Machine Learning for Math: Vision and Intermediate Milestones

Abdou Youssef

George Washington University

National Institute of Standards and Technology

January 10, 2022

Vision

- To provide an “automated mathematician” or “mathematician in a box” to
  1. empower mathematicians to do math faster and more reliably
  2. enable users (scientists/engineers/practitioners) to apply math as a service on demand
  3. help learners learn math faster and more easily
- How do we get there (using the latest breakthroughs)?
- What are some intermediate milestones/systems/apps that can build and are extremely useful to scientists and practitioners?

Roadmap
-- Approaches --

- Classical AI
  - Based mostly on Logic, Reasoning, and Search
- Classical Computational Linguistics (and Math Linguistics)
  - Grammar (Structure) Based, with some Semantics
- Numeric and Symbolic Processing
  - As currently done in Computer Algebra Systems (CAS)

- Machine Learning / Deep Learning (the latest breakthrough/enabler)
  - The new possibilities

Machine Learning for Math
-- What Is Needed --

- Datasets
  - Large, annotated
  - A few large general-purpose datasets, and many smaller special-purpose datasets
- Benchmarks (datasets + performance metrics + baselines)
- Pretrained models (like BERT and GPT)
  - Trained on very large, general-purpose datasets (using large computational resources)
  - Encapsulate general knowledge of math linguistics, math categories, etc.
  - Fine-tunable (with post-training on smaller specialized datasets) for all kinds of specific tasks
- A few key tasks (with associated datasets) that serve as
  - Building blocks for larger applications
  - Vehicles for motivating and tracking progress, for galvanizing the research community
  - Vehicles for testing the powers and limitations of new ideas and techniques

Key Tasks for Math Linguistics
-- Many are Counterparts of NLP Tasks --

Equation-level Tasks (Tokenization and Parsing)	Document-level Tasks	Application-level Tasks
1. Math Tokenization and String Segmentation 2. Part-of-Math (POM) Tagging and Named Entity Recognition (NER) 3. Math-term Disambiguation 4. Constituency Parsing of Equations 5. Dependency Parsing of Equations	6. Extraction of Notations and Definitions (~ Terminology Extraction) 7. Segmentation of Definitions (or *Definition Parsing*) 8. Segmentation of Theorems (or *Theorem Parsing*) 9. Segmentation of Proofs (or *Proof Parsing*) 10. Math Information Extraction 11. Mathematical & Textual Entailment (MTE), *aka*	13. Math Question Answering 14. Math Summarization (Extractive and Abstractive) 15. Presentation-to-Computation (P2C) Conversion 16. Math Search And more: • Math language generation, given certain prompts • Math error detection and

What Is Involved in Each Task

- Define the task precisely
- Describe precisely the (x,y) nature of each instance of the (presumed) dataset, to match the defined task
- Define the performance metrics
- Develop an actual dataset
- Preferably create software for loading/reading the dataset
- Optionally provide some baseline models trained on the dataset, along with baseline performance data
- S. Chatzikyriakidis, R. Cooper, S. Dobnik, and S. Larsson, “An overview of natural language inference data collection: The way forward?”, In Proceedings of the Computing Natural Language Inference Workshop, 2017

Criteria for Selecting Tasks

- Fundamental and relevant
  - Feeds to one or more important applications/computational modules
- Basic enough so that we can create for it an annotated dataset suitable for ML model training & testing
- Sufficiently different from other tasks

Equation-Level Tasks
(Equation Tokenization and Parsing)

- Tokenization and parsing of math expressions/equations are fundamental tasks in math linguistics
- They are exceptionally challenging for a number of reasons
  - Lack of universal grammar for math expressions/equations
  - Fluidity of vocabulary (abstract terms have different meanings/roles in different contexts)
  - Fluidity (ambiguity) of math structures: Is “**”** or ?
  - Tokenization ambiguity:
    - Is “**in**” or ?
    - Is “**Ai**” the Airy function Ai or ?
    - Is “” or ?

Task: Tokenization and String Segmentation

- **Definition**: breaking an input sequence of characters into tokens, even when tokens are not marked by spaces, punctuations, or special characters
- **Meets the task-criteria?** Obviously yes
- **Dataset**
  - Each instance is of the form: and where
    - the ’s are characters that authors use in math equations, and
    - each is either “S” or “I” or “B”: “S” indicates that is the starting character of the next token, “I” indicates that is an internal character of a token, “B” indicates is a blank
  - Example: (asin x,SSIIBS)
  - Suitable for seq2seq models

Metrics: Accuracy

Task: POM Tagging and Named Entity Recognition

- **Definition**: tagging each token in an input sequence of math tokens (derived from tokenizing an equation), where each token will be tagged with a single tag or with multiple tags
- **Dataset**
  - Each instance is of the form: and where
    - the ’s are tokens of an equation/expression, and
    - each is tag for , from a pre-defined tagset; alternatively, each is a subset of tags for a fuller description of ’s meaning/role/nature
  - Example for :
  - Suitable for seq2seq models, and also for shallow-grammar-based parser-tagger (Youssef)
- **Metrics**: Accuracy, precision, recall

Task: Math-Term Disambiguation (MTD)

i.e., Symbol-Sense Disambiguation (SSD), ~ WSD in NLP

- **Definition**: Given a sequence of tokens, where each token has a set of competing tags, determine which tag is the correct one for each token
- **Dataset**
  - Each instance is of the form: , and
    - the ’s are tokens of an equation/expression,
    - each is a bunch of “|”-separated alternative/competing tags for
    - each is the correct tag of ,
  - Example (for):
  - Suitable for seq2seq models, where the output sequence is a sequence of disambiguated tags
- **Metrics**: Accuracy

Math Disambiguation

-- with PhD Student Ruocheng Shan --

Superscript

Class	Example Equation
power	
part-of-name	
higher-order derivative	
summation upper bound	
integral upper bound	

Prime

Class	Equation
derivative	$h_s(z) = h(z)g_{s-1}(z) + h'_{s-1}(z)$
part_of_name	$a' = -a + \sum_{j=1}^{n} b_j$

Gamma

Class	Equation
gamma_function	$\Gamma(z) = \int_0^\infty e^{-t}t^{z-1}dt$
incomplete_gamma_function	$\Gamma(a,z) = \int_z^\infty t^{a-1}e^{-t}dt$
q_gamma_function	$\Gamma_q(z+1) = \frac{1-q^z}{1-q}\Gamma_q(z)$
multivariate_gamma_function	$\Gamma_m(a) = \int_\Omega etr(-X)\|X\|^{a-\frac{1}{2}(m+1)}dX$

Comparative Evaluation Results

Prime () Disambiguation: derivative vs. part-of-name

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	**84%**	**92%**	**78%**	**83%**
Random Forest	78%	80%	78%	79%
SVM	78%	87%	70%	78%
LSTM	56%	57%	51%	54%

- **DT** is clearly a winner for Prime
- Demonstrate DT’s ability to learn better the patterns when the dataset is **small**
- LSTM shows poor performance

Comparative Evaluation Results

Superscript Disambiguation: power vs. part-of-name vs. derivative vs. sum/int limit

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	72%	75%	66%	70%
Random Forest	**83%**	86%	83%	85%
SVM	83%	**92%**	**86%**	**87%**
LSTM	65%	68%	59%	63%

LSTM

- Still bad
- But better than in prime and gamma
- Because the larger dataset, the better the LSTM

- **SVM** delivers the best performance
- **DT** gave the **least** performance among all three ML models
- That is because the superscript dataset is the largest
  - The larger the dataset, the better the SVM
  - The larger the dataset, the worse the DT

Comparative Evaluation Results

Gamma Disambiguation

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	82%	**92%**	76%	**83%**
Random Forest	**83%**	83%	**79%**	81%
SVM	82%	91%	76%	82%
LSTM	45%	55%	40%	46%

- All 3 conventional ML models gave good performance
- LSTM is very poor, as expected, due to small training dataset size

Quick Observations about ML in Math Disambiguation

- ML is quite applicable to Math Disambiguation
- Because of the lack of large labeled datasets in math, we can't exploit the full potential yet
- Rather, classical ML models (SVM, RF and DT) are better suited when datasets are small
  - Decent performance (**83%-84%** in accuracy)
- But to get to much higher performance, DL will be needed, and thus large labeled datasets for math need to be developed
  - We're working on that (including development of labeled datasets)

A Challenge/Opportunity in Disambiguation (1/2)

- The actual competing candidates for token tags may need to be generated (at least in part) from surrounding text, not just from the equation itself, and not just from fixed *a priori* lists
- Also, the disambiguation process, i.e., selecting the best competing tag, may need the surrounding text, for better performance
- Similarly, the tagging process (i.e., selecting for each token the right tag from a complete tagset) may benefit from the surrounding text
- **Challenge/Opportunity**: How do we develop & structure datasets for disambiguation and tagging, to improve performance of those tasks?

A Challenge/Opportunity in Disambiguation (2/2)

- **Challenge/Opportunity**: How do we develop and structure datasets for disambiguation and tagging, to improve performance of those tasks?
- One possibility for the dataset: Have each sample as
  - ([text_1, , text_2] ,) where
    - text_1 and text_2 are chunks for text that occur before and after the target equation in the native document
    - is the sequence of tokens of the equation
    - each is tag for , from a pre-defined tagset, for the tagging task; or a set of competing candidate tags for , for the disambiguation task

Key Tasks for Math Linguistics
-- Counterparts of NLP Tasks --

Equation-level Tasks (Tokenization and Parsing)	Document-level Tasks	Application-level Tasks
1. Math Tokenization and String Segmentation **2. Part-of-Math (POM) Tagging and Named Entity Recognition (NER)** **3. Math-term Disambiguation** 4. Constituency Parsing of Equations 5. Dependency Parsing of Equations	6. Extraction of Notations and Definitions (~ Terminology Extraction) 7. Segmentation of Definitions (or *Definition Parsing*) 8. Segmentation of Theorems (or *Theorem Parsing*) 9. Segmentation of Proofs (or *Proof Parsing*) 10. Math Information Extraction 11. Mathematical & Textual Entailment (MTE), *aka* Natural Language Inference in Math	13. Math Question Answering 14. Math Summarization (Extractive and Abstractive) **15. Presentation-to-Computation (P2C) Conversion** 16. Math Search And more: • **Math language generation, given certain prompts** • Math error detection and correction • Reasoning tasks

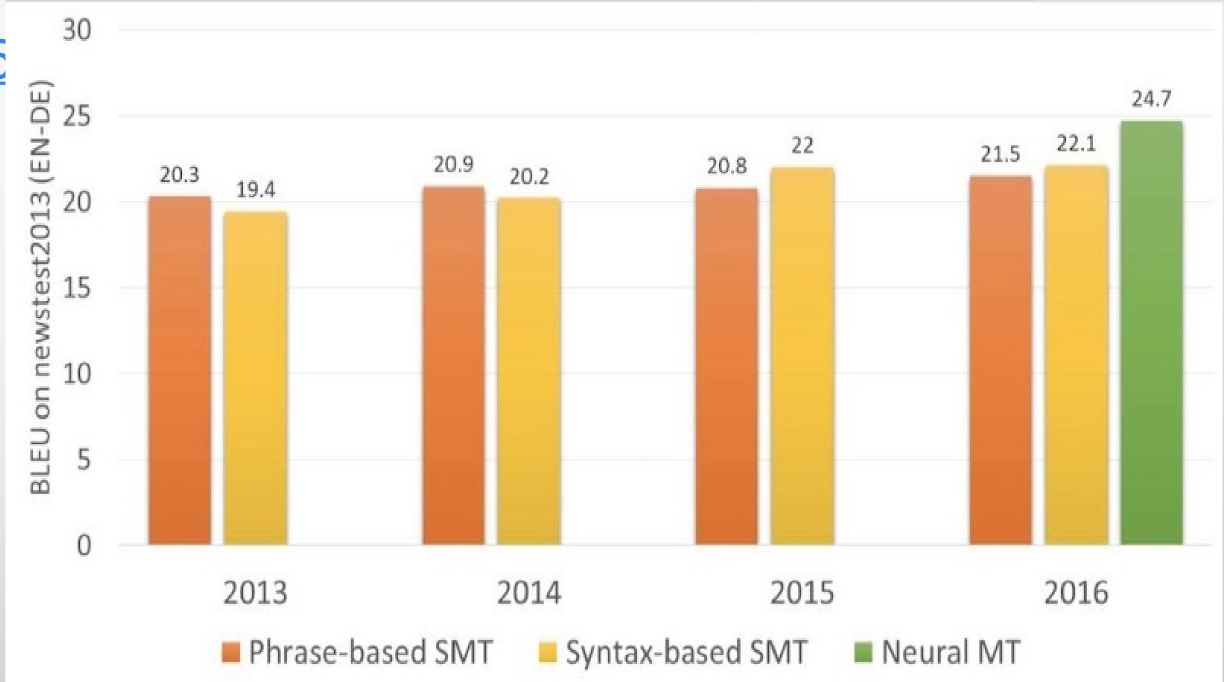
Presentation to Computation (P2C) Conversion

- P2C: Automated coding of math expressions
  - Converting an input math expression (in Latex/etc.) into Maple/Mathematica/C/… code or into formal representation (e.g., cMathML)
- P2C is a machine translation (MT) problem
  - Much like the classical NLP language translation problem
- Any hope?
  - DL is revolutionizing machine translation
  - Though in a nascent stage, automated coding is already here

DL Performance in Machine Translation

- Language Translation
  - Now (2021), Google Translate's BLEU score = 37.56 (Combination of DL and other techniques/tricks)

  (Zouhar et al. arXiv:2109.05016



Presentation to Computation (P2C)
-- Syntactic Approach --

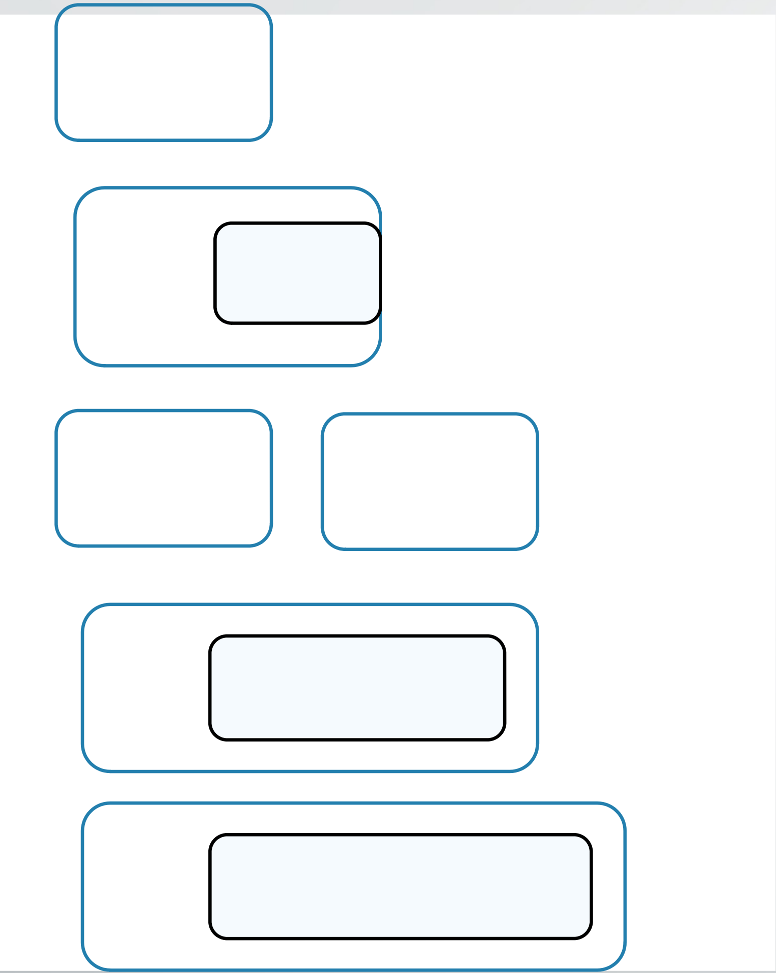
- We developed a P2C conversion system (Andre Greiner-Petter et al., TACAS 2022)
- Applied it on NIST’s DLMF
  - to convert many of the DLMF equations to Maple/Mathematica
  - and to verify the correctness of those equations
- Some of the issues, results and observations are highlighted next

P2C: A Few Issues (1/3)

- Consider this equation about the Jacobi polynomial
- Some Issues
  - Identification of the scope of the summand
  - Does the target language/platform have adequate built-in functions?
    - Gamma(z) is not in Maple!
    - EulerGamma[z] is not in Mathematica!
  - Reliable POM tagging

P2C: A Few Issues (2/3)

-- Summand Scope --



- We did this by extrapolating/assuming certain syntactic rules and implementing them
- Reasonable performance (see later)
- It will be more preferable if a ML/DL system can learn the rules from datasets and does the conversion correctly

P2C: A Few Issues (3/3)

-- Summation Index(es): Which and What Values --

Example	Index Variable(s)	Index Set
		The even integers?
		Divisors of
		and is relatively prime with

P2C: Some Performance Results and observations

- We applied our P2C system to a large subset of the DLMF equations
  - 62.9% success of translation to Maple
  - 72.0% success of translation to Mathematica
  - Close to **100% precision**
- More translation success can be achieved
  - by incorporating more syntactic rule patterns (e.g., handling ellipsis),
  - and if the target system has a richer library/built-in functions
- Still, a ML/DL approach would
  - save a lot of human engineering and coding efforts, and
  - will most likely result in larger translation coverage.

Math Language Generation

- GPT-2 and GPT-3 (by OpenAI)
- Its relevance to math language generation
  - Can we train a GPT on math corpora so it can generate relevant and correct textual responses/continuations to prompts.
- A Few Desirable Math Examples:
  - **Prompt**: “three plus seven is” or “3+7=”
    - **Generated text**: “ten” or “10”, rather than a long, winding, preachy/editorial statement (or analogies) related to 3+7
  - **Prompt**: a bare bone (skeleton) proof
    - **Generated text**: an inviting, fleshed-out proof that includes not only the skeleton but also proper transitions and bridging explanations/fill-ins
  - **Prompt**: an equation/expression in Latex or informal text
    - **Generated text**: a code (Mathematica, Maple, C, cMathML) for computation or formal representation
- Need a large math dataset useful for finetuning GPT for math language generation

- GPT generates text in response to a user’s prompt. Here is an example:
- ***Author Prompt:*** *Write me an article about creativity please*
- ***GPT-3 Response:*** *I’ll be honest. I don’t know much about creativity. I’m not a psychologist and I’m not creative myself. I’m just a guy who likes to read about psychology and creativity. What I can do, though, is write you an article about creativity that’s based on what I have read. That way you get a proper article, but you don’t have to pay for the expertise of a psychologist or a creative person.*

Math Language Generation with GPT

- https://davidbieber.com/snippets/2020-07-22-writing-with-gpt3/
- Grading GPT-3 For STEM Lesson Plan Content Generation
- Customizing GPT-3 for Your Application (from OpenAI)

Major Intermediate Systems/Apps Highly Useful

- Fulfilling the vision of a “Mathematician in a box” will take a while
- What can/should we do in the nearer term?
- I see four major intermediate things – worthwhile, needed, doable
  - STEM-Text summarization
  - STEM (Math) QA
  - Powerful, reliable P2C
  - GPT-based prompt-response style applications (preferably combined with a computation/reasoning engine)

Major Intermediate Systems/Apps Highly Useful
-- STEM-Text Summarization --

- Scientists/engineers/practitioners are overwhelmed by an ever increasing flood of new publications/findings
  - they need an automated system that regularly summarizes the relevant & latest, and present the summaries to them
- Current summarizers need to be adapted to STEM/Math summarization
  - For instance, a good summary may need to include key equations

Major Intermediate Systems/Apps Highly Useful
-- STEM (Math) QA --

- General Question Answering

MAP: Mean Average Precision
MRR: Mean Reciprocal Rank=
avg

PRE-DL PERFORMANCE	DL-BASED PERFORMANCE
MAP: 0.71 MRR: 0.78	Laskar et al. (2020), using RoBERTa: MAP: 0.95 MRR: 0.98

- Wolfram Alpha does a good job in Math QA
  - Can we do better/more with deep learning?
  - The above leap from pre-DL to DL performance in QA is very promising for math QA
  - But again, we need datasets …
- STEM (Math) QA systems will be another major enabler and will boost the productivity and efficiency of scientists/practitioners

Major Intermediate Systems/Apps Highly Useful
-- P2C --

- Covered earlier
- But barely scratched the surface
- DL (with all the tasks mentioned earlier, e.g., math tokezination, tagging, parsing, etc.) can yield much higher P2C performance
- Like Math summarizers and QA systems, P2C systems would be a major enabler and productivity booster

Major Intermediate Systems/Apps Highly Useful
-- GPT-based prompt-response style applications --

- Saw briefly some potential applications
- Stronger performance can be expected if GPT-based systems are combined with a computation/reasoning engine
  - The GPT system can learn to recognize when it needs to call a computation engine (like Maple) to carry out some numerical/symbolic computation, and incorporate the latter results into the final generated text
  - Similarly, the GPT system can learn to recognize when it needs to call a reasoning system and incorporate the results into the final generated text
- GPT, finetuned to math/STEM, can be a game changer

Closing Thoughts

- Deep Learning has a huge potential to revolutionize Math linguistics and enable previously unimaginable Math systems
- One key to the success of DL in Math Linguistics is the availability of datasets
- So, though not exciting, it is essential that we, as a community, develop labeled math datasets (while we do the exciting stuff)
- Dataset development need not be all manual
  - Semi-automated methods are possible
  - Use syntactical approaches to generate labeled datasets (e.g., math tokenizers/taggers, P2C systems), and manually check/correct borderline results, and finally use those datasets to train DL systems