HMS: A Hierarchical Solver with Dependency-Enhanced Understanding for Math Word Problem

Xin Lin, Zhenya Huang, Hongke Zhao, Enhong Chen, Qi Liu, Hao Wang, Shijin Wang

AAAI 2021

Reporter: Xin Lin
Outline

- Background
- Problem Definition
- Model Framework
- Experiments
- Conclusion
Background

- **Math word problem solver**
  - A crucial issue in AI and NLP
  - A good testbed to evaluate the intellectual ability of machines
    - e.g. NLU and logical reasoning
  - A milestone towards general AI

- **Requirements**
  - **Language comprehension**: to understand semantics of problems
  - **Mathematical inference**: to generate the expression towards the answer

<table>
<thead>
<tr>
<th>Problem</th>
<th>A rectangle is 4 cm wide, and its length is 3 cm longer than its width. What is the perimeter of the rectangle?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Answer</td>
<td>22</td>
</tr>
<tr>
<td>Expression</td>
<td>2 × (4+3+4)</td>
</tr>
</tbody>
</table>
Related Works

- Rule-based methods
  - E.g. Fletcher 1985
- Statistical machine learning
  - E.g. Hosseini et al. 2014
- Semantic parsing
  - E.g. Koncel-Kedziorski et al. 2015
- Deep learning methods
  - E.g. DNS, T-RNN, GTS et al.

Exploit language structure (e.g. semantic dependency) imitating human reading habits

Tremendous human effort and low generality

Poor understanding of problems due to neglect of language structure
Challenges

- How to mimic human part-by-part reading and pay attention to local details
  - E.g. parts with different colors in the example

- How to capture semantic dependency in each part
  - E.g. the relation between “3”, “longer” and “width”

- How to translate logic in local semantics into expression segment
  - E.g. “3 cm longer” being projected to “+3”

- How to utilize human knowledge in math domain
  - E.g. rectangle perimeter formula in the example, not given in the problem
Outline

- Background

- **Problem Definition**

- Model Framework

- Experiments

- Conclusion
Problem Definition

- **Math word problem**
  - Problem: $P = \{p_1, p_2, \ldots, p_n\}$, where $p_i$ is a word token or a numeric value
    - E.g. “A rectangle is 4 cm … its length is 3 cm … the perimeter of the rectangle?”
  - Answer: the numeric value of required variable
    - Derived by a mathematical expression
    - E.g. “22” for the “perimeter”
  - Expression: $E_P = \{y_1, y_2, \ldots, y_m\}$, where $y_i$ comes from $V_P = V_O \cup V_C \cup N_P$
    - $V_O$: mathematical operators set, e.g. $\{+,-,\times,\div\}$
    - $V_C$: mathematical constants set, e.g. $\{1, 2, \pi\}$
    - $N_P$: numeric values set from $P$, e.g. $\{3, 4\}$ in the example
    - E.g. “$2 \times (4 + 3 + 4)$”

- **Goal**
  - Read input tokens from $P = \{p_1, p_2, \ldots, p_n\}$
  - Generate output sequence of expression $\hat{E}_P = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m\}$
Outline

- Background
- Problem Definition
- Model Framework
- Experiments
- Conclusion
Model Framework

- **HMS framework**
  - Hierarchical encoder
    - Split problem into several clauses to simplify semantics
    - Model problem semantics from local to global in different levels
  - Tree-based decoder
    - Exploit semantics from different levels
    - Infer knowledge & copy information
Hierarchical Encoder

- **Problem split**
  - Goal: imitate human part-to-part reading
  - Split $P$ into several clauses $C_P = \{C_1, C_2, ..., C_m\}$ by commas and periods
    - $P$: “A rectangle … wide, and its length … width. What is … rectangle?”
    - $C_P$: {“A rectangle … wide,”, “and its length … width.”, “What is … rectangle?”}
  - Semantics in each clause is relatively complete and simple
    - Complete: contains almost all necessary semantic elements
    - Simple: describes only one relation

- **Hierarchical Encoder**
  - Goal: model semantics from local to global in different levels
    - Word-level: words and context
    - Clause-level: local details and dependency within clause
    - Problem-level: global goal and relations between clauses
Word Embedding

- Goal: semantics of words and contextual information
- Word semantics: word embedding vector
- Contextual information: Bidirectional GRU

\[
\begin{align*}
  x_s &= \text{Embed}(p_s) \\
  h_s &= \text{BiGRU}(x_s, h_-, h_+) 
\end{align*}
\]
Clause Module

- Goal: local details and dependency structure in each clause

- Dependency tree
  - Goal: capture semantic dependency
  - Construction: dependency parsing
    - E.g. Stanford corenlp toolkit
  - Node: a word in the clause
  - Edge: child depends on its parent and provides a supplementary detail
    - E.g. “rectangle” depends on “perimeter” and describes its subject
Clause Module

- Dependency-based module
  - Goal: exploit local details from children for each parent
  - Semantics exploitation
    - Leaf node (e.g. “rectangle”)
      \[ h^d_l = h_l \]
    - Inner node (e.g. “perimeter”)
      \[ h^d_p = \text{Attn}(h_p, h^d_{c^*}, h^d_{c^*}) \]
  - Bottom-up from leaves to the root
    - From ① to ⑥
  - Clause representation: the root node
    - E.g. “is”
      \[ c^d_k = h^d_{\text{root}} \]
Problem Module

- Goal: global goal and relations between clauses
- Relations
  - Sequential relation: positional encoding
  - Semantic relation: self-attention
    - E.g. the value of the width in $C_1$ & the relation between the length and the width in $C_2$
- Problem representation: the question part, usually the last one

$$c_k^p = c_k^d + \text{PE}(k)$$

$$c_k^s = \text{Attn}(c_k^p, c_*^p, c_*^p)$$

$$h^s = c_m^s$$
Tree-Based Decoder

- Goal: exploit tree structure of expression
- Prefix expression
  - By predicting expression tree
A rectangle is 4 cm … its length is 3 cm … the perimeter of the rectangle?

Tree prediction

Tree-expression map

Encoder

Decoder

\[ h^s / c^s / h_* \]
Tree-Based Decoder

- **Goal:** exploit tree structure of expression
  - **Prefix expression**
    - By predicting expression tree
  - **Tree node prediction**
    - Generate goal vector $q$ for node follow GTS
      - Determine tree structure
      - Exploit problem semantics & structure of predicted tree nodes
    - Extract context vector $c$ according to goal vector $q$
      - Hierarchical attention mechanism
    - Predict symbol distribution according to $q$ & $c$
      - Pointer-generator network
Hierarchical Attention Mechanism

- Goal: exploit local details in semantics from different levels
- Extract clause-level semantics according to clause relevance
- Extract word-level semantics according to both clause and word relevance

\[ w^c_* = \text{Attn}_\text{weight}(q, c^s_*) \]
\[ w^{w}_{k,*} = \text{Attn}_\text{weight}(q, h_{k,*}) \]
\[ c = \Sigma_k w^c_k (c^s_k + \Sigma_t w^w_{k,t} \cdot h_{k,t}) \]
Goal: copy information & infer knowledge

Split search space into knowledge $Y_g$ & given information $Y_p$

- $Y_g = V_o \cup V_c$, e.g. 2 and $\times$, $+$ in rectangle perimeter formula
- $Y_p = N_p$, e.g. width 4 and difference 3 in problem

$P_c$ weighted search on $Y_g$ & $Y_p$ with predicted $P_{gen}$

Predict symbol according to $P_c$

\[
\begin{align*}
P_{gen} &= \sigma(MLP(q,c)) \\
P_p(Y_p) &= \text{Softmax}(MLP(q,c,h_{p_y})) \\
P_g(Y_g) &= \text{Softmax}(MLP(q,c,e_{y_g})) \\
P_c(Y = y) &= \begin{cases} 
(1 - P_{gen}) \cdot P_p(y), y \in Y_p \\
P_{gen} \cdot P_g(y), y \in Y_g 
\end{cases}
\end{align*}
\]
Outline

- Background
- Problem Definition
- Model Framework
- Experiments
- Conclusion
Experiments

- Dataset
  - Math23K
  - MAWPS

- Baseline methods
  - DNS
  - Math-EN
  - T-RNN
  - GROUP-ATT
  - GTS

- Evaluate metric
  - ACC
Experiments

- Overall result & ablation study
  - Observations
    - HMS outperforms baseline models
      - HMS effectively model problems and enhance expression inference
    - DNS has the worst performance
      - Exploitation of related features in MWP is important
    - Accuracy in MAWPS is higher than Math23K
      - Higher difficulty in Math23K from longer expressions and problems and varied templates
    - Absence of each component degrades the performance
      - Effectiveness of each component

<table>
<thead>
<tr>
<th></th>
<th>Math23K</th>
<th>MAWPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>0.581</td>
<td>0.595</td>
</tr>
<tr>
<td>Math-EN</td>
<td>0.667</td>
<td>0.692</td>
</tr>
<tr>
<td>T-RNN</td>
<td>0.669</td>
<td>0.668</td>
</tr>
<tr>
<td>GROUP-ATT</td>
<td>0.695</td>
<td>0.761</td>
</tr>
<tr>
<td>GTS</td>
<td>0.743</td>
<td>0.786</td>
</tr>
<tr>
<td>HMS</td>
<td><strong>0.761</strong></td>
<td><strong>0.803</strong></td>
</tr>
<tr>
<td>HMS w/o hierarchy</td>
<td>0.750</td>
<td>0.788</td>
</tr>
<tr>
<td>HMS w/o dependency</td>
<td>0.755</td>
<td>0.791</td>
</tr>
<tr>
<td>HMS w/o pointer-generator</td>
<td>0.756</td>
<td>0.789</td>
</tr>
</tbody>
</table>
Experiments

- **Performance over clause number**
  - **Observations**
    - Performance decreases with increasing clause number due to increasing difficulty.
    - In group “6” and “7” HMS outperforms GTS obviously:
      - Performance drops sharply due to complex problems.
      - HMS is able to capture more semantics from complex problems.
    - In group “1” HMS outperforms GTS:
      - The main difference is dependency-based module, whose effectiveness is proved.
Outline

- Background
- Problem Definition
- Model Framework
- Experiments
- Conclusion
Conclusion

Summary
- HMS framework
  - A hierarchical encoder with dependency-based module to model problem semantics
  - A tree-based decoder with hierarchical attention mechanism and pointer-generator network to generate expressions
- Experimental results proved the effectiveness

Future work
- Investigate more relations in problems to enhance MWP solver
  - E.g. different reference of the same entity
- Improve exploitation of human knowledge in MWP
  - E.g. mathematical knowledge, common sense
Thanks for your listening!

Q&A