Discrete Reasoning Templates for Natural Language Understanding

Hadeel Al-Negheimish, Pranava Madhyastha, Alessandra Russo

Imperial College London

Motivation

- Reasoning about information from multiple parts of a passage to derive an answer is an open challenge for reading-comprehension models.
- We work on developing a complementary approach based on *reasoning templates* that exploits the power of the contextualized representations large language models provide with symbolic reasoning.
- We show that our approach is competitive with the state-of-the-art on subtraction-type questions while being interpretable and requires little supervision.

Task: DROP dataset

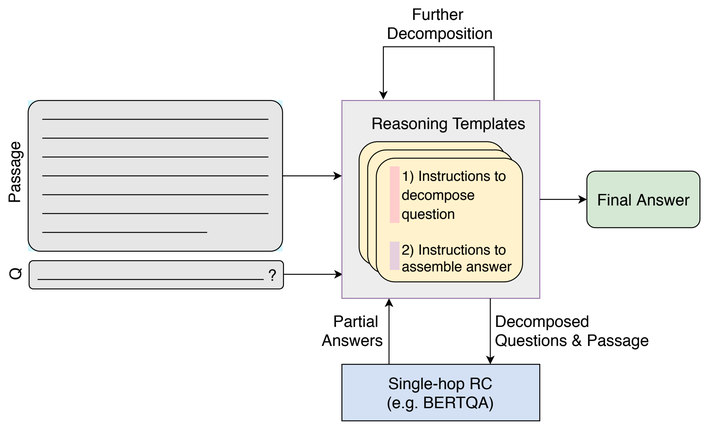
DROP (**D**iscrete **R**easoning **O**ver the content of **P**aragraphs) is a reading comprehension dataset that is inspired by the semantic parsing literature. It contains questions that involve possibly multiple steps of discrete reasoning over the contents of paragraphs, including numerical reasoning.

Main idea: Iteratively decompose complex questions into simpler ones

Each reasoning type is associated with a single template, atomic questions can be answered with a single-span extraction LM

The building block of our approach is a *reasoning template*, where each template contains two main components:

1) Instructions on how to decompose a question
2) How to derive the final answer given partial answers to subquestions



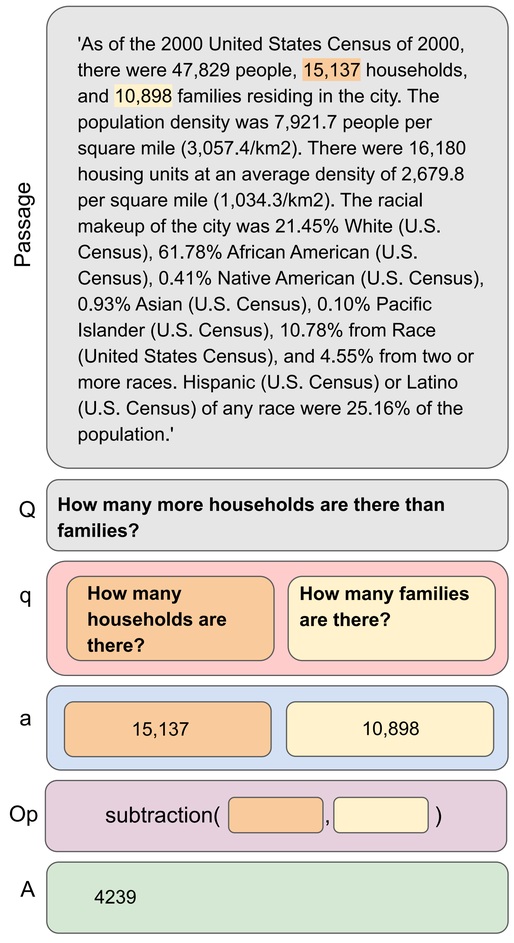
Proof of Concept: Subtraction template

We start by demonstrating our approach for a single type of reasoning, *subtraction*. Here is a breakdown of the components of the template:

Question decomposition: We build upon the method used in DecompRC for decomposing questions, using a pointer model to extract relevant spans from the question and write a heuristic procedure to generate subquestions given these pointers.

Single-hop question answering: We use off-the-shelf LMs to extract answers to the subquestions generated in the first step.

Operation: In this template, the operation used to find the final answer is the absolute difference of the numbers retrieved as partial answers.



Experiments

1) How good are the generated subquestions?

For most questions, the two generated subquestions perfectly match the gold annotations, but sometimes the final verb is omitted.

Similarity Measure	WMD_{max}	WMD_{avg}	WMD_{median}
q1	3.56	0.23	0
q2	4.43	0.67	0

Word Mover's Distance between the two generated subquestions and gold annotations

2) How good is this approach at finding the final answer?

Datasets

Clean: A manually curated subset of 52 subtraction questions of DROP's devset, these have been inspected for the correctness (5 mislabeled) of the annotation, and gold decompositions have been crafted for each.

Noisy: Heuristically collected subset of 892 subtraction questions. Questions were filtered based on trigrams at the beginning of the question: *'How many more'* or *'How many fewer'*.

SoTA

We compare our approach with the state-of-the-art; MTMSN the best performing model with specialized modules; and NeRd, the most recent work based on program induction.

	Model	Acc_{Clean}	Acc^{-}_{Clean}	#MM	Acc_{Noisy}
SoTA	MTMSN	86.5	89.4	3	81.3
	NeRd	73	76.6	2	62.3
Ours	$Decomp_{Gold}$	78.8	85.1	1	-
	$Decomp_{Learn}$	74.4± 2.4	79.9±2.6	1	64

This table shows accuracy (EM) on clean (including w/o mislabeled Acc^{-}_{Clean}) and noisy datasets, and number of matched mislabeled (**#MM**) questions by each model.

For our work, we evaluate two different variations: We run the pipeline on the gold decompositions that have been manually rewritten, and automatically-decomposed questions generated by our approach, using BERT single-hop RC fine-tuned for SQuAD. For both gold-decompositions($Decomp_{Gold}$) and learned-decompositions ($Decomp_{Learn}$) we get promising results that are on par with the state-of-the-art on this dataset.