Modelo Computacional para Correção Automatizada de Questões Discursivas

Fernando Fogliato, Júlio De Pieri

Curso de Ciência da Computação – Universidade do Sul de Santa Catarina (UNISUL)
Caixa Postal 370 – 88.704-900 – Tubarão – SC – Brasil

{fernando.fogliato,julio.pieri}@unisul.br

Abstract. *This paper describes a computational model with the objective of automating the correction of discursive questions of applied evaluations at the Universidade do Sul de Santa Catarina (UNISUL). Text mining techniques, similarity between documents and regression models Support Vector Regression and Random Forest are applied to predict the grades. The regression models did not show large differences in the predictions, where Random Forest had 76% prediction accuracy and SVR 73%. The mean difference between the grade given by the teacher and that predicted by the model was 0.38 points using Random Forest and 0.45 points using the SVR. Values that indicate the potentiality of the model.*

Resumo. *Este artigo descreve um modelo computacional com o objetivo de automatizar a correção de questões discursivas de avaliações aplicadas na Universidade do Sul de Santa Catarina (UNISUL). São utilizadas técnicas de mineração de textos, similaridade entre documentos e modelos de regressão Support Vector Regression (SVR) e Random Forest para predição das notas. Os modelos de regressão não apresentaram grandes diferenças nas predições, onde o Random Forest obteve 76% de acerto das predições e o SVR 73%. A diferença média entre a nota dada pelo professor e a predita pelo modelo foi de 0,38 pontos empregando o Random Forest e 0,45 pontos utilizando o SVR. Valores que indicam a potencialidade do modelo.*

1. INTRODUÇÃO

No processo de ensino, a utilização de métodos de avaliação de conhecimento se faz necessária para que se possa obter um *feedback* sobre a aprendizagem dos alunos. Segundo Libâneo (1994), a avaliação é uma tarefa didática necessária e permanente do trabalho docente, que deve acompanhar o processo de ensino e aprendizagem. Através das avaliações os resultados obtidos no decorrer do trabalho conjunto entre o professor e aluno são comparados com os objetivos propostos a fim de constatar progressos, dificuldades e se necessário, orientar para correções.

Entre as possíveis abordagens de avaliação de conhecimento dos alunos encontram-se às questões discursivas, que segundo Novak e Gowin (1984), Günther e Júnior (1990), avaliam melhor o raciocínio, pois demanda um entendimento e síntese do conteúdo para que seja elaborada uma resposta com argumentos coerentes ao contexto da pergunta.

A avaliação de conhecimento também é utilizada na modalidade de ensino a distância (EaD), na qual utiliza ambientes virtuais, como chats, fóruns e e-mails para conectar alunos e professores. O método de avaliação é similar ao presencial, porém, os alunos comparecem aos polos apenas para realização das avaliações, que empregam questões objetivas e discursivas.

A modalidade EaD no Brasil cresce a cada ano, e ocupa cada vez mais espaço no ensino superior [INEP 2017]. O Censo da Educação Superior de 2016, demonstrou que a modalidade EaD teve expansão de 7,2% no número de matrículas em relação a 2015, e participação de 18,6% do total de matrículas do ensino superior em 2016 [INEP 2017]. Este crescimento também é evidenciado pelo Censo da Associação Brasileira de Educação a Distância (ABED), que apontou cerca de 3,7 milhões de estudantes matriculados em algum curso de EaD no ano de 2016 [ABED 2017].

Com o crescimento acelerado do EaD, consequentemente aumentou o número de avaliações, assim, os professores têm dedicado mais tempo na correção das avaliações do que de fato na preparação das aulas. De acordo com um estudo publicado pela companhia de tecnologias linguísticas ABBYY em 2014, o tempo gasto por todos os professores brasileiros corrigindo provas chega anualmente a pelo menos 21,4 milhões de horas (894 mil dias) para verificar um total de 322 milhões de testes e exames, portanto, cada professor investe mais de 23 horas (3,9 dias de seis horas de trabalho) por ano na tarefa de corrigir provas [Batimarchi 2014].

Somente na Universidade do Sul de Santa Catarina, no seu programa de EaD UnisulVirtual no primeiro semestre de 2016, foram aplicadas 42.706 avaliações presenciais para um total de 14.286 alunos. A correção manual de questões discursivas dessas avaliações é um processo moroso que demanda muita atenção dos educadores envolvidos, já que precisam avaliar questão por questão utilizando um caminho de resposta como base.

Reconhecendo esta pesada carga de trabalho e visando aproximar-se das notas dadas pelos professores foi desenvolvido um modelo e método computacional para automatizar este processo utilizando a mineração de textos e modelos de regressão.

A mineração de texto (*Text Mining*) consiste na descoberta de informações utilizando técnicas de análise e extração de dados a partir de textos, frases ou apenas palavras escritas em linguagem natural [Weiss et al. 2005]. Essa ferramenta permite a análise qualitativa e quantitativa das respostas, e a melhor compreensão do conteúdo disponível nesses documentos textuais. Por se tratar de um problema de predição de valores numéricos (notas) é necessário aplicar modelos de regressão que possibilitem encontrar uma relação de razão razoável entre as variáveis de entrada (similaridade) e saída (nota) [Weisberg 2005].

O artigo está organizado em seções, as quais são descritas a seguir: a segunda seção apresenta os trabalhos relacionados ao modelo desenvolvido. A terceira seção refere-se à contextualização teórica sobre mineração de textos e modelos de regressão. A quarta seção descreve as ferramentas utilizadas no desenvolvimento do modelo e o método computacional. A quinta seção apresenta os resultados e suas discussões. Por fim, a sexta seção apresenta as conclusões relacionadas a aplicação do modelo e ideias para trabalhos futuros.

2. TRABALHOS RELACIONADOS

O primeiro sistema de correção automatizada de questões discursivas surgiu em 1966, conhecido como Project Essay Grader (PEG), foi desenvolvido pelo professor Ellis Batten Page com o intuito de ser uma possível solução para aliviar a carga dos professores na correção de provas. O modelo empregado pelo PEG para a correção das questões é dividido em uma etapa de treino e uma de pontuação, necessitando por volta de 100 a 400 respostas para treinar o modelo [Dikli 2006].

Segundo Chung e O'Neil (1997), o sistema PEG não leva em consideração o significado ou relevância do conteúdo das respostas para realizar a pontuação, o foco deste modelo é voltado para avaliar as características do estilo de escrita do texto chamadas de proxy, como comprimento médio de palavras, quantidade de palavras no texto, quantidade de vírgulas, quantidade de preposições, quantidade de palavras incomuns, entre outras. No treinamento do modelo PEG são aplicadas técnicas estatísticas de regressão linear múltipla sobre as respostas para se obter as combinações de pesos que definem a relação entre as características avaliadas e a nota do professor, buscando montar uma estrutura que forneça a melhor aproximação das pontuações dadas pelos professores [Ferster 2014].

Em um dos testes iniciais do sistema PEG foram alcançadas taxas de correlação entre as notas do sistema e dos professores de 0,64 a 0,78 com uma quantidade de 276 exemplos, sendo que a correlação entre os professores fica entre 0,72 e 0,85 [Chung e O'Neil 1997]. No entanto, os críticos argumentam que o sistema analisa superficialmente os textos, dessa forma, podendo ser enganado pelos estudantes ao dar pontuações altas para textos longos [Dikli 2006].

A empresa norte americana Measurement Inc. adquiriu os diretos do projeto PEG em 2003, após dez anos o software foi remodelado utilizando as últimas técnicas em linguística, aprendizado de máquina e processamento de linguagens naturais. A última versão do software analisa as respostas calculando mais de 300 pontos que refletem as características da escrita, como fluência, gramática, construção, entre outros, e obteve bons resultados em comparação com as notas dadas por avaliadores humanos. Atualmente o PEG está sendo utilizado como ferramenta de auxílio em mil escolas dos EUA para fins de validação e melhoria do processo de avaliação [PEG 2017].

Outro sistema de correção automatizado de questões que ganhou destaque nos últimos anos foi o edX AES System da plataforma online de ensino edX, que precisa de apenas 100 questões corrigidas pelo professor para começar a funcionar [Markoff 2013]. Os resultados obtidos até então variam muito, sendo constantemente criticados pela sua eficácia em textos longos.

Existem atualmente duas formas conhecidas de pontuação de questões discursivas utilizadas pelos softwares de correção, que são a holística e a rubrica (também chamada de analítica). Pontuação holística envolve fazer uma avaliação sobre os aspectos gerais do texto, se o mesmo é coeso, coerente, se está dentro do assunto proposto ou não. Enquanto analítica refere-se à atribuição de múltiplas contagens com base em várias características de uma resposta, como clareza, organização, gramática e ortografia [Burstein; Leacock e Swartz 2001]. O sistema da edX utiliza ambos os métodos, gerando a pontuação tanto analítica como holística para respostas dos alunos, mas a pontuação

holística é dada como a nota final da resposta.

Um estudo de eficiência do software da edX realizado pela Universidade do Texas, EUA, demonstrou uma correlação entre as notas dadas pelo sistema e por professores humanos entre 73,89% e 79,31%. O estudo foi realizado utilizando questões dos cursos online da edX, farmácia e filosofia, comparando as notas obtidas pelo edX-Holistíco e edX-Rubrico com as notas dadas pelos instrutores da disciplina. O sistema da edX tende a dar notas com diferença de um a dois pontos, para mais ou para menos, em relação às dadas pelos instrutores, portanto, estes resultados indicam a necessidade de melhorias nos critérios de pontuação utilizados pelo software para reduzir essa diferença.

Outra forma de corrigir questões foi apresentada pelo pesquisador Santos (2016) em sua tese de doutorado, onde criou um modelo usando *Latent Semantic Analysis* (LSA) na avaliação automática de respostas curtas, com média de 25 a 70 palavras, de questões discursivas. Nesta pesquisa foram utilizados dois domínios de aplicação: uma questão discursiva de natureza conceitual de Biologia e outra de natureza argumentativa de Geografia.

Os melhores resultados foram obtidos com as respostas para a questão de Geografia, embora fosse usada a mesma arquitetura do modelo LSA para ambos conjuntos de respostas. A diferença se deu por ter sido considerada para a questão de Geografia as três melhores respostas avaliadas pelos professores como a resposta de referência, enquanto para a questão de Biologia foi um texto dado por um especialista humano. O modelo proposto trabalhou tanto com unigramas[1] quanto com bigramas, sendo que o uso dos bigramas não trouxe qualquer melhoria significativa ao modelo [Santos 2016].

Para avaliar o desempenho do modelo foi comparada a acurácia do modelo frente a dois especialistas humanos. Para a questão de Biologia a diferença entre os índices de acurácia do avaliador humano e do programa LSA foi um pouco mais de 10%. Uma provável justificativa segundo o autor é que esta diferença foi o fato de ter considerado a resposta referência como sendo um único texto dado por um especialista humano.

Para a questão de geografia os índices de acurácia entre dois especialistas humanos e do programa LSA foi praticamente o mesmo. Uma provável justificativa segundo o autor é que para este fato foi considerado como resposta referência a concatenação das respostas mais bem avaliadas por especialistas humanos. Portanto, o sistema tem melhor desempenho quando ocorre um aumento do vocabulário da resposta referência.

3. CONTEXTUALIZAÇÃO

3.1 Análise de Textos

A mineração de texto (*Text Mining*) é uma evolução da área de Recuperação de Informações (RI), que consiste na descoberta de informações utilizando técnicas de análise e extração de dados a partir de textos, frases ou apenas palavras escritas em linguagem natural [Weiss et al. 2005]. Envolve a aplicação de algoritmos computacionais

[1] Unigramas são palavras únicas dentro do corpus (tokens), já bigramas é a combinação de dois unigramas em sequência, trigramas a combinação de três unigramas e assim sucessivamente.

que processam textos e identificam informações úteis e implícitas, que normalmente não poderiam ser recuperadas utilizando métodos tradicionais de análise, pois as informações contidas nestes textos estão por vezes armazenadas em formato não estruturado [Feldman e Sanger 2007].

Os benefícios dessa técnica podem se estender a qualquer domínio que utilize textos. As principais contribuições estão relacionadas à busca de informações específicas em documentos, à análise qualitativa e quantitativa de grandes volumes de textos, e a melhor compreensão do conteúdo disponível em documentos textuais.

3.2 Random Forest

Inventado e desenvolvido por Leo Breiman, *Random Forest* é um método de aprendizado *ensemble* que faz uso de uma combinação de árvores de decisões para aumentar a taxa de acerto na predição dos resultados [Breiman 2001]. Neste modelo são utilizadas coleções de árvores de decisão aleatórias (*Random Trees*) como preditores, sendo que cada árvore depende dos valores de um vetor aleatório de amostras independentes (amostras *bootstraps*) e com a mesma distribuição para todas as árvores na floresta.

Métodos de *ensemble* são algoritmos de aprendizado que constroem um conjunto de classificadores e então, com base em um voto (ponderado) de suas previsões, classificam novos exemplos [Dietterich 2000]. O voto corresponde a classe predita pelos classificadores para o novo exemplo, sendo que para um modelo de regressão ao invés de uma classe o modelo irá predizer um valor numérico.

Bootstrap [Efron 1979] é um método de amostragem com reposição, neste método novos subconjuntos são criados aleatoriamente com reposição a partir do conjunto original. Cada subconjunto é criado com o mesmo tamanho do conjunto original e devido a reposição, alguns exemplos podem ser selecionados mais de uma vez e outros não serem selecionados.

Bagging (*bootstrap aggregating*) segundo [Breiman 1996] é um método *ensemble* para gerar múltiplas versões de um preditor e usá-las para obter um preditor agregado. É feita a média das múltiplas versões dos preditores ao se tentar prever um resultado numérico e se faz uma pluralidade de votos no caso de os resultados serem uma classe. As múltiplas versões são formadas criando amostragens *bootstrap* do conjunto de treinamento principal e usando estes como novos conjuntos de treinamento. Os testes em conjuntos de dados reais e simulados usando árvores de classificação, regressão e seleção de subconjuntos em regressão linear mostram que o *bagging* pode dar ganhos substanciais de precisão [Breiman 1996]. A precisão deste método é aumentada se o método de predição é instável, como no caso de árvores de decisão, onde pequenas mudanças no conjunto de treinamento ou nos parâmetros utilizados na construção podem resultar em grandes mudanças no resultado da árvore. A cada nó da árvore, um subconjunto de m atributos é selecionado aleatoriamente do conjunto total de atributos e avaliado. O melhor atributo é, então, escolhido para dividir o nó. O valor m é fixado para todos os nós. As árvores crescem sem poda [Oshiro 2013].

O resultado do Random Forest para regressão é formado tomando a média sobre os resultados das árvores individuais. A figura 1 mostra o funcionamento do modelo *Random Forest*, nesta imagem podemos ver como as amostragens *bootstrap* são

utilizadas no treinamento do modelo, estas alimentam as árvores de decisão geradas aleatoriamente (*Random Trees*), as árvores de decisão com base nestes conjuntos de treinamento predizem um valor e a média destas previsões se torna o resultado do modelo, sendo que ao gerar as árvores são selecionados n atributos aleatoriamente para as árvores, diferenciando assim o modelo *Random Forest* do *bagging*.

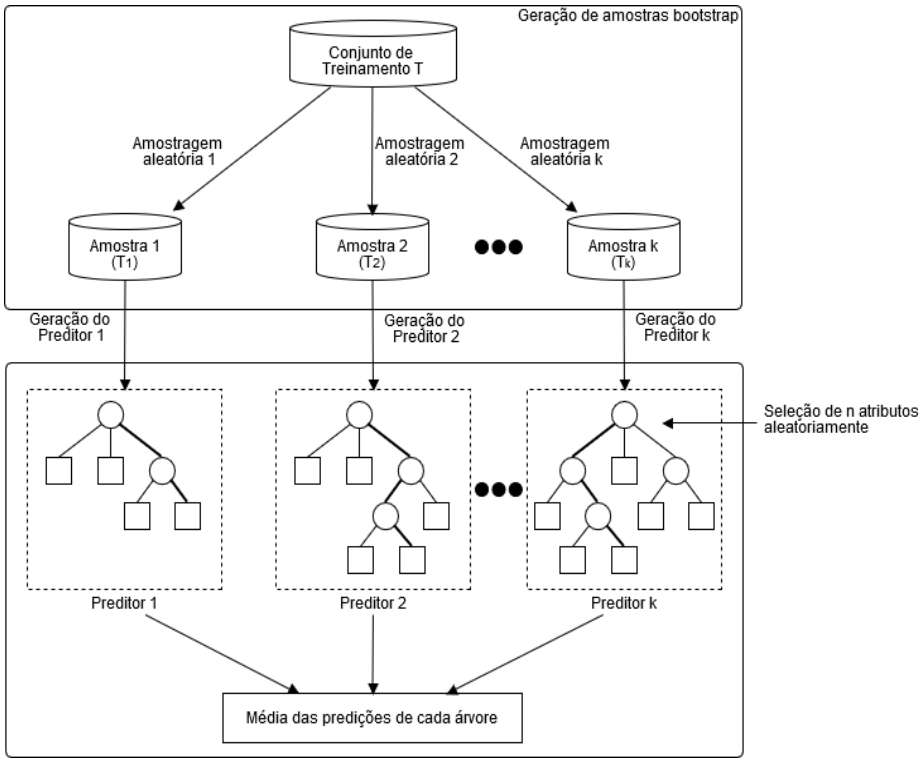

Figura 1. Funcionamento do método Random Forest

3.3 Support Vector Machine - Regression (SVR)

Inventado por Vapnik e seus colaboradores [Boser et al. 1992] as máquinas de vetores de suporte (SVM) são sistemas de aprendizagem de máquina que usam espaços de hipóteses de funções lineares na alta dimensionalidade característica do espaço, treinado como algoritmo de aprendizagem da Teoria de Otimização, que implementa um limite derivado da Teoria da Aprendizagem Estatística. [Gevert et al. 2010].

A SVM trabalha com o princípio da minimização do risco estrutural, buscando minimizar um limite superior do erro de generalização em vez de minimizar o erro de treinamento, que é o princípio seguido pelas redes neurais artificiais (RNAs). A ideia básica do SVR consiste em imaginar um tubo (margem) em volta do traçado da função (separador de margem) de aproximação, encontrando uma função que tenha no máximo um desvio de ϵ sobre todos os exemplos, procurando-se assim obter um tubo que seja o mais estreito possível. Os pontos mais próximos do separador são chamados de vetores de suporte porque sustentam o plano de separação como mostra a figura 2.

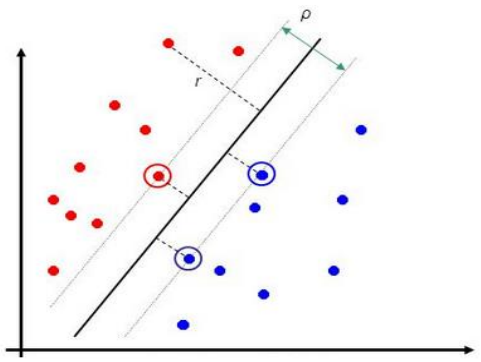

Figura 2. Máquina de vetor de suporte (SVM)

As SVMs criam uma separação linear em hiperplano, mas tem a capacidade de incorporar os dados em um espaço de dimensão superior, usando o assim chamado truque de Kernel [Hofmann 2006]. Assim os dados que não são separáveis linearmente no espaço de entrada original são facilmente separáveis em um espaço de dimensão superior [Russell e Norvig 2013].

Uma das abordagens para se trabalhar com problemas de regressão não linear é utilizar a teoria do Lagrangeano [Bertsekas 1982] para obter a forma *dual* da SVM [Haykin 2001]:

$$Max(\alpha_i \alpha'_i) = \sum_{i=1}^{N} d_i(\alpha_i - \alpha'_i) - \epsilon \sum_{i=1}^{N} (\alpha_i + \alpha'_i) - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha'_i)(\alpha_j - \alpha'_j) k(x_i, x_j)$$

Sujeito as seguintes restrições:

$$\sum_{i=1}^{N} (\alpha_i - \alpha'_i) = 0$$

$$0 \leq \alpha_i \leq C, \qquad i = 1,2, \dots, N$$
$$0 \leq \alpha'_i \leq C, \qquad i = 1,2, \dots, N$$

Onde: C = Constante especificada pelo usuário para penalização
$\alpha \; e \; \alpha'$ = Multiplicadores de Lagrange
ϵ = Valor de parâmetro que define uma margem de tolerância onde nenhuma penalidade é aplicada aos erros no treinamento.
x_i e x_j= Valor de amostra do vetor de entrada x
d_i= Valor correspondente da saída do modelo
N = Número de exemplos
k = Função Kernel

A SVM pode ser utilizada tanto para problemas de classificação como de regressão, sendo as vezes referenciada como SVR (*Support Vector Regression*). A ideia principal é a mesma: minimizar o erro e individualizar o hiperplano que maximiza as margens. O SVR é baseado na construção de uma função de regressão linear em um espaço de recursos de alta dimensão onde os dados de entrada são mapeados por uma função não linear.

4. MATERIAIS E MÉTODOS

4.1 Ferramentas

Para o desenvolvimento do modelo computacional foi utilizada a linguagem de programação Python versão 3.6, o banco de dados PostgreSQL versão 9.4 e as seguintes bibliotecas CoGrOO 4.0, Scikit-learn 0.18.1 e Nltk 3.2.2.

O CoGrOO é um corretor gramatical de código aberto da suíte de escritório Open Office. Ele é capaz de identificar erros como colocação pronominal, concordância nominal, concordância sujeito-verbo, uso da crase, concordância nominal, e verbal e outros erros comuns de escrita em português do Brasil.

Scikit-learn é uma biblioteca de código aberto implementada em Python contando com inúmeros algoritmos e ferramentas para data mining e análise de dados.

O NLTK (*Natural Language Toolkit*) é a biblioteca de código aberto mais utilizada na criação de programas Python para trabalhar com dados de linguagem humana. Ele fornece interfaces fáceis de usar para mais de 50 corpus e recursos léxicos como o WordNet, juntamente com um conjunto de bibliotecas de processamento de texto para classificação, tokenização, derivação, marcação, análise e raciocínio semântico.

4.2 Modelo Computacional

4.2.1 Dados

Para o desenvolvimento do modelo foi necessária uma base de histórico de correções de avaliações, dessa forma, a base de dados foi construída a partir de questões de cursos da modalidade EaD da UNISUL e avaliações aplicadas presencialmente para alunos das disciplinas de Arquitetura de Computadores, Grafos e Desenvolvimento de Jogos do curso de Ciência da Computação da UNISUL. As informações foram tratadas de maneira confidencial, preservando a identidade dos alunos.

Como os dados foram exportados pela equipe da Gestão de Processos e Tecnologia da Informação (GPTI) da UNISUL em formato de imagem, foi necessário transcreve-las para texto. Essa transcrição ocorreu de forma manual conforme mostra a figura 3.

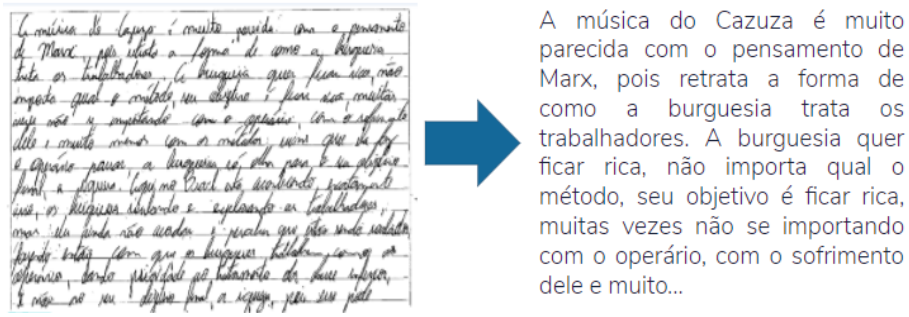

Figura 3. Exemplo de transcrição da resposta no formato imagem para texto

Após a transcrição das respostas as mesmas foram agrupadas em 3 classes de notas: baixa, média e alta. Essas classes foram atribuídas a partir de um limiar dependendo da nota máxima possível na questão (tabela 1). Foi necessário realizar esta

divisão para que na etapa de treinamento do modelo houvesse uma distribuição proporcional de exemplos nos conjuntos de treino/teste, pois como o volume de informações é pequeno, a técnica de separação de dados aleatória acabaria distribuindo os exemplos de forma irregular, afetando assim, a generalização do modelo.

Tabela 1. Classificação das notas em três classes possíveis: baixa, média e alta.

Questão	Condições onde x = nota	Qtd. Baixa	Qtd. Média	Qtd. Alta
Q1	$f(x) = \begin{cases} Baixa, & x \leq 1{,}6 \\ Média, & x > 1{,}6 \; e \; x \leq 3{,}5 \\ Alta, & x > 3{,}5 \end{cases}$	9	10	18
Q2	$f(x) = \begin{cases} Baixa, & x \leq 1{,}0 \\ Média, & x > 1{,}0 \; e \; x \leq 2{,}0 \\ Alta, & x > 2{,}0 \end{cases}$	11	7	13
Q3	$f(x) = \begin{cases} Baixa, & x \leq 1{,}0 \\ Média, & x > 1{,}0 \; e \; x \leq 2{,}5 \\ Alta, & x > 2{,}5 \end{cases}$	9	6	9
Q4	$f(x) = \begin{cases} Baixa, & x \leq 1{,}0 \\ Média, & x > 1{,}0 \; e \; x \leq 2{,}0 \\ Alta, & x > 2{,}0 \end{cases}$	13	11	32
Q5	$f(x) = \begin{cases} Baixa, & x \leq 1{,}0 \\ Média, & x > 1{,}0 \; e \; x \leq 2{,}0 \\ Alta, & x > 2{,}0 \end{cases}$	9	51	97

A tabela 2 mostra as quantidades de respostas utilizadas para treinamento/teste do modelo. E o gráfico 1 apresenta a quantidade de respostas distribuídas por classes. Nota-se pela tabela 2 que as respostas variam de muito curtas, como a questão 2 com média de 14 palavras, a muito longas, como no caso da questão 5 com média de 119 palavras.

Tabela 2. Relação de questões utilizadas no modelo

Questão	Quant. Respostas	Média palavras por resposta	Maior resposta	Desvio padrão
Q1	37	15,56	49	9,72
Q2	31	13,93	29	5,02
Q3	24	80,45	193	42,86
Q4	56	81,60	193	33,13
Q5	157	119,5	220	38,96
Total/Média	**305**	**62,21**	**136,8**	**25,94**

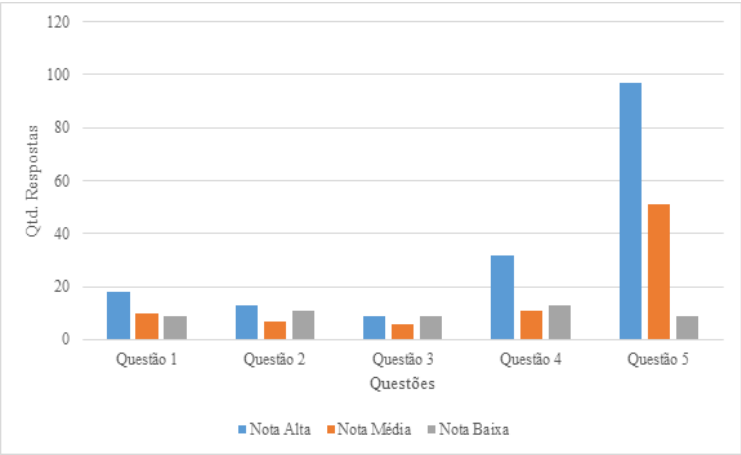

Gráfico 1. Distribuição das respostas conforme a divisão por classes

Abordagem das cinco questões utilizadas no modelo:

- A questão 1 é referente a disciplina presencial de Arquitetura de Computadores do curso de Ciência da Computação, composta por 37 respostas, com nota variando de 0 a 5, possuindo um caminho de resposta curto e objetivo. As respostas em sua maioria são curtas, média 15 palavras e fazem uso de siglas para descrever componentes da arquitetura.
- A questão 2 é referente a disciplina presencial de Grafos do curso de Ciência da Computação, composta por 31 respostas curtas, média 14 palavras, que possuem nota entre 0 e 3.
- A questão 3 é relativa à disciplina presencial de Desenvolvimento de Jogos do curso de Ciência da Computação, conta com 24 respostas variando de 0 a 3 e com média de 80 palavras por resposta.
- A questão 4 é referente a disciplina de Sociologia, que apresenta duas notícias de jornais do Brasil, e os alunos devem, segundo o sociólogo Anthony Giddens, identificar qual o tipo de exclusão apresentada nos textos e explicar o porquê da escolha. Composta por 57 respostas variando entre 0 e 2, 5 e com uma variação de respostas com média de 81 palavras por resposta.
- A questão 5 é referente a disciplina de Sociologia, com 157 respostas e com uma média de 119 palavras por resposta, o que apresenta uma grande variação de termos e possibilidades. As notas variam entre 0 e 2,5.

4.2.2 Processamento

Para determinar a nota do aluno o sistema utiliza o caminho da resposta dado pelo professor como resposta ideal e efetua comparações com outras boas respostas dadas por outros alunos, atribuindo uma similaridade para a resposta. O modelo é dividido em seis etapas sequenciais conforme visualizado na figura 4.

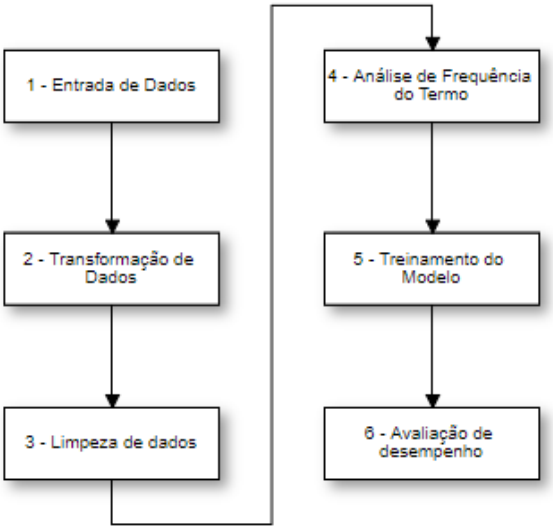

Figura 4. Etapas do modelo proposto

Etapa 1 - Entrada de dados

Consiste em carregar informações da questão e respostas dos alunos do banco de dados para que fiquem disponíveis para o modelo.

Etapa 2 - Transformação de dados

Nesta etapa é realizada a normalização morfológica (lematização) que consiste em transformar as palavras em seus termos primitivos, para isso é utilizada a biblioteca CoGrOO 4.0 conforme mostrado na figura 5.

```
In [1]: from cogroo_interface import Cogroo
        cogroo = Cogroo.Instance()
        texto_lematizado = cogroo.lemmatize('fui, vou, foram, iremos'.lower())
        texto_lematizado

Out[1]: 'ir , ir , ir , ir'
```

Figura 5. Exemplo de lematização utilizando a biblioteca Cogroo

Lematização

A lematização consiste em obter o lema das palavras de um texto. O lema é uma versão da palavra que representa todas as suas flexões, os verbos são alterados para o infinitivo e substantivos e adjetivos são flexionados para o masculino do singular.

Exemplo de utilização da lematização:

- correram, correm, corre, corro, corremos = correr
- fui, vou, foram, iremos = ir
- gato, gatinho, gatão, gata, gatos = gato
- processava, processávamos, processastes, processaremos = processar

A lematização é uma alternativa ao *stemming*, um algoritmo que tenta detectar sufixos e removê-los com base nas regras comuns da língua, mas que falha em tratar exceções. A lematização, diferentemente do *stemming*, usa um dicionário morfológico para encontrar o radical das palavras [Balakrishnan e Lloyd-Yemoh 2014].

Etapa 3 - Limpeza de dados

Para o correto funcionamento das próximas etapas do modelo é necessário realizar uma limpeza de pontuações e acentuações das respostas para evitar que o algoritmo diferencie palavras que na realidade são as mesmas, por exemplo, gramática e gramatica, apesar de igualar, também palavras que são diferenciadas justamente pelo acento, caso das palavras pelo, pélo e pêlo, mas como são casos raros não afetaram o modelo. Um exemplo da limpeza de pontuação realizada no modelo pode ser visto na figura 6 e limpeza de acentos na figura 7.

Nessa fase também são removidas as palavras irrelevantes (*stopwords*) da resposta, como mostrado na figura 8. São consideradas palavras irrelevantes artigos (um, uma, a), preposições (de, em, para, por meio de) e pronomes (ele, seu, sua), estas que não possuem significado e aparecem inúmeras vezes nas respostas.

```
In [1]: import re

        re.sub(u'[^a-zA-ZàèìòùáéíóúÁÉÍÓÚâêîôÂÊÎÔãõÃÕçÇ]',
               ' ',
               'Unidade lógica aritmética, memória, processador e unidade de controle.')

Out[1]: 'Unidade lógica aritmética  memória  processador e unidade de controle '
```

Figura 6. Exemplo de substituição de pontuação por um espaço

```
In [1]: from unicodedata import normalize

        normalize('NFKD', 'Sucessão lógica memória você.').encode('ASCII','ignore')

Out[1]: b'Sucessao logica memoria voce.'
```

Figura 7. Exemplo de limpeza de acentos

```
In [1]: import nltk
        from nltk.corpus import stopwords

        texto = 'Unidade logica e aritmetica, memoria, processador e unidade de controle.'
        palavras = nltk.word_tokenize(texto)
        texto = ' '.join([palavra for palavra in palavras if palavra not in stopwords.words('portuguese')])
        texto

Out[1]: 'Unidade logica aritmetica memoria processador unidade controle'
```

Figura 8. Exemplo de remoção de stopwords utilizando NLTK

Etapa 4 - Análise de frequência do termo

Esta etapa consiste em montar uma matriz com as palavras e pesos relacionados aos documentos para que se possa calcular a similaridade entre as respostas.

TF - Term Frequency

É uma medida estatística que determina quantas vezes a palavra apareceu no documento atribuindo um peso. Essa frequência deve ser normalizada para evitar distorções em documentos longos e determinar uma medida de importância do termo t_i no documento d_j [Salton e Buckley 1988], representado pela equação:

$$tf_{i,j} = \frac{f_{i,j}}{\sum_{K=1}^{K} f_{k,j}}$$

Onde: $f_{i,j}$ = quantidade de ocorrências do termo t_i no documento d_j.

K = quantidade de termos distintos.

A utilização desse algoritmo é importante para que se possa, a partir dos vetores com os pesos de cada palavra, realizar comparações, já que apenas com as palavras não seria possível. Na figura 9 é demonstrado a utilização do TfidfVectorizer da biblioteca scikit-learn para montagem da matriz com as frequências (TF) de cada palavra.

```
In [1]: from sklearn.feature_extraction.text import TfidfVectorizer

        dados = []
        dados.append('cpu memoria dado instrucao programa')
        dados.append('responsavel comunicacao meio externo computador')

        tf = TfidfVectorizer(tokenizer=nltk.word_tokenize, min_df=0, use_idf=False)
        tf_matriz = tf.fit_transform(dados)
        tf_matriz
```

Figura 9. Utilização do algoritmo TF

Similaridade entre Documentos

Para se calcular a similaridade entre dois documentos em um Sistema de Recuperação de

Informação utilizando VSM (*Vector Space Model*), calcula-se o cosseno do ângulo formado no vetor termo-por-documento [Huang 2008]. A métrica de similaridade entre vetores (v1, v2):

$$cos\ (v1, v2) = \frac{v1\ .\ v2}{|v1||v2|}$$

Onde $|v1| = \sqrt{v1.\ v1}$.

O resultado do cálculo é um valor real no intervalo 0 a 1 para cada par origem/destino, em que 0 (zero) significa ausência total de similaridade e 1 (um) totalmente similar. Na figura 10 é apresentado um exemplo da aplicação do sckit-learn para o cálculo de similaridade entre os vetores com as frequências dos termos.

```
In [1]: from sklearn.metrics.pairwise import cosine_similarity

        cosine_similarity(tf_matriz[origem], tf_matriz[destino]).flatten()

Out[1]: 0.54967024
```

Figura 10. Cálculo de similaridade entre dois vetores

No modelo a similaridade é obtida da soma de duas variáveis, $SimC$ que é a similaridade entre o caminho da resposta x resposta alvo e $SimB$ que é a média de similaridade entre a resposta alvo e N boas respostas. A utilização das boas respostas no cálculo é importante para que se possa suavizar a similaridade, já que há respostas com notas altas atribuídas, mas que possuem baixa similaridade com o caminho de resposta.

O vetor de N respostas é gerado a partir de 25% do total de boas respostas da questão, uma resposta é considerada boa quando pertence à classe alta, ou seja, se numa questão existem 37 respostas das quais 18 são consideradas notas altas, então pega-se 5 ou 25% de 18 (arredondando para cima) para ser utilizada no cálculo de similaridade. O percentual de 25% foi definido após inúmeros testes com outros valores e este percentual apresentou melhor custo/desempenho. Os cálculos de similaridade utilizados pelo modelo foram:

$$SimC = \cos(c, r)$$

Onde: $SimC$ = Similaridade caminho resposta x resposta aluno
c = vetor TF caminho resposta
r = vetor TF resposta do aluno

$$SimB = \frac{\sum_{i=1}^{n} \cos(b_i, r)}{n}$$

Onde: $SimB$ = Similaridade boas respostas x resposta aluno
b_i = vetor TF boa resposta na posição i
r = vetor TF resposta do aluno
n = número de boas respostas

$$Similaridade\ média = \frac{SimC + SimB}{2}$$

Onde: $SimC$ = Similaridade caminho resposta x resposta aluno
$SimB$ = Similaridade boas respostas x resposta aluno

Após realizar o cálculo de similaridade é montada uma nova matriz relacionando a nota e a similaridade como apresentado no gráfico 2, onde os pontos com cores frias são referentes a similaridade baixa e as cores quentes para maior similaridade. Essa matriz será utilizada pelos modelos de regressão na próxima etapa.

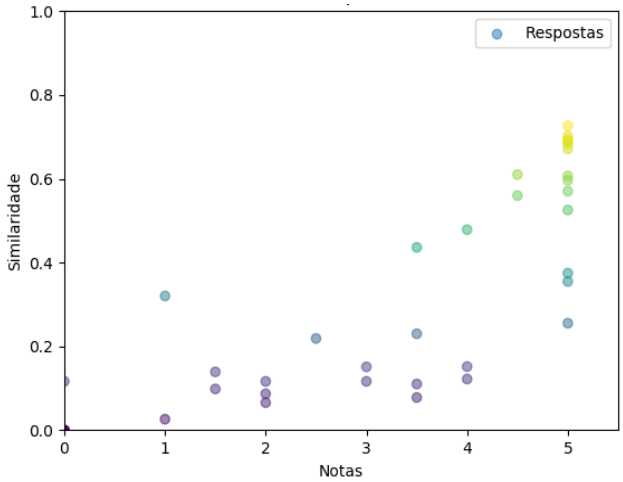

Gráfico 2. Similaridade x Nota

Etapa 5 - Treinamento do Modelo

Nesta etapa é realizado o treinamento/teste dos modelos de regressão SVR (*Support Vector Regression*) e Random Forest utilizando validação cruzada. Foi determinado o valor de 5 para o *k fold* devido ao tamanho limitado de dados. A escolha desses algoritmos se deu por trabalharem bem com poucos exemplos [Oshiro 2013] [Boser et al. 1992].

O processo de distribuição das respostas nos cinco subconjuntos foi realizado de forma a manter um equilíbrio entre as classes, ou seja, a distribuição dos exemplos ao longo dos subconjuntos é proporcional ao número de exemplos de cada classe baixa, média ou alta. Assim, garantindo que todos os subconjuntos apresentem exemplos de todas as classes para ocorrer uma melhor predição.

A figura 11 ilustra o processo de validação cruzada, onde a cada iteração é separado um subconjunto para teste e os demais são utilizados para treino, dessa forma, permitindo uma análise mais profunda da generalização do modelo.

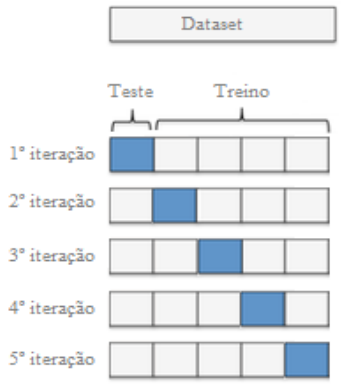

Figura 11. Exemplo de validação cruzada com k fold = 5

Na figura 12 e figura 13 é possível observar a utilização do algoritmo SVR (*Support Vector Regression*) e *Random Forest* da biblioteca sckit-learn para realizar a predição das notas. Ambos algoritmos contam com métodos de mesma nomenclatura para treino (*fit*), teste (*predict*) e métrica de avaliação das previsões (*score*).

O modelo SVR foi configurado utilizando o *kernel* padrão *Radial Basis Function* (RBF), onde o parâmetro *gamma* define o alcance da influência de um único exemplo de treinamento e o parâmetro C troca a classificação errada de exemplos com da superfície de decisão [Scholkopf et al. 1997].

O modelo *Random Forest* foi configurado com 100 árvores no parâmetro *n_estimators*, 4 como o nível de profundidade *(max_depth) e 3* (*min_samples_leaf*) como número mínimo de exemplos necessários para continuar a dividir um nó da árvore. Todas essas configurações foram realizas para diminuir o tamanho da árvore gerada.

```
from sklearn.svm import SVR

# Inicialização do modelo SVR
svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)

# Treino
regressao_rbf = svr_rbf.fit(conjunto_treino.similaridade, conjunto_treino.nota)

# Teste
predicao = regressao_rbf.predict(conjunto_teste.similaridade)

# Coeficiente de determinação
r2_svr = svr_rbf.score(conjunto_teste.similaridade, conjunto_teste.nota)
```

Figura 12. Inicialização e utilização do modelo de regressão SVR

```
from sklearn.ensemble import RandomForestRegressor

# Inicialização do modelo Random Forest
random_forest = RandomForestRegressor(n_estimators=100, max_depth=4,
                            random_state=1, min_samples_leaf=3)

# Treino
random_forest.fit(conjunto_treino.similaridade, conjunto_treino.nota)

# Teste
predicao = random_forest.predict(conjunto_teste.similaridade)

# Coeficiente de determinação
r2_rf = random_forest.score(conjunto_teste.similaridade, conjunto_teste.nota)
```

Figura 13. Inicialização e utilização do modelo de regressão Random Forest

Etapa 6 - Avaliação de desempenho

Etapa responsável pela avaliação da qualidade das previsões do modelo utilizando duas métricas, erro médio absoluto e coeficiente de determinação. O erro médio absoluto é o desvio médio das previsões em relação aos valores efetivos. Essa métrica expõe o quão errado estão as previsões (para mais ou para menos), mas não passa a ideia de direção (*underfitting/overfitting*). É dada pela seguinte equação:

$$EMA = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Onde: y_i = valor previsto
x_i = valor efetivo

n = total de observações

O coeficiente de determinação (R^2) é um valor entre 0 e 1 que indica quanto o modelo foi capaz de explicar os dados, ou seja, quanto a similaridade das respostas (variável independente) pode explicar as notas (variável dependente). É a razão entre a soma de quadrados da regressão e a soma de quadrados total dada pela seguinte equação:

$$R^2 = \frac{SQR}{SQT} = 1 - \frac{\sum_{i=1}^{n}(x_i - \bar{x})Y_i}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

Onde: n = total de observações
x_i = valor previsto
$\bar{x}$ = média das observações
Y_i = valor efetivo
$\bar{Y}$ = média dos valores efetivos

5. Resultados e Discussões

Os resultados foram obtidos a partir de experimentos utilizando uma combinação entre unigramas, bigramas e trigramas, executados em uma máquina com processador Intel Core i7-6700K 4.00 Ghz (4 núcleos), 16GB de memória RAM e sistema operacional Windows 10 (x64).

A tabela 3 apresenta o erro médio absoluto (EMA), coeficiente de determinação (R^2) e configurações dos melhores resultados encontrados pelos modelos de regressão SVR e *Random Forest*. Foi possível identificar uma diferença na média de erro absoluto de 0,45 pontos entre as notas preditas pelo modelo e as dadas pelo professor. Também percebe-se que a análise de contexto (n-gramas), obteve bons resultados em respostas curtas, já os unigramas e bigramas tiveram um desempenho melhor nas respostas longas. Isso se deve ao fato de as respostas curtas, por se tratarem de questões mais diretas e objetivas, apresentarem um agrupamento de palavras em sequência que se repetem em muitas respostas, assim aumentando a similaridade ou diminuindo conforme a nota. Por outro lado, as respostas longas não apresentaram um agrupamento de palavras em sequência recorrente, dado que são questões mais amplas e permitem uma grande variação de respostas, por isso, os unigramas e bigramas apresentaram melhores resultados.

Tabela 3. Melhores resultados experimentos

Questão	EMA SVR	EMA RF	R^2 SVR	R^2 RF	N-gramas	Quant. Boas Respostas
Q1	0,67	0,60	0,74	0,78	Trigramas	5
Q2	0,62	0,33	0,65	0,81	Trigramas	3
Q3	0,45	0,51	0,73	0,67	Unigramas	3
Q4	0,21	0,21	0,80	0,76	Unigramas	8
Q5	0,28	0,25	0,75	0,77	Bigramas	24
Média	**0,45**	**0,38**	**0,73**	**0,76**	-	-

O gráfico 3 demonstra o erro médio absoluto (EMA) por questão e modelo de regressão. Verificou-se que a maior diferença entre as notas preditas x reais foi na questão 1, com SVR atingindo quase 0,70 pontos de diferença seguido pelo *Random Forest* com 0,60 pontos. A diferença se dá pela questão 1 apresentar notas esparsas que variam de 0 a 5, o que aumenta a distância entre as respostas e consequentemente afeta a predição do

modelo, já que a questão conta com apenas 37 exemplos disponíveis para análise.

Outra informação importante que pode ser retirada do gráfico 3, é a diferença de 0,29 pontos entre os modelos de regressão na predição da questão 2, onde as respostas, mesmo as de notas baixas, apresentam similaridade com boas respostas. O *Random Forest* por explorar mais profundamente as características obteve um desempenho melhor, mas houve queda na generalização (*overfitting*), já o SVR manteve certa generalização nas predições.

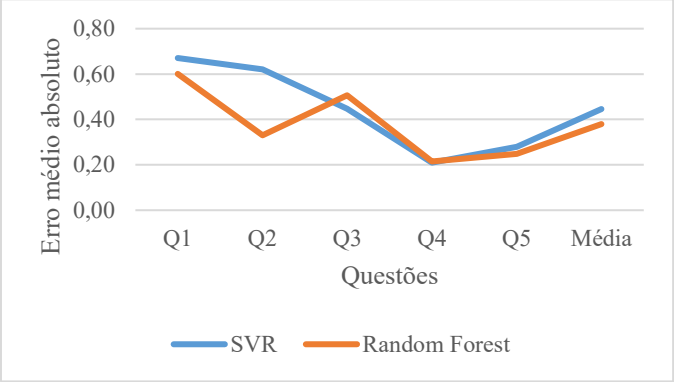

Gráfico 3. Erro médio absoluto (EMA) – SVR x Random Forest

No gráfico 4 percebe-se que o *Random Forest* obteve melhores resultados em respostas curtas comparado ao SVR que se saiu bem contra questões maiores. Isso se deve pelo fato do *Random Forest* necessitar de poucos exemplos para construir as estruturas das árvores e a amostragem *bootstrap* criar vários conjuntos diferentes de treinamento para o modelo, enquanto o SVR trabalha com treinamento fixo. Por fim, constata-se que o SVR obteve uma média de coeficiente de determinação de 73% enquanto o *Random Forest* obteve 76%, ou seja, o *Random Forest* conseguiu explicar um pouco melhor a relação entre similaridade (variável independente) e nota (variável dependente).

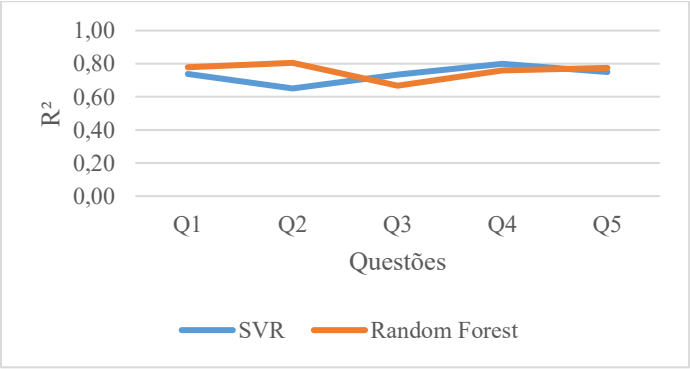

Gráfico 4. Coeficiente de determinação R² - SVR x Random Forest

O gráfico 5 apresenta um comparativo entre os modelos de regressão *Random Forest* e SVR ao analisar a questão 4. Ambos ficam muito próximos não apresentando grandes diferenças.

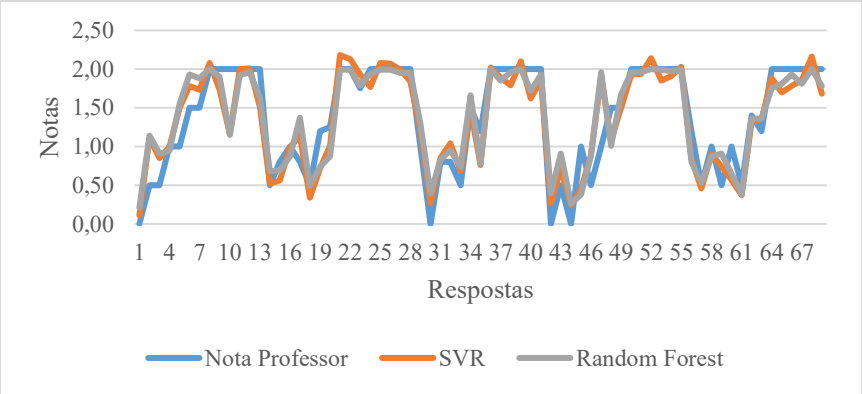

Gráfico 5. Diferença entre modelos SVR e Random Forest ao analisar questão 4.

6. Conclusão

Neste estudo foi apresentado um modelo computacional composto por seis etapas para a predição de notas de cinco questões discursivas de quatro disciplinas diferentes na modalidade EaD e presencial da Universidade do Sul de Santa Catarina. Nos experimentos realizados o modelo que apresentou melhor resultado foi a regressão *Random Forest* com 0,76 de coeficiente de determinação, o que conclui-se que, a similaridade entre as respostas pôde ser capaz de determinar 76% das notas de forma correta. Por outro lado, a regressão SVR ficou levemente atrás do *Random Forest* com coeficiente de determinação igual a 0,73. Portanto, não houve grandes diferença entre os dois modelos de regressão, pois foi utilizada apenas a similaridade como variável independente para explicar a nota.

Outro dado importante do modelo é a média de erro absoluto nas predições de 0,38 pontos com o *Random Forest* e 0,45 pontos com o SVR, que é uma diferença pequena para uma primeira versão do modelo. A análise de contexto também se mostrou importante na avaliação das respostas, porque permitiu identificar agrupamentos de palavras mais relevantes e consequentemente um cálculo de similaridade mais preciso.

O modelo em comparação ao edX e PEG, necessita de poucas respostas para seu funcionamento, visto a questão 3 que possui apenas 24 exemplos e obteve uma diferença em relação as notas dadas pelos professores de 0,45 pontos com SVR e 0,51 com o *Random Forest*. Comparado com o modelo LSA apresentado por Santos (2016) que utilizou respostas curtas com uma média de 25 a 70 palavras, o modelo aqui apresentado, trabalhou com tamanhos diversos, onde o conjunto de treino/teste variou de respostas curtas com 14 palavras (questão 2), a respostas longas com 120 palavras (questão 5).

Para trabalhos futuros pretende-se conseguir um valor maior e mais diversificado de dados para treinamento e teste, pois com uma quantidade de 305 respostas o modelo apresentou dados razoáveis, assim, planeja-se implementar uma etapa para *feedback*, onde o modelo irá atribuir um comentário a uma correção para se assemelhar a um processo natural de correção.

7. Referências

ABED, Associação Brasileira de Educação a Distância; (2017) Relatório Analítico da

Aprendizagem a Distância no Brasil. Disponível em: <http://abed.org.br/censoead 2016/Censo_EAD_2016_portugues.pdf>, p. 80, Acesso em: 22 out. 2017.

Balakrishnan, V.; Lloyd-Yemoh, E.; (2014) Stemming and Lemmatization: A Comparison of Retrieval Performances, Lecture Notes on Software Engineering, v. 2, n. 3. p. 262-267.

Batimarchi, S.; (2014) Estudo mostra que professores brasileiros gastam 894 mil dias só com correções de provas, Disponível em: <http://docmanagement.com.br/11/25/2014 /estudo-mostra-que-professores-brasileiros-gastam-894-mil-dias-so-com-correcoes-de-provas/>. Acesso em: 15 ago. 2017.

Bertsekas, D. P.; (1982) Constrained Optimization and Lagrange Multiplier Methods, Belmont: Athena Scientific.

Boser, B. E.; Guyon I. M.; Vapnik V.; (1992) A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory. Pittsburgh: ed. ACM Press, p. 144–152.

Breiman, L.; (1996) Bagging predictors. Machine Learning, Hingham, v.24, n.2, p.123-140.

Breiman, L.; (2001) Random Forests. Machine Learning, Hingham, v. 45, n. 1, p. 5-32.

Burstein, J., Leacock, C.; Swartz, R.; (2001) Automated evaluation of essays and short answers. In: CAA Conference, 5. Ed. Loughborough University. p. 1-13.

Chung, G. K. W. K.; O'Neil, H. F.; (1997) Methodological Approaches to Online Scoring of Essays. Disponível em: <http://cresst.org/wp-content/uploads/TECH461.pdf>. Acesso em: 15 out 2017.

Dietterich, T. G.; (2000) Ensemble Methods in Machine Learning, Lecture Notes in Computer Science, Berlin, v.1857, p. 1-15.

Dikli, S.; (2006) An overview of automated scoring of essays. The Journal of Technology, Learning and Assessment, v. 5, n. 1, p. 1-36.

Efron, B.; (1979) Bootstrap Methods: Another Look at the Jackknife. The Annals of Statistics, v. 7, n. 1, 1-26.

Feldman, R.; Sanger, J.; (2007) Introduction to Text Mining, In: The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data, New York: Cambridge University Press, p. 1-13.

Ferster, B.; (2014) From The Cloud. In: Teaching Machines: Learning from the Intersection of Education and Technology. Baltimore, Johns Hopkins University Press. p. 152.

Gevert, V. G.; Silva, A. C. L.; Gevert, F.; Ales, V. T.; (2010) Modelos de Regressão Logística, Redes Neurais e Support Vector Machine (SVMs) na Análise de Crédito a Pessoas Jurídicas. Revista Ciências Exatas e Naturais, v. 12, n. 2, p. 269-293.

Günther, H.; Júnior, J. L.; (1990) Perguntas Abertas Versus Perguntas Fechadas: Uma Comparação Empírica, Psicologia: Teoria e Pesquisa, Brasília, v. 6, n. 2, p. 203-213.

Haykin, S.; (2001) Redes Neurais: Máquinas de Vetor de Suporte. In: Redes Neurais:

Princípios e Prática. 2. ed. Porto Alegre: Bookman, p. 349-384

Hofmann, M.; (2006) Support Vector Machines - Kernels and the Kernel Trick: An elaboration for the Haupt seminar “Reading Club: Support Vector Machines”.

Huang, A.; (2008) Similarity Measures for Text Document Clustering, In: New Zealand Computer Science Research Student Conference 6, University of Waikato, p. 49-56.

INEP, Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira. (2017) Notas Sobre o Censo da Educação Superior 2016. Disponível em: <http://download.inep.gov.br/educacao_superior/censo_superior/documentos/2016/notas_sobre_o_censo_da_educacao_superior_2016.pdf>. Acesso em: 22 out. 2017

Libâneo, J. C.; (1994) Didática. 2. ed. Coleção Magistério 2° Grau Série Formando Professor. São Paulo: Cortez.

Markoff, J.; (2013) Essay-Grading Software Offers Professors a Break. Disponível em: http://www.nytimes.com/2013/04/05/science/new-test-for-computers-grading-essays-at-college-level.html?pagewanted=all>. Acesso em: 11 nov. 2017.

Novak, J.D.; Gowin D.B.; (1984) Learning how to learn. New York: Cambridge University Press.

Oshiro, T. M.; (2013) Uma abordagem para a construção de uma única árvore a partir de uma Random Forest para classificação de bases de expressão gênica. Dissertação (Mestrado em Bioinformática) – Universidade de São Paulo, Ribeirão Preto.

PEG. (2017) Project Essay Grade: Automated Essay Scoring. Disponível em: <http://measurementinc.com/products-services/automated-essay-scoring>. Acesso em: 15 out. 2017.

Reilly, E. D.; Stafford, R. E.; Williams, K. M.; Corliss, S. B; (2014) Evaluating the Validity and Applicability of Automated Essay Scoring in Two Massive Open Online Courses. The International Review of Research in Open and Distance Learning, v. 15, n. 5. p. 83-98.

Russell, S.; Norvig, P.; (2013) Máquinas de Vetores de Suporte. In: Inteligência Artificial, 3. ed., Rio de Janeiro: Elsevier. p. 648-652.

Salton, G.; Buckley, C.; (1988) Term-weighting Approaches in Automatic Text Retrieval, Information Processing and Management, v. 24, n. 5, p. 513-523.

Santos, J. C. A.; Avaliação Automática de Questões Discursivas Usando LSA. (2016). 118 f. Tese (Doutorado) – Universidade Federal do Pará, Belém.

Scholkopf, B.; Kah-Kay, S.; Burges, C.J.C.; Girosi, F.; Niyogi, P.; Poggio T.; Vapnik V. (1997) Comparing support vector machines with Gaussian kernels to radial basis function classifiers. IEEE Transactions on Signal Processing. v. 45, n. 11.

Weisberg, S.; (2005) Scatterplots and Regression In: Applied Linear Regression, 3 ed, Hoboken: John Wiley & Sons, Inc., p. 1-17.

Weiss, S. M.; Indurkhya, N.; Zhang, T.; Damerau, F. J.; (2005) Overview of Text Mining In: Text Mining: Predictive Methods for Analyzing Unstructured Information, New York: Springer, p. 1-13.