MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
Seção de Engenharia de Computação / SE8

JOILSON CISNE DO NASCIMENTO

IMPLEMENTAÇÃO DE UMA INTERFACE GRÁFICA REMOTA PARA A REALIZAÇÃO DE EXPERIMENTOS DE APRENDIZADO DE MÁQUINA

Rio de Janeiro
2013

INSTITUTO MILITAR DE ENGENHARIA

JOILSON CISNE DO NASCIMENTO

IMPLEMENTAÇÃO DE UMA INTERFACE GRÁFICA REMOTA PARA A REALIZAÇÃO DE EXPERIMENTOS DE APRENDIZADO DE MÁQUINA

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia.

Orientador: Prof. Julio Cesar Duarte – D.C.

Rio de Janeiro
2013

INSTITUTO MILITAR DE ENGENHARIA

JOILSON CISNE DO NASCIMENTO

IMPLEMENTAÇÃO DE UMA INTERFACE GRÁFICA REMOTA PARA A REALIZAÇÃO DE EXPERIMENTOS DE APRENDIZADO DE MÁQUINA

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia.
Orientador: Prof. Julio Cesar Duarte – D. C.

Aprovada em 19 de agosto de 2013 pela seguinte Banca Examinadora:

_____

Prof. Julio Cesar Duarte – D. C., do IME

_____

Prof. Ricardo Choren Noya – D. C., do IME

_____

Prof.ª. Maria Claudia Reis Cavalcanti – D. C., do IME

Rio de Janeiro
2013

SUMÁRIO

RESUMO

Aprendizado de Máquina é um dos ramos da Inteligência Artificial que mais recebe destaque nos dias atuais. Baseados na aplicação de diversos algoritmos, os experimentos nesse campo mostram-se bastante importantes para a consagração das pesquisas feitas e algoritmos desenvolvidos. Para realmente garantir que há aprendizado, e não memorização, o uso da validação cruzada mostra-se imprescindível.

O presente trabalho desenvolve uma interface gráfica para acesso remoto de um sistema, previamente desenvolvido, de aprendizado de máquina. Tal website possibilitará a passagem dos parâmetros apropriados à utilização do sistema, instalado em um servidor remoto, e a visualização dos resultados pelo usuário, seja na própria tela, seja via e-mail.

A criação de ferramentas que auxiliem na execução de experimentos em aprendizado de máquina é um grande impulso para a área. A contribuição principal deste trabalho está no desenvolvimento de uma ferramenta que pode ser usada à distância, aumentando a flexibilidade de uso pelo usuário.

ABSTRACT

Machine Learning is a branch of Artificial Intelligence that most receives attention nowadays. Based on the application of different algorithms, experiments in this field are very important for the consecration of the research done and algorithms developed. To really ensure there is learning, not memorization, the use of cross validation proves indispensable.

This paper develops a graphical interface for remote access to a machine learning system, previously developed. This website will allow the passage of appropriate parameters to use the system, installed on a remote server, and the visualization of the results by the user, either on the screen itself, or via e-mail.

The creation of tools that assist in the execution of experiments in machine learning is a great boost for the area. The main contribution of this work is to develop a tool that can be used remotely, increasing the flexibility of use by the user.

1 INTRODUÇÃO

Neste trabalho abordam-se os temas de interoperabilidade entre sistemas escritos em diferentes linguagens de programação e do desenvolvimento de aplicações *web* para um sistema já desenvolvido.

Utilizando os conceitos supracitados, este trabalho pretende prover o acesso remoto a um sistema de algoritmos de aprendizado de máquina, o FAMA.

1.1 MOTIVAÇÃO

O acesso remoto via internet eleva qualquer aplicação a outro nível de utilização, em termos de alcance em número de usuários e, além disso, em termos de facilidade de acesso. Prova disso é o fato de que hoje, no Brasil, o número pessoas que acessam a internet diariamente supera os 70 milhões, e, além disso, 27 milhões de pessoas utilizam smartphones e, portanto, qualquer sistema disponibilizado na rede possui um vasto número de possíveis usuários e acesso praticamente irrestrito.

O website desenvolvido nesse trabalho possibilitará o recolhimento da entrada de um usuário remoto, a passagem dos parâmetros necessários ao sistema, em C++, e o recebimento dos resultados de forma imediata, ou, se o tempo de processamento for longo, o envio do envio do resultado via e-mail. Assim, a importância desse trabalho se dá no fato que este website servirá de subsídio à implementação de qualquer website que tenha como objetivo o acesso remoto via internet a sistemas implementados em C++, aumentando a visibilidade e alcance do sistema.

1.2 OBJETIVO

O objetivo desse trabalho é a implementação de um website para o acesso remoto a um sistema de aprendizado de máquina. Como alguns dos algoritmos usados pode apresentar um longo tempo de execução, pretende-se criar uma funcionalidade para envio dos resultados via e-mail, tornando assim o uso do sistema mais conveniente ao usuário. Além do acesso remoto, esse *website* facilitará o acesso a uma ferramenta da área de Inteligência Artificial, podendo ser usado até uma pessoa leiga no assunto.

Esse projeto tem uma grande relevância no contexto de pesquisa no IME, visto que Inteligência Artificial ainda é uma área pouco explorada dentro do âmbito graduação. E esse trabalho é mais um a contribuir com esforços para o desenvolvimento da área dentro da instituição.

1.3 METODOLOGIA

Este trabalho está estruturado em três partes: estudo e decisão da melhor solução para a implementação do website, a implementação propriamente dita do website, de acordo com solução escolhida, e teste de funcionamento do sistema para diversas aplicações, como Processamento de Linguagem Natural (PLN) e identificação de *malwares*.

A escolha da solução adotada, principalmente no que se refere a linguagem de programação usada, foi feita na primeira parte por meio da comparação entre algumas soluções disponíveis. Foi levado em consideração tanto as necessidades técnicas do projeto, como o tempo disponível para sua implementação.

A segunda e terceira partes tem um relacionamento muito estreito. À medida em que foram implementados com sucesso os primeiros protótipos de uma funcionalidade, passou-se à implementação daquela funcionalidade no sistema final.

1.4 ESTRUTURA DA MONOGRAFIA

Este trabalho está estruturado em sete capítulos. O capítulo dois discorrerá sobre o *framework* (FAMA), para o qual será desenvolvida a interface para acesso remoto.

O terceiro capítulo discorrerá sobre as abordagens que poderão ser usadas na resolução do problema proposto. Dando um enfoque na escolha da linguagem de programação a ser usada durante o decorrer do trabalho.

O quarto capítulo apresentará a experimentação feita e os resultados obtidos.

O quinto capítulo descreverá as funcionalidades implementadas no *website* (FAMA-*Online*) e como o usuário deve proceder para a sua utilização.

O sexto capítulo apresenta um cronograma das atividades do trabalho.

O sétimo capítulo apresentará as conclusões e propostas de trabalhos futuros.

2 *FRAMEWORK* DE APRENDIZADO DE MÁQUINA (FAMA)

Segundo (NASCIMENTO e OLIVEIRA, 2012), o *framework* desenvolvido tem a função de classificar textos usando algoritmos de Aprendizado de Máquina. Para isso, o FAMA recebe um arquivo de texto já processado em atributos (*corpus*) e com uma classificação prévia de cada instância executada por um sistema especialista de base humana.

A produção do FAMA ocorreu na plataforma C++, onde foram criadas classes especializadas. A execução do programa se dá pela troca de mensagens entre essas classes. O *framework* possibilitou a instanciação de três abordagens no qual foram implementados diferentes algoritmos de treinamento, um é bem simples e classifica apenas baseado na classificação mais comum de uma palavra (MaisProvavel), outro age baseado em diagramas de estados de classificações possíveis (HMM) (RABINER, 1989) e o último refina a classificação de outro método aplicando ao *corpus* um conjunto de regras otimizadas (TBL) (BRILL, 1992). A figura 2.1 mostra um diagrama UML da estrutura original do *framework*, apresentando as interfaces Avaliador, Treinador e Classificador, e a classe abstrata Corpus.

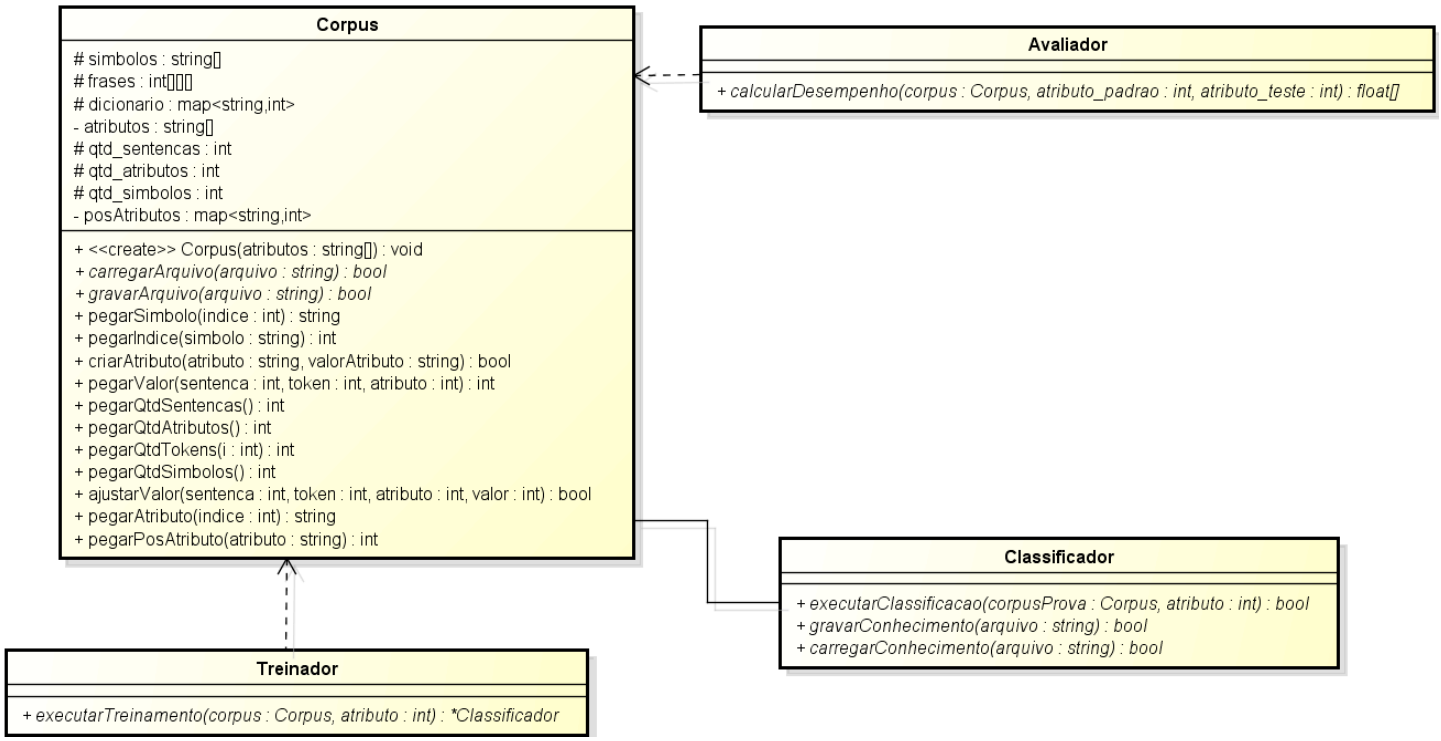

FIG. 2.1 – Classes abstratas do *framework*.

3 ABORDAGENS

Este capítulo trata de algumas das possíveis linhas de ação que podem ser adotadas para a resolução do problema proposto de criação de uma interface para acesso remoto do FAMA. Como o exposto nesse capítulo, pretende-se auxiliar a tomada de decisão sobre as etapas futuras do projeto, como linguagem de programação e tecnologias a serem utilizadas.

3.1 LINGUAGEM DE PROGRAMAÇÃO

Com o objetivo, dentre outros, de ganho de desempenho, o sistema de classificação morfossintática a ser acessado remotamente foi desenvolvido em C++. Por isso, logo num primeiro momento, descartou-se a possibilidade de conversão desse sistema para alguma linguagem que fornecesse uma maior facilidade e rapidez na implementação de uma página *web*. Isso porque tais linguagens são de mais alto nível do que C++, proporcionando um rápido desenvolvimento por parte do programador, em detrimento do desempenho global do sistema.

Procurou-se então, dentro de algumas das linguagens mais conhecidas por suas facilidades para o desenvolvimento *web* (C#, Python e Java), analisar qual o modo como cada uma deles oferece interoperabilidade com a linguagem C++.

3.1.1 INTEROPERABILIDADE C#/C++

A linguagem C# pertence ao conjunto maior chamado plataforma *.NET*. São três as principais formas de interoperabilidade na plataforma *.NET*: *Platform Invoke*, ou *P/Invoke*, é uma tecnologia de interoperabilidade de código gerenciado para nativo que permite chamar APIs nativas no estilo C de código gerenciado; a

interoperabilidade COM é uma tecnologia que permite consumir interfaces COM nativas de código gerenciado ou exportar essas interfaces a partir de APIs gerenciadas; por último, a C++/CLI (anteriormente chamada de C++ gerenciado), com a qual é possível criar *assemblies* que contêm uma combinação de código compilado C++ gerenciado e nativo e funciona como uma ponte entre o código gerenciado e o nativo.

3.1.1.1 P/INVOKE

Sendo a mais simples das três tecnologias citadas, a *P/Invoke* foi projetada para permitir acesso gerenciado a APIs no estilo C. Ela pode ser uma ótima escolha se houver algumas APIs para encapsular e se as respectivas assinaturas não forem muito complexas. No entanto, o uso da *P/Invoke* fica consideravelmente mais difícil se as APIs não gerenciadas têm muitos argumentos que não possuem bons equivalentes gerenciados, como estruturas de tamanho variável, ponteiros (*p) nulos, uniões sobrepostas etc.

Existem alguns recursos valiosos que podem facilitar o uso de *P/Invoke*. Por exemplo, uma ferramenta geradora de assinaturas *P/Invoke* chamada Assistente de Interoperabilidade de *P/Invoke*, que automaticamente cria assinaturas para APIs nativas com base em um arquivo de cabeçalho.

3.1.1.2 INTEROPERABILIDADE COM

A interoperabilidade *COM* (Component Object Model) pode ser uma ótima solução, caso já se esteja usando *COM* no aplicativo ou como modelo de extensibilidade. Também é a forma mais fácil de manter total fidelidade da semântica *COM* entre código gerenciado e nativo. Especificamente, a interoperabilidade *COM* é uma excelente opção se se deseja interoperar com um componente baseado no Visual Basic 6.0, uma vez que o *CLR (Common Language Runtime)* segue basicamente as mesmas regras de *COM* que o Visual Basic 6.0.

3.1.1.3 C++/CLI

Sendo utilizado como uma ponte entre códigos nativos e gerenciados, a C++/CLI é uma linguagem da plataforma *.NET* que permite a compilação de código C++ gerenciado e nativo dentro do mesmo *assembly* e fazer chamadas C++ padrão entre as duas partes do *assembly*. Ao usar C++/CLI, pode-se escolher qual parte do *assembly* deve ser de código gerenciado e qual deve ser de código nativo. O *assembly* resultante é uma combinação de *MSIL* (*Microsoft Intermediate Language*, encontrada em todos os *assemblies* gerenciados) e código de *assembly* nativo. A C++/CLI é uma tecnologia de interoperabilidade que permite controle total sobre o limite de interoperabilidade. O aspecto negativo é que ela obriga ao desenvolvedor a assumir o controle quase que total do limite.

A C++/CLI poderá ser uma boa ponte se a verificação de tipo estático for necessária, se o desempenho estrito for uma exigência e se se precisa de uma finalização mais previsível. Caso P/Invoke ou a interoperabilidade COM atenda às suas necessidades, no geral elas são mais fáceis de usar, principalmente se os desenvolvedores não estão familiarizados com a linguagem.

3.1.2 INTEROPERABILIDADE Python/C++

Assim como em C# também é possível promover interoperabilidade entre Python e C/C++. Das alternativas pesquisadas, destacam-se três: através de linguagens *Pyrex* (ou Cython, uma versão melhorada da *Pyrex*); usando a biblioteca *ctypes*; ou ainda, através da biblioteca *Boost.Python*.

3.1.2.1 PYREX/CYTHON

A primeira forma de interoperabilidade pesquisada trata do uso da *Pyrex*, uma linguagem *Python-like* que permite a implementação de módulos de extensão C para Python. Em outra palavras, pode-se descrevê-la como Python com tipos de dados

de C. Já a *Cython* é uma linguagem baseada em *Pyrex*, que combina Python e C, permitindo entre outras funcionalidades: escrever código em Python que chama e é chamado por código nativo em C ou C++ em qualquer ponto; usar *debugging* combinado a nível de código fonte para encontrar erros no seu código em Python, *Cython* e C. A linguagem *Cython* é um superconjunto da linguagem Python que adicionalmente suporta chamadas a funções em C e declaração de tipos C em variáveis e atributos de classe.

3.1.2.2 BIBLIOTECA CTYPES

De acordo com a documentação de Python, *ctypes* exporta os objetos *cdll*, *windll* and oledll (os dois últimos somente em ambiente Windows), para carregar *DLLs*. Então, utilizando o método *LoadLibrary* de *cdll*, pode-se carregar a *DLL* desejada dentro do código em Python. A partir de então, pode-se acessar os métodos providos pela biblioteca. A biblioteca *ctypes*, como o próprio nome já sugere, provê tipos de dados compatíveis com C, e permite a chamada de funções em *DLLs* ou bibliotecas compartilhadas. Ela pode ser usada para envolver essas bibliotecas em Python puro.

3.1.2.3 BOOST.PYTHON

De acordo com seu criador Dave Abrahams, *Boost.Python* é uma biblioteca que fornece uma suave interoperabilidade entre C++ e Ptyhon. Atualmente na segunda versão, essa biblioteca fornece entre outras funcionalidades, suporte: a referência e ponteiros; coerção de tipo globalmente registrados; tradução de exceções de C++ para Python; argumentos *default*; e manipulação de objetos Python em C++.

Essa ferramenta foi projetada criar *wrappers* sobre interfaces em C++, de forma tal que não seja necessário realizar nenhuma alteração no código nativo em C++. Ideal para usar na integração de bibliotecas de terceiros, as quais não podem ter seu código alterado.

3.1.3 INTEROPERABILIDADE Java/C++

Java Native Interface (JNI) fornece um suporte, em tempo de compilação e execução, a chamadas de código nativo (código não escrito em Java), a partir de um programa em Java.

Em tempo de compilação, *JNI* define como será o mapeamento entre os tipos de dados em Java e os tipos de dados em C/C++ (código nativo tratado no presente trabalho). Os programas em C/C++ recebem esta informação dos arquivos de cabeçalho do *JNI*.

Em tempo de execução, *JNI* permite que os objetos Java sejam passados para o código em C/C++, e permite também que o código em C/C++ acesse as propriedades e métodos Java.

Para fazer chamadas ao código nativo a partir de Java, deve-se seguir três passos:

1. Escrever uma declaração do código nativo em Java;
2. Criar um arquivo de cabeçalho que será usado pelo código nativo. Este cabeçalho deve conter uma série de argumentos e declarações que definem como as duas linguagens irão se comunicar;
3. Implementar o método nativo em C/C++. Esta função usa o arquivo de cabeçalho do passo 2, faz as chamadas necessárias para as funções da biblioteca, e retorna os resultados para o programa em Java.

3.2 COMPARAÇÃO

Além das funcionalidades exigidas para a eficaz resolução do problema tratado neste trabalho – como: prover uma comunicação eficaz entre o FAMA (C++) e o *website* desenvolvido (C#); e agilizar o desenvolvimento *web* – foi levado em conta, em grande medida, aspectos subjetivos do autor, tais como: Experiência com a linguagem; interesse pela linguagem; e experiência com a *IDE* utilizada. Tais aspectos foram considerados devido ao relativo curto tempo disponível para a

realização do trabalho. Com isso evitar-se-ia, por exemplo, o gasto de tempo para a familiarização com uma outra linguagem de programação não dominada pelo autor.

A tabela comparativa (TAB. 3.1) mostra que os aspectos subjetivos supracitados foram decisivos na escolha da linguagem adotada para a implementação do sistema proposto.

TAB. 3. 1 – Tabela comparativa entre as possíveis linguagens.

Linguagem	***Interoperabilidade***	***Web***	***Experiência***	***Interesse***	***IDE***
C#	Alta	Alta	Alta	Alta	Alta
Python	Alta	Média	Baixa	Baixa	Baixa
Java	Alta	Média	Média	Baixa	Média

Os aspectos de comparação foram cinco: Interoperabilidade – capacidade da linguagem de prover uma correta comunicação com C++; *Web* – agilidade fornecida pela linguagem para a criação de páginas *web*; Experiência – relacionado com a experiência do autor no uso da linguagem; Interesse – que reflete o interesse pessoal do autor em aprofundar seus conhecimento na linguagem; e *IDE* – experiência do autor com *IDEs* mais comuns disponíveis para a linguagem.

A cada um dos aspectos acima descritos foi atribuído um grau de prioridade. Os possíveis graus de prioridade eram: Alta – a linguagem apresenta grande relevância considerando o aspecto analisado; Média – a linguagem apresenta média relevância; Baixa – a linguagem apresenta baixa relevância.

Pelo exposto acima, concluiu-se pela utilização da linguagem de programação C#, em detrimento de Python e Java, para início dos trabalhos de implementação.

4 EXPERIMENTOS

A partir da definição da linguagem a ser utilizada, pôde-se dar início aos primeiros experimentos.

A parte experimental foi dividida em quatro partes:

1. Criação da classe *Calculadora*, em C++, que possui um único método público *somar*, que recebe dois inteiros de parâmetro e retorna a soma desses inteiros (ver FIG. 4.1);

```
1  int Calculadora::somar(int a, int b)
2  {
3       return a + b;
4  }
```

FIG. 4.1 – Definição do método *somar* da classe *Calculadora*.

2. Criação de uma DLL (CppClassDll) que exporta o método *somar* da classe *Calculadora* (ver FIG. 4.2)*;*

```
1  extern "C" _declspec(dllexport) int somar(int a, int b)
2  {
3       return Calculadora().somar(a, b)
4  }
```

FIG. 4.2 – *DLL* que exporta o método *somar* da classe *Calculadora*.

3. Criação de uma aplicação *console*, em C#, que chama o método *somar* da DLL em C++ e retorna o valor correto da soma de dois inteiros (ver FIG 4.3);

```
1  class Program
2  {
3      [DllImport(@"myPath\CppClassDll.dll")]
4      public static extern int somar(int a, int b);
5
6      static void Main(string[] args)
7      {
8          var result = somar(10, 10);
9          Console.WriteLine(result);
10     }
11 }
```

FIG. 4.3 – Aplicação C# para console.

4. Criação de uma aplicação *web*, em C#, que faz a chamada do método *somar* da DLL e apresenta o correto valor de retorno na tela (ver figuras 4.4, 4.5 e 4.6).

```
1  // Aplicação C# para web
2  // Controle
3  public class HomeController : Controller
4  {
5      [DllImport(@"myPath\CppClassDll.dll")]
6      public static extern int somar(int a, int b);
7
8      public ActionResult Test()
9      {
10         ViewBag.Result = somar(200, 300);
11         return View();
12     }
13 }
```

FIG. 4.4 – Aplicação C# para *web* (controle).

```
1  @* Aplicação C# para web *@
2  @* Visão (Test.cshtml) *@
3
4  <h2>Test</h2>
5  <h2>ViewBag.Result</h2>
```

FIG. 4.5 – Aplicação C# para *web* (visão).

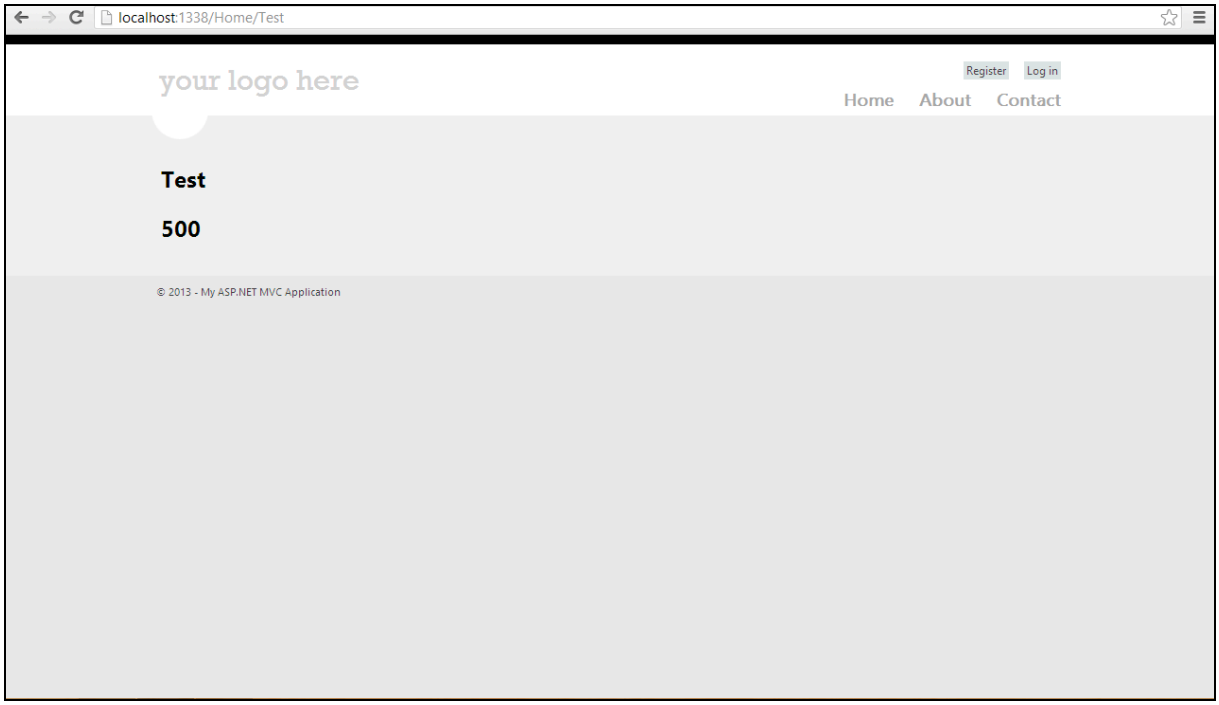

FIG. 4.6 – Resultado da chamada o método *somar* pela aplicação *web*.

5 FAMA-*ONLINE*

Após o sucesso dos experimentos iniciais, iniciou-se a implementação do *website* final, doravante denominado FAMA-*Online*. Nele implementou-se a chamada e execução do algoritmo *Mais Provável* (NASCIMENTO e OLIVEIRA, 2012).

Na tela inicial do FAMA-*Online* (FIG. 5.1) o usuário pode selecionar o método *Mais Provável* no canto superior direito da tela. Com isso o usuário irá para a tela de parâmetros (FIG. 5.2), na qual deverão ser inseridos todos os parâmetros necessários para a correta execução do algoritmo, além do e-mail para o envio dos resultados. Nessa mesma tela, há dois parâmetros que são arquivos que devem ser selecionados pelo usuário. O usuário tem a opção de fazer o *upload* de um novo arquivo ainda não disponível no servidor, clicando no link “Upload novo arquivo”, o que o levará para a *pop-up* de *upload* (FIG. 5.3).

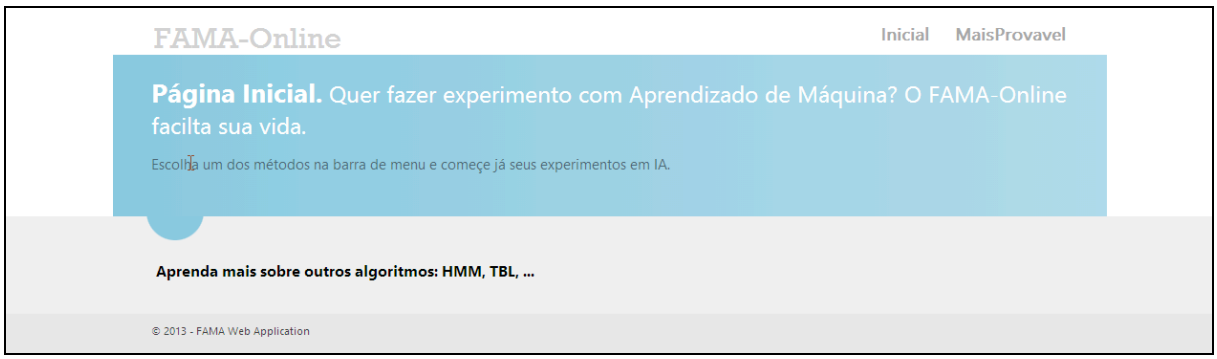

FIG. 5. 1 – Página inicial do website.

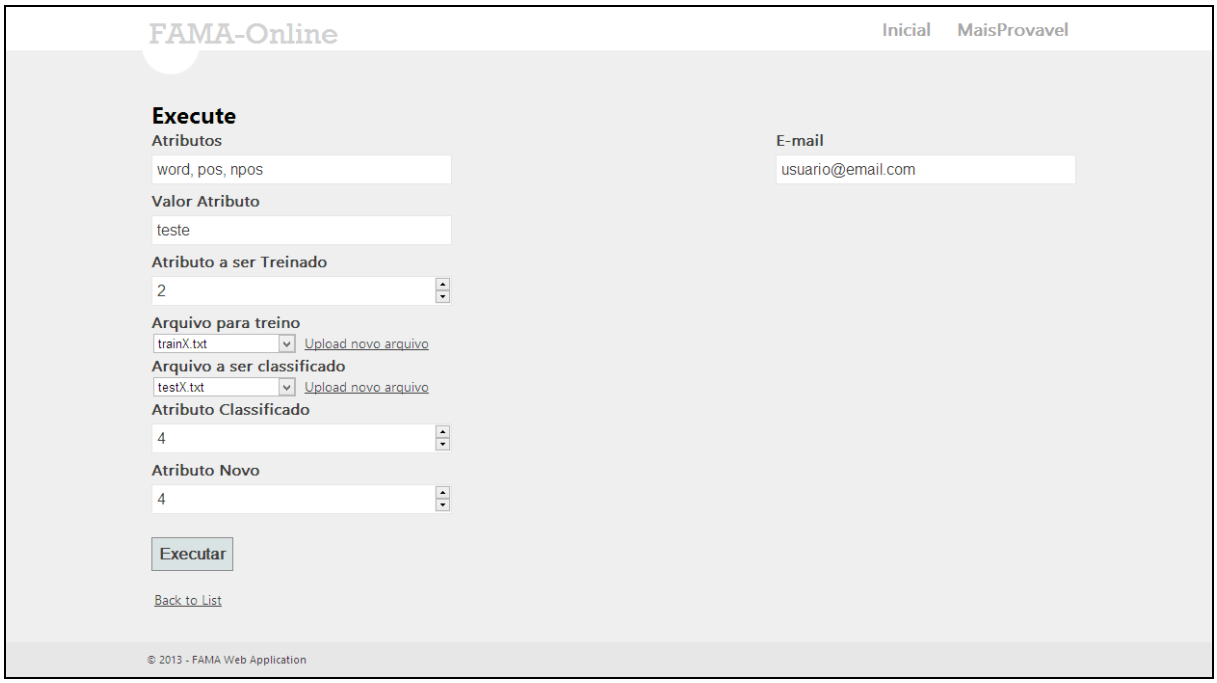

FIG. 5. 2 – Página para a inserção de parâmetros.

FIG. 5. 3 – *Pop-up* para *upload* de novo arquivo.

Selecionado todos os parâmetros exigidos, o usuário poderá clicar no botão *Executar* na tela de parâmetros (FIG. 5.2) para executar o algoritmo *Mais Próvavel*. Será então redirecionado para uma página de confirmação de envio da requisição (FIG. 5.4).

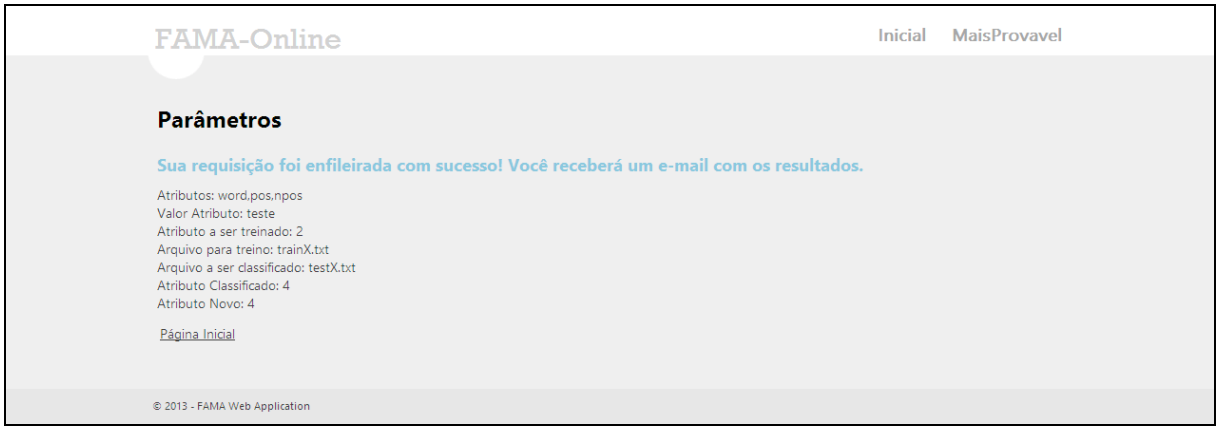

FIG. 5. 4 – Tela de parâmetros.

Após o processamento, cria-se um arquivo na pasta *Outputs* do servidor. Um sistema de monitoramento dessa pasta foi implementado para enviar a resposta para o e-mail fornecido pelo usuário, quando da alteração dos arquivos contidos nela. O sistema de monitoramento é ativado quando o usuário manda executar algum método do *website*. A figura 5.5 mostra o manipulador que trata o evento gerado quando da alteração num arquivo da pasta monitorada. Ele simplesmente monta o e-mail (linhas 4 a 10), envia o e-mail ao destinatário (linhas 12 e 13) e desativa o monitoramento que havia sido criado (linha 15).

```
1 private static void OnChanged(object source, FileSystemEventArgs e)
2         {
3
4            var message = new MailMessage(
5                to:_destinatario,
6                from:"fama.framework@gmail.com",
7                subject:"Teste do FAMA",
8                body:"Este é o corpo do email!");
9
10           message.Attachments.Add(new Attachment(@"~\resultado.txt"));
11
12           var client = new SmtpClient();
13           client.Send(message);
14
15           ((FileSystemWatcher)source).Dispose();
16
17
18        }
```

FIG. 5. 5 – Manipulador de eventos. Dispara o envio de e-mails.

6 CRONOGRAMA

	FEV	MAR	ABR	MAI	JUN	JUL	AGO	SET	OUT
Documentação									
Análise e definição da linguagem utilizada									
Implementação dos protótipos de interoperabilidade									
Estudo e implementação do protótipo de envio de e-mail									
Estudo e definição da arquitetura da solução									
Implementação do website									
Implementação das chamadas aos algoritmos									
Implementação da validação cruzada									

FIG. 5.1 – Cronograma.

Documentação – registros ocasionais e confecção do relatório.

Análise e definição da linguagem utilizada – Determinar que linguagem de programação melhor se adequa a resolução do problema tratado.

Implementação dos protótipos de interoperabilidade – Escolhida uma linguagem, realizar experimentos que comprovem a eficácia dessa para o problema tratado.

Estudo e implementação do protótipo de envio de e-mail – Pesquisar tecnologias de envio de e-mail disponíveis e realizar experimentos básicos de funcionamento da tecnologia escolhida.

Estudo e definição da arquitetura da solução – Analisar as classes do FAMA de forma a determinar que métodos serão expostos para acesso via *website*.

Implementação do website – Compreende praticamente todo o período do trabalho (exceto a etapa inicial de experimentos e definição da linguagem), onde serão realizadas as implementações finais do sistema (como formas de apresentação para o usuário, sistemas de controle do *website* e modelos de armazenamento de dados, se necessário).

Implementação das chamadas aos algoritmos – Realizar implementação para prover a correta chamada aos algoritmo implementados no FAMA (via *website*) e

obter os resultados corretos para o usuário.

Implementação da validação cruzada – Realizar implementação para prover experimentos de validação cruzada pelo usuário.

7 CONCLUSÃO

Em Inteligência Artificial, especialmente na área de Aprendizado de Máquina, o uso de ferramentas que facilitam a execução de experimentos e aplicação das diversas técnicas estudadas por especialista é cada vez mais importante.

Neste trabalho, pretende-se desenvolver uma ferramenta, na forma de *website*, que proverá acesso remoto a um sistema de aprendizado de máquina (FAMA). Na fase atual o *website* já fornece uma correta comunicação com o FAMA.

Criar uma ferramenta como a proposta neste trabalho facilitará a execução de experimentos de validação cruzada de aprendizado de máquina de duas maneiras marcantes: possibilitando a execução de experimentos via Internet; e permitindo que até mesmo leigos no assunto possam usar o sistema.

Desenvolvidas algumas das principais funcionalidades propostas para o FAMA-*Online*, como acesso remoto ao FAMA e envio dos resultados ao usuário via e-mail, pretende-se na etapa posterior implementar uma funcionalidade que permita ao usuário realizar experimentos de validação cruzada através do *website*.

8 REFERÊNCIAS BIBLIOGRÁFICAS

BOOST. **Boost.Python**. Disponível em: <http://www.boost.org/doc/libs/1_54_0/libs/python/doc/index.html>. Acesso em : 10 de agosto de 2013.

BRILL, E. **A simple rule-based part of speech tagger**. Em Proceedings of the Third Conference on Applied Natural Language Process, Trento, Italy, 1992. Association for Computational Linguistics.

CYTHON. **Cython – C-Extensions for Python**. Disponível em: <http://docs.cython.org>. Acesso em: 28 de julho de 2013.

KAPLAN, J. **Práticas recomendadas de interoperabilidade entre código gerenciado e nativo**. MSDN Magazine. Disponível em: <http://msdn.microsoft.com/pt-br/magazine/dd315414.aspx>. Acesso em : 17 de março de 2013.

KOLMAN, M. **DLL (C++) for C Sharp (C#)**. Disponível em: <www.youtube.com/watch?v=hwmRtnJag4A>. Acesso em : 17 de março de 2013.

NASCIMENTO, J. C. e OLIVEIRA, F. A. **Algoritmos de aprendizado de máquina para tarefas de classificação morfossintática**. Rio de Janeiro, 2012. Monografia – Apresentada ao curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia.

PONT, M. **Calling C Library Routines from Java**. Disponível em: <http://www.nag.co.uk/IndustryArticles/CallingCLibraryRoutinesfromJava.pdf>. Acesso em: 19 de março de 2013.

PYREX. **Pyrex - a Language for Writing Python Extension Modules**. Disponível em: <http://www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex/>. Acesso em: 28 de julho de 2013.

PYTHON. **ctypes – A foreign function library for Python**. Disponível em: <http://docs.python.org/2/library/ctypes.html>. Acesso em: 19 de março de 2013.

RABINER, L. R. **A tutorial on hidden Markov models and selected applications in speech recognition**. Proceedings of the IEEE, 1989.