura 5 o fluxograma criado pela interpretação da linguagem de programação exibida na Figura 4.

Para compor o fluxograma, a IA ELISA após a segunda passagem de interpretação de uma linha de linguagem natural forma uma linha de linguagem de programação, sendo então este resultado conferido e usado de base para gerar um bloco de fluxograma.

Após a finalização do fluxograma, uma outra passagem é efetuada para verificar e desenhar as setas de fluxo e pontos de ligação entre cada bloco.

Para cumprir o foco didático do projeto, a construção do fluxograma ocorre em tempo real em uma janela distinta da interface padrão, permitindo ao usuário a visualização das modificações nos blocos ao mesmo tempo em que a linguagem natural e linguagem de programação são alteradas.

A interface do HUMANI é responsável pela coordenação das interações do usuário com o sistema, requisições geradas, ações nas áreas de texto e comunicação interna com a IA. Conforme a Figura 6, a interface é apresentada ao usuário por meio de áreas de texto que compõe a linguagem natural, programa gerado, mensagens do sistema, menu principal e ícones de acesso rápido.

Note que há quatro linhas selecionadas no programa apresentadas na região direita da janela. Esta marcação é automaticamente efetuada pela IA ELISA, que relaciona a linha de texto da linguagem natural em que o cursor estiver posicionado com a(s) linha(s) de linguagem computacional gerada(s) a partir daquela posição.

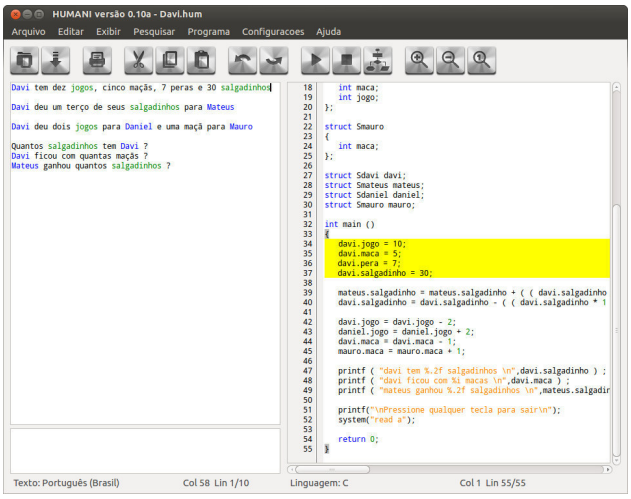

Figura 6: Janela de interface visual do HUMANI

Outro detalhe a ser percebido na mesma figura, é que há duas linhas na área de texto da linguagem computacional que não estão sendo exibidas integralmente. Se o monitor do usuário não oferecer resolução suficiente para resolver o problema, é possível utilizar a barra de rolagem, ou pode-se mudar o estilo da tela, que neste caso poderia ser adequado optar pelo *layout* horizontal duplo, conforme exibido na Figura 7.

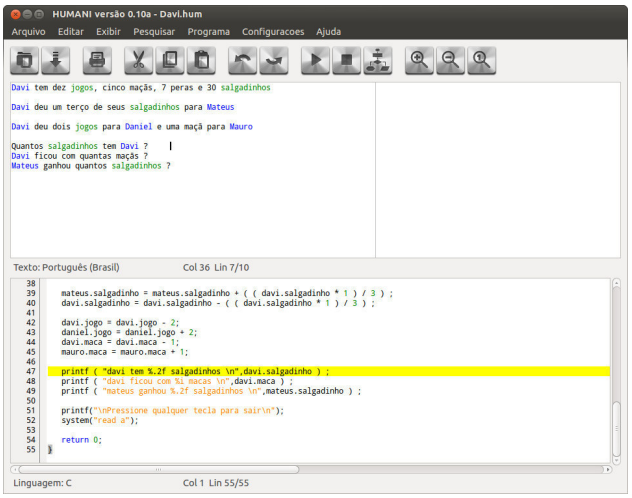

Figura 7: Interface visual com layout horizontal duplo

Por meio da interface padrão é possível acessar algumas janelas de configuração do sistema, dando liberdade ao usuário para alterar os temas, tamanhos das fontes, disposição do layout, e linguagem de programação a ser gerada.

Muitas das opções ainda não tem sua funcionalidade implementada, porém, para a conclusão da primeira fase do projeto foi decido implementar por completo a configuração do nível de interpretação, que pode ser resumido em:

- Em tempo real: efetua a conversão a cada letra. Recomendado para microcomputadores a partir de dois núcleos a 1.2 Ghz.
- Moderada: efetua a conversão a cada parágrafo.
- Na execução: efetua a conversão antes de compilar o programa. Recomendado para microcomputadores modestos ou obsoletos.

Para que o sistema execute o código gerado, é necessário configurar a ferramenta externa que será utilizada. Optou-se pelo emprego de softwares livres como o GCC e Python, sendo que em ambientes Linux basta que o usuário possua os pacotes instalados para que o HUMANI utilize-os automaticamente, e no caso de sistemas Windows, o usuário poderá apontar o caminho dos *softwares* externos.

É possível afirmar que a primeira fase do desenvolvimento do projeto HUMANI atingiu seu objetivo, pois o *software* desenvolvido é capaz de realizar tarefas em tempo real, como a interpretação da linguagem natural para linguagem de programação e criação de fluxogramas, incluindo a possibilidade de executar os códigos gerados.

A estrutura do código começou de forma singular e pouco tempo após se tornou um *software* modular capaz de abraçar novos blocos de código, o que futuramente pode incluir agentes de IA, reprodutores de mídia, editores de imagem etc. Além da codificação interna, é possível integrar o projeto com outros *sofwares* externos, seja para objetivo acadêmico, corporativo ou de entretenimento.

O agente ELISA, apesar de não utilizar algoritmos canônicos e consagrados de IA, atingiu performance e eficiência satisfatórias para a primeira fase do projeto.

Como resultado foi desenvolvido um *software* que oferece um método de ensino para pessoas que nunca programaram, permitindo que o usuário observe o código fonte criado e verifique passo a passo a formação do *software* com seu fluxo de processamento e decisões.

Ainda há muito o que ser desenvolvido para as próximas fases do projeto, incluindo aspectos importantes da interpretação, como avaliação de estados (frio, quente, aberto, fechado etc), aprendizagem de palavras por interação do usuário, gráficos cartesianos etc, porém, acredita-se que o projeto HUMANI esteja apenas começando.

Um complemento importante para aprimorar exponencialmente a capacidade de interpretação da ELISA sem comprometer a performance, é a criação de um outro agente de IA que possibilite um armazenamento online global e centralizado de todos os conceitos aprendidos em cada instância do HUMANI utilizado por cada usuário. O nome deste agente de IA é BRAINIAC (*Base of Rational Analysis of Important Natural Inferences with Advanced Comparisons* - base de análise racional de inferências naturais importantes com comparações avançadas).

A grande vantagem da existência de um agente de IA online e global é que os dados poderão ser processados sem a preocupação com a performance ou realização de operações em tempo real, sendo assim, BRAINIAC poderá ser executado por um parque de máquinas multiprocessadas. O objetivo desta IA é absorver uma grande quantidade de árvores de possibilidades e palavras, e por meio de técnicas de mineração de dados (*data mining*) será gerado um dicionário de dados otimizado, que será baixado localmente pela interface de comunicação do HUMANI e estará pronto para a ELISA consultar em tempo real, mantendo a velocidade do sistema e aumentando profundamente as chances de sucesso na interpretação.

O usuário poderá optar pela colaboração com a ampliação da base de conhecimentos, e em caso positivo, a cada encerramento de sessão será efetuado o *upload* compactado de suas experiências para a base de conhecimento do BRAINIAC.

Conclui-se portanto que o projeto HUMANI obteve sucesso em realizar seu objetivo inicial proposto em sua primeira fase de desenvolvimento, e apresentou resultados que permitem a expansão do projeto futuramente para abraçar novos horizontes.

2. REFERÊNCIAS

[1] Simas, Anna. Gazeta do Povo, As graduações campeãs de desistência. Disponível em: http://www.gazetadopovo.com.br/vida-universidade/nocampus/conteudo.phtml?id=1248860. Acesso em 3 set. 2014.

[2] Norvig, Peter ; Russel, Stuart J. (2010). Artificial Intelligence: a modern approach. Third Edition. New Jersey. Pearson Education, Inc., 2010.

[3] Rich, Elaine (1988). Inteligência artificial. Trad. Vasconcellos, Newton. São Paulo: McGraw-Hill, 1988.

[4] Whitby, Blay. Inteligência artificial: um Guia para Iniciantes. Trad. Blanc, Claudio. São Paulo: Madras, 2004.