Análise Automática de Exercícios de Programação como Forma de Avaliar a Cobertura de Tópicos da Disciplina

Matheus Gaudencio[1], **Ayla Dantas**[2], **Dalton D. S. Guerrero**[1]

[1]Laboratório de Práticas de Software
Universidade Federal de Campina Grande
Av. Aprígio Veloso, s/n, SPLab, Bodocongó
58429-900 – Campina Grande - PB – Brasil

[2]Departamento de Ciências Exatas
Universidade Federal da Paraíba (UFPB) – Campus IV
Rua da Mangueira, S/N. Companhia de Tecidos Rio Tinto
58297-000 – Rio Tinto - PB – Brasil

matheusgr@copin.ufcg.edu.br, ayla@dce.ufpb.br, dalton@dsc.ufcg.edu.br

Abstract. *Exercises are a common activity during the learning process. But few discussions are made about how effective are those exercises to cover proposed subjects of a course. In this work, we propose a process based on automatic analysis of students codes to help teachers construct better exercises. Our propose works using automatic detection of topics and also similarity comparison among students solutions. We applied such strategy to one semester of CS1 which allowed us to see how related are topics from our course in our exercises. Our work can be extend to other courses to provide teachers with information to prepare them for a better classroom experience.*

Resumo. *A produção de listas de exercícios é uma atividade comum no ensino, mas pouco se discute sobre sua efetividade em cobrir os tópicos trabalhados. Neste trabalho nós propomos uma metodologia que se baseia na análise automática das respostas dos alunos para auxiliar o professor na reformulação das listas de exercícios para futuras turmas. Nós utilizamos nossa proposta ao final de um semestre da disciplina de ensino de programação e isso nos permitiu visualizar como tópicos da disciplina foram explorados através das listas de exercícios ao longo do semestre. Este método pode ser aplicado a outras matérias e permite ao professor um melhor planejamento dos exercícios a serem apresentados a uma nova turma da disciplina.*

1. Introdução

A lista de exercícios é uma atividade que dá ao aluno a oportunidade de praticar e aprofundar os conteúdos explorados numa disciplina. O professor costumeiramente disponibiliza ao aluno um conjunto de exercícios que considera relevantes para praticar o que se está trabalhando em um curso. Na medida que os alunos apresentam suas respostas, o professor pode acompanhar as soluções apresentadas e fazer ajustes nos próximos exercícios de uma lista para melhor cobrir tópicos passados que não foram bem trabalhados ou nos quais os alunos apresentaram mais dificuldades.

DOI: 10.5753/CBIE.SBIE.2013.617

O professor também pode colocar exercícios de forma livre ou complementar à disciplina. No entanto, nem sempre aquilo que o professor planeja explorar se reflete nas soluções dos alunos. Por vezes, a habilidade ou conhecimento específico que o professor pensa estar estimulando em um dado exercício é apenas parte do que o exercício efetivamente requer do estudante, desfocando ou confundindo o estudante e a avaliação que se faz dele a partir de seu desempenho nos exercícios. Assim, é importante que o professor disponha de ferramentas e mecanismos automáticos para lhe auxiliar na análise de suas listas de exercícios de forma a permitir ajustes no próprio processo pedagógico que vem sendo adotado.

Neste trabalho, propomos uma metodologia para auxiliar o professor no processo de reflexão sobre suas listas de exercícios. Primeiramente, identifica-se, através das respostas dos alunos, quais são os tópicos explorados nos exercícios. Estes tópicos servem para identificar conjuntos de exercícios que exploram conceitos semelhantes e são definidos pelos termos mais relevantes extraídos das soluções dos alunos.

Em seguida, cada tópico detectado é confrontado com aqueles planejados para serem exercitados pela lista. Por fim, compara-se a similaridade das soluções das diferentes questões de uma lista de exercícios. Esta comparação permite a detecção de possíveis semelhanças entre questões de tópicos distintos. Essa metodologia é aplicável para qualquer disciplina em que elementos (termos) das respostas dos alunos tenham significado para a disciplina.

Para concretizar nossa proposta, nós aplicamos nossa metodologia ao final de um semestre da disciplina introdutória de programação do Curso de Ciência da Computação da Universidade Federal de Campina Grande. De acordo com Chamillard e Braun (2000) , a prática da lista de exercícios com a produção de código é uma atividade bastante comum no ensino de programação. Ao trabalhar diretamente com o computador, o aluno produz um artefato automaticamente manipulável e passível de análise. Ainda, os termos sintáticos de um código, como exemplo, o FOR, WHILE e IF estão bastante ligados aos conteúdos explorados na disciplina, o que nos permite fazer uso da análise proposta. Por exemplo, ao identificar um tópico onde o IF seja o termo prevalente, o professor sabe que o exercício explorado foca no estudo de comandos condicionais.

Com a aplicação da nossa metodologia nós identificamos como os conceitos da disciplina são explorados pelas questões de uma lista de exercícios em um semestre. Encontramos que as questões, através dos tópicos identificados, exploram boa parte do que foi planejado para a disciplina. Detectamos que um dos conteúdos (uso e criação de funções) não foi explorado por um conjunto único de questões da lista, mas teve seu conteúdo sendo trabalhado em diferentes momentos da disciplina. Ainda, foi possível observar alguns padrões nos exercícios que foram propostos como, por exemplo, o pouco retorno aos conceitos iniciais da disciplina em questões publicadas ao final da lista de exercícios.

Na Seção 2 discutiremos o referêncial teórico existente na área de análise semântica e similaridade, bem como a de avaliação de listas de exercícios. Em seguida, apresentamos a nossa metodologia para a avaliação automática baseando-se neste referencial. Na Seção de Avaliação, apresentamos uma avaliação existente aplicada em um semestre da disciplina de ensino de programação. Por fim, apresentamos nossas

conclusões e possíveis trabalhos futuros.

2. Referêncial Teórico

De acordo com Oliveira (2002) é importante que exercícios pedidos aos alunos sejam de qualidade e sejam capazes de proporcionar, em um primeiro momento, o início do processo de ensino-aprendizagem. O autor afirma ainda a importância de que a avaliação seja capaz de refletir sobre o planejamento pedagógico da disciplina na sua prática. Neste sentido, é importante que o professor possa analisar as avaliações e exercícios que utilizou, especialmente considerando os resultados alcançados ou não em um semestre de acordo com as produções dos alunos.

Por vezes, o próprio professor prepara de antemão exercícios com a descrição dos tópicos a serem abordados nas questões propostas. Esta prática é especialmente comum em sistemas de tutores inteligentes, onde a definição destes tópicos é importante para que questões sejam selecionadas para o aluno, como exemplificado na proposta de Schuck e Giraffa (2001) . No entanto, não há ainda uma discussão sobre a qualidade dos exercícios propostos para cobrir os tópicos planejados na disciplina considerando o que os alunos produziram para responder tais exercícios.

Nossa metodologia proposta faz, primeiramente, uso da detecção automática dos tópicos que estão sendo trabalhados nos exercícios com base nas soluções apresentadas pelos alunos. Esta detecção é bastante discutida na literatura e já foi abordada em outros contextos. Na proposta de Dalmolin et al. (2009) , tópicos são extraídos de documentos de referência de um curso para a organização de mapas conceituais. Esta mesma abordagem é utilizada na área de Engenharia de Software, como no trabalho de Maletic e Marcus (2000) , onde códigos são analisados para identificar tópicos associados a componentes de código. Nós fazemos uso do mesmo arcabouço teórico de análise de código proposto por Maletic e o aplicamos em soluções apresentadas por alunos a determinadas listas de exercícios para identificar tópicos relevantes para exercícios.

Mais especificamente, para a identificação de tópicos, utilizamos da estratégia de análise semântica proposta por Blei et al. (2003) , a Alocação Latente de Dirichlet. Neste modelo, cada documento analisado é visto como uma composição de tópicos, e cada tópico é gerado a partir de uma coleção de termos relevantes. Este modelo apresenta vantagens ao considerar que cada documento é composto de uma mescla de tópicos, e por considerar que cada termo existente pode apenas contribuir positivamente ou de forma nula a cada tópico. Assim, cada tópico identificado pela Alocação Latente de Dirichlet é composto por termos relevantes para a geração daquele tópico.

Além da detecção de tópicos, nossa metodologia faz uso da detecção de similaridade para a identificação da semelhança entre soluções de referência das questões das listas de exercícios. O uso de detecção de similaridade é bastante utilizado na área de detecção de plágio, como discutido por Pears et al. (2007) . Maciel et al. (2013) faz uso da análise da comparação de códigos para extrair informação útil sobre um conjunto de questões.Nós fazemos uso do mesmo princípio proposto, mas agora com objetivo de extrair informação relevante de como as questões e de como os tópicos identificados se relacionam. Especialmente, nós fazemos uso da detecção de similaridade entre termos utilizando a técnica da distância de Jaccard explorada ostensivamente na área de mineração de dados [Tan et al. 2005].

3. Metodologia

Para começar o processo de avaliação sobre a lista de exercícios, é preciso que o professor tenha em mãos a ementa do curso em questão, bem como o planejamento que foi proposto para a disciplina. A idéia de nosso trabalho se inicia com a comparação de tópicos automaticamente identificados a partir da análise automática das soluções dos alunos com os tópicos que o professor planejou para o curso.

Esta análise começa com o pré-processamento das respostas coletadas dos exercícios. Nesta etapa, para cada questão, as soluções propostas são processadas de forma a identificar automaticamente uma única solução de referência. Esta solução será aquela que melhor representa o que foi explorado pela turma. Em seguida, é feita a extração dos tópicos de cada questão a partir das soluções de referência. Esta geração de tópicos é também automática e oferece informação útil a repeito de que aspectos estão sendo explorados na lista. Por fim, gera-se uma comparação entre as questões para guiar como tais temas foram explorados ao longo da lista de exercícios.

Um dos aspectos relevantes deste trabalho é que a metodologia proposta é replicável para outras áreas além de programação, bastando para isto que se adapte a fase de pré-processamento de acordo com cada necessidade. Na nossa avaliação exploramos o uso de "tokens" sintáticos (palavras-chave e símbolos) da linguagem de programação Python como os termos relevantes para a identificação de tópicos.

3.1. Solução de referência

Cada solução enviada pelos alunos é convertida em um multiconjunto (conjunto com repetições) contendo os termos utilizados em cada resposta. Depois, as soluções de cada questão são comparadas par-a-par utilizando a distância de Jaccard definida pela interseção de termos em comum sobre o total de termos. Tal distância permite identificar aquelas soluções que tem mais termos semelhantes levando em consideração os diferentes tamanhos de soluções. As soluções mais parecidas terão distância 0 e as completamente diferentes, distância 1.

Com cada distância par-a-par calculada, escolhemos como solução de referência de uma questão aquela que é central em relação as demais soluções. Para tanto, procuramos a solução que apresenta a menor distância euclidiana das demais. Isto permite que tal solução de referência possa ser uma melhor representação de todas as soluções para uma determinada questão.

3.2. Detecção dos tópicos

A partir das soluções de referência, uma para cada questão, a análise automática de tópicos é trivial utilizando a Análise Latente de Dirichlet. O primeiro parâmetro necessário para este algoritmo é o número de tópicos a serem identificados. Nossa recomendação é de que esta escolha seja o número de tópicos planejados para a própria disciplina mais um valor de tolerância que representa a possível detecção de tópicos não planejados pelo professor mas que possam vir a ser detectados pela análise automática. Esta escolha pode vir a ser refinada de acordo com os resultados obtidos de análises iniciais.

O segundo parâmetro necessário pela Análise Latente de Dirichlet é o fator de convergência. Por ser um método iterativo, é necessário que se determine o momento de

parada de sua execução. Entretanto, não há um valor ideal para tais parâmetros. Em geral, o valor padrão da ferramenta em uso costuma ser suficiente na medida que a convergência para um número baixo de documentos é facilmente obtida.

Como resultado, a Alocação Latente de Dirichlet retorna um conjunto de tópicos, cada um associado a um conjunto de termos e a relevância de cada termo para a geração de cada tópico. A análise semântica também retorna o quanto cada tópico está associado com cada documento. Para a nossa análise, fazemos uso apenas do tópico mais relevante por documento. Isto permite uma discussão sobre qual o foco principal de cada questão ao longo do semestre.

3.3. Similaridade entre-questões

Nesta etapa, nós calculamos a distância de Jaccard entre as soluções de referência das diferentes questões. Questões semelhantes terão soluções de referência semelhantes e com esta informação é possível identificar a similaridade de questões de mesmo tópico e de tópicos distintos. Para melhor visualizar esta informação, fazemos uso de um mapa de calor, que faz um cruzamento das questões entre si e cada célula representa, com sua cor, a intensidade da similaridade entre duas questões. Quanto mais clara for a cor numa célula representa, maior a similaridade entre as questões. Um exemplo de mapa de calor é apresentado na Figura 1 onde 5 questões são cruzadas. A diagonal representa maior similaridade (a questão com ela mesma). Na figura de exemplo é possível observar que a quarta questão apresenta uma maior similaridade apenas com a quinta questão. Já a quinta questão apresenta maior semelhança com as questões 2, 3 e 4.

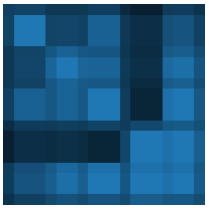

Figura 1. Exemplo de Mapa de Calor

4. Avaliação

A lista de exercícios é uma das atividades disponíveis aos alunos para treinar a prática de programação. Semestralmente, para a disciplina de Introdução à Programação da Universidade Federal de Campina Grande, compomos uma lista de questões abertas que têm como resposta um código em Python. Cada aluno pode submeter suas respostas quantas vezes achar necessário e cada submissão recebe um feedback gerado através de testes automáticos definidos pelos professores. Cada questão tem uma especificação fechada e qualquer ambiguidade é discutida por lista de emails (ou presencialmente) podendo o enunciado da questão ser revisado quando necessário.

Neste contexto, a produção de questões por parte dos professores foi guiada por duas necessidades específicas, exercícios que trabalham o conteúdo atualmente sendo explorado ou exercícios de revisão. Em geral, o professor procurava publicar na lista de exercícios, após cada aula teórica, um conjunto de questões práticas referente ao conteúdo visto. Dependendo do acompanhamento da turma e das avaliações, o professor preparava questões para explorar conceitos já vistos.

A lista de exercícios permitia ao aluno submeter livremente respostas de qualquer questão que já estivesse publicada. Em sala, o aluno era guiado a trabalhar exercícios de conteúdos que tinha dificuldade e, de preferência, seguindo a ordem proposta dos exercícios. Neste trabalho, nós consideramos todas as questões e respostas de todos os alunos do primeiro semestre de 2012 do curso de Introdução à Programação da Universidade Federal de Campina Grande. No total, 193 questões foram disponibilizadas para 102 alunos, o que resultou na coleta de 23.692 respostas, ou uma média de 1,2 submissões por aluno para cada questão. É importante observar que os resultados obtidos desta avaliação são referentes ao semestre avaliado. Outros semestres podem apresentar diferentes resultados, cabendo a quem aplica a metodologia proposta ser capaz de detectar padrões como os citados nesta avaliação ou novos comportamentos que não foram explorados por este trabalho.

4.1. Soluções de referência

Na nossa avaliação, cada termo extraído era um 'token' da árvore abstrata sintática do código. Valores como constantes ou variáveis, eram traduzidos para sua representação simbólica CONST ou ASSNAME, o que anonimizava nomes de variáveis ou valores das mesmas ou de constantes. De todas as soluções, 499 continham erros sintáticos, o que impossibilitava sua extração automática de termos, e foram descartadas da nossa avaliação.

4.2. Detecção dos tópicos

Para a escolha do número inicial de tópicos, nós consideramos o número de unidades da disciplina (9) e acrescentamos 2 tópicos para permitir a detecção de tópicos não programados nas unidades, mas que poderiam existir. Os tópicos planejados para a disciplina foram, em ordem: Shell, Programas, Comandos e Programas, Alternativas, Repetições com for, Repetições com while, Formatação de Strings, Funções, e Tuplas e Dicionários.

Assim, após a execução da Análise Latente de Dirichlet, onze tópicos foram identificados. Na Tabela 1 é possível visualizar cada tópico, bem como a quantidade de questões identificadas primariamente com estes tópicos. Junto a cada tópico, há também quais os termos e qual a relevância (de 0 a 1) de cada termo para a identificação daquele tópico.

Quatro dos tópicos identificados aparecem com pouca frequência na lista de exercícios (menos que 5%). São esses os tópicos 2, 7, 9 e 10. Tais tópicos apresentam termos de alta relevância que são pouco explorados como assunto primário (==, in, lower e split). Assim, as 15 questões com tais tópicos exploraram principalmente termos que não estão associados diretamente a nenhum dos conceitos planejados para a disciplina. Isto indica a presença de 15 questões que exploram majoritariamente aspectos secundários da linguagem.

A Figura 2 apresenta a distribuição dos tópicos nas questões publicadas. Cada questão, apresentada no eixo X está associado primariamente com um dos tópicos do eixo Y. As questões estão colocadas em ordem de publicação e nela, é possível visualizar que, ao longo da disciplina, houve pouco retorno a questões de tópicos iniciais. Especificamente, é possível ver o começo da disciplina com predominância das questões

Tabela 1. Tópico, quantidades de questões associadas e termos relevantes para a definição do tópico

Tópico	Questões	Termos relevantes
0	32	$(0.44, print)$, $(0.23, raw_input)$, $(0.19, float)$
1	35	$(0.26, print)$, $(0.23, int)$, $(0.19raw_input)$
2	8	$(0.40, ==)$, $(0.29, mod)$, $(0.16, if)$
3	12	$(0.18, if)$, $(0.15, len)$, $(0.11, >=)$, $(0.10, <)$, $(0.10, return)$
4	27	$(0.24, subscript)$, $(0.10, split)$
5	40	$(0.33, subscript)$, $(0.12, len)$, $(0.10, ==)$
6	14	$(0.17, if)$, $(0.14, break)$, $(0.14, print)$, $(0.13, ==)$, $(0.11, while)$
7	3	$(0.26, in)$, $(0.18, return)$, $(0.15, subscript)$, $(0.10, if)$
8	32	$(0.17, subscript)$, $(0.10, for)$, $(0.10, del)$, $(0.10, list)$, $(0.10, append)$
9	3	$(0.22, lower)$, $(0.10, for)$, $(0.10, if)$
10	1	$(0.22, split)$, $(0.15, map)$, $(0.14, raw_input)$, $(0.12, assignment\ tuple)$

dos tópicos 0 e 1 (print) seguido por questões dos tópicos 4 e 6. Depois as questões se alternam entre os tópicos 5 e 8.

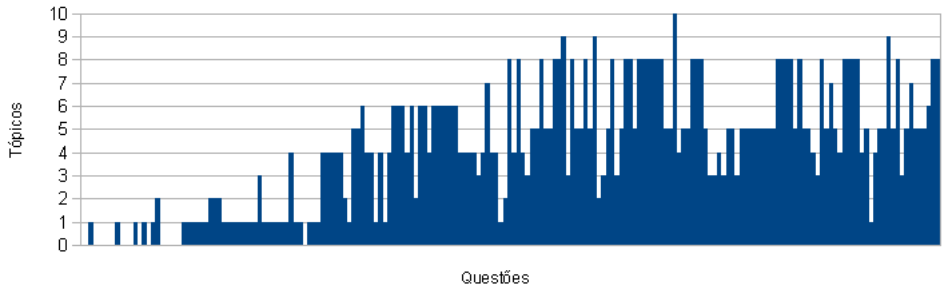

Figura 2. Questões e os tópicos detectados

Os dois primeiros tópicos explorados apresentam print como principal construção e um grande peso para o uso do raw_input. Em seguida, o tópico 3 explora o uso de comando condicional if. Depois, dois tópicos (4 e 5) exigem muito do aluno no uso de acesso a itens de uma estrutura de dados (subscript). Pela sequencia planejada pelos professores, o curso focava inicialmente no uso de programas simples de entrada-e-saída, seguindo por comandos condicionais para entrar em estruturas de repetição (for).

Observe que, nos tópicos 4 e 5, não há a presença do termo for, apesar de ser, neste momento que este comando de repetição é explorado em sala. Ao olharmos as soluções de referência, foi possível observar que o uso do comando de repetição for vem sempre associado com o termo para uso de acesso a estrutura de dados (subscript). Assim, o subscript passou a ser o termo mais relevante durante o período que exploramos tal

conteúdo. É importante observar que, pelos exercícios produzidos, é interessante dar ênfase ao uso de estruturas de dados ao explorar comandos de repetição.

O tópico 6 apresenta novamente o if como termo principal, mas difere do tema anteriormente explorado pela presença do break, do print, do == e do while. Isto acontece porque, pelo conteúdo programado, o aluno passa agora a explorar o comando de repetição while. Foi interessante observar a presença do if e break predominando nas questões planejadas de while. Ao examinar as soluções existentes, observa-se que a essência dos programas que fazem uso deste termo (while) está justamente na definição das condições de parada adequadas.

Em seguida, ao contrário do planejado pelo curso, não foi possível detectar um tópico único para o tratamento de funções. Esperava-se que, questões de criações de funções poderiam aparecer como tópico principal, com o uso do termo def (definição de funções). Isto indica que o uso de funções foi um conteúdo ausente ou diluído nos demais tópicos presentes. Na avaliação de similaridade encontramos que o uso de funções aparece distribuído entre os demais tópicos.

O próximo tópico explorado (tópico 8) aparece com o termo mais relevante sendo acesso a um item. Nesta etapa do curso, era planejado o uso de tuplas e dicionários. Entretanto, não foi possível observar a presença do termo de criação de dicionários. Desta vez, o acesso a item teve um menor peso no tópico, passando a dar importância a termos de manipulação de estruturas de dados (for, del, list e append), com destaque a relevância do termo de repetição for. Ao analisar tais dados, é possível observar uma concentração de outra atividade, não planejada, que foi o uso de questões de matrizes.

4.3. Similaridade entre-questões

Espera-se da similaridade entre-questões que as questões de mesmo tópico sejam semelhantes entre si. Para visualizar isto, nós desenhamos o mapa de calor de similaridade entre as questões da lista. A linha diagonal representa similaridade máxima (a questão com ela mesma) e a matriz é simétrica por ser um mapa de questões x questões. Para cada célula, quanto mais clara a cor, maior similaridade. Acima do mapa de calor, está o gráfico da distribuição de tópicos para cada questão como mostra a Figura 3. Ao cruzar a informação dos tópicos detectados e o gráfico de similaridade, é possível observar uma maior similaridade existente das questões de um mesmo tópico.

Entretanto, o resultado mais surpreendente acontece quando consideramos dois grandes blocos que são extremamente semelhantes entre si. Estes blocos, 1 e 2 estão destacados na Figura 4. Ao investigar as soluções de referência das questões destes blocos, nós encontramos que, exatamente na metade do curso, as questões deixavam de explorar aspectos de entrada-e-saída para explorar predominantemente funções. Enquanto não foi possível fazer a identificação automática do tópico semântico próprio pra funções, a avaliação de similaridade ajuda para identificar a presença deste tópico de forma distribuída entre os exercícios.

5. Conclusões

Neste trabalho nós propusemos uma metodologia para auxiliar o professor a reavaliar a lista de exercícios utilizando as soluções propostas pelos alunos ao longo do semestre. Nós aplicamos nossa proposta a um semestre de um curso de Introdução à Programação,

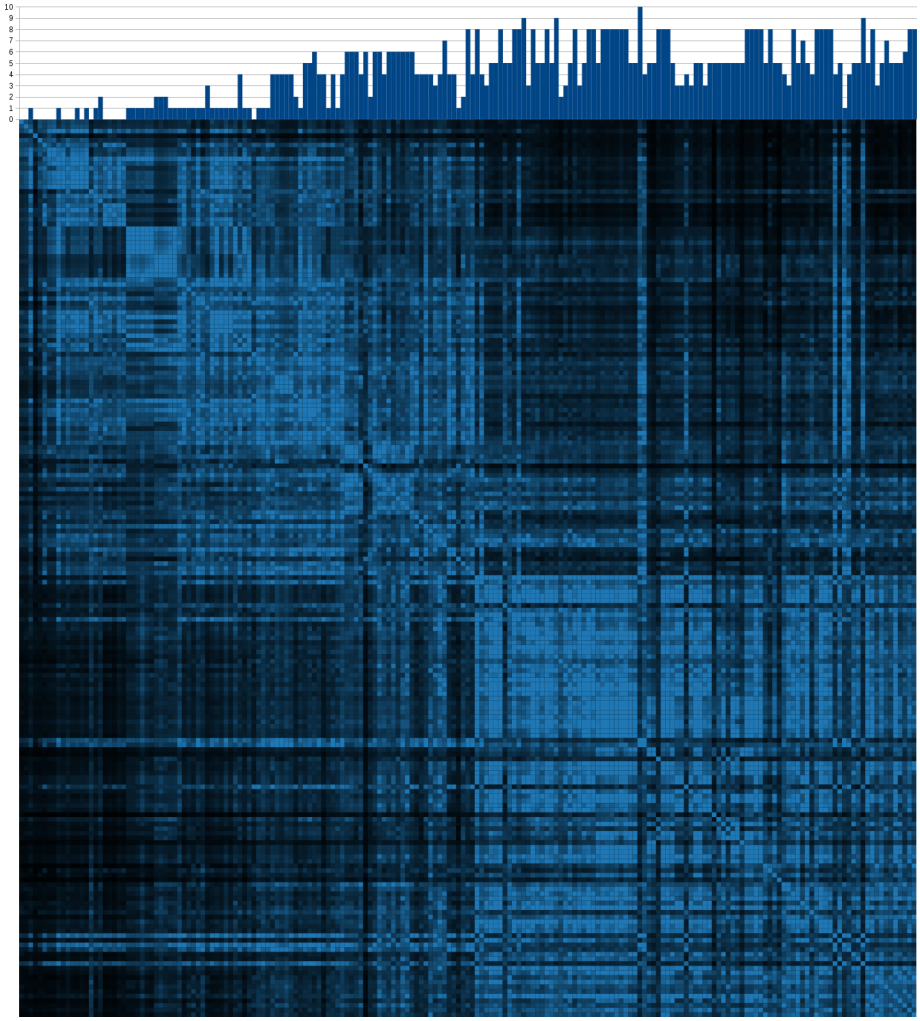

Figura 3. Similaridade entre questões e os tópicos associados

o que nos permitiu ver como os tópicos das disciplinas eram explorados ao longo desse semestre e detectar determinados padrões, mesmo que não planejados, que ocorriam na construção da lista de exercícios, como por exemplo, no uso de funções em questões dispersadas em diferentes tópicos da disciplina.

Como trabalhos futuros, propomos a avaliação do uso de tópicos secundários detectados em cada documento pela análise de tópicos para enriquecer a reflexão sobre lista de exercícios. Além disso, nós pretendemos aplicar a mesma experiência em outros semestres e em outras áreas além de programação. Como exemplo, esperamos aplicar nossa metodologia para um curso de línguas estrangeiras, obtendo informação relevante para o professor repensar os exercícios propostos.

Referências

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Chamillard, A. T. and Braun, K. A. (2000). Evaluating programming ability in an introductory computer science course. In *Proceedings of the thirty-first SIGCSE*

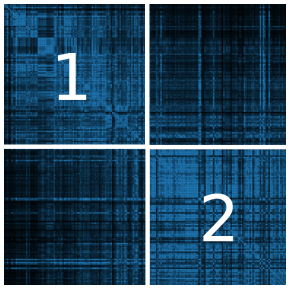

Figura 4. Dois grupos de questões detectados

technical symposium on Computer science education, SIGCSE '00, pages 212–216, New York, NY, USA. ACM.

Dalmolin, L. C. D., Nassar, S. M., Bastos, R. C., and Mateus, G. P. (2009). A concept map extractor tool for teaching and learning. *Advanced Learning Technologies, IEEE International Conference on*, 0:18–20.

Maciel, D., França, A., and Soares, J. (2013). Sistema de apoio a atividades de laboratório de programação via moodle com suporte ao balanceamento de carga e análise de similaridade de código. *Revista Brasileira de Informática na Educação*, 21(01).

Maletic, J. I. and Marcus, A. (2000). Using latent semantic analysis to identify similarities in source code to support program understanding. In *Tools with Artificial Intelligence, 2000. ICTAI 2000. Proceedings. 12th IEEE International Conference on*, pages 46–53.

Oliveira, G. P. d. Avaliação formativa nos cursos superiores: verificações qualitativas no processo de ensino-aprendizagem e a autonomia dos educandos. Disponível em: `http://www.rieoei.org/deloslectores/261Pastre.PDF`. Acesso em: 30 set. 2013.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., and Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *SIGCSE Bull.*, 39(4):204–223.

Schuck, P. W. and Giraffa, L. M. M. (2001). Construindo um sistema tutor inteligente para suporte ao ensino de matemática financeira: da modelagem à validação. In *Anais do XII Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 418–426, Vitória, ES.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.