Revista Brasileira de Informática na Educação – RBIE
Brazilian Journal of Computers in Education
(ISSN online: 2317-6121; print: 1414-5685)
http://br-ie.org/pub/index.php/rbie

Submission: 14/02/2019; 1st round notif.: 20/05/2019; New version: 12/07/2019; 2nd round notif.: 01/10/2019
Camera ready: 10/10/2019; Edition review: 05/01/2020; Available online: 16/02/2020; Published: 16/02/2020

Assistente conversacional para resolução de Problemas Trigonométricos em Linguagem Natural

Title: Conversational assistant Trigonometric Solving Problems in Natural Language

Neiva Larisane Kuyven
Centro Universitário Uniftec
neivakuyven@acad.ftec.com.br

Vinícius João de Barros Vanzin
Centro Universitário Uniftec
vini.vanzin.rs@gmail.com

Carlos André Antunes
Centro Universitário Uniftec
c4rlo4@gmail.com

Alexandra Cemin
Centro Universitário Uniftec
alexandracemin@acad.ftec.com.br

João Luis Tavares da Silva
Centro Universitário Uniftec
joaosilva@acad.ftec.com.br

Liane Margarida Rockenbach Tarouco
UFRGS
liane@penta.ufrgs.br

Resumo

Sistemas Tutores Inteligentes (STI) constituem-se, atualmente, em uma área interdisciplinar que investiga como representar modelos de conteúdo instrucional baseado em decisões pedagógicas que consideram as interações dos estudantes. Normalmente, estas decisões são estruturadas a partir da modelagem do estudante, regras que representam o conhecimento do domínio, de um especialista e as estratégias de ensino. Entretanto, a maioria dos STI preconiza a inferência de nível de necessidade do estudante através do feedback de exercícios e atividades planejados pelo STI, limitando assim a possibilidade interativa do estudante em relação a interface conversacional. Chatbots são então agregados aos STI fornecendo uma abordagem mais "assistente", usando técnicas de processamento de linguagem natural e de inteligência artificial, em geral, para gerenciar estas interações. Este artigo apresenta uma abordagem que proporciona maior grau de liberdade ao estudante, quando possibilita a entrada livre, em linguagem natural, de problemas do domínio não necessariamente pertencente a base de atividades do STI. Neste trabalho, uma abordagem baseada em aprendizagem profunda (Deep Learning) é usada para mapear automaticamente estes problemas trigonométricos, fornecidos como entrada livre pelo usuário, em modelos de equações, como parte de um projeto maior de um STI na área da Trigonometria (STIT). Esta abordagem traduz diretamente problemas matemáticos em modelos de equações usando um modelo de Rede Neural Recorrente (RNR), combinando um tratamento do conhecimento linguístico para modelar o contexto trigonométrico. Os experimentos conduzidos demonstraram que o modelo proposto classificou corretamente grande parte dos problemas enunciados por um grupo de alunos, fornecendo respostas para os problemas propostos em um formato de resolução passo a passo de um CAS (Computer Algebra System)

Palavras-Chave: *Chatbot; Deep Learning; Sistemas Tutores Inteligentes; Trigonometria; Ensino e Aprendizagem.*

Abstract

Intelligent Tutoring Systems (ITS) are an interdisciplinary area aiming to investigate how to model instructional content based on pedagogical decisions and students' interactions. Those decisions use student modeling, based-rules domain knowledge and teaching strategies. Most ITS advocate that the feedback of exercises and activities planned by the system define the knowledge students' level. Therefore, limiting the student's interactive possibility through the conversational interface. A more "wizard" approach integrates Chatbots into the ITS, using artificial intelligence techniques to manage interactions. This article intends to provide a greater degree of freedom for the

Cite as: Kuyven, N.L., Vanzin, J.B.V., Antunes, C.A., Cemin, A., Silva, J.L.T. & Tarouco, L.M.R. (2020). Conversational assistant Trigonometric Solving Problems in Natural Language (Agente Coversacional para Resolução de Problemas Trigonométricos em Linguagem Natural). Brazilian Journal of Computers in Education (Revista Brasileira de Informática na Educação - RBIE), 28, 208-228. DOI: 10.5753/RBIE.2020.28.0.208

student through natural language inputs of problems out of the activity set of the ITS. This work presents a Recurrent Neural Network (RNR) model, which translates trigonometric problems, entered by students, into equation models as part of a larger ITS Trigonometry project. The experiments conducted showed that the proposed model correctly classified a large part of the problems posed by students, providing answers to the proposed problems in a step-by-step resolution format of a Computer Algebra System (CAS).

Keywords: *Chatbot; Deep Learning; Sistemas Tutores Inteligentes; Trigonometria; Ensino e Aprendizagem.*

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 STI agregam muitos elementos de conhecimento tradicionais como textos, imagens, animações e hipermídia, entre outros. Além disso, os STI vêm incorporando agentes de diálogo conversacional para melhorar os ganhos de aprendizado (Graesser et al., 2001; Martins et al., 2003; Moraes e Machado, 2016). Paralelamente, os agentes conversacionais (*chatbots* ou *chatterbots*)[1] 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 à necessidade em instrumentalizar STI com diálogos personalizados e adaptativos, este trabalho propõe o desenvolvimento e aplicação de um *chatbot* como tutor para o ensino de Trigonometria, aliando tecnologias de Aprendizagem Profunda, Processamento de Linguagem Natural, e modelos de Agentes Conversacionais, tendo como público-alvo alunos de cursos superiores de engenharia.

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. 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, o ensino da Trigonometria e os aspectos psicopedagógicos na Seção 3. Em seguida, a visão geral sobre a arquitetura do projeto Sistema Tutor Inteligente, que inclui o *chatbot* de resolução trigonométrica, na Seção 4. A Seção 5 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 6. Os resultados experimentais são analisados na Seção 7 e as considerações finais na Seção 8.

2 Trabalhos Relacionados

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

[1] Ao longo do restante do artigo o termo usado será Chatbot, por questões de simplicidade.

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. O espaço de equações possíveis é definido por um conjunto de modelos de sistema de equações, inferido a partir de exemplos de treinamento, onde cada modelo é constituído por *templates* possuindo um conjunto de *slots*. Os *Slots* são de dois tipos: numéricos, que são preenchidos por números do texto; e, *slots "unknow"* que são preenchidos pelos substantivos encontrados no enunciado. O conjunto de exemplos foi obtido a partir de estratégias de supervisão usando vários anotadores humanos, relacionando a resposta aos enunciados aos sistemas de equações.

Esta abordagem gera um enorme espaço de possíveis sistemas de equações e para encontrar o melhor alinhamento, um modelo de distribuição log-linear sobre estes sistemas completos de equações e o texto é utilizado. Desambiguações do sistema de tradução são tratadas adicionando-se características (*features*) adicionais ao modelo log-linear, como *unigramas* e *bigramas*, se o *slot* é objeto de uma questão ou pertence a uma questão, se um lema está próximo de uma constante, se pares de *slots* contém o mesmo lema, comparações numéricas, equivalência de classes gramaticais, entre outros. Embora a proposta dos autores tenha melhorado a acurácia em relação ao sistema ARIS (usado como avaliação), alguns problemas ainda restam insolúveis pois dependem da correta interpretação semântica inerente à linguagem natural.

Diferente de Kushman et al. (2014), nossa abordagem não considera diretamente características semânticas do problema, pois não lida com apenas um modelo matemático tal como o sistema de equações. Ao elencarmos quatro tipos básicos de problemas trigonométricos, nos afastamos de um *template* simplificado qual o de problemas algébricos e não é possível considerar tantas características linguísticas no modelo probabilístico. Portanto, aproveitamos o modelo de alinhamento de *template* e *slots* para mapearmos, também em nossa proposta, números próximos de objetos conceituais tradicionais da trigonometria, como "catetos" e "hipotenusas", por exemplo, mas inferido a partir de um modelo de tradução neural artificial.

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.

Esta abordagem é aplicada também sobre problemas algébricos associando sua tradução a sistemas de equações. Os autores criaram uma linguagem intermediária para construir árvores semânticas dos enunciados matemáticos, a ideia não é gerar árvores de equações, mas sim uma linguagem de representação semântica, chamada DOL (*Dolphin*) para representar a estrutura semântica do enunciado. O componente principal é um analisador semântico que transforma o texto do enunciado em árvores DOL. Neste caso, para projetar uma espécie de parser semântico, as regras léxico-gramaticais forma geradas de maneira semi-supervisionada. A associação entre os conceitos relacionados à matemática e suas regras gramaticais foi manualmente construído. A gramática possui 9.600 regras e durante a análise, uma pontuação é calculada para cada nó DOL, para que a derivação das árvores DOL com a maior pontuação seja selecionada e o resultado, resolvido por meio de um módulo adicional de raciocínio.

A abordagem em Shi et al. (2015) reconhece apenas um subconjunto de problemas algébricos relacionados a identificação de números nos enunciados. Além disso, a tradução precisa gerar uma árvore válida na linguagem DOL, restringindo a problemas com equações pré-definidas. O *dataset* também foi gerado manualmente pela seleção do subconjunto específico de identificação de números em enunciados algébricos. As limitações para nosso uso residem na gramática em Inglês e na possível explosão combinatória de regras gramaticais para a língua Portuguesa e, também, no domínio da Trigonometria.

Koncel-Kedziorski et al. (2015) utilizaram técnicas de programação linear inteira (ILP) 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. Diferente dos trabalhos anteriores, o sistema ALGES proposto por Koncel-Kedziorski et al. (2015) gera árvores de equações para cada enunciado algébrico. A partir do texto, são gerados *QSets* (*Quantified Set*), o qual é uma *tupla* contendo o trecho de texto correspondendo a quantidades com suas propriedades relacionadas (por exemplo, adjetivos, locações, verbos, funções sintáticas). O mapeamento é feito através da identificação da correspondência entre trechos de textos e operadores matemáticos, examinando tais textos e os *Qsets* dos operandos envolvidos. A informação sintática para a geração de propriedades do *QSet* é obtida com o *Stanford Dependency Parser* e o modelo ILP usado é o *CPLEX 12.6.1* da IBM (IBM ILOG, 2013). A aprendizagem fracamente supervisionada usa o modelo ILP para gerar árvores de equações candidatas para instâncias de *QSets* avaliando um conjunto de características derivadas de relações semânticas e intertextuais entre *QSets*, além de traços lexicais e sintáticos, como por exemplo, objetos como argumentos de verbos, distância de termos matemáticos aos verbos, relações entre entidades, adjetivos, distâncias e similaridades entre múltiplos verbos, entre outros.

Embora o modelo proposto em Koncel-Kedziorski et al. (2015) fosse mais abrangente em termos linguísticos do que aquele em Kushman et al. (2014), erros de *parsing* geram muitos conjuntos *QSets* errôneos, além das limitações semânticas de ordenamento e construção do *parser*. O sistema ALGES é aplicado para equações algébricas simples, envolvendo poucas variáveis. Embora seja um pouco mais robusto do que os previamente apresentados, ainda precisa ser estendido para enunciados algébricos com múltiplas equações. Em nossa abordagem, precisamos considerar o mapeamento para múltiplas equações trigonométricas sem a possibilidade de aumentar a complexidade da análise através do uso de *parsers* de dependência para o Português.

Roy e Roth (2016) aplicaram uma combinação de analisador semântico baseado em gramática livre de contexto 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 et al. (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*[2] de cinco camadas com células *Gated-Recurrent-Unit* (GRU) para o *encoder* e células LSTM para o *decoder*.

Nesta abordagem a solução também encontra-se em linguagem natural, portanto os autores extraem sistemas de equações e respostas das soluções através de um método de *parsing* baseado em regras, para obterem um *dataset* rotulado de equações e respostas bem estruturadas. Quando um enunciado de entrada é analisado, o sistema rotula quantidades mapeando-as em *templates* de equações lineares. Para melhorar a acurácia do modelo, os autores propõem um abordagem híbrida em que uma RNR auxiliar identifica um número significativo nos enunciados, ou seja, validas os números que são realmente usados nas equações de solução.

[2] 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.

O modelo *seq2seq* proposto em Wang et al. (2017) é aplicado apenas em equação lineares contendo uma única variável desconhecida. Uma camada de vetorização de palavras é aplicada à RNR após um tratamento da localização de números significativos, reduzindo assim o conjunto de possibilidades. Embora a proposta supere outros trabalhos em termos de acurácia, continua a aplicação em subconjuntos muito simples de equações lineares e sentenças em Inglês ou Chinês. Em nossa abordagem, procuramos examinar o máximo de sentenças trigonométricas encontradas em Português, e aproveitamos a ideia da vetorização de palavras ou *word embedding* para representação de entrada e saída, construindo um modelo *seq2seq* tradicional para monitorar a quantidade de discrepâncias em vocabulários não controlados.

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 Aprendizagem de Máquina. Analisando os cinco trabalhos relacionados, podemos observar, na maioria, a simplificação do contexto do problema para equações algébricas de uma ou duas variáveis ou a necessidade de representações intermediárias. Por exemplo, no trabalho de Shi et al. (2015), é utilizada uma representação semântica intermediária propondo uma gramática extensa de regras e, portanto, os problemas são restritos a equações algébricas pré-definidas. Em Koncel-Kedziorski et al. (2015) a abordagem gera outra estrutura intermediária, uma árvore de equações, portanto necessitando de *parsers* específicos. Isto levou a limitação das entidades semânticas identificadas e restringe o uso para equações mais complexas como as da Trigonometria. O modelo *seq2seq* proposto em Wang et al. (2017) é aplicado apenas em equação lineares contendo uma única variável desconhecida. E, finalmente, todos os modelos analisados são preparados para línguas diferentes do Português.

A presente proposta enquadra-se nos trabalhos que aplicam Aprendizagem Profunda para tradução de linguagem natural em equações, justamente tentando evitar uma possível explosão combinatória de regras gramaticais para o Português no domínio da Trigonometria. Entretanto, aplicamos um modelo *seq2seq* semelhante ao de Wang et al. (2017), sem a inferência de números significativos pois a área da Trigonometria gera soluções muito diferentes. Também formalizamos o tratamento dos textos através da vetorização de termos, construindo um modelo *seq2seq* mais tradicional, com *encoder* e *decoder* baseados em LSTM. A ideia de estruturação do *dataset* é importante e incorporamos um modelo similar de *template* e slots, semelhantes aos usados em Kushman et al. (2014), para mapeamento de números e objetos trigonométricos.

3 O Ensino da Trigonometria e os Aspectos Psicopedagógicos

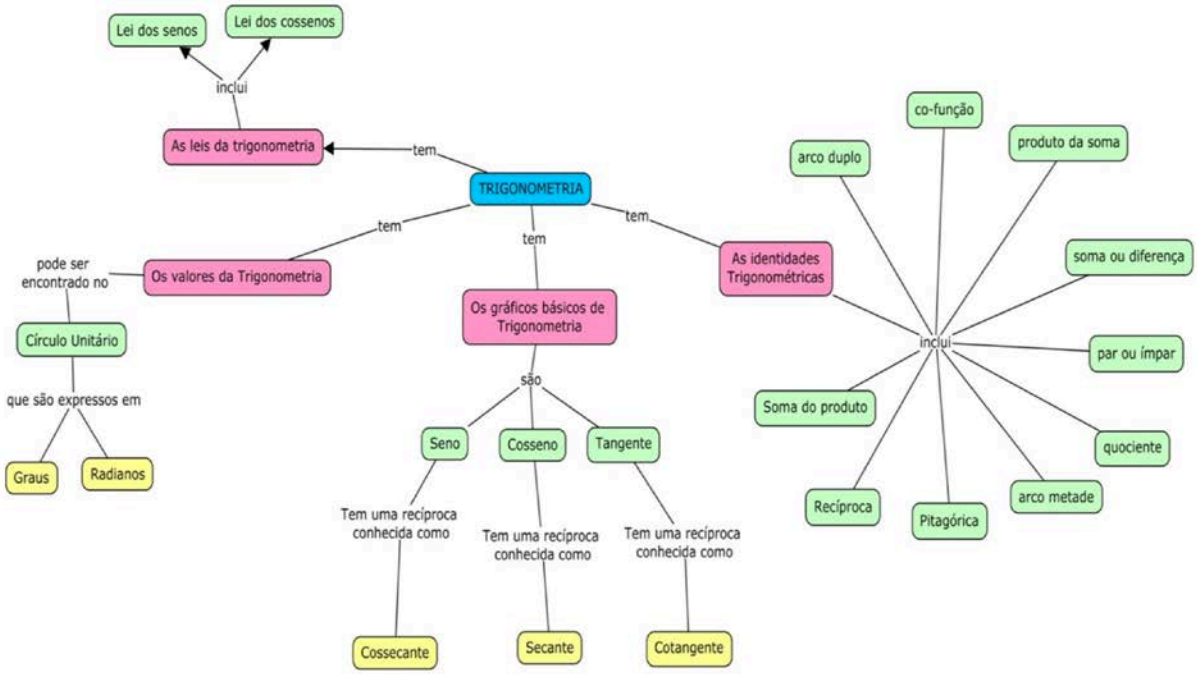
A escassez das pesquisas sobre as dificuldades enfrentadas ao se aprender Trigonometria não só no Brasil, mas no mundo (Weber, 2005; Demir e Heck, 2013; Ishartono et al., 2016; Kamber e Takaci, 2018; Feijó, 2018) contribuiu para investir-se na proposta de mapear automaticamente problemas trigonométricos em modelos de equações, para resolução destes, usando sistemas de álgebra computacional. Além da motivação inicial para realização do estudo, a qual baseava-se nas dificuldades vivenciadas pelos professores e alunos dos cursos de engenharia do centro universitário envolvido na pesquisa. Sendo que a Trigonometria é um dos temas centrais para a compreensão de "[...] tópicos em física, arquitetura, [...] e muitos ramos da engenharia" (Weber, 2005, p.91), tornando-se fundamental investir na formação dos alunos e, caso existam, os obstáculos enfrentados por eles no aprendizado, auxiliando-os na compreensão deste tema.

Constata-se que o estudo da Trigonometria possui relevância na atualidade, influenciando significativamente a Astronomia, a Trigonometria plana, como uma ferramenta extremamente poderosa na mensuração de distâncias, ou nas funções trigonométricas, com modelagens de situações cotidianas e uma infinidade de situações problemas que envolvem este conceito. Além disso, como a Trigonometria é um dos primeiros tópicos de Matemática que relaciona o raciocínio

algébrico, geométrico e gráfico, ela pode servir como um precursor importante para a compreensão do pré-cálculo e do cálculo. (Weber, 2005, p.91)

As dificuldades enfrentadas pelos alunos na compreensão da Trigonometria podem ser influenciadas pela dificuldade do docente em compreender a amplitude deste conceito, visto que envolve uma série de raciocínios. Chigonga (2016, p.169) apresenta em sua pesquisa a fala de um dos professores do ensino básico em uma escola no sul da África: “Parece que muitos professores têm problemas para ensinar aos alunos como resolver equações trigonométricas ... e, como resultado, os estudantes tem fobia das equações trigonométricas e não é um tópico favorito para eles ...”

Chigonga (2016, p.174) apresenta um mapa conceitual, conforme a Figura 1, expondo a abrangência da Trigonometria.



Fonte: Feijó (2018)
Figura 1: Mapa conceitual de Trigonometria.

Através do mapa conceitual com as diversas áreas da Trigonometria percebe-se a importância do investimento em explorar formas de aprendizagem que efetivem o conhecimento, visto que existe uma série de fragmentos independentes. Tall e Vinner (1981), definem imagem conceitual como:

> *“(...) a completa estrutura cognitiva que é associada com o conceito, que inclui todas as imagens mentais e propriedades e processos associados. Ela é construída ao longo dos anos, por meio de experiências de todos os tipos, mudando à medida que o indivíduo encontra novos estímulos e maturidade”.*

No presente trabalho defende-se a proposta de que a expansão do conhecimento na Trigonometria pode ser atingida através da conexão clara entre os conceitos, utilizando-se técnicas de inteligência artificial e outros recursos que favoreçam a compreensão dos problemas trigonométricos através do apoio de um sistema capaz de colaborar com o estudante na compreensão dos conceitos inerentes.

Este apoio ao indivíduo aprendente é importante pois, conforme Vygotsky (2007), o aprendizado e desenvolvimento estão inter-relacionados desde o início da vida. Através dos

processos de interações o indivíduo menos experiente passa a contar com o conhecimento daquele com mais experiências. Para estabelecer as relações reais entre o processo de desenvolvimento e a capacidade de aprendizado de um indivíduo, é necessário compreender dois níveis de desenvolvimento, que Vygotsky denominou: nível de desenvolvimento real e nível de desenvolvimento potencial. Segundo Vygotsky (2007), o nível de desenvolvimento real se refere ao desenvolvimento já consolidado pelo indivíduo. O nível de desenvolvimento potencial é aquele que o sujeito poderá construir, através da ajuda de pessoas mais experientes (professor ou pares). A partir destes dois níveis de desenvolvimento, pode ser melhor compreendida a zona de desenvolvimento proximal, definida por Vygotsky, como "a distância entre o nível de desenvolvimento real, determinado através da solução independente de problemas, e o nível de desenvolvimento potencial, determinado através da solução de problemas sob a orientação de um adulto ou em colaboração com companheiros mais capazes".

O *chatbot* tutor assume o papel do professor ou tutor na interação com o aluno, criando, desta forma, um canal de diálogo (nem sempre possível de ser estabelecido com todos os alunos em sala de aula pelo número excessivo de atendimentos e de alunos nas turmas). O chatbot assume o papel do mediador da aprendizagem, promovendo a aprendizagem pela interação com o aluno. Neste sentido, o chatbot possui condições de interferir na Zona de Desenvolvimento Proximal ampliando-a.

No contexto em estudo, a utilização de *chatbot* tutor, aliado a tecnologias de Aprendizagem Profunda e Processamento de Linguagem Natural, podem auxiliar no desenvolvimento de habilidades do domínio da Trigonometria e desta forma, propiciar ao aluno um atendimento personalizado e adaptado conforme seu nível de conhecimento e dificuldades individuais. O uso destes recursos permite atender e acompanhar o aluno a qualquer hora e local, oferecer feedback personalizado, buscando, desta forma, auxiliar na abstração e resolução de problemas no domínio da trigonometria.

O modelo proposto traduz diretamente problemas de matemática para modelos de equações, sendo categorizado por tipos específicos de problemas a serem resolvidos, resultando na definição das categorias trigonométricas. A partir desta definição, implementou-se uma RNR para cada categoria com a intenção de melhoramento nas especificidades de reconhecimento e tradução do enunciado de forma mais assertiva.

4 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 2.

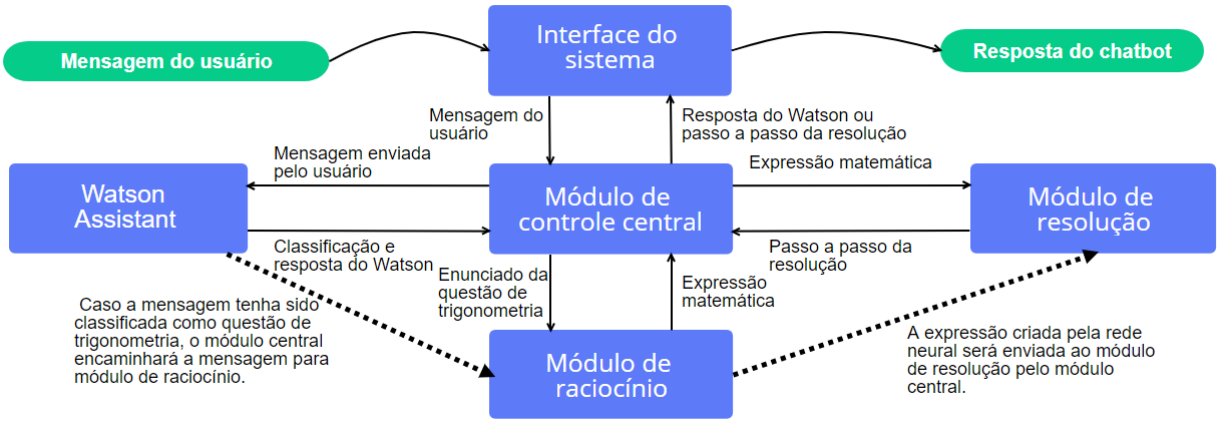


Figura 2. Diagrama do fluxo geral do sistema.

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, no qual o usuário submete uma sentença ao *chatbot* em linguagem natural. Em seguida, o módulo de controle central encaminha a sentença para a API do *Watson Assistant*.

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.

Um dos primeiros objetivos do Assistente é compreender a intenção do usuário. Neste caso o *Watson Assistant* reconhece expressões de simples conversação, como ilustrado na Figura 3(A), questões conceituais sobre o domínio da Trigonometria, Figura 3(B) ou um problema trigonométrico enunciado, Figura 3(C). Deste modo, o assistente inserido no STI é capaz de manter o foco do aluno no domínio, auxiliando-o com questões conceituais e também direcionando as questões práticas para o módulo de raciocínio. Assim, mantém-se a premissa de deixar transparente ao aluno a interação com o assistente virtual, capaz de responder a uma ampla gama de assuntos teóricos e práticos.

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 métodos de Aprendizagem Profunda, e a envia ao módulo de resolução, que retorna o passo a passo de solução criado pelo *SymPy*[3] e o envia ao usuário de forma amigável, como em uma conversa de *chat*, juntamente com a representação gráfica do enunciado, conforme ilustrado na Figura 4.

Aprendizagem profunda tem sua origem nos tradicionais *Perceptrons* Multicamadas[4] representa um conjunto de métodos de aproximação baseados em múltiplos níveis de representação, obtidos pela composição de módulos simples e não lineares, que transformam

[3] *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)

[4] Também conhecidos como *Feedforward Neural Networks*, são representados por grafos acíclicos dirigidos em que a resolução de funções de aproximação dependentes são calculadas em camadas de iteração, geralmente retroalimentando a próxima camada com as saídas da camada anterior. [Goodfellow et al., 2016]

representações de baixo nível em representações mais abstratas (LeCun et al., 2015). Normalmente, este aprendizado em profundidade é retroalimentado através das camadas, na qual a saída de uma camada é a entrada da próxima, proporcionando um elevado nível de extração de características, que torna essa classe de algoritmos capaz de resolver problemas considerados extremamente complexos. Exemplos clássicos da abordagem de resolução neural em camadas são as Redes Convolucionais (*convnets*) e as Redes Neurais Recorrentes (RNR), entre outras. Devido a essa capacidade de lidar com entradas complexas, as *convnets* podem ser utilizadas para resolver os mais variados tipos de problemas, desde a identificação de objetos através de imagens, até a análise de sons distintos ou interpretação de textos contidos em documentos (Karpathy, 2017).

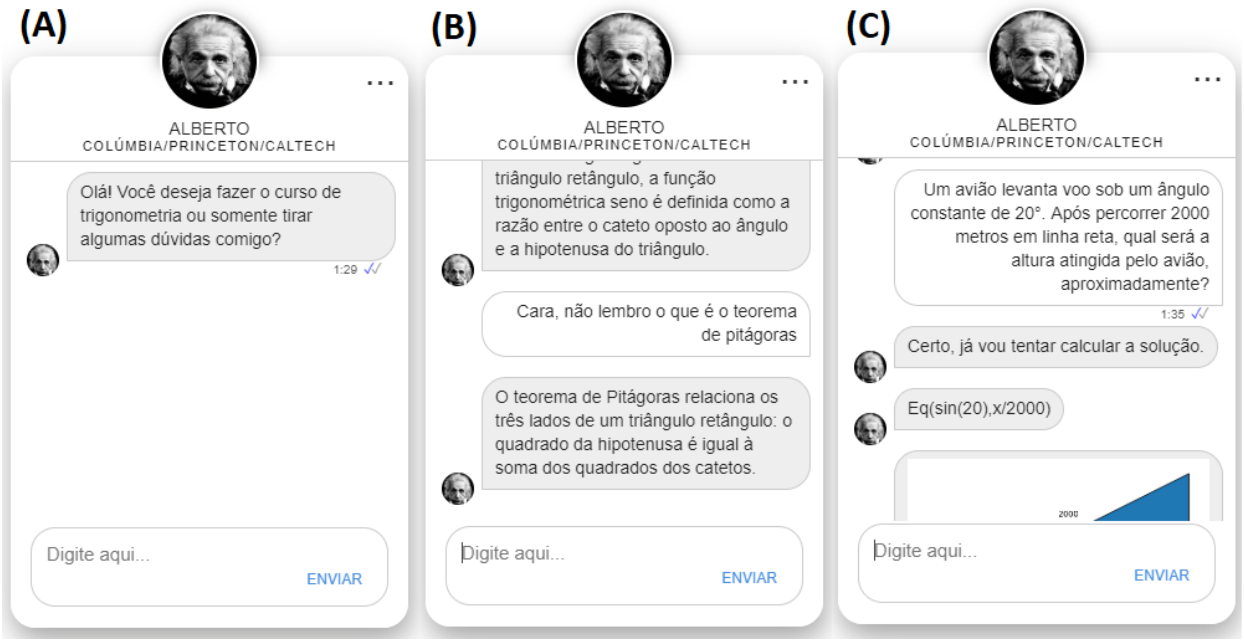


Figura 3. Exemplos de interação com o assistente. (A) interação inicial com o aluno; (B) explicações usando domínio conceitual; (C) interação do aluno com o assistente no domínio prático.

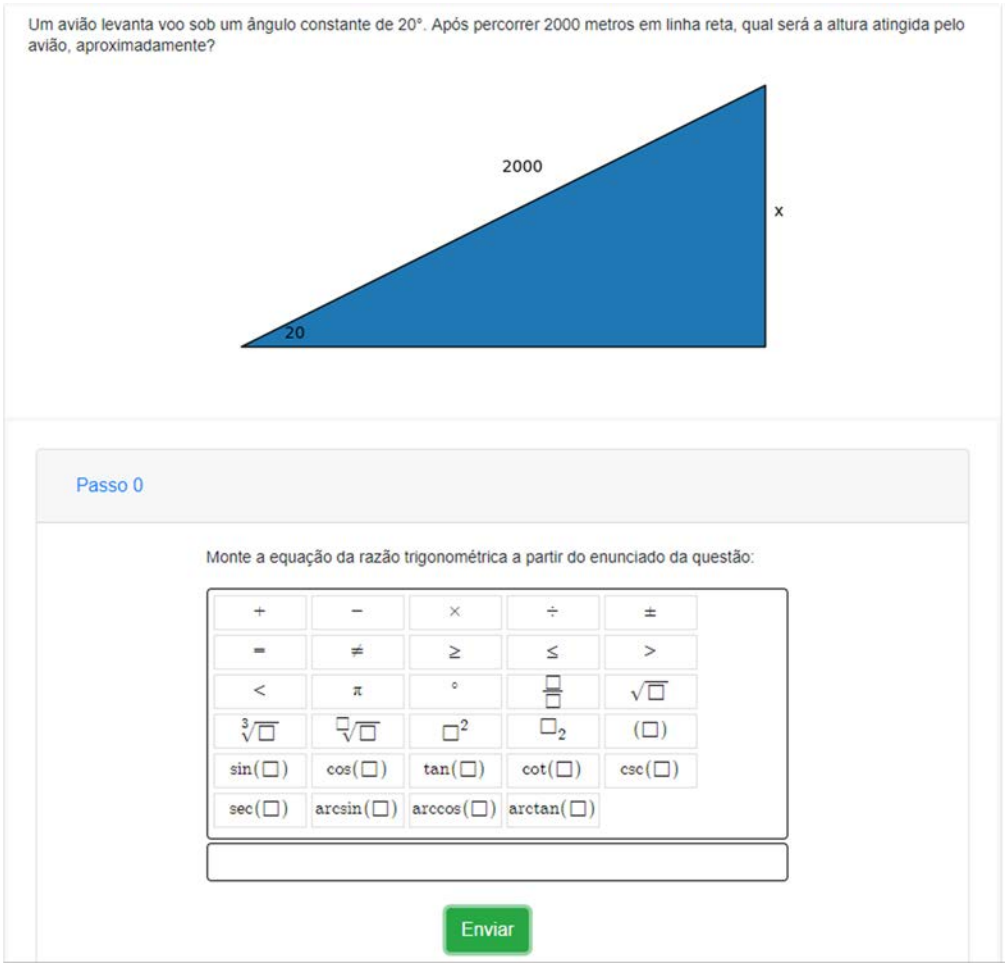


Figura 4. Exemplo de ambiente de interação com o aluno logo após o Assistente haver analisado o enunciado do usuário e traduzido o problema.

Na Figura 4, o *chatbot* assistente fez a tradução do enunciado para a equação trigonométrica adequada e inicia um fluxo de interações com o aluno em que a resolução é subdividida em passos. Nesta etapa, a expressão matemática gerada é passada para o módulo de resolução que gera um passo a passo através de equações. O método de resolução segue uma abordagem tradicional entre as ferramentas CAS (*Computer Algebra System*), primeiramente são tentadas regras de simplificação e resolução até que a expressão não se modifique mais. Para cada passo, todas as regras implementadas são testadas e a primeira que causar alguma modificação na expressão é adicionada à lista de regras. Tenta-se aplicar as regras até que nenhuma delas produza alteração ou se o número de passos de resolução ultrapassar 20 regras.

A ordem de aplicação das regras é de grande importância, já que simulam a resolução humana. Possivelmente, se a ordem fosse diferente, o resultado seria o mesmo, porém o passo a passo não "pareceria" natural ao usuário. Em termos gerais, a sequência adotada inicia pela simplificação da expressão para sua posterior resolução, passando por operações aritméticas, operações com frações, simplificação de polinômios, funções trigonométricas e manipulação de variáveis para resolução de equações. A biblioteca SymPy utilizada envolve não apenas as funções trigonométricas, mas também operações de aritmética, frações e álgebra básica. No SymPy, as expressões são representadas como árvores, nas quais os nodos são as operações, constantes e variáveis. Deste modo, cada uma das regras percorre a árvore e soluciona sua respectiva operação, possibilitando o controle da resolução através de uma sequência de passos.

Portanto, na Figura 4, o aluno já recebe uma atividade de resolução com controle do domínio e controle temporal (sequência) do tutor. Na Figura 4, a metade inferior ilustra como o Tutor

devolve o controle ao aluno para sua resposta ao primeiro passo de resolução usando um editor de equações. A abordagem epistemológica neste processo baseia-se na contingência instrucional e de domínio proposto em Wood (2003) e o tutor é capaz de interagir com o aluno avaliando cada passo de resolução do aluno contra os passos produzidos automaticamente pelo SymPy. Eventualmente, o tutor é capaz de interagir com o aluno demonstrando os passos, durante a sequência de resolução conforme ilustrado na Figura 5.

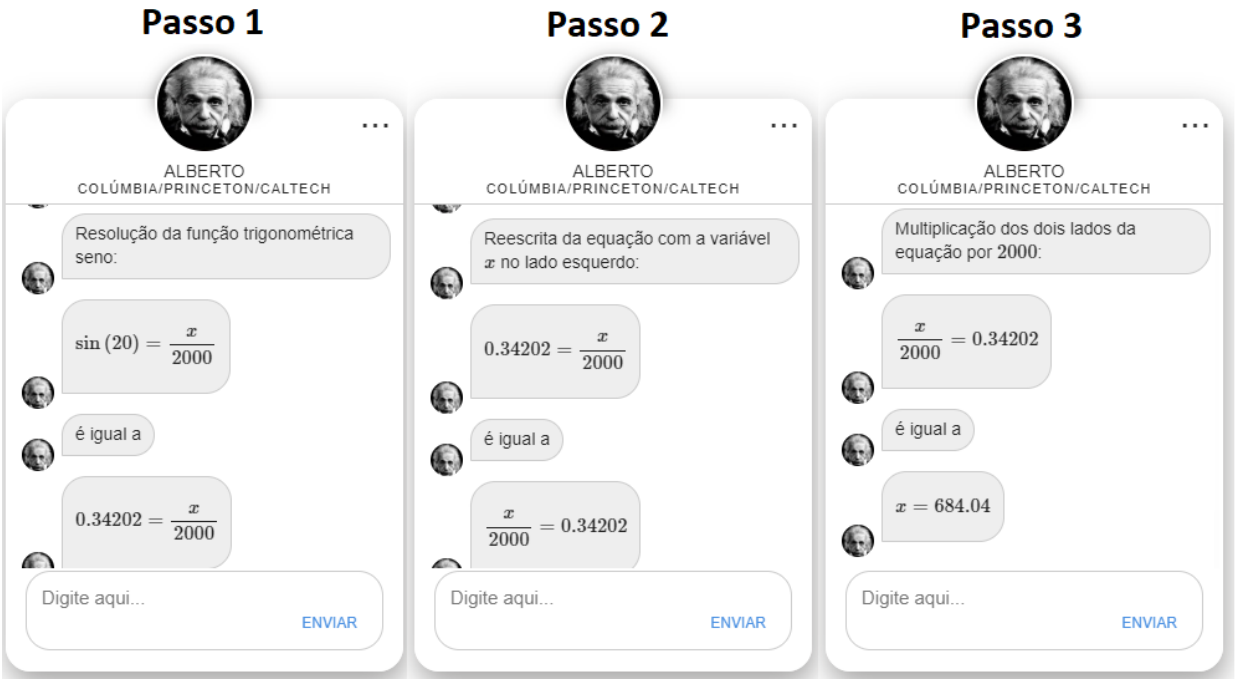


Figura 5. Exemplo de resolução passo a passo.

Na Figura 5, a sequência de resolução executada pelo Tutor é exibida ao aluno na forma de diálogo, o mais próximo possível de uma interação em linguagem natural. O fato do *chatbot* ilustrar o problema gerando uma representação gráfica do problema (triângulo-retângulo na Figura 4) e guiando a resolução através de interações pergunta-resposta com o aluno através dos vários passos de resolução algébrica, auxilia o aluno com a necessária abstração do problema enunciado em relação aos conceitos trigonométricos estudados no domínio.

5 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* $T_p = \{t_1, \ldots, t_m, x_1, \ldots, x_m\}$ em que $t_1, \ldots, t_m$ 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)*

5.1 Pré-processamento Linguístico

O *dataset* é subdividido nas quatro categorias trigonométricas definidas nesta proposta: funções trigonométricas básicas, funções trigonométricas inversas, teorema de Pitágoras e semelhança de triângulos, descritas na Tabela 2.

Nos trabalhos relacionados, o domínio do problema é o dos *Math Word Problems*, que levam em consideração a formulação verbal, relações matemáticas subjacentes e, certamente, a expressão simbólica de representação, identificando características e propriedades próprias ao problema. É assim que um problema algébrico relaciona apenas as quantidades, principais e auxiliares, a entidades concretas na frase como substantivos e objetos, associando *Question words* aos substantivos próximos destas quantidades e derivando *templates* de equações com *1* ou *2* variáveis, no máximo. Problemas mais comuns como cálculo de distâncias, idades, porcentagens, todas relacionadas com quantidades que podem ser associadas a sistemas de equações, possuem características linguísticas mais simples de serem aprendidas por uma RNA.

Assim, traduzir “Some 5 com 4” ou “qual o resultado da multiplicação de 5 por 4” é menos complexo do que deduzir um problema trigonométrico que envolve uma abstração espacial de figuras geométricas. Em problemas trigonométricos, a abstração da semântica está relacionada a configurações gráficas no espaço Euclidiano, no caso de círculo trigonométrico, por exemplo, enquanto associações de ângulos especiais, abstraem razões trigonométricas em triângulos, novamente um objeto geométrico.

Neste trabalho, lidamos com uma hipótese inicial de que manter o máximo de palavras do domínio trigonométrico na frase, fornece pistas para classificar o problema em uma das quatro categorias que definimos na seção anterior. Assim, uma versão preliminar de nossa solução usa quatro RNR distintas em função do tipo de saída esperado.

Nosso *Dataset* foi gerado manualmente e devido ao seu tamanho limitado, é necessário usar uma abordagem de expansão artificial. Um dos problemas mais frequentemente mencionados no campo de aprendizado de máquina, principalmente quando se trata da utilização de Aprendizagem Profunda ou RNR, é a quantidade insuficiente de dados de treinamento ou o equilíbrio de classes desiguais dentro dos conjuntos de dados. Em visão computacional, estes problemas são resolvidos usando técnicas de aumento de dados (*Data Augmentation*) para aumentar a robustez e melhorar o aprendizado de objetos com um número limitado de exemplos de treinamento.

Nas aplicações usando imagens, as técnicas de aumento utilizadas são, por exemplo, inversão do horizonte, corte aleatório, inclinação da imagem, alternância dos canais RGB das imagens originais, entre outros (Fadaee et al., 2017). Entretanto, se o aumento de dados é uma técnica padrão para treinar redes profundas para processamento de imagens, não é tão comum em treinamento com dados textuais, como Tradução Automática. Para aumentar o *dataset* em processamento de textos, existem técnicas de uso de sinonímia ou anotação do texto. No presente trabalho, optou-se por anotar as classes gramaticais dos termos linguísticos e substituir algumas palavras por identificadores específicos listados em dicionários de *tags*. Para evitar a etiquetagem manual, o pré-processamento linguístico proposto aqui consiste em uma classificação supervisionada, usando um pacote de etiquetagem, e posterior codificação dos marcadores linguísticos próprios para o domínio da trigonometria.

Uma primeira etapa submete o enunciado do problema a um etiquetador que classifica cada palavra em classes morfossintáticas, usando o *CoreNLP*[5] de *Stanford* e o pacote *NLTK*[6]. 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)***

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#>*** em que ***#*** 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

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

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

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(<funcaobasica> (<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,<funcaoinversa>(<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; *<funcaobasica>* é uma das três funções trigonométricas básicas em sintaxe *Python*: *sin* para seno, *cos* para cosseno e *tan* para tangente; *<funcaoinversa>* 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.

5.2 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, em que *m* indica 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, na qual 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.

6 Rede Neural LSTM para Resolução Trigonométrica

Neste trabalho, implementou-se uma rede neural recorrente com células *LSTM,* que utiliza um tipo de neurônio conhecido como célula de memória, capaz de memorizar valores em um espaço de tempo arbitrário. (GREFF et al., 2016). A arquitetura da rede proposta neste artigo é baseada em abordagens, atualmente, tradicionais de aplicações de Máquinas Neurais de Tradução (*Neural Machine Translation ou NMT*) que representam de forma mais adequada uma sequência contínua de informações interdependentes. Basicamente, uma NMT usa um modelo *seq2seq* que consiste de duas RNN, um *Encoder*, que codifica a sentença de entrada (de uma língua fonte) gerando uma sequência de representações distribuídas, e um *Decoder* que gera a tradução desta sequência em uma língua alvo. Diferente de abordagens puramente probabilísticas, uma NMT é capaz de generalizar melhor os trechos de texto ainda não identificados, explorando similaridades de palavras em vetores de palavras (*word embeddings*). Estes vetores distribuídos, não esparsos e fixos podem representar características de palavras e determinar a proximidade de outros vetores com características semelhantes, por exemplo, palavras em diferentes línguas com a mesma função sintática.

Entretanto, usando uma RNR simples ainda é difícil manter a conexão entre termos muito distantes em sentenças longas, desta forma, uma estrutura reorganizada com camadas e células LSTM conseguem capturar uma reordenação de longa distância, condicionando de forma contínua os contextos maiores.

Para modelar a tradução do nosso problema trigonométrico em equações simbólicas, foi utilizado um modelo *seq2seq* consistindo em camadas *Encoder* e *Decoder* na RNR, utilizando células LSTM em ambas. 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 incorpora o contexto da entrada para gerar o contexto respectivo de saída (*context vector*). No lado esquerdo da Figura 6, 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 6, o *Decoder* corresponde a sentença de saída <**GO**> **Eq(...** <**EOF**>, também mapeados para os respectivos identificadores *{1, 3, ..., 9}*.

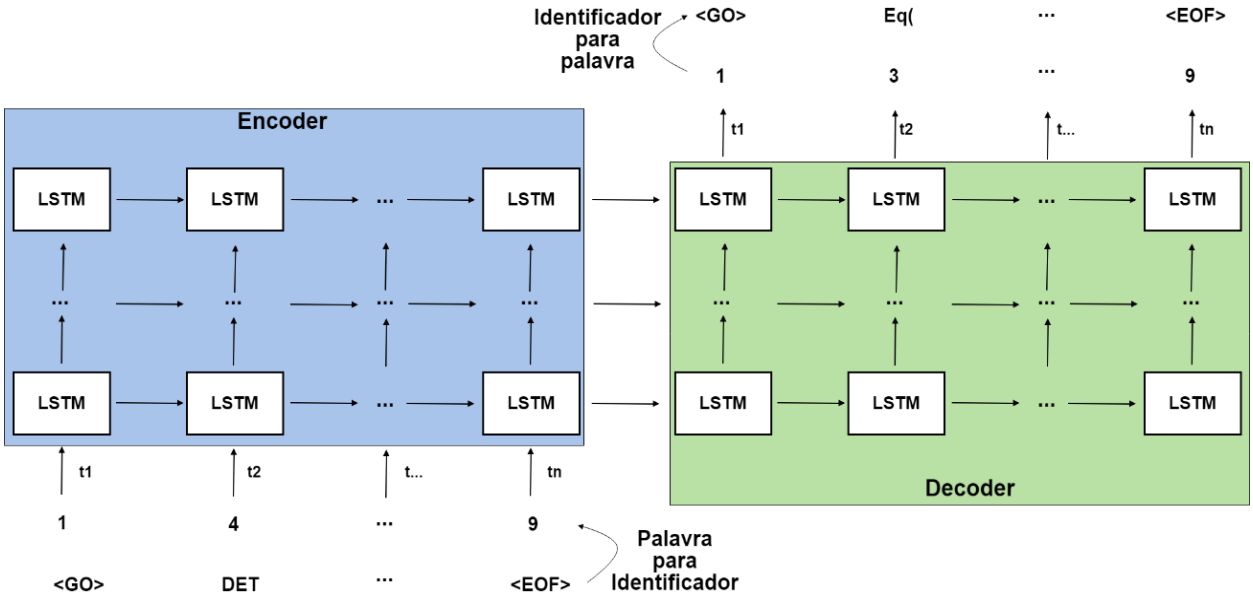


Figura 6. Arquitetura usando um modelo *Encoder-Decoder* RNR.

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*).

Pelo fato de nosso dataset ser muito limitado, o processo de seleção e validação para os hiperparâmetros de cada RNR foi manual. À medida que testes intermediários e incrementais eram realizados e a perda (*loss*) diminuía junto com a adequação da acurácia (como exemplificado na Tabela 3), os valores iam sendo ajustados para definir um modelo mais adequado para cada categoria. Para a proposta do *dataset* aumentado, não foi possível aplicar um método de validação cruzada, por exemplo, em tempo hábil. Sem garantias preliminares de que os hiperparâmetros do melhor modelo pudessem ser selecionados para este *dataset* aumentado, decidimos por fazer validação cruzada nos trabalhos futuros.

Portanto, para as etapas de treinamento e validação a 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.

O treinamento foi propositalmente feito de forma que as redes realmente ficassem super adaptadas, já que a aplicação é para um domínio bastante fechado e restrito em situações problema

na área de Trigonometria (teorema de Pitágoras, funções básicas, funções inversas e semelhança de triângulos) e não seria necessário uma generalização complexa por parte da rede. Além disso, usamos um *Dataset* bastante limitado em termos de exemplos de problemas Trigonométricos em Português e, em sua grande maioria, as entradas mantinham muitas semelhanças estruturais.

Outro fato que faz a rede ficar super adaptada é que nosso aumento artificial de *dataset* e nosso pré-processamento já generaliza os dados, trocando os substantivos por sinônimos, transformado sentenças de ordem direta para sentenças de ordem indireta e vice-versa, além de utilizar classes morfossintáticas como características básicas de treinamento.

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. Este balanço nos hiperparâmetros foram alcançados através de vários testes manuais com o objetivo de gerar o modelo mais adaptado ao tipo de *Dataset* construído.

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

7 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.

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*. Estas novas questões não passaram pelo treinamento e validação das redes, e foram considerados conjuntos de testes,

embora não tenham sido avaliadas em termos de acurácia para determinação da eficiência do modelo, estas medições são objeto de trabalhos futuros.

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 "aprende" 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

8 Conclusão

Este artigo propôs uma abordagem baseada em Aprendizagem Profunda para mapear automaticamente problemas trigonométricos em modelos de equações, para resolução usando sistemas de álgebra computacional (CAS). 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 isto foi necessária a categorização de tipos específicos de problemas a serem resolvidos, resultando na definição de quatro categorias trigonométricas (funções trigonométricas básicas, funções trigonométricas inversas, teorema de Pitágoras e semelhança de triângulos). A partir desta definição, foi implementada uma RNR para cada categoria a fim de melhor

especializar o reconhecimento e tradução do enunciado de forma mais assertiva. Para o treinamento foi necessário construir um grande *dataset* através de transformações linguísticas aliando uma aplicação ainda inédita na área de Trigonometria e língua portuguesa brasileira, já que não foram encontrados *datasets* específicos para este domínio. Conforme foi demonstrado, 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 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. Na definição do *dataset*, é necessário melhorar o *Word Embedding* através da agregação de novas características relacionadas ao domínio de aplicação. Os vetores de palavras de entrada atuais contemplam apenas um tratamento morfossintático através de um *Tagger*, relacionando informações de quantidades com substantivos próprios do domínio da Trigonometria e suas posições relativas. É preciso utilizar um conjunto de modelos do tipo *Word2Vec*, para identificar relacionamentos semânticos do domínio da Trigonometria. O grande problema é encontrar modelos para o PT-BR, e portanto, a ideia é pesquisar a possibilidade de uso de arquiteturas do tipo *Continuous Bag-of-Words* (CBOW) ou *Continuous Skip-Gram* (Mikolov et al., 2013) para encontrar as vizinhanças mais prováveis de um conjunto de palavras.

A integração com o STI ora descrita ainda não contempla como novos problemas introduzidos pelos alunos devam ser considerados no módulo de domínio e pedagógico. É necessário definir um modelo de avaliação em que a representação do domínio do especialista, implementado no STI, deva levar em consideração novos problemas externos que não participam da construção global do currículo do curso. A arquitetura dos modelos pedagógico e do estudante a serem considerados em estudos futuros precisam ser flexíveis o suficiente para agregar esta "atividade avaliativa" em seu fluxo pedagógico.

Referências

Chigonga, B. (2016). Learners' errors when solving trigonometric equations and suggested interventions from grade 12 mathematics teachers. *Proceedings of ISTE International Conference on Mathematics, Science and Technology Education*. Limpopo, South Africa. pp. 163-176. [GS Search]

Demir, O., Heck, A. (2013). A new learning trajectory for trigonometric functions. *In E. Faggiano, & A. Montone (Eds.), Proceedings of the 11th International Conference on Technology in Mathematics Teaching*, pp. 119-124. [GS Search]

Fadaee, M, Bisazza, A. & Monz, M. (2017). Data Augmentation for Low-Resource Neural Machine Translation. *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vancouver, Canada, pp. 567-573. doi: 10.18653/v1/P17-2090 [GS Search]

Feijó, R.S.A.A. (2018). Dificuldades e obstáculos no aprendizado de Trigonometria: um estudo com alunos do ensino médio do Distrito Federal. Dissertação de Mestrado Profissional em Matemática em Rede Nacional (PROFMAT) da Universidade de Brasília (UNB), Brasília. 108p.

Gliozzo, A.,Ackerson, C., Bhattacharya, R., Goering, A., Jumba, A., Kim, S.Y., Krishnamurthy, L., Lam, T., Littera, A., McIntosh, I., Murthy, S. & Ribas, M.. (2017). Building Cognitive Applications with IBM Watson Services. IBM Redbooks. 132 p. Retrieved From: http://www.redbooks.ibm.com/redbooks/pdfs/sg248387.pdf

Graesser, A. C., , VanLehn, K., Rosé, C. P., Jordan, P. W. & Harter, D. (2001). Intelligent tutoring systems with conversational dialogue. AI Magazine, 22(4), pp. 39-52. doi: 10.1609/aimag.v22i4.1591 [GS Search]

Goodfellow, I., Bengio Y. & Courville, A. Deep Learning. Adaptive Computation and Machine Learning series. MIT Press. 2016. ISBN-13: 978-0262035613 [GS Search]

Greff, K., Srivastava, R.K., Koutník, J. Steunebrink, B. R. & Schmidhuber, J. (2016). LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10). doi: 10.1109/TNNLS.2016.2582924 [GS Search]

IBM ILOG. 2013. IBM ILOG CPLEX Optimization Studio 12.6. IBM Europe, Middle East, and Africa Software Announcement ZP13-0553, dated October 1, 2013.

Ishartono, N.; Juniati, D.; Lukito, A. (2016). Developing Mathematics Teaching Devices in the Topic of Trigonometry Based on Guided Discovery Teaching Method, Journal of Research and Advances in Mathematics Education, 2016,1(2), pp. 154-171.

Kamber, D; Takaci, D. (2018). On problematic aspects in learning trigonometry, International Journal of Mathematical Education in Science and Technology, 49(2), pp. 161-175. doi: 10.1080/0020739X.2017.1357846.

Karpathy, A. (2017). Convolutional neural networks. Retrieved from: http://cs231n.github.io/convolutional-networks. Access: Dez/2018.

Koncel-Kedziorski, R., Hajishirzi, H., Sabharwal, A., Etzioni, O. & Ang, S. D. (2015). Parsing algebraic word problems into equations. Transactions of the Association for Computational Linguistics, 3, pp. 585-597. ISSN: 2307-387X. [GS Search]

Kushman, N., Artzi, Y., Zettlemoyer, L. & 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, pp. 271-281. ISBN: 978-1-937284-72-5. [GS Search]

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep Learning. *Nature*, Vol. 521, pp. 436-444. doi: 10.1038/nature14539. doi: https://doi.org/10.1038/nature14539. [GS Search]

Martins, F. J., Ferrari, D.N. & 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*, pp.445-454. doi: 10.5753/cbie.sbie.2003.445-454 [GS Search]

Moraes, S. & 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*, pp.1347-1351. doi: 10.5753/cbie.sbie.2016.1347 [GS Search]

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Roy, S. & 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, pp. 52-56. doi: https://doi.org/10.18653/v1/N16-3011 [GS Search]

Shi, S., Wang, Y., Lin, C., Liu, X. & 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*, pp. 1132-1142. doi: 10.18653/v1/D15-1135 [GS Search]

SymPy Development Team. (2018). SymPy Documentation. Retrieved from: http://docs.sympy.org/latest/index.html . Access: Jun/2018.

Tall, D., Vinner, S. (1981). Concept image and concept definition in mathematics with particular reference to limits and continuity. Educational Studies in Mathematics, 12(2), pp.151-169. [GS Search]

Vygotsky, L. S. (2007). *A formação social da mente: o desenvolvimento dos processos psicológicos superiores.* 7 edição. São Paulo: Martins Fontes.

Wang, Y., Liu, X. & Shi, S. (2017). Deep Neural Solver for Math Word Problems. *In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp.845-854. doi: 10.18653/v1/D17-1088 [GS Search]

Weber, K. (2005). Students' Understanding of Trigonometric Functions. *Mathematics Education Research Journal*, 17(3), pp.91-112. doi: 10.1007/BF03217423. [GS Search]

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

Wood, D. (2003). The why? what? when? and how? of tutoring: The development of helping and tutoring skills in children. *Literacy Teaching and Learning: An International Journal of Early Reading and Writing*, 7(1-2), pp.1-30. [GS Search]