Mapeamento de Problemas Trigonométricos usando Deep Learning

Neiva Larisane Kuyven[1 2], Vinícius João de Barros Vanzin[1], Carlos André Antunes[1], Alexandra Cemin[1], João Luis Tavares da Silva[1] , Liane Margarida Rockenbach Tarouco[2]

[1] Centro Universitário UNIFTEC - Caxias do Sul - RS - Brasil

[2] Universidade Federal do Rio Grande do Sul (UFRGS) - Porto Alegre - RS - Brasil

{neivakuyven,alexandracemin,joaosilva}@acad.ftec.com.br,
{vini.vanzin.rs,c4rlo4}@gmail.com,liane@penta.ufrgs.br

Abstract. *This paper presents a deep neural solver to automatically map trigonometric problems to equation models as part of a larger project of an Intelligent Tutor System in the area of Trigonometry. This approach directly translates mathematical problems into equation templates using a Recurrent Neural Network (RNN) model, combining a model based on linguistic knowledge to treat the trigonometric context. Experiments were conducted using a Chatbot for interaction with students.*

Resumo. *Este artigo apresenta uma abordagem baseada em Deep Learning para mapear automaticamente problemas trigonométricos em modelos de equações, como parte de um projeto maior de um Sistema Tutor Inteligente na área da Trigonometria. Esta abordagem traduz diretamente problemas matemáticos em modelos de equações usando um modelo de Rede Neural Recorrente (RNR), combinando um modelo baseado em conhecimento linguístico para tratar o contexto trigonométrico. Foram conduzidos experimentos usando Chatbot para a interação com os alunos.*

1. Introdução

Sistemas Tutores Inteligentes (STI) computacionais tratam com modelos de conteúdo instrucional baseado em decisões pedagógicas sobre as interações com o aluno. Normalmente, estas decisões são estruturadas a partir de regras que representam o conhecimento do tutor a respeito do domínio, e as estratégias de ensino como metodologia de interação (Wenger, 1987).

Como instrumento de interação computacional, as interfaces de STIs agregam muitos elementos de conhecimento tradicionais como texto, imagens, animações e hipermídia, entre outros. Os STIs vêm incorporando agentes de diálogo conversacional para melhorar os ganhos de aprendizado (Graesser et al., 2001; Martins et al., 2003; Moraes e Machado, 2016). Os agentes conversacionais (*chatbots* ou *chatterbots*) são sistemas capazes de dialogar com o usuário em linguagem natural. As técnicas de processamento de linguagem natural (PLN) e de inteligência artificial permitem que o *chatbot* classifique as interações com o usuário, extraia informações relevantes para sua compreensão e responda de maneira apropriada ao contexto.

Aliado à recente popularização do uso de *chatbots* e da necessidade em instrumentalizar STIs com diálogos personalizados e adaptativos, este trabalho propõe o desenvolvimento e aplicação de um *chatbot* como tutor para o ensino de Trigonometria,

DOI: 10.5753/cbie.sbie.2018.1513

aliando tecnologias de *Deep Learning*, processamento de linguagem natural, e agentes conversacionais, tendo como público-alvo alunos de cursos superiores de engenharia.

Deep learning são métodos de aprendizagem profunda baseados em múltiplos níveis de representação, obtidos pela composição de módulos simples e não lineares, que transformam representações de baixo nível em representações de níveis mais altos e abstratos (LeCun et al., 2015).

Diferentemente de outros trabalhos baseados em linguagem natural (Shi et al., 2015; Koncel-Kedziorski et al., 2015; Roy e Roth, 2016) ou abordagens de aprendizagem estatística (Kushman et al., 2014) para tradução de enunciados matemáticos, o presente trabalho traduz diretamente problemas de matemática para modelos de equações usando um modelo de Rede Neural Recorrente (RNR). Uma solução híbrida que combina um modelo RNR e um pré-processamento linguístico é proposta, para melhorar a representação dos vetores de entrada através de conhecimento linguístico (português brasileiro) e considerando o domínio da Trigonometria.

O restante deste artigo apresenta uma análise dos principais trabalhos relacionados, na Seção 2. Em seguida, a visão geral sobre a arquitetura do projeto Sistema Tutor Inteligente, que inclui o *chatbot* de resolução trigonométrica, na Seção 3. A Seção 4 apresenta a metodologia de mapeamento dos problemas trigonométricos, preparando o conjunto de entrada e saída para ser processado na RNR de resolução trigonométrica, tratado na Seção 5. Os resultados experimentais são analisados na Seção 6 e as considerações finais na Seção 7.

2. Trabalhos Relacionados

Este trabalho está inserido no contexto da interpretação semântica e mapeamento de linguagem natural para representações formais de problemas matemáticos.

Kushman et al. (2014) desenvolveram uma abordagem baseada em *templates* para resolução de problemas de álgebra na qual o enunciado das questões é alinhado a um sistema de equações. Os parâmetros do modelo estatístico são calculados por aprendizagem supervisionada usando características linguísticas extraídas do enunciado como n-gramas, morfologia e comparações numéricas, equivalência de classes gramaticais, entre outros.

Shi et al. (2015) propuseram um sistema de tradução de problemas de aritmética básica em linguagem natural, convertendo questões em árvores semânticas através de uma gramática livre de contexto que contém regras de produção associadas a funções matemáticas e a classes de uma base de conhecimento. As árvores são pontuadas e analisadas e, a partir da melhor árvore, é derivada a expressão matemática correspondente e sua resposta.

Koncel-Kedziorski et al. (2015) utilizaram técnicas de programação linear inteira para gerar árvores de equação de enunciados de problemas de álgebra básica. O modelo de otimização é treinado de forma supervisionada em características morfológicas, sintáticas e semânticas extraídas dos enunciados, com a adição de restrições em relação à validade sintática, consistência e simplicidade das expressões das árvores geradas.

Roy e Roth (2016) aplicaram uma combinação de analisador semântico e de classificador binário para resolução automática de problemas de aritmética básica. O

solucionador decompõe o enunciado da questão em problemas de decisão e treina classificadores, que predizem as operações aritméticas e a relevância dos termos para a solução. Estas predições são combinadas em uma árvore binária que representa a expressão matemática associada ao exercício em linguagem natural.

Wang, Liu e Shi (2017) propuseram um modelo de RNR para tradução direta de problemas de álgebra em linguagem natural para suas respectivas equações. O sistema é composto por uma RNR de identificação que é constituída por uma camada de células *Long-Short-Term Memory* (LSTM) e indica quais dos números encontrados no enunciado da questão devem fazer parte da equação de resolução, e outra RNR de tradução que usa um modelo *seq2seq*[1] de cinco camadas com células *Gated-Recurrent-Unit* (GRU) para o *encoder* e células LSTM para o *decoder*.

Observa-se que as abordagens para inferência de expressões matemáticas a partir de enunciados de questões em linguagem natural podem ser divididas em métodos de análise semântica e em técnicas de *machine learning*. Nesse sentido, o presente trabalho enquadra-se na segunda categoria, aplicando *deep learning* para tradução de linguagem natural em equações de forma semelhante ao utilizado por Wang, Liu e Shi (2017). Por outro lado, este trabalho diferencia-se dos demais no domínio matemático abordado (trigonometria básica) e no idioma (português brasileiro).

3. Arquitetura Básica do *Chatbot*

A arquitetura básica do *chatbot* para resolução de problemas de trigonometria é composta por: Interface do sistema, Módulo de raciocínio matemático, Módulo de resolução e um Módulo de controle central integrado ao *Watson Assistant* (Gliozzo et al., 2017), conforme ilustra a Figura 1.

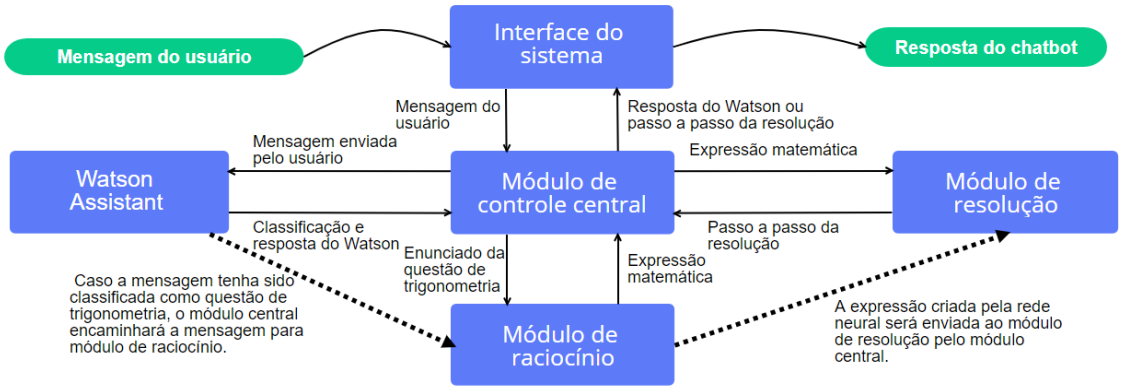

Figura 1. Diagrama do fluxo geral do sistema

A interface do sistema realiza a comunicação entre o *chatbot* e o *webservice*, responsável por trocar mensagens entre usuário e o módulo do controle central. A API do *Watson Assistant* atua como um classificador de texto. O usuário submete uma sentença ao *chatbot* em linguagem natural. O módulo de controle central encaminha para a API do *Watson Assistant*.

[1] Modelo que consiste de duas redes neurais recorrentes (RNR), um "*encoder*" que processa a entrada e um "*decoder*" que gera uma saída correspondente, em sequência.

A mensagem é então classificada em *intents* (intenções do usuário) em uma das quatro categorias trigonométricas (funções trigonométricas básicas, funções trigonométricas inversas, teorema de Pitágoras e semelhança de triângulos) ou em conversação comum, e envia esta classificação ao módulo de raciocínio.

O Módulo de raciocínio avalia o enunciado dos usuários no *chatbot* e infere qual é a expressão matemática correspondente a ser utilizada para a resolução do problema. O controle central recebe a expressão matemática traduzida, usando RNRs, e a envia ao módulo de resolução, que retorna o passo a passo de resolução criado pelo *SymPy*[2] e o envia ao usuário de forma amigável, como em uma conversa de *chat*, juntamente com a representação gráfica do enunciado.

4. Mapeamento dos Problemas Trigonométricos

Para resolver os problemas trigonométricos, a entrada do problema em linguagem natural precisa passar por um processo de transformação para sua expressão matemática associada. Neste trabalho, um *problema Trigonométrico* é modelado como uma sequência W_p de palavras, vetorizado a partir de um pré-processamento linguístico, associando certos termos com um conjunto de *tags* $T_p = \{t_1, \ldots, t_m, x_1, \ldots, x_m\}$ onde $t_1, \ldots, t_m$ é um conjunto de *tags*, que representa uma categoria conceitual predefinida, e $x_1, \ldots, x_m$ são números sequenciais identificando a posição da *tag* em um dicionário de *tags*.

O processo de transformação matemática resulta em uma equação no formato *SymPy*, que é um CAS (*Computer Algebra System)* para manipulação simbólica de expressões matemáticas com resolução (SYMPY, 2018). O exemplo abaixo ilustra uma possível sentença de entrada de um problema trigonométrico (à esquerda) com sua respectiva transformação simbólica para uma equação *SymPy* (à direita).

Uma Pessoa se posiciona a 100m de uma torre de rádio. Com um instrumento de medição, ela determina que o topo da torre faz um ângulo de 75° com o chão. Qual a altura da torre?	*Eq(tan(75) , x / 100)*

4.1. Pré-processamento Linguístico

O *dataset* é subdividido nas quatro categorias trigonométricas supracitadas e devido ao seu tamanho limitado, é feita uma expansão artificial que consiste em trocar algumas palavras por identificadores específicos no dicionário de *tags*.

Uma primeira etapa submete o enunciado do problema a um etiquetador que classifica cada palavra em classes morfossintáticas, usando o *CoreNLP*[3] de Stanford e o pacote *NLTK*[4]. As palavras são substituídas por suas *classes*, mantendo números e substantivos. Por exemplo, a sentença *S1* a seguir:

"Em um triângulo retângulo os catetos medem 7 cm e 24 cm. Determine a medida da hipotenusa."	***S1***

[2] *SymPy* - é uma biblioteca Python para computação simbólica, e fornece ferramentas de álgebra computacional tanto como uma aplicação independente como, também, uma biblioteca para outras aplicações (http://www.sympy.org/pt/index.html)

[3] Conjunto de ferramentas de software para o processamento da linguagem humana: disponível em www.stanfordnlp.github.io/CoreNLP/

[4] Biblioteca de processamento de linguagem natural desenvolvida para a linguagem de programação Python: disponível em www.nltk.org

Ao ser anotada no etiquetador, produz a sentença *S2* pós-processada:

"ADP DET triângulo retângulo DET catetos VERB 7 cm CCONJ 24 cm PUNCT VERB DET medida DET hipotenusa PUNCT" ***S2***

Os números na sentença são substituídos pela *tag* ***<NUM#>*** onde ***#*** é a posição da *tag* no dicionário, assim como todas as formas de se referenciar as unidades pela *tag* ***<UN#>***, criando desta forma, um dicionário de números e unidades. Neste dicionário são mantidos os valores originais da sentença e a *tag* correspondente. Logo, os valores da sentença ***S2*** estão armazenados no índice *1* da lista, como por exemplo:

[{'<NUM1>':7, '<NUM2>':24}], [{'<UN1>':'cm', '<UN2>':'cm'}]

Em seguida um dicionário é criado com todas as palavras únicas neste conjunto de sentenças, onde cada *tag* se torna uma chave única e o seu valor é o número sequencial do dicionário. Também são adicionadas outras *tags* especiais para marcar o início e o fim das frases, *padding* e palavras desconhecidas, como ilustrado na Tabela 1.

Tabela 1. Descrição da lista de *tags* utilizadas nos dicionários.

Tag Especial	***Descrição***
<PAD>	Completa as entradas e saídas quando a sentença é menor que o tamanho do vetor de entrada ou quando a resposta da rede é menor que o tamanho do vetor de saída.
<GO>	Sinaliza para a RNR o início de uma sentença.
<EOS>	Sinaliza para a RNR o fim de uma sentença.
<UNK>	Substitui as palavras que não existem no dicionário de vetorização de entradas criando conjuntos de possibilidades.

Após este pré-processamento, a sentença final resultará no formato da ***S3***:

"<GO> ADP DET triângulo retângulo DET catetos VERB <NUM1> <UN1> CCONJ <NUM2> <UN2> PUNCT VERB DET medida DET hipotenusa PUNCT <EOS>" ***S3***

Com o seguinte dicionário associado

words_to_num = {' <GO>':0, '<EOS>':1, '<PAD>':2, '<UNK>:3, 'ADP':4, 'DET':5, 'triângulo':6, 'retângulo':7, 'catetos':8, 'VERB':9, '<NUM1>':10, '<UN1>':11, 'CCONJ': 12, '<NUM2>':13, <UN2>: 14, 'PUNCT':15, 'medida':16, 'hipotenusa':17}

As saídas do conjunto de treinamento são as equações matemáticas, também vetorizadas, que representam o problema descrito no enunciado, na sintaxe do *SymPy*. As equações matemáticas são escritas na sintaxe *Python*, para que seja possível processá-las com o pacote de computação simbólica *SymPy*. Cada uma das categorias de problemas de trigonometria gera uma expressão matemática alinhada a um *template*, de acordo com a Tabela 2.

Tabela 2. Descrição da lista de *tags* utilizadas nos dicionários.

Categoria	***Template* de saída**
Funções Básicas	Eq(<funçãobásica> (<num#>), [<num#> **OU** x]/[<num#> **OU** x])
Teorema de Pitágoras	Eq([<num#> **OU** x**2], [<num#> **OU** x**2] + [<num#> **OU** x**2])
Funções Inversas	Eq(x,<funçãoinversa>(<num#> /<num#>))
Semelhança de Triângulos	Eq(tan(<num#>),(<numerador> /<denominador>)

Na qual, ***<numerador>*** e ***<denominador>*** podem assumir uma das seguintes expressões:

<numerador>	*<num#> | x | x*tan(<num#>) | tan(<num#>)*x+<num#> | tan(<num#>)*<num#>+x*
<denominador>	*<num#> | x | x+<num#> | (x/tan(<num#>))+<num#> | <num#>/tan(<num#>)+x | (x+<num#>)/tan(<num#>)*

Onde: *<num#>* é o número do dicionário de números na posição #; *x* é a incógnita da equação; *<função básica>* é uma das três funções trigonométricas básicas em sintaxe *Python*: *sin* para seno, *cos* para cosseno e *tan* para tangente; *<função inversa>* é uma das três funções trigonométricas inversas em sintaxe *Python*: *asin* para arcosseno, *acos* para arcocosseno e *atan* para arcotangente; e ***2* é a função de potenciação com expoente 2.

Na primeira categoria, a incógnita pode aparecer somente uma vez na equação, no numerador ou no denominador do lado direito da equação. Na segunda categoria, a incógnita somente pode aparecer duas vezes, caso esteja nas duas posições do lado direito da equação. Na quarta categoria, a incógnita pode aparecer uma ou duas vezes no lado direito da equação e a função *tan* deve aparecer exatamente duas vezes na equação, já que os problemas de semelhança de triângulos tratam sempre de dois triângulos retângulos.

4.2. Construção do *Dataset* Trigonométrico

Após a construção dos dicionários de palavras únicas, da lista de dicionários de números e de unidades, o conjunto de questões é dividido em conjunto de treinamento e conjunto de validação, sendo aproximadamente 30% do conjunto total para validação e 70% do conjunto total para treinamento. O conjunto de validação verifica se a rede está generalizando, tendo em vista que a mesma não conhece as sentenças deste conjunto.

Para que a rede seja treinada nas inúmeras possibilidades de combinações entre substantivos, é gerado um conjunto completo de amostras representando as sentenças, mas mantendo um padrão de enunciado. Neste caso, os substantivos nas sentenças são substituídos pela *tag* da vetorização *<UNK>* (palavra fora do vocabulário) para todas as possibilidades de enunciados. A quantidade de amostras é 2^m, onde *m* é a quantidade de substantivos na frase original. Exemplo de um conjunto possível de enunciados gerados:

1) *ADP DET <UNK> retângulo DET catetos VERB 7 cm DET 24 cm PUNCT VERB DET medida DET hipotenusa PUNCT*
2) *ADP DET triângulo <UNK> DET catetos VERB 7 cm DET 24 cm PUNCT VERB DET medida DET hipotenusa PUNCT*

...

m) *ADP DET <UNK><UNK>DET <UNK>VERB 7 <UNK> DET 24 <UNK> PUNCT VERB DET <UNK> DET <UNK> PUNCT*

Desta forma, é possível gerar um grande número de sentenças, respeitando a estrutura sintática do português brasileiro, populando o *dataset* com grande número de possibilidades sentenciais para cada categoria de função trigonométrica básica.

A respectiva saída esperada para a sentença ***S3*** está representada na sentença ***S4***:

*"<GO> Eq(x **2 , <NUM1> **2 + <NUM2> **2) <EOS>"* ***S4***

Com o seguinte dicionário de saída associado:

```
words_to_num = {'<GO>':0, '<EOS>':1, '<PAD>':2, '<UNK>:3, 'Eq(':4, 'x':5, '**2':6, ',':7,
'<NUM1>':8, '+':9, '<NUM2>':10, ')':11}
```

Cada categoria tem sua própria RNR especialista, onde cada *corpus* é treinado. Diminui-se assim os ruídos e erros por parte da rede, porém demanda a criação de um classificador de sentenças para selecionar à qual rede neural a questão deve ser enviada.

5. Rede Neural LSTM para Resolução Trigonométrica

Neste trabalho, implementou-se uma rede *LSTM* que utiliza um tipo de neurônio conhecido como célula de memória, capaz de memorizar valores em um espaço de tempo arbitrário (KLAUS et al, 2016).

A arquitetura para o mapeamento do problema trigonométrico é do tipo *Encoder-Decoder RNR,* ilustrado na Figura 3, utilizada com sucesso em softwares de tradução de linguagens.

Para modelar a tradução do problema trigonométrico em equações simbólicas, foi utilizado um modelo *seq2seq* consistindo em camadas "*Encoder*" e "*Decoder*" na RNR. O *Encoder* deriva uma sequência de *tokens* contextuais a partir da sentença de entrada vetorizada e atualiza a camada escondida da RNR. Após o processamento da sequência da sentença, a rede produz um estado final escondido que, de certa forma, incorpora o contexto da entrada para gerar o contexto respectivo de saída (*context vector*). No lado esquerdo da Figura 3, o *Encoder* examina **<GO> DET ... <EOF>**, mapeados para os respectivos identificadores discretos.

O *Decoder* obtém a representação do contexto de entrada, incorporado no estado final escondido, a partir do "*context vector*" e gera a respectiva tradução (ou mapeamento trigonométrico). Ou seja, a cada passo t_i, esta camada transforma o estado escondido do *decoder*, em uma distribuição normal sobre uma camada *Dense*. Na direita da Figura 3, o *Decoder* corresponde a sentença de saída **<GO> Eq(... <EOF>**, também mapeados para os respectivos identificadores *{1, 3, ..., 9}*.

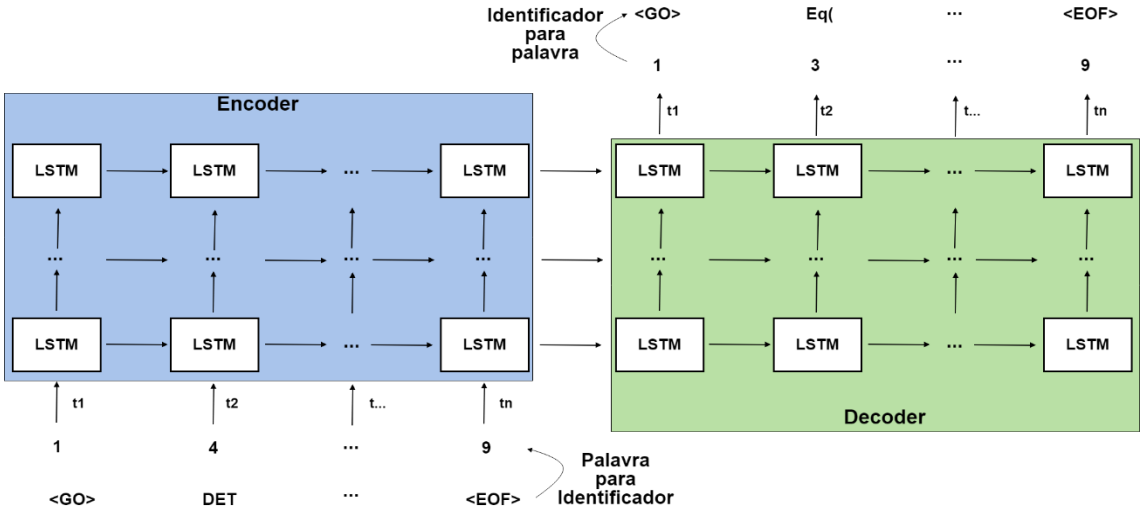

Figura 3. Arquitetura usando um modelo Encoder-Decoder RNR

A RNR utiliza hiperparâmetros para melhorar a performance da rede de acordo com o tamanho do *dataset* e a complexidade dos padrões existentes, por exemplo:

tamanho da amostra de treinamento (*batch*) quantidade de camadas (*layers*), quantidade de células por camada, número de passagens de treinamento (*epochs*), probabilidade de ignorar neurônios (*dropout*) e a taxa de aprendizado da rede (*learning_rate*). A configuração dos hiperparâmetros é discutida na próxima seção.

6. Resultados Experimentais e Discussão

Os experimentos foram realizados com uma turma de 23 alunos da disciplina de Cálculo I assistido por computador. Solicitou-se aos alunos realizar a resolução manual de 20 questões de trigonometria básica no triângulo retângulo e depois as submeter ao *chatbot* para comparar as respostas e o modo de resolução.

Neste experimento apenas a etapa de etiquetagem não foi utilizada. Para o treinamento utilizou-se 79 questões de teorema de pitágoras, 40 questões de funções inversas, 123 questões de funções básicas e 17 questões de semelhança de triângulos. A Tabela 3 apresenta os resultados de acurácia em treinamento e validação.

Tabela 3 - Acurácia dos modelos antes do experimento

Categoria	Treinamento	Validação
Teorema de Pitágoras	89%	86%
Funções Básicas	91%	85%
Funções Inversas	89%	80%
Semelhança de Triângulos	90%	81%

Todos os modelos foram treinados por 1000 épocas, com o vetor de entrada do *encoding* igual a *100* e vetor de entrada de *decoding* de *30*. Para contornar o problema do uso de *corpora* limitados (*overfitting*) ou mais extensos (*underfitting*) definiu-se um balanço de alguns parâmetros conforme a Tabela 4.

Tabela 4 - Hiperparâmetros dos modelos

Hiperparâmetros	Teorema de Pitágoras	Funções Básicas	Funções Inversas	Semelhança de Triângulos
Tamanho do Batch	2	8	2	5
Nº de células por camada	64	128	32	32
Taxa de aprendizado	1	1	1	0.01
Dropout	0.25	0.2	0.4	0.2

Na prática, cada aluno resolveu uma média de *21,65* questões dos *498* exercícios de trigonometria. Das *498* questões enviadas para o *chatbot*, foram selecionadas *240* questões que foram resolvidas manualmente por um especialista. O sistema encontrou classificações válidas para *186* questões, das quais *84* estavam nas categorias trigonométricas corretas. *51* questões foram classificadas em categorias erradas e *3* questões tiveram problemas de infraestrutura. A entrada do enunciado com problemas gramaticais, ortográficos e dados incompletos são as principais origens deste erro de classificação. O restante advém de erros de classificação do *Watson*, erros de tradução das RNRs e erros de cálculo do *SymPy*.

Outras *44* questões apresentaram erros de classificação trigonométrica pelo *Watson* ocasionando envio de sentenças para a RNR errada. A RNR traduziu de forma errônea *58* questões que, embora classificadas corretamente pelo *Watson*, não

produziram equações válidas. O *SymPy* não apresentou erro, tendo sido capaz de resolver todas as equações que foram apresentadas ao módulo de resolução.

Estas questões foram adicionadas ao *corpus* de sua categoria e um retreinamento nas RNRs foi realizado utilizando a metodologia apresentada neste trabalho. Como resultado, conforme a tabela 5, os níveis de acurácia da rede foram maiores em treinamento e validação:

Tabela 5 - Acurácia da rede após o retreinamento com as questões do experimento

Categoria	Treinamento	Validação
Teorema de Pitágoras	100%	94%
Funções Básicas	99%	96%
Funções Inversas	100%	90%
Semelhança de Triângulos	100%	93%

Ao atingir 100% de acurácia a rede "decora" o conjunto de treinamento, não errando nenhuma tradução das sentenças daquele conjunto. Entretanto, a métrica mais importante é a acurácia de validação, já que o conjunto de validação é constituído de sentenças que nunca passaram pela RNR, demonstrando sua capacidade de generalizar. Observa-se um ganho considerável de acurácia de validação do novo modelo em comparação ao modelo anterior, parametrizados na Tabela 4.

Os hiperparâmetros da rede também foram modificados, para atender ao novo tamanho dos *corpora*, conforme a tabela 6.

Tabela 6 - Hiperparâmetros dos novos modelos

Hiperparâmetros	Teorema de Pitágoras	Funções Básicas	Funções Inversas	Semelhança de Triângulos
Tamanho do Batch	256	256	128	128
Nº de células por camada	256	512	256	256
Taxa de aprendizado	0.01	0.01	0.01	0.01
Dropout	0.4	0.2	0.2	0.4

7. Conclusão

Este artigo propôs uma abordagem baseada em *Deep Learning* para mapear automaticamente problemas trigonométricos em modelos de equações, para resolução em sistemas de álgebra computacional. Diferente de outros trabalhos puramente baseados em tradução de linguagem natural ou abordagens de aprendizagem estatística, este modelo traduz diretamente problemas de matemática para modelos de equações usando *RNRs* com células *LSTM* e *seq2seq*. Para o treinamento foi construído um grande *dataset* através de transformações linguísticas aliando uma aplicação ainda inédita na área de trigonometria e língua portuguesa brasileira. Demonstrou-se que este modelo classificou corretamente grande parte dos problemas enunciados por um grupo de alunos, fornecendo respostas para os problemas propostos.

O modelo de mapeamento de problemas trigonométricos usando RNRs proposto aqui pode ser integrado a um STI conversacional com interação via *Chatbots*. Inicialmente, o mapeamento foi definido para a área de Trigonometria e sua aplicação em outro domínio matemático demandaria a redefinição dos *intents* de classificação das

entradas e reestruturação das RNRs no módulo de raciocínio. Com este modelo é possível agregar um modelo especialista a um STI, que acompanha o aluno na resolução passo a passo das equações propostas, auxiliando-o na construção da abstração de um problema enunciado. Trabalhos futuros visam melhorias de performance da RNR e a integração ao STI completo.

Referências

Gliozzo, A. et al. Building Cognitive Applications with IBM Watson Services. IBM Redbooks. 2017. 132 p.

Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W. e Harter, D. (2001). Intelligent tutoring systems with conversational dialogue. AI Magazine, 22(4), 39-52.

Klaus, G. et al. (2016). LSTM: A Search Space Odyssey, IEEE Transactions on Neural Networks and Learning Systems.

Koncel-Kedziorski, R., Hajishirzi, H., Sabharwal, A., Etzioni, O. e Ang, S. D. (2015). Parsing algebraic word problems into equations. Transactions of the Association for Computational Linguistics, 3, 585-597.

Kushman, N., Artzi, Y., Zettlemoyer, L. e Barzilay, R. (2014). Learning to automatically solve algebra word problems. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) Vol. 1, p. 271-281.

LeCun, Y., Bengio, Y. e Hinton, G. E. (2015). Deep Learning. Nature, Vol. 521, p. 436-444.

Martins, F. J., Ferrari, D.N. e Geyer, C.F.R. (2003). jXChat - Um Sistema de Comunicação Eletrônica Inteligente para apoio a Educação a Distância. In SBIE - Brazilian Symposium on Computers in Education, p.445-454.

Moraes, S. e Machado, R. (2016). Chatterbot for Education: a Study based on Formal Concept Analysis for Instructional Material Recommendation. In SBIE - Brazilian Symposium on Computers in Education, p.1347-1351.

Roy, S. e Roth, D. (2016). Illinois Math Solver: Math Reasoning on the Web. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, p. 52-56.

Shi, S., Wang, Y., Lin, C. Y., Liu, X. e Rui, Y. (2015). Automatically solving number word problems by semantic parsing and reasoning. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, p. 1132-1142.

Sympy. Sympy Development Team. (2018). SymPy Documentation. Disponível em: <http://docs.sympy.org/latest/index.html>. Acesso em: 15 de junho de 2018.

Wang, Y., Liu, X. e Shi, S. (2017). Deep Neural Solver for Math Word Problems. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing , p. 845-854.

Wenger, E. (1987). Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge. San Francisco: Morgan Kaufmann.