Aumentando a Interatividade no Ensino a Distância via Geração Automática de Questões: Desafios, Soluções via Aprendizado por Máquina e Um Estudo de Caso no CEDERJ

Marcus Paulo Amorim[1], Jefferson Elbert Simões[2], Felipe Assis[1], João Ismael Pinheiro[1], Daniel S. Menasché[1], Claudia Motta[1], Ageu Pacheco[1]

[1]Universidade Federal do Rio de Janeiro (UFRJ)

[2]Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

Abstract. *Active learning has revolutionized education. By fostering greater interaction between students and teachers, active learning increases learning effectiveness and may help reducing the student dropout rate. However, active learning is challenging in an online environment, where students interact through online platforms such as those offered by CEDERJ. In this paper, we propose the use of interactive questionnaires to increase the interaction between students and teachers in CEDERJ distance learning course. Then, one of the challenges is to determine the difficulty of questions which, for the most part, are answered only once (i.e., as the number of questions is very large, we have at most one sample answer for each question). In order to overcome this challenge, we propose the use of machine learning to extract attributes from the questions and classify them according to these attributes. With the help of decision trees and a naive Bayes classifier, preliminary results indicate that the difficulty level of questions can be automatically inferred.*

Resumo. *O aprendizado ativo vem revolucionando a educação. Promovendo a maior interação entre alunos e professores, o aprendizado ativo diminui a taxa de evasão de alunos e aumenta a eficácia do aprendizado. Entretanto, o aprendizado ativo é desafiador num ambiente online, em que os alunos interagem por uma plataforma de ensino semi-presencial no estilo do CEDERJ. Neste artigo, propomos o uso de questionários interativos para aproximar alunos e professores do curso de ensino a distância do CEDERJ. As questões, no entanto, precisam ser apresentadas aos alunos no nível de dificuldade adequado. Um dos desafios consiste, então, em determinar a dificuldade de questões que, em sua maioria, são respondidas apenas uma única vez. Para contornar tal desafio, propomos o uso de aprendizado por máquina, a fim de extrair atributos das questões e classificá-las segundo tais atributos. Usando uma árvore de decisão e um classificar naive Bayes, resultados preliminares indicam a possibilidade de classificar as questões de acordo com seu nível de dificuldade.*

1. Introdução

O ensino a distância está transformando a educação moderna ao permitir que alunos evoluam no seu próprio ritmo, e interajam entre si e com os professores por plataformas online. O ensino a distância é um dos pilares para aumentar o acesso da população ao

conhecimento. Entretanto, garantir a *escalabilidade* do ensino a distância, e de modalidades similares como o ensino semipresencial, de forma que todos os participantes tenham acesso a um ensino de qualidade e continuem motivados é ainda um grande desafio, principalmente frente a novas evidências de que o aprendizado ativo pode beneficiar significativamente o processo de aprendizado [de Castro 2018].

A grande taxa de evasão dos cursos de ensino a distância indica que existe uma série de desafios a serem enfrentados no âmbito de tais cursos [Silva Filho et al. 2007, Abbad et al. 2005, Schwartzman 2006, Prietch and Pazeto 2010]. Dentre eles, destacamos a importância de ajudar os alunos a identificarem os pontos das disciplinas nos quais precisam evoluir. Após identificar tais pontos, os alunos podem sentir-se mais seguros para procurar ajuda com os professores ou com os colegas. Neste contexto, uma das ferramentas mais recentes a ser incorporada aos sistemas de ensino a distância consiste na geração automatizada de questões. Tais ferramentas, que têm o objetivo de produzir questões sob demanda sobre conteúdo pré-determinado, têm sido aplicadas principalmente na avaliação de aprendizado, constituindo ferramentas para *e-assessment*.

Neste trabalho, propomos uma aplicação de geração automatizada de questões para auxiliar os alunos a identificar suas dificuldades. Nós vislumbramos que tal mecanismo pode promover maior interação entre alunos e professores, uma vez que o primeiro passo para interação é a detecção de pontos que merecem atenção. Em particular, visamos responder a seguinte pergunta: *como estimar, de forma automatizada e escalável, a dificuldade de novas perguntas geradas, que possivelmente nunca tenham sido respondidas?* Em um ambiente tradicional de livro texto, é fácil identificar quais são as questões mais fáceis e díficeis – após um número limitado de turmas ter respondido as questões, os professores percebem quais são aquelas em que os alunos costumam ter mais dificuldade. No caso de questões geradas automaticamente, no entanto, isso torna-se mais difícil, pois as questões podem ser novas.

Existe uma ampla literatura sobre aprendizado ativo, indicando que a interação dos alunos, entre si e com os professores, tende a favorecer o aprendizado [Johnson and Johnson 2008]. O uso de questões para promover o debate entre alunos e professores também não é novo [Singhal et al. 2016, Pérez et al. 2012], e o uso de aprendizado por máquina para auxiliar no ensino também vem sendo amplamente discutido [Bruno et al. 2017]. Entretanto, não é de nosso conhecimento nenhum trabalho anterior que tenha feito uso da geração automática de questões para promover maior interação entre alunos e professores, e que tenha se aproveitado do aprendizado por máquina para automaticamente identificar o nível de dificuldade de questões que potencialmente ainda não tenham sido apresentadas a nenhum aluno (tal lacuna é apontada como trabalho futuro, p.ex., em [Singhal et al. 2016, Pérez et al. 2012]).

1.1. Metodologia

Implementamos um sistema de geração automática de questões para os alunos, e como estudo de caso consideramos o problema de simplificação de expressões lógicas. O estudo de caso foi então apresentado aos alunos do Centro de Educação a Distância do Estado do Rio de Janeiro (CEDERJ). Em seguida, estendemos também o estudo para alunos de outros cursos (não identificados aqui para preservar o anonimato dos autores). Colhemos as respostas e depoimentos dos alunos, que foram usados para alimentar algoritmos de

aprendizado por máquina para automaticamente identificar os níveis de dificuldade das questões. Reportamos aqui os resultados obtidos.

Um dos principais *insights* de nosso trabalho consiste no uso de atributos da questão para identificar sua dificuldade. Ao invés de identificarmos a dificuldade de cada questão individualmente, "aprendemos" de forma automatizada os atributos que tornam uma questão mais fácil ou difícil, e usamos tais atributos para classificar e gerar novas questões. A escolha adequada dos atributos é uma etapa importante desde a concepção da questão até a aferição de sua dificuldade.

1.2. Contribuições

Construímos um sistema para geração de questões com o qual os alunos interagem através de uma interface Web. Este sistema pode ser utilizado para promover maior interação entre tutores e alunos (aprendizado semipresencial) e para os alunos treinarem de forma individualizada (aprendizado à distância). A interação é favorecida tendo em vista que os alunos podem identificar novas dúvidas e questões a serem debatidas em grupo.

Utilizamos como estudo de caso os alunos do curso de Tecnologia em Sistemas de Informação do CEDERJ. Mais precisamente, utilizamos o sistema[1] para apresentar aos alunos questões sobre o tópico de simplificação de expressões lógicas com Mapas de Karnaugh, que faz parte da ementa da disciplina de Introdução à Informática. Foram coletados dados referentes às questões geradas e às interações dos alunos com estas questões.

Desenvolvemos um mecanismo de aprendizado por máquina para identificar a dificuldade das questões. Tal mecanismo se baseia no uso de árvores de decisão, que tradicionalmente têm a finalidade de classificação, mas foram utilizadas neste trabalho para fins *generativos*, ou seja, para *gerar* novas questões.

O restante deste artigo está organizado da seguinte forma. Na Seção 2, discutimos os principais conceitos sobre sistemas de geração de questões, e na Seção 3, descrevemos o sistema para geração de questões desenvolvido neste trabalho. Na Seção 4 descrevemos o experimento conduzido e na Seção 5 apresentamos resultados sobre a classificação automática da dificuldade das questões. A Seção 6 cobre trabalhos relacionados e discute limitações do estudo, e as conclusões seguem na Seção 7.

2. Geração automatizada de questões

Principalmente nas ciências exatas, a formulação de questões é um dos elementos mais fundamentais do processo didático, tendo em vista sua aplicação em listas de exercícios, na preparação de avaliações de aprendizagem, e até mesmo durante a exposição inicial de um conteúdo. Diante de tal ubiquidade, é natural que surjam padrões ou similaridades nas questões preparadas pelos professores ao longo dos anos. Por exemplo, um professor de Matemática para alunos do Ensino Fundamental frequentemente deverá solicitar que seus alunos resolvam equações de 2º grau, embora a equação específica a ser apresentada possa mudar de turma para turma. Neste trabalho, tomamos como premissa que questões são ocorrências ou instâncias de um "modelo" concebido pelo professor, que prepara uma questão individual preenchendo os elementos faltantes neste modelo — e.g.,

[1] O sistema está online, mas para preservar anonimidade iremos compartilhar o link a posteriori.

selecionando os coeficientes da equação no nosso exemplo anterior. Neste contexto, o objetivo de um sistema de geração de questões, ou *gerador de questões*, é permitir que esta instanciação seja feita de maneira automatizada, apoiando assim o professor em seu processo didático. Um gerador de questões difere de um banco de questões, no qual a seleção de uma questão é feita a partir de uma listagem previamente elaborada. Embora o primeiro possa ser visto formalmente como uma generalização do segundo, em um gerador de questões, as questões propriamente ditas somente são formuladas no momento de sua instanciação, em geral de forma aleatória; além disso, um modelo de questão pode, potencialmente, originar um número ilimitado de questões, enquanto um banco de questões está forçosamente limitado pelo tamanho de sua listagem.

O desenvolvimento de um sistema de geração de questões requer que os modelos de questão de interesse, concebidos por um professor da área, possam ser programados ou descritos em alguma linguagem de computador. Esta restrição leva a uma tendência de uso destes sistemas no contexto de disciplinas das STEM (Ciência, Tecnologia, Engenharia e Matemática), nas quais os modelos de questões podem ser mais naturalmente traduzidos nestas linguagens. Além disso, consideramos que a utilização adequada de um gerador de questões requer a união de três saberes distintos, quais sejam: didática, conhecimento de domínio na disciplina, e habilidades de programação, embora a necessidade deste último possa ser mitigada através de um arquitetura de sistema que priorize a interação com o usuário. Citamos, como exemplos de sistemas comerciais voltados para educação nas STEM que possuem funcionalidades de geração de questões, os sistemas Numbas[2], criado pela Universidade de Newcastle em 2015, e o Maple T.A.[3], da empresa canadense MapleSoft.

A aplicação mais direta de um gerador de questões reside na avaliação de aprendizagem, em particular permitindo a preparação de uma avaliação individualizada. Isto permite a minimização de problemas como plágio e memorização de soluções por parte dos alunos. Um benefício adicional desta aplicação é a possibilidade de correções automatizadas de avaliações, especialmente para questões seguindo determinados formatos, como associação ou múltipla escolha, em que as respostas corretas podem ser determinadas no momento da instanciação da questão. Ambos os benefícios são particularmente interessantes tanto em iniciativas de educação em massa como em educação a distância.

Neste trabalho, estamos interessados em como a geração de questões pode ser utilizada no decorrer do processo de aprendizagem. Acreditamos que um gerador de questões pode servir aos alunos como uma ferramenta para identificar lacunas no aprendizado de conteúdos específicos de forma mais precisa, e que tais lacunas podem ser mais facilmente sanadas mediante interações com monitores, tutores, e até mesmo colegas de turma. No contexto desta aplicação, um dos principais desafios, consiste em avaliar o grau de dificuldade de uma questão gerada de forma automatizada, problema que abordamos neste trabalho.

3. Sistema de geração de questões

Nesta seção, apresentamos a arquitetura do sistema de geração de questões desenvolvido e descrevemos o modelo de questões implementado neste sistema e utilizado posterior-

[2]https://numbas.mathcentre.ac.uk/

[3]https://www.maplesoft.com/products/mapleta/index1a.aspx

mente em nosso estudo de caso.

3.1. Arquitetura do sistema

O núcleo do sistema de geração de questões utilizado neste trabalho foi desenvolvido primariamente na linguagem Python. As questões geradas por este núcleo são exportadas em formato JSON e servem de entrada para um sistema Web, escrito em PHP, que apresenta o enunciado das questões e as alternativas de resposta aos alunos, usuários do sistema. Esta arquitetura é ilustrada na Figura 1.

Ao iniciar o conjunto de exercícios, o aluno pode voluntariamente identificar-se. O conjunto de exercícios é composto por 10 questões, cada uma podendo admitir nenhuma, 1 ou no máximo 2 respostas corretas. O aluno tem 2 chances para acertar cada questão. Ao final é exibido um relatório sobre o seu desempenho.

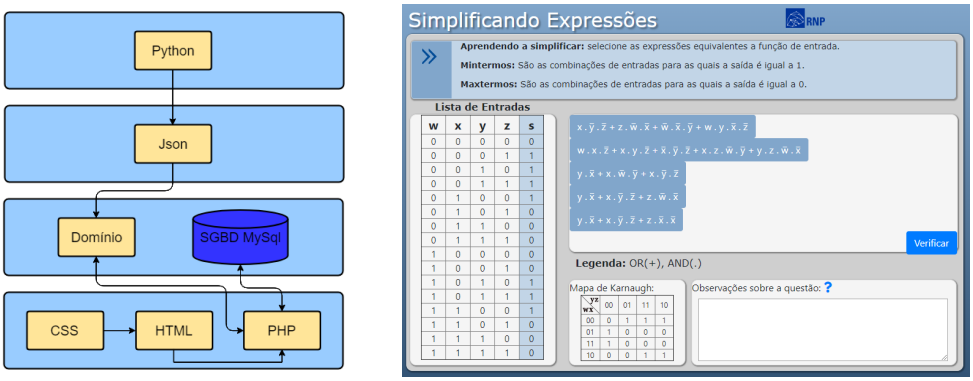


Figura 1. Arquitetura do sistema e tela ilustrando uma questão.

Todos os resultados colhidos são armazenados em um banco de dados MySQL. Um dos subprodutos desta pesquisa é o *dataset* com as respostas dos alunos, que será compartilhado a posteriori, para preservar o anonimato dos autores.

O modelo que optamos por implementar neste sistema, para fins do nosso estudo, versa sobre simplificação de expressões booleanas a partir da tabela verdade. Este é um tópico fundamental em cursos de níveis superior e médio na área de computação, e um dos tópicos em que os alunos costumam ter mais dificuldade durante o primeiro ano de estudos no curso de Tecnologia em Sistemas de Computação do CEDERJ. Para gerar questões automaticamente, seguimos os seguinte passos: (i) construímos aleatoriamente uma tabela verdade, com um pre-determinado número de mintermos (ou seja, um pre-determinando número de linhas da tabela verdade cuja saída é igual a 1); (ii) produzimos duas formas simples (soma de produtos e produto de somas) que caracterizam a tabela verdade. Essas são duas possíveis respostas certas apresentadas aos alunos. Esta etapa foi realizada utilizando a biblioteca *sympy* para a linguagem Python; (iii) aplicamos modificações aleatórias para gerar respostas erradas a partir das respostas certas; (iv) escolhemos aleatoriamente algumas respostas certas e erradas, e concluímos a montagem de uma questão.

O número de mintermos na tabela verdade é um dos elementos que determina a dificuldade das questões. O processo de introdução de erros e perturbações nas respostas é outro dos pontos chaves que distinguem uma questão fácil de uma questão difícil. Ambos

estes elementos foram considerados e incorporados ao modelo de questão utilizado. Neste trabalho, consideramos os seguintes tipos de erros para gerar respostas incorretas: (i) **nenhum erro (correta)**: resposta correta; (ii) **inversão de termo na tabela verdade (invertida):** trocamos um número 0 por um número 1 na tabela verdade, e recalculamos a expressão lógica correspondente (para a tabela verdade modificada/errada); (iii) **gerada aleatoriamente (aleatoria):** criada aleatoriamente com mintermos que não têm relação com os mintermos originais utilizados para gerar a questão; (iv) **modificação de termo na resposta (modificada):** trocamos aleatoriamente uma letra (e.g., trocamos x por y) na reposta correta.

Note que a introdução dos erros acima gera níveis distintos de dificuldade. Entretanto, não sabemos de antemão quais irão gerar questões mais fáceis ou mais difíceis. Pretendemos aprender isso de forma automatizada.

A seguir, listamos todos os atributos coletados. Os primeiros atributos estão relacionados à tabela verdade: **questão**, nome do arquivo JSON que contém a questão; **qtentrada**, (valor inteiro, 3 ou 4) número de variáveis de entrada. Os valores 3 e 4 geram, respectivamente, tabelas verdade com 8 e 16 linhas [igual a 4 na Figura 1]; **qtmintermos**, (valor inteiro entre 4 e 7) quantidade de mintermos utilizados na questão, ou seja, quantidade de linhas com saída igual a 1 na tabela verdade [igual a 7 na Figura 1].

Os atributos seguintes correspondem a erros propositais introduzidos pelo desenvolvedor da questão, conforme indicado na Seção 3.1: **qtdcorretas**, número de respostas corretas; **qtdmodificada**, número de respostas com string modificada; **qtdaleatoria**, número de respostas com erro aleatório; **qtdinvertida**, número de respostas com entrada invertida (vide Seção 3.1).

Os atributos finais correspondem a respostas oferecidas pelo usuário. O **tempo_gasto** é o tempo gasto para resolver a questão e **acertou** determina se a questão foi respondida corretamente.

Cabe destacar que o tempo de resposta é uma aproximação para o tempo que os alunos levaram para responder as questões. Não sabemos realmente se ao iniciar a questão o aluno ficou focado nela tentando responder ou se foi realizar alguma outra atividade, deixando o exercício em aberto em seu *browser*. Em trabalhos futuros, pretendemos também medir, presencialmente, o desempenho dos alunos em um ambiente controlado para melhor aferir o tempo gasto, mas destacamos que para os fins deste trabalho a metodologia adotada já se mostrou promissora (vide Seção 5).

3.2. Atributos para classificação da dificuldade usando aprendizado por máquina

A seguir, descrevemos os atributos utilizados para fins de classificação das questões em fáceis ou difíceis. Os seis atributos sublinhados na Seção 3.1 são usados para classificar as questões em fáceis ou difíceis. Para fins de aprendizado supervisionado, tomamos o atributo **acertou** (indicando se o usuário acertou ou não aquestão) como nossa variável resposta (alvo).

Note que segundo a descrição do parágrafo anterior, todos os atributos utilizados para determinar se uma questão é fácil ou difícil estão disponíveis *a priori* antes de a questão ser apresentada ao aluno. Para fins de recomendação de questões, esses são os atributos mais relevantes, que alimentamos numa árvore de decisão. A árvore de decisão

Tabela 1. Média (μ) e desvio padrão (σ) de cada variável

	qtdentrada	qtdmintermos	qtdcorretas	qtdmodificada	qtdaleatoria	qtdinvertida	tempo (seg)
μ	3,69	5,35	1,29	0,47	1,93	1,31	94,40
σ	0,46	1,00	0,48	0,50	0,75	0,78	192,12

é facilmente interpretável, e permite entender quais variáveis do algoritmo de geração de questões são mais relevantes na hora de criar uma questão fácil ou difícil. Assim, pode-se ajustar estes parâmetros ao longo do exercício para gerar questões de acordo com o nível do aluno.

Considerar o tempo que levou para os alunos responderem uma questão pode ser valioso para fazer um ajuste fino do algoritmo de aprendizado. Intuitivamente, se o aluno demora muito para responder uma questão é um sinal de dificuldade. Assim, para fins de explicar, em retrospectiva, o desempenho dos alunos, é interessante também entender a relação entre **tempo_gasto** e a taxa de acerto. Por esse motivo, levamos em conta tal variável na Seção 5. Vislumbramos também que pode-se usar o tempo que o aluno demorou para responder questões anteriores como um indicativo do tempo que o aluno irá despender para responder questões futuras, e deixamos esta análise para trabalhos futuros. Finalmente, o tempo de resposta também pode ser usado para remoção de *outliers*, tendo em vista que se um aluno respondeu uma questão muito rápido provavelmente ele ”chutou”a resposta.

4. Experimento realizado

Os alunos, primordialmente do CEDERJ e de mais uma instituição do Rio de Janeiro (não identificada para preservar o anonimato dos autores), utilizaram o sistema durante o período de setembro de 2018 a março de 2019 e responderam centenas de questões. Essas questões foram utilizadas como base para o treinamento dos algoritmos de aprendizado por máquina descritos na seção seguinte.

Um de nossos maiores desafios consistiu em coletar uma quantidade estatisticamente significativa de respostas dos alunos. No início, o processo de levantamento de dados não estava evoluindo no ritmo esperado e, por isso, alguns ajustes foram necessários. Dentre as estratégias que adotamos para contornar a dificuldade de coletar um conjunto estatisticamente significativo de respostas, anunciamos nosso experimento online na plataforma Moodle do CEDERJ e criamos vídeos online no Youtube (cujo link não incluímos aqui para preservar o anonimato dos autores). Consideramos que nosso *dataset* de respostas é uma contribuição relevante do trabalho. Após a aplicação das estratégias acima, foi possível povoar o banco de dados utilizado com 670 questões respondidas, sendo 499 respostas incorretas e 171 respostas corretas.

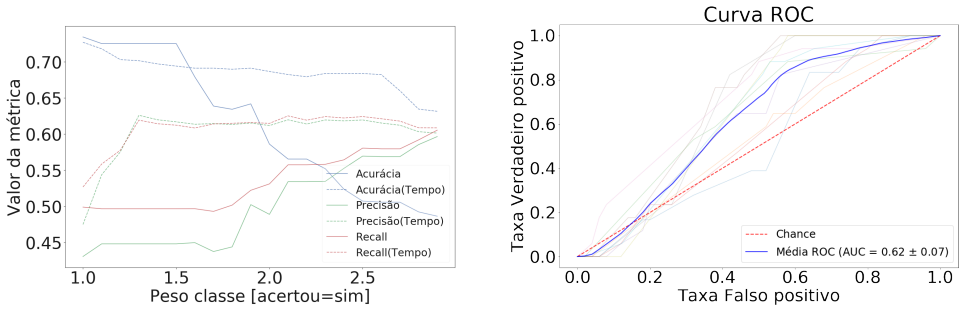
Na Tabela 1 temos as estatísticas básicas dos *atributos* do conjunto de dados. A maior variabilidade corresponde à variável tempo_gasto: alguns alunos responderam muito rápido algumas questões (possivelmente indicando chute) enquanto outros levaram muito tempo (possivelmente indicando que esqueceram a janela do *browser* aberta com a questão). Note também que, como esperado, a soma do número médio de respostas de cada tipo (1,29+0,47+1,93+1,31) é igual a 5, que corresponde ao número de respostas apresentadas por questão.

5. Avaliando a dificuldade das questões

A seguir, descrevemos a abordagem proposta para avaliar a dificuldade das questões. Iniciamos considerando árvores de decisão (Seção 5). e consideramos em seguida o método naive Bayes, contemplando o uso do tempo de resposta para fins de classificação da dificuldade das questões (Seção 5).

Para avaliar o impacto dos múltiplos atributos (*features*) sobre as dificuldades das questões, nós treinamos um modelo de árvore de decisão para prever se um aluno irá acertar ou errar uma questão.

A acurácia de um classificador é igual ao número de registros corretamente classificados, sobre o total de registros. Os atributos das questões usados para fins de treinamento dos algoritmos de aprendizado por máquina estão listados na Seção 3.2. Para efeito comparativo, consideramos a árvore C4.5 (usado por padrão pelo RapidMiner) e a árvore CART (usada por padrão no scikitleran) [Anyanwu and Shiva 2009]. Utilizamos parâmetros padrões em todos os casos. Obtivemos então acurácias de 80% e 75%, respectivamente. Consideramos também o uso de KNN, e obtivemos acurácia igual a 76% na melhor parametrização desse algoritmo. Dada a superioridade da acurácia da árvore C4.5 em nosso problema, focamos neste método no restante dessa seção, com excecção da análise a seguir, sobre desbalanceamento dos dados, executada com scikitlearn.



(a) Validação com e sem uso do *tempo* (b) Curva ROC

Figura 2. *Tradeoff* entre acurácia, *recall* e precisão na árvore de decisão.

O desbalanceamento dos dados (i.e., o fato de termos muito mais respostas erradas do que certas) é um desafio intrínseco ao problema que estamos considerando. Espera-se que a maioria dos alunos erre algumas questões, antes de começar uma fase de acertos, tendo em vista que estão em fase de aprendizado. Assim, discutimos a seguir algumas formas de se lidar com tal desbalanceamento sob a perspectiva da árvore de decisão. Na Seção 5 consideramos alternativas (via abordagem naive Bayes) para se lidar com o desbalanceamento.

Por termos muito mais registros da classe *acertou=não* (em que o aluno errou a questão), o treinamento da árvore de decisão fica enviesado e valoriza a classe dominante (classificando, com alta chance, uma nova questão como sendo da classe *não*). Existem inúmeras estratégias na literatura para contornar tal desafio. Para fins de ilustração, a seguir consideramos o uso de *pesos* para as classes. Ajustando os *pesos*, aumentamos o valor da classe que possui menos registros.

Denotando as classes *sim* e *não* como *positiva* e *negativa*, respectivamente, um falso (resp., verdadeiro) positivo corresponde à classificação de um elemento *não* (resp.,

sim) como *sim*. A sensibilidade (resp., precisão) da classe *sim* é igual ao número de verdadeiros positivos sobre o número de registros na classe *sim* (resp., número de registros classificados como *sim*). Na medida em que aumentamos o peso da classe *sim*, aumentamos o número de verdadeiros positivos, o que favorece um aumento da sensibilidade, podendo ou não diminuir tanto a precisão quanto a acurácia geral do sistema. A sensibilidade e precisão da classe *não* podem ser definidas de forma análoga. Nas figuras a seguir, apresentamos a média simples entre as métricas das duas classes, e.g., a curva de precisão corresponde à média entre as precisões das classes *sim* e *não*.

A Figura 2(a) mostra como a acurácia, precisão e sensibilidade (*recall*) variam em função do peso da classe *sim*. Linhas cheias (resp., pontilhadas) são obtidas com a árvore de decisão sem considerar-se a variável *tempo* (resp., considerando-se todos os atributos). No canto esquerdo da figura, a sensibilidade é igual a 0,5: acerta-se quase todos registros da classe *não* e erra-se quase todos da classe *sim*, produzindo um *recall* médio $\approx (1+0)/2$. A Figura 2(a) mostra que na medida em que aumentamos o peso da classe *acertou=sim*, aumentamos o *recall* (sensibilidade) da classificação, como esperado. A figura indica que a precisão também aumenta. Entretanto, como acabamos por classificar erroneamente registros da classe *acertou = não* como sendo da classe *acertou=sim*, a acurácia, i.e., a taxa de acertos geral cai.Temos então um *tradeoff* entre a qualidade das métricas de precisão e *recall* (obtidas via médias das classes) e a acurácia (obtida inerentemente para os registros como um todo). Tal *tradeoff* também se verifica quando consideramos o atributo *tempo* ao construir a árvore de decisão, porém de forma mais sutil (linhas pontilhadas na Figura 2(a)).

A Figura 2(b) mostra evidências adicionais relativas ao tradeoff entre falsos positivos e acurácia (assumindo que a classe *positiva* corresponde a *acerto=sim*). Dada uma certa taxa de falsos postivos (eixo x da curva ROC) temos a respectiva taxa de verdadeiros positivos factível (eixo y). Note que se admitirmos uma taxa de falsos positivos de 40%, por exemplo, alcançamos uma taxa de verdadeiros positivos em torno de 60%. A dificuldade de alcançarmos concomitantemente uma baixa taxa de falsos positivos e alta taxa de verdadeiros positivos advém do fato de nossos dados serem muito desbalanceados, e discutimos no restante desta seção formas de se lidar com tal desafio.

A Figura 3 ilustra a árvore de decisão obtida (via RapidMiner). Cada nó da árvore, exceto as folhas, contém uma regra de decisão binária usada para dividir o conjunto de dados. Além disso, cada nó também contém o número de amostras nesse estágio, indicando o número de questões que os alunos acertaram / erraram em tal ponto. Para a construção desta árvore, consideramos *pesos* iguais para as duas classes.

O nó raiz da árvore divide o conjunto de dados usando a quantidade de respostas corretas (QtdCorretas). Quando temos mais de uma resposta correta, quase sempre os alunos erram a questão. Os alunos tendem a esperar que um dos itens esteja correto, mas para fins de experimentação, admitimos que algumas questões tenham mais de uma resposta certa. A árvore consegue capturar, de forma automática, nossa intuição de que os alunos em geral esperam encontrar uma resposta certa.

Em seguida, a quantidade de respostas com erros de inversão de símbolos (QtdInvertida) também é relevante. Quando não há erros desse tipo, os alunos acertaram aproximadamente 50% das vezes. Por outro lado, quando há ao menos um erro desse

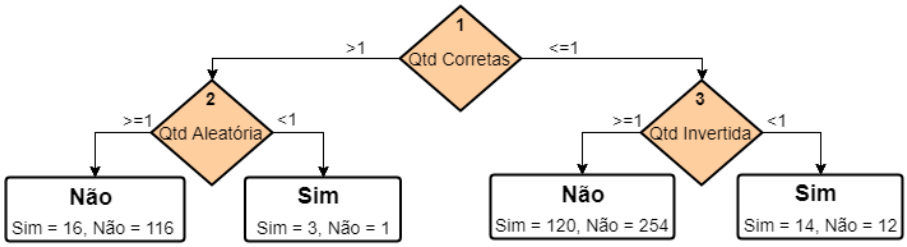


Figura 3. Árvore de decisão.

tipo, a taxa de acertos cai para aproximadamente 33%.

Propomos que a árvore seja usada para classificar e gerar questões em *níveis de dificuldade*, de acordo com a *fração* de respostas sim e não em cada uma de suas folhas, conforme descrito a seguir.

A árvore de decisão é usada, classicamente, para classificar novas instâncias em função de dados históricos. Desta forma, ela é lida *de cima para baixo, da raiz até as folhas*. Em nosso sistema, isso pode ser feito, por exemplo, gerando-se novas questões aleatoriamente e classificando-as de acordo com as recomendações da árvore. Alternativamente, pode-se usar as folhas da árvore para determinar, a partir da fração de respostas *sim* (corretas) e *não* (incorretas) de cada conjunto subjacente a uma folha, o nível de dificuldade daquele conjunto.

Percorrendo-se um ramo da árvore *de baixo para cima, de uma folha até a raiz,* pode-se determinar os atributos que geraram questões com determinado grau de dificuldade, e gerar novas questões de acordo. Por exemplo, a folha da árvore correspondente ao nó no qual 120 questões são da classe *sim* e 254 da classe *não* pode ser classificada como nível médio. Percorrendo-se a árvore *desta folha até a raiz*, pode-se gerar questões com dificuldade média gerando 2 respostas com erros do tipo *invertida* (nó 3 da árvore), 1 resposta correta (nó 1 da árvore) e 2 respostas *aleatórias* (não indicado na árvore).

Propomos, assim, o uso da árvore de decisão para fins generativos (i.e., para gerar novas questões). Cabe destacar que classicamente árvores de decisão são usadas para fins não-generativos, i.e., para classificação de amostras e não para geração de novas amostras. Assim, uma característica que costuma ser indesejável no projeto de uma árvore de decisão, que consiste em ter incerteza sobre as classes das folhas, é usado ao nosso favor, para estabelecer diferentes níveis (graus) de dificuldade e gerar questões de acordo.

A seguir, consideramos o uso da abordagem naive Bayes para determinação do grau de dificuldade das questões. A abordagem naive Bayes, como o nome indica, é ingenua (e intuitiva) permitindo-nos ganhar uma compreensão maior sobre como os diferentes atributos afetam o grau de dificuldade das questões. Nesta análise, consideramos também o tempo de resposta para fins de classificação do grau de dificuldade das questões.

A Tabela 2 é uma tabela de contingência, cruzando cada atributo do problema contra as classes de interesse (referente ao acerto da questão, *sim* ou *não*). Indicamos também na tabela o p-valor associado a cada atributo. O p-valor é calculado pelo teste qui-quadrado de independência entre um atributo e a variável resposta. Quanto menor o p-valor, maior tende a ser a capacidade preditiva desse atributo. Claramente, a Tabela 2

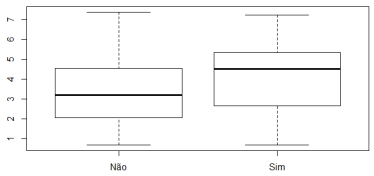
indica que o atributo Qtdcorretas (número de respostas corretas dentre as oferecidas como múltipla escolha na questão) é o mais relevante (vide Seção 5). É interessante também observar que os demais atributos que constam na árvore da Figura 5 são condicionalmente relevantes dado que Qtdcorretas é incluído na árvore. Entretanto, a Tabela 2 indica que tais atributos, por si só, são menos relevantes que Qtdmodificada e Qtdmintermos. Usando florestas de decisão (ao invés de uma simples árvore de decisão) corroboramos tal fato, que é discutido mais adiante nesta seção (Figura 5).

Tabela 2. Importância dos atributos na inferência da resposta correta. Quanto menor o p-valor, maior a relevância. Atributos com p-valor $>$ 0,3 ignorados.

Acertou?	Qtdmodificada		Qtdmintermos				Qtdcorretas		Total
	3	4	4	5	6	7	0 ou 1	2	
Não	281	218	120	173	120	86	331	168	499
Sim	80	91	50	45	42	34	159	12	171
p-valor	0,03861		0,2038				2,31e-11		

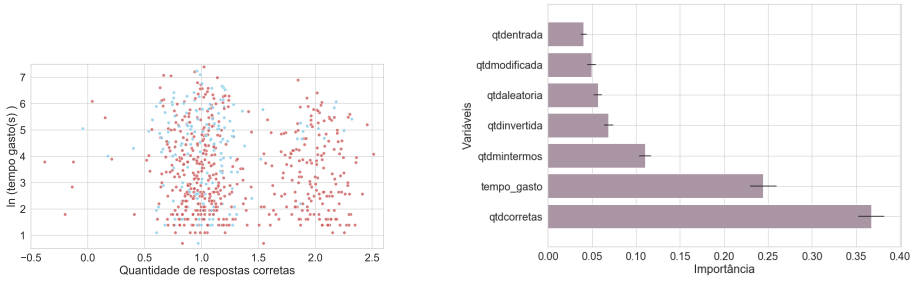
5.1. Tempo gasto

A Figura 4 apresenta um box plot do tempo gasto pelos alunos para responder as questões, em escala logarítmica, para cada valor da variável resposta. Claramente, dentre os alunos que acertam as questões o tempo de resposta tende a ser maior. Observando a Figura 4, escolhemos $\ln(tempo) = 4$ como ponto de corte. Indicamos no lado direito da Figura 4 a respectiva tabela de contingência. Calculamos então o p-valor correspondente ao uso de $\ln(tempo) \leq 4$ como critério de classificação de acerto. O p-valor obtido é baixo, igual a 8.693e-10, indicando a grande relevância do tempo consumido pelos alunos para prever o acerto das questões (e o correspondente nível de dificuldade das mesmas).



Classe de tempo	Não	Sim	Total
ln(tempo) $<$ 4	328	66	394
ln(tempo) $>$ 4	171	105	276
Total	499	171	670

Figura 4. Box plots do atributo $\ln(tempo)$ e tabela do tempo versus resposta.



(a) (b)

Figura 5. (a) *Scatter plot* entre as variáveis ln(tempo_gasto) e qtd_respostas; (b) importância das variáveis considerando o modelo *Random Forest*.

A Figura 5(a) apresenta um gráfico de dispersão (*scatter plot*) dos dois atributos identificados como mais relevantes para determinar se um aluno irá acertar uma questão:

Tabela 3. Probabilidades condicionais das condições e matriz de confusão.

Condição ($\mathcal{C}$)	P($\mathcal{C}$| acertou)	P($\mathcal{C}$| errou)
Qtdcorretas = 2	0,070	0,337
ln(tempo) > 4	0,614	0,343
Qtdmodificada = 1	0,257	0,437
Qtdmintermos > 5	0,444	0,413

Acertou via regra A	Acertou de fato Não	Sim	Total
Não	387	77	464
Sim	112	94	206
Total	499	171	670

quantidade de respostas corretas (Qtdcorretas) contra tempo gasto ($\ln(tempo)$). Note que como a quantidade de respostas corretas é uma variável discreta, introduzimos um pequeno ruido aleatório para gerar a Figura 5 de tal forma que não tenhamos todos os pontos sobrepostos. Os pontos marcados em vermelho correspondem a respostas incorretas. Segundo a Figura 5, quando a quantidade de respostas corretas é igual a 1, e o tempo gasto é alto, os alunos tendem a ter maior chance de acerto.

Com o objetivo de obter evidências adicionais sobre os atributos relevantes, adotamos uma abordagem alternativa para identificá-los. Usando uma floresta aleatória (*random forest*), pesquisamos os atributos que aparecem como separadores mais importantes. O resultado é apresentado na Figura 5(b). Mais uma vez, vemos que Qtdcorretas e o tempo gasto são os dois atributos mais relevantes. Logo em seguida, Qtdmintermos consta como o terceiro atributo mais significativo. A partir de então, as diferenças entre relevâncias são menores.

5.2. Naive Bayes

Com base nos p-valores acima, propomos um modelo preditivo do tipo naive Bayes, envolvendo os 4 atributos mais relevantes (com menores p-valores): Qtdcorretas, tempo, Qtdmodificada e Qtdmintermos. Buscamos uma relação do tipo,

$$P(acertou|x) = \frac{P(x|acertou)P(acertou)}{P(x|acertou)P(acertou) + P(x|errou)P(errou)} \quad (1)$$

onde x é o vetor formato pelos 4 atributos escolhidos. Todas as probabilidades aqui envolvidas foram estimadas a partir da base de dados. Temos que $P(acertou) = 0,255$ e $P(errou) = 0,745$.

Após inspeção dos dados e experimentação, convergimos no seguinte conjunto de condições a serem impostas para promover bipartições segundo os atributos em questão: (i) Qtdcorretas = 2; (ii) ln(tempo) > 4; (iii) Qtdmodificada = 1; (iv) Qtdmintermos > 5. A Tabela 3 contém, para cada uma dessas condições, as probabilidades condicionais de que ela tenha sido atendida para cada classe.

Seja x o vetor que caracteriza uma determinada condição de classificação segundo os 4 critérios acima. O vetor x é um vetor de dimensão 4, cujas coordenadas são iguais a 0 ou 1. A Tabela 4 contém, para cada valor de x, (i) a probabilidade condicional do vetor x, dado resposta correta; (ii) a probabilidade condicional do vetor x, dado resposta incorreta; (iii) a probabilidade condicional de resposta correta, dado o vetor x. Em conjunto, as Tabelas 3 e 4 apresentam todos os ingredientes necessários para se paramterizar a equação chave do algoritmo de classificação naive Bayes (eq. (1)).

Tabela 4. Probabilidades condicionais para cada possível vetor x de condições (vetores x com as 3 probabilidades condicionais menores que 10^{-1} removidos).

Condição	Valores (x)									
QtdCorretas = 2	0	0	0	0	0	0	0	0	1	1
$\ln(tempo) > 4$	0	0	0	0	1	1	1	1	1	1
QtdModificada=1	0	0	1	1	0	0	1	1	0	0
QtdMintermos > 5	0	1	0	1	0	1	0	1	0	1
P($x \mid$ acertou)	0,148	0,118	0,051	0,041	0,236	0,188	0,082	0,065	0,018	0,014
P($x \mid$ errou)	0,144	0,101	0,112	0,079	0,075	0,053	0,058	0,041	0,038	0,027
P(acertou $\mid x$)	0,260	0,286	0,136	0,152	0,518	0,550	0,324	0,353	0,138	0,154

Segundo a Tabela 4, as quatro maiores probabilidades a posteriori $P(acertou|x)$ correspondem aos casos em que $Qtdcorretas < 2$ e $\ln(tempo) > 4$. Então podemos considerar que uma regra de decisão adequada, e bastante simples, consiste em prever que o aluno irá acertar uma questão se $Qtdcorretas < 2$ e $\ln(tempo) > 4$ (regra A, na primeira linha da Tabela 5). Aplicando essa regra de classificação bem parcimoniosa aos dados, obtivemos uma acurácia de 0,72, apenas um pouco mais baixa que as acurácias obtidas por outros critérios mais elaborados. A matriz de confusão é apresentada na Tabela 3. Conforme podemos inferir a partir da matriz de confusão, e indicado na Tabela 5, o *recall* correspondente a essa regra é relativamente baixo, a saber, 0,55.

Visando aumentar o *recall*, testamos outras regras de classificação, B e C, ligeiramente mais complexas que a regra A. As regras B e C também são motivadas pela Tabela 4, i.e., são obtidas identificando combinações das condições sobre os atributos para as quais a probabilidade a posteriori $P(acertou|x)$ é alta. Como se pode observar na Tabela 5, para essas novas regras a taxa global de acerto é mais baixa, porém o *recall* é mais alto. Conforme discutido na Seção 5 (Figura 2), o desbalanceamento entre classes acarreta num significativo *tradeoff* entre acurácia e *recall*.

Um outro aspecto a ser considerado é o fato de que o tempo gasto no experimento não é um atributo que se conhece de antemão. Sendo assim, repetimos todo procedimento excluindo o atributo *tempo*. As regras de classificação D, E e F, excluindo o tempo, foram então derivadas usando a mesma metodologia descrita acima. Seus desempenhos refletem mais uma vez o *tradeoff* entre acurácia e *recall* (vide três últimas linhas da Tabela 5).

5.3. Mensagem chave desta seção

Nesta seção, apresentamos resultados preliminares que indicam que é possível, de forma automática, identificar o grau de dificuldade de questões. Em particular, indicamos que usando árvores de decisão ou o método naive Bayes obtemos critérios, alguns dos quais intuitivos, sobre o que torna uma questão de simplificação de expressões booleanas fácil ou difícil. Vislumbramos que tais critérios, quantititativos, obtidos de forma automática,

Tabela 5. Desempenho das regras de classificação baseadas em Naive Bayes

[Regra] Classificar como *acerto* se	Acurácia	Recall	Comentário
[A] Qtdcorretas < 2 e ln(tempo) > 4	0,72	0,55	referência parcimoniosa
[B] (Qtdcorretas < 2 e ln(tempo) > 4) ou (Qtdcorretas < 2 e ln(tempo) ≤ 4 e Qtdmodificada = 0)	0,66	0,64	aumenta *recall* diminui *acurácia*
[C] (Qtdcorretas < 2 e ln(tempo) > 4) ou (Qtdcorretas < 2 e ln(tempo) ≤ 4 e Qtdmodificada = 0 e Qtdmintermos ≤ 5)	0,60	0,73	
[D] Qtdcorretas < 2	0,49	0,93	não usa tempo
[E] Qtdcorretas < 2 e Qtdmodificada = 0	0,61	0,43	
[F] Qtdcorretas < 2 e Qtdmodificada = 0 e Qtdmintermos > 5	0,69	0,19	

podem ser usados em algoritmos de geração e recomendação de questões para alunos.

6. Trabalhos relacionados

O uso de aprendizado por máquina para auxiliar no ensino está em voga [Racy 2017, Bruno et al. 2017]. Neste trabalho, propomos uma nova forma de se utilizar o aprendizado por máquina neste contexto, para classificação da dificuldade de questões.

A literatura sobre aprendizado baseado em questões (*problem based learning*) é ampla [Araújo et al. 2015, Gavaza et al. 2017], mas os trabalhos sobre geração automática de questões são mais escassos [Singhal et al. 2016, Pérez et al. 2012, Kampff and Alves 2010]. Em particular, não é de nosso conhecimento trabalho anterior que tenha se aproveitado do aprendizado por máquina para automaticamente identificar o nível de dificuldade de questões que potencialmente ainda não tenham sido apresentadas a nenhum aluno (tal lacuna é apontada, direta ou indiretamente, como trabalho futuro em [Singhal et al. 2016, Pérez et al. 2012, Francisco et al. 2017]).

7. Conclusões

Nesta seção, discutimos alguns aspectos de nossos resultados que merecem destaque. *Primeiro*, classificamos a dificuldade de conjuntos de questões em função do fato de a maioria dos alunos terem acertado ou errado questões daquele conjunto. Entretanto, alunos novatos podem errar questões fáceis. Para levar em conta tal fato, em trabalhos futuros pretendemos analisar o histórico de acertos de cada aluno a fim de determinar o papel dos erros e acertos deste aluno na classificação da dificuldade de questões. *Segundo*, ainda não experimentamos mecanismos de recomendação de questões para avaliar como os alunos reagem a questões oferecidas em nível crescente de dificuldade, e medir sua efetividade. Usando informações sobre as notas dos alunos nas provas da disciplina, e correlacionando-as com o fato de terem participado de nosso experimento, pretendo adicionalmente avaliar a efetividade da proposta aqui apresentada. *Terceiro*, nosso estudo de caso envolve apenas um tipo de questão. Estamos no momento ampliando o menu de questões, tendo em vista as considerações apresentadas na Seção 3.1. *Quarto*, todas as questões consideradas são do tipo múltipla-escolha. Colocamos um campo livre para o aluno discutir como chegou ao resultado, e colocar comentários e dúvidas. Entretanto, poucos alunos preencheram tal campo. Pretendemos estimular tal preenchimento em iterações futuras do experimento. *Quinto*, colhemos, de forma voluntária, os contatos dos alunos. Com esses contatos, podemos recomendar grupos de estudos ou pareamento de alunos entre si, de forma a promover maior interação entre os mesmos.

Apresentamos neste trabalho uma nova abordagem para aumentar de forma escalável a interatividade no ensino a distância. A abordagem combina dois aspectos chaves: (i) geração automática de questões e (ii) identificação de dificuldade das mesmas. Esta abordagem conjunta visa aumentar a interatividade entre os alunos, professores e tutores. Não é de nosso conhecimento nenhum trabalho anterior que tenha combinado estas duas abordagens acima em um *framework* unificado. Usando árvores de decisão, conseguimots interpretar o que faz uma questão ser considerada fácil ou difícil e, de forma quantitativa, decidir e prever se uma nova questão tem maior chance de ser fácil ou difícil. Vislumbramos que esse é um passo adicional para promover maior interação entre alunos e professores, e.g., recomendando o pareamento entre alunos em função de suas disponi-

bilidades e de seus históricos de erros e acertos ou sugerindo que os alunos procurem os tutores com determinadas questões para serem resolvidas em conjunto.

Agradecimentos: Este projeto foi parcialmente financiado pelo CEDERJ, FAPERJ, CAPES e CNPq.

Referências

Abbad, G., Carvalho, R., and Zerbini, T. (2005). Evasão em curso a distância via internet: explorando variáveis explicativas. *Encontro da ANPAD*, 29.

Anyanwu, M. N. and Shiva, S. G. (2009). Comparative analysis of serial decision tree classification algorithms. *Journal of Computer Science and Security*, 3(3):230–240.

Araújo, D., Rodrigues, A., Silva, C., and Soares, L. (2015). O ensino de computação na educação básica apoiado por problemas. In *Anais do WEI*.

Bruno, G., de Souza e Silva, E., and Menasché, D. (2017). *Ciência para Educação*. Atheneu. Máquinas que ensinam: o que aprendemos com elas?

de Castro, R. M. (2018). Desenvolvimento e avaliação de uma metodologia de aprendizagem ativa apoiada pelo uso de qr code para ensino de banco de dados. In *26º Workshop sobre Educação em Computação (WEI 2018)*, volume 26. SBC.

Francisco, R. E., Ambrósio, A. P., et al. (2017). Grau de dificuldade de problemas de programação introdutória: Uma revisão sistemática. In *WEI*, volume 25. SBC.

Gavaza, L. O. R., do Nascimento Salvador, L., and dos Santos, D. M. B. (2017). Uma experiência de aplicação de uma abordagem baseada em problemas no ensino de teoria da computação em sala de aula tradicional. In *WEI*, volume 25. SBC.

Johnson, R. T. and Johnson, D. W. (2008). Active learning: Cooperation in the classroom. *The annual report of educational psychology in Japan*, 47:29–30.

Kampff, A. and Alves, G. (2010). NetAula-geração automática de avaliações. *RENOTE*, 8(3).

Pérez, E. V., Santos, L. M. R., Pérez, M. J. V., de Castro Fernández, J. P., and Martín, R. G. (2012). Automatic classification of question difficulty level: Teachers' estimation vs. students' perception. In *Frontiers in Education*, pages 1–5. IEEE.

Prietch, S. S. and Pazeto, T. A. (2010). Estudo sobre a evasão em um curso de licenciatura em informática e considerações para melhorias. In *Anais do WEI*.

Racy, S. (2017). Governo usará inteligência artificial para prevenir evasão escolar. Jornal Estadão.

Schwartzman, S. (2006). Redução da desigualdade, da pobreza, e os programas de transferência de renda. *Instituto de Estudos do Trabalho e Sociedade (IETS)*.

Silva Filho, R. L. L., Motejunas, P. R., Hipólito, O., and Lobo, M. B. C. M. (2007). A evasão no ensino superior brasileiro. *Cadernos de pesquisa*, 37(132):641–659.

Singhal, R., Goyal, S., and Henz, M. (2016). User-defined difficulty levels for automated question generation. In *Conf. Tools with Artificial Intelligence*, pages 828–835. IEEE.