Implementação de Modelos de Covariância no arcabouço ToPS

Renato Cordeiro Ferreira

Relatório Parcial

apresentado à disciplina

MAC0499

(Trabalho de Formatura Supervisionado)

Supervisor: Prof. Dr. Alan Mitchell Durham

São Paulo, Agosto de 2015

1 Plano de trabalho inicial

No início do projeto, dividimos a realização deste trabalho em 6 etapas:

1. Criar novo repositório para o ToPS, portando o código para uma nova biblioteca.
2. Realizar refatorações na hierarquia de classes. principal do arcabouço, facilitando a reutilização de código e a inserção de novos modelos.
3. Implementar algoritmos de modelo de covariância (*inside*, *outside*, CYK, *inside-outside* e análise comparativa automatizada)
4. Realizar testes com outras ferramentas que utilizam modelos de covariância.
5. Escrever monografia
6. Montar apresentação e pôster

Dispostas no seguinte cronograma:

	mar	abr	mai	jun	jul	ago	set	out	nov
1	X	X							
2	X	X	X	X					
3					X	X			
4						X	X	X	X
5						X	X	X	X
6								X	X

Tabela 1: *Cronograma do plano de trabalho inicial*

2 Atividades realizadas no primeiro semestre

Durante os meses de Março a Julho, tínhamos por objetivo concluir as etapas 1 e 2 descritas na Tabela 2. Com estas atividades, obteríamos uma nova versão do arcabouço que facilitaria tanto o seu uso, por parte das aplicações construídas sobre ele, quanto sua extensão, no que diz respeito aos desenvolvedores que realizarem novas atividades. Com isso, conseguiríamos conhecer a estrutura do código, suas classes e algoritmos, e obteríamos um ambiente mais amigável para começar a etapa número 3.

Os trabalhos realizados foram feitos em parceria com o aluno de doutorado Ígor Bonadio, que auxiliou com conhecimentos sobre utilização do ToPS e das bibliotecas externas usadas no projeto.

Nas subseções seguintes, listamos detalhadamente as atividades realizadas durante esse semestre para o cumprimento das etapas 1 e 2.

2.1 Etapa 1: Porte do código do ToPS para um novo repositório

- Criar uma organização para o projeto de desenvolvimento do ToPS do Github (topsframework) e adicionar os principais desenvolvedores.
- Criar um novo repositório (tops-refactoring) para a versão refatorada do ToPS.
- Alterar a documentação do código do repositório, atualizando o leiame (`README.md`) e a lista de contribuidores (`AUTHORS`).
- Propor novo estilo padronizado de mensagens de *commit*, utilizando o tempo imperativo do inglês, conforme recomendado na documentação do git.
- Portar código dos modelos antigos para a versão refatorada, eliminando dependências externas desnecessárias (biblioteca boost) no novo padrão do C++.
- Separar biblioteca de modelos probabilísticos em um *namespace* próprio (`model`), diferente dos da linguagem de descrição dos modelos (`lang`).
- Trocar sistema de *build* do CMake para o Make Framework.
- Padronizar o estilo de código usando as diretrizes do Google Code Standard.
- Dar suporte para o uso de bibliotecas de testes de unidade (GMock e GTest) e de benchmarks (GBenchmark) com a biblioteca de modelos probabilísticos do ToPS.
- Adicionar testes de unidade para os métodos públicos das classes da hierarquia principal.

- Utilizar um *webhook* do GitHub como sistema de integração contínua (Travis CI) com suporte para Linux e OSX, de modo a permitir a refatoração mantendo as funcionalidades originais. Rodar testes utilizando os compiladores gcc e clang para maior compatibilidade.

- Dar suporte para o uso de ferramentas de cobertura de código (`gcov` e `lcov`) e *profiling* (`gprof`).

- Conectar o sistema de integração continua com um *webhook* do Github para cobertura de testes (Coveralls).

- Adicionar documentação básica de exemplo com o Doxygen, permitindo geração automática de documentação de funções no estilo Javadoc em página web e em documento LaTeX.

- Portar o site original do ToPS do SourceForge para o Github Pages.

- Incluir documentação auto-gerada pelo Doxygen no novo site para consulta *online*.

- Propor novo sistema de contribuição para o repositório, baseado na criação de um *branch* (contribuidores principais) ou *fork* (contribuidores externos) para cada nova funcionalidade ou refatoração, sendo posteriormente integradas à versão estável (*branch master*) por meio de *pull requests* aceitos após revisão. Descrever o fluxo de contribuição no arquivo `CONTRIBUTING.md`.

- Propor nova técnica de entrega do código baseada em *tags* seguindo as regras do versionamento semântico (semver). Colocar a *tag* atual como versão `2.0.0-alpha` (instável, suscetível a modificações a qualquer momento).

- Utilizar o Github Issues para listagem de tarefas e rastreamento de *bugs*. Conectar um *webhook* de visualização em forma de Kanban (Waffle.io) para separar histórias de usuário conforme proposto nas práticas de Programação Extrema.

- Remover inconsistências e avisos apontados no nível máximo de *warning* dos compiladores gcc e clang (flags `-Wall -Wextra -pedantic`) e tornar *warnings* erros de compilação (flag `-Werror`).

- Remover erros apontados pelo lint (analisador estático de código) cppchek e pelo *webhook* Coverity.

- Garantir a ausência de vazamento de memória nos testes de unidade, por meio da utilização da ferramenta Valgrind.

2.2 Etapa 2: Refatoração da hierarquia de classes do ToPS

- Unificar a subclasse `ProbabilisticModel` e `FactorableModel` da hierarquia de classes de modelos probabilísticos.
- Renomear o modelo `InhomogeneusFactorableModel` para `InhomogeneusMarkovChain`, e dividí-la em duas implementações: com fase e sem fase.
- Utilizar *smart pointers* para automatizar a gestão de memória de objetos alocados dinamicamente.
- Criar repositório auxilar (tops-architecture) para prover um exemplo mínimo do funcionamento da nova arquitetura. Realizar experimentos em pequena escala nele antes de implementá-los no ToPS.
- Encapsular o armazenamento de estado (*cache*) em classes auxiliares (denominadas *front-ends*), permitindo que os modelos probabilísticos (*back-ends*) rodem algoritmos de forma paralela sem conflitos (condição de corrida) na escrita dos *caches*.
- Maximizar o reuso de código dentro dos *front-ends*, utilizando *templates*, macros e o padrão de projeto *double dispatch* para delegar a implementação dos algoritmos dos métodos dos *front-ends* para a versão correspondente desses métodos nos *back-ends*.
- Implementar os *front-ends* de geração (`Generators`) e avaliação (`Evaluators`).
- Utilizar o *Curiously Recurring Template Pattern* (CRTP) e herança em tempo de compilação (herança estática) para reutilizar os *template methods* que alocam *front-ends* nas classes da hierarquia principal.
- Criar nomes de métodos mais genéricos, escondendo as implementações algorítmicas (com nomes diferentes) que realizam a mesma tarefa sob uma mesma interface genérica.
- Utilizar *alias* para abstrair tipos concretos (`double`, `int`, `vector<int>`) para nomes mais semânticos (`Probability`, `Symbol`, `Sequence`).

3 Atividades atuais e alterações no cronograma

Durante o primeiro semestre, focamos nas alterações no repositório e na refatoração do ToPS, utilizando todo o tempo hábil para melhorar a qualidade do código em termos de projeto, implementação e práticas de programação adotadas. Por esse motivo, não iniciamos a etapa 3, e estamos no processo de concluir a refatoração. No momento, as seguintes atividades das etapas 1 e 2 estão sendo realizadas:

3.1 Etapa 1: Porte do código do ToPS para um novo repositório

- Dar suporte para a compilação do ToPS no sistema operacional Windows, usando os sistemas Cygwin ou MSYS2.
- Utilizar um *webhook* do GitHub como sistema de integração continua (Appveyor) com suporte para Windows, permitindo a compilação do ToPS nos principais sistemas operacionais utilizados atualmente.

3.2 Etapa 2: Refatoração da hierarquia de classes do ToPS

- Concluir a implementação do *front-end* de treinamentos (*Trainers*). Criar um *Builder* para treinamento postergado dos modelos.
- Implementar um *Visitor* que permitirá treinar os modelos recursivamente no *Composite* da hierarquia de modelos probabilísticos.

4 Atividades planejadas

Nos próximos meses, temos por objetivo concluir a refatoração, e começar a implementação do Modelo de Covariância. Paralelamente, focaremos na elaboração dos trabalhos escritos necessários para a disciplina - em especial, dedicando tempo para a elaboração da monografia.

4.1 Etapa 2: Refatoração da hierarquia de classes do ToPS

- Interligar o *front-end* de treinamento com o novo *parser* da linguagem do ToPS feito pelo doutorando ígor Bonadio, com o uso das ferramentas Flexc++ e Bisonc++.
- Auxiliar os alunos de mestrado Bruno Tenório e Ademar Alves de Oliveira na criação de uma hierarquia de exceções do ToPS, a ser utilizada como erros dentro da biblioteca de modelos probabilísticos.
- Auxiliar os mesmos alunos de mestrado na criação dos aplicativos (executáveis) do ToPS, que permitirão criar e realizar cálculos com modelos probabilísticos.

4.2 Etapa 3: Implementação do Modelo de Covariância

- Criar um novo modelo decodificável para o Modelo de Covariância (CM).
- Implementar o algoritmos *inside*, *outside*, CYK, *inside-outside* e análise comparativa automatizada, colocando-os dentro dos métodos dos *front-ends* utilizados nos modelos decodificáveis.
- Adicionar testes de unidade para os métodos do novo modelo.
- Validar o programa alinhador do ToPS, comparando-a com outra implementação (Infernal). Utilizar sequências reais e simuladas, verificando se há similaridade entre os modelos treinados as probabilidades dos alinhamentos.
- Tentar portar o Modelo Oculto de Markov Sensível ao Contexto (csHMM) para o novo repositório do ToPS, e adequá-lo às modificações já realizadas na arquitetura do arcabouço. Reutilizar código entre a implementação do CM e do csHMM.

4.3 Etapa 4: Elaboração dos trabalhos escritos

- Escrever a monografia do trabalho.
- Elaborar a apresentação de slides e o pôster, expondo os principais resultados obtidos durante projeto.

5 Plano de trabalho atualizado

Considerando as atividades já realizadas e planejadas para o próximo semestre, podemos redividir o projeto em 4 grandes etapas:

1. Porte do código do ToPS para um novo repositório.
2. Refatoração da hierarquia de classes do ToPS.
3. Implementação do Modelo de Covariância.
4. Elaboração dos trabalhos escritos.

E atualizar o cronograma do plano de trabalho:

	mar	abr	mai	jun	jul	ago	set	out	nov
1	X	X	X	X	X	X			
2	X	X	X	X	X	X			
3						X	X	X	
4						X	X	X	X

Tabela 2: *Cronograma do plano de trabalho atualizado*