Application of Natural Language Processing Toward Scientific Text Understanding

Aida Amini

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

University of Washington

2021

Reading Committee:

Hannaneh Hajishirzi, Chair

Yejin Choi

Bhavana Dalvi

Program Authorized to Offer Degree:

Computer Science

University of Washington

Abstract

Application of Natural Language Processing Toward Scientific Text Understanding

Aida Amini

Chair of the Supervisory Committee:

Professor Hannaneh Hajishirzi

Computer Science and Engineering

Natural language processing over scientific text is useful for many downstream tasks, such as mathematical question answering, understanding chemical processes, and extracting knowledge base relations from biology-related articles. Scientific text, which consists of continuous formal statements, technical lexicon, and implicit relations differs significantly from general narratives. Its analysis and labeling requires laborious effort from domain experts with sufficient background knowledge. Therefore, the labeled data in scientific domains can be more scarce and noisy. Further, answering technical questions related to these domains (e.g., algebra or physics queries) requires reasoning capabilities beyond those available for conventional natural language processing.

The emergence of pre-trained language models has improved the quality of predictions in many tasks, but due to the use of technical terms and formal statements, these models do not perform well over scientific tasks. Therefore, additional methods are needed both for collecting data and navigating textual implications in order to achieve better outcomes. Furthermore, scientific texts usually contain implicit references to background knowledge within the same or different scientific domain, and as the result of sparsity in training sets, models lack the ability to encode the necessary background knowledge at training time.

This dissertation presents applications of natural language processing (NLP) toward better understanding and grounding the scientific literature in terms of (a) data curation, (b) modeling approaches, (c) large-scale application construction, and (d) evaluation. We present the results of

three research endeavors on scientific text analysis. Specifically, we tackle (1) answering questions in the mathematical domain, (2) understanding and following the elements of a scientific process (e.g., natural, chemistry), and (3) knowledge base (KB) construction of functional relations over diverse and interdisciplinary scientific domains. Each task poses challenges due to the scarcity of annotated data and the need for a higher level of inference. There still remain limitations to completely comprehend and understand the scientific text using NLP techniques, but our proposed approaches can further push the boundaries of scientific text understanding.

Acknowledgement

First and foremost, I wish to express my deepest gratitude and thanks to my advisor, Prof. Hannah Hajishirzi, who has supported me throughout my endeavor. Her invaluable support, knowledge and supervision has helped me immensely through my studies. She provided me with the guidance and freedom to explore and encourage me to be the more confident in myself.

I would also like to thank the UW CSE community, the most intelligent and supportive people I met during last six years of mys studies. I always felt as being part of a great and welcoming family. Specially, I want to express my deepest gratitude to our graduate advisor, Elise Degoede Dorough, who has helped me alot with all the technical aspects of going through the PhD program.

I am really grateful to my thesis committee, Yejin Choi, Bhavana Dalvi Mishra, and Meliha Yetisgen for their time and insightful comments. I am honored to have them as my committee.

I would also like to extend my thanks to all my fantastic co-authors who have contributed throughout different projects: Yejin Choi, Dan Weld, Eric Horvitz, Madeleine van Zuylen, Tom Hope, David Wadden, Roy Schwartz, Sravanthi Paras, Saadia Gabriel, Rik Koncel-Kedziorski, Peter Lin, Antoine Bosselut, Bhavana Dalvi Mishra.

I would also like to acknowledge my family, specially my mother, Zohreh Azima and my father, Abdolreza Amini for always believing that I can achieve whatever I set my mind to and for encouraging me to follow what I cherish. I would also like to extend my special thanks to Kiana Ehsani, my dearest friend. We started our journey as graduate students at the same time and she has always supported me throughout my graduate studies.

To my beloved grandmother.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We are witnessing significant progress in natural language processing (NLP) in many application domains, including query-based search, question answering applications, and translation. This progress chiefly results from the availability of large annotated training datasets and the emergence of deep learning models and pre-trained language models. Current NLP approaches focus on English text from news articles and Wikipedia mainly because these types of texts are comprehensible to the general public and are easily collected using crowd sourcing.

However, many applications (e.g., educational systems, scientific search engines) require the understanding and processing of scientific text in resources such as scientific journal articles, textbooks, and technical examinations. Understanding scientific text differs from general text understanding in terms of data collection and modeling challenges due to: (1) the presence of a technical lexicon of the underlying domain, (2) the use of formal narratives to describe facts, and (3) the need for implicit reasoning based on background knowledge to comprehend and process the text. Furthermore, collecting and annotating scientific data to train machine learning (ML) models is laborious and time consuming, requiring a labor force with expertise in the underlying domain. As a result, existing labeled datasets for scientific and technical domains are usually both scarce and noisy, motivating efforts that address automatic labeling and improved modeling to make use of

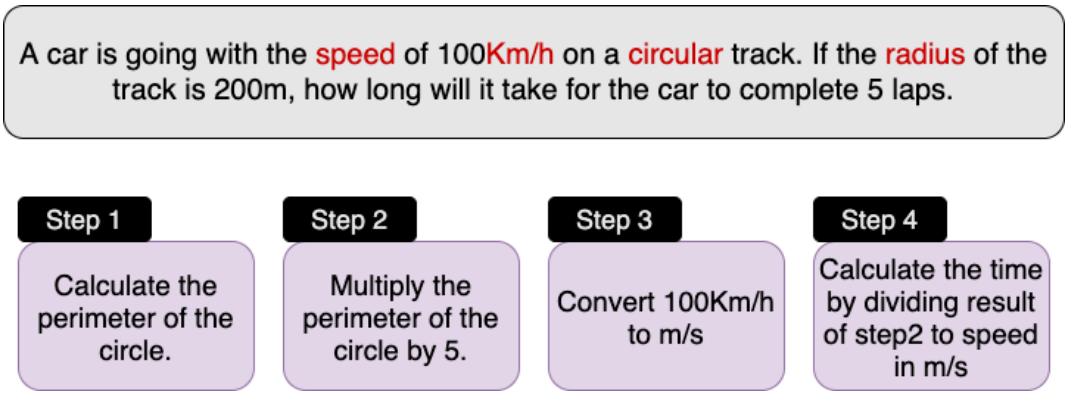


Figure 1.1: Example of a math word problem and the intermediate reasoning for problem-solving.

underlying patterns.

Further, the improved modeling of scientific and technical literature can also improve our understanding of general narrative literature. Numerous instances of general narratives use technical terms in many downstream applications, for example, social effects on financial markets, weather forecasting, and product assembly instructions.

In this dissertation, we focus on NLP approaches toward the tasks of reading comprehension and information extraction in scientific literature. Our research focuses on three main categories of scientific domains: math problems, scientific processes and interdisciplinary science domains. Within these categories and evaluations across two tasks of question answering and information extraction we explore data curation, model architecture, applications, and evaluations.

The domain of **mathematical word problem solving** has received substantial attention and many question answering algorithms have explored their performance within the domain. Math word problem refers to **algebra** and **physics** questions, which include textual and numerical information required to solve the problem. Due to annotation challenges, current datasets in this domain have been either relatively small in scale or did not offer precise operational annotations over diverse problem types.

The other challenge regarding this domain includes the background knowledge necessary for finding the solution of the problems. This knowledge may include mathematical or physics formula,

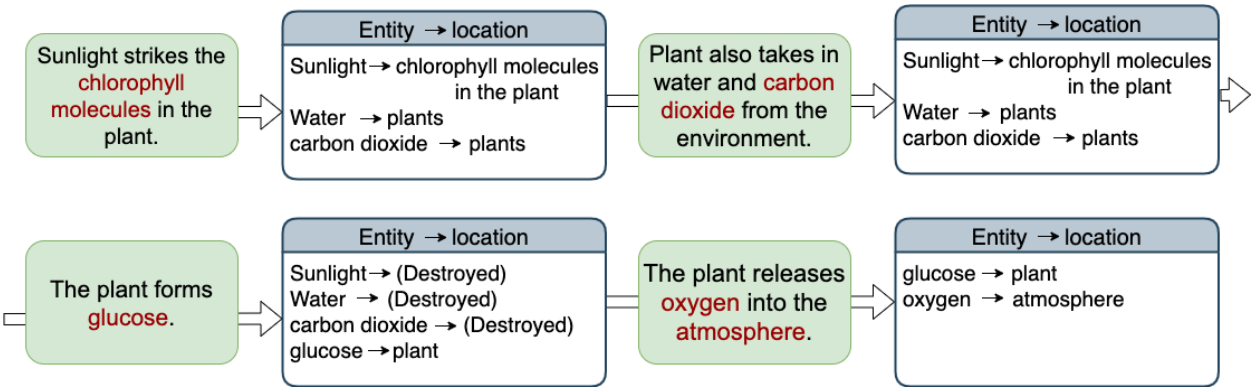


Figure 1.2: Example of scientific process explaining how elements like water, sunlight and CO_2 are mixed to create sugar and O_2.

implicit common-sense reasoning, etc. For instance, in the problem in figure 1.1, a model reasoning from scratch, not only should conclude that calculating circle's perimeter is part of the solution, but also reason for the correct formula for such calculation. While the first challenge is interesting and needs semantic grounding and understanding of the task, the second one can be handled as a retrieval task. In this dissertation, we have introduced a new representation language for representing the mathematical solutions in sequential steps. Each step consists of a meaningful mathematical calculation where the background knowledge related to required formulas and constants is provided. This formalism facilitates the overall data annotation and results in more interpretable modelings through the stepwise and accumulative solutions.

Algebra word problems have quantitative solutions and numeric results. In addition, to answer questions regarding other scientific domains (e.g., chemistry, biology, geology, and etc.), the models should retain the ability to explore qualitative and textual values. Understanding and the ability to ground information and answer questions within these domains can play a crucial role in many downstream applications such as educational systems. The information within these domains is usually in the form of procedural texts. Hence, in this dissertation we address a more general task of reading and comprehending procedural text across multiple domains.

Procedural reading comprehension: Procedural texts describe processes (e.g., photosynthesis, cooking instruction) through the state changes undergone by the entities (e.g., plant, food) involved in them. Procedural reading comprehension consists of assigning values to attributes related to the entities of process at every timepoint within the process. By tracking these values, the model can draw conclusions regarding a higher picture of the context. As illustrated in Figure 1.2 the procedural text consists of a few sentences, a few main participants (e.g., `water`, `sunlight`, and CO_2) and the attributes that we are interested in (e.g., `location`). By keeping track of attribute values, we are enabled to answer higher level questions such as the input, outputs and the conversions of the process.

Recent works in understanding procedural text develop domain-specific models for tracking entities in scientific processes (Mishra et al. [2018]) or cooking recipes (Bosselut et al. [2018]). More recently, Gupta and Durrett [2019a] leverage pre-trained language models to obtain general entity-aware representations of procedural text and predict entity transitions from a set of pre-defined classes that are independent of entity attributes. Pre-defining the set of entity states limits the model's general applicability, since entity attribute values might be arbitrary spans of text. Moreover, entity attributes can be used for tracking entity state transitions. For instance, a change in the `location` value of an entity leads to `movement`; however, there may be no explicit reference to `movement` since it might have to be inferred from the change in the attribute's value.

Interdisciplinary information extraction: Recently, the COVID-19 pandemic has spawned a diverse body of scientific literature that is challenging to navigate, stimulating interest in automated tools to help find useful knowledge. The emergence of the COVID-19 virus has motivated scientists from interdisciplinary backgrounds, to focus their research endeavor toward mitigating the pandemic. This results in a large body of text with similar focus but from various domains of science. The sheer number of the accumulated publications, and the technical background required to read through each article in order to extract information, enhances the usefulness of a large-scale application that

enables scientific queries.

Recent information extraction schemas are usually designed to capture entities and relations based on an ontology related to the underlying domain. Yet, they fail to translate smoothly to other domains. On the other hand, recent works on open information extraction have focused primarily on coverage, and therefore, the resulting relations lack a certain amount of desired informativeness.

Additionally, most recent works have focused on well defined entities, though not all information is represented in specific format. As an illustration, in sentence

- *Respirators can be in short supply , though they are necessary to protect workers from SARS - CoV-2 exposure.*

The phrase `protect workers from SARS - CoV-2 exposure` can play a meaningful role in showcasing the use-case of `respirators`.

As part of this dissertation, we introduced the notation of `mechanisms`, a fundamental concept across the sciences, which encompasses activities, functions and causal relations, ranging from cellular processes to economic impacts. In addition, to allow for more coverage over concepts, it is necessary to allow for either side of the mechanism based relation to be from open and free form spans of text.

We pursue the construction of a knowledge base (KB) of `mechanisms`. We extract this information from the natural language of scientific papers by developing a broad, unified schema that strikes a balance between relevance and breadth. We annotate a dataset of mechanisms with our schema and train a model to extract mechanism relations from papers.

1.1 Contributions

In each of these problem areas, we confronted the key issues and produced notable results. Specifically, we:

- Explore the task of question answering in the domain of math word problems, where the language structure and lexicon differ substantially from other forms of narrative text and data annotation is challenging:

  - ⋆ By defining a **sequential formalism** that captures mathematical concepts, we demonstrate the ability to navigate the challenging task of mathematical data collection.
  - ⋆ We have curated a large-scale and diverse dataset carefully annotated with our formalism containing 37K GRE and GMAT level problems.
  - ⋆ Using this formalism, the simple model we propose improves the solution accuracy for every mathematical category by an average of 2% compared to the state-of-the-art model.

- Tackle the task of entity tracking for procedural reading comprehension, where we show that **using the implicit relation between participants' states and their changes** helped our model outperform previous state-of-the-art models by 3 point of F1 score.

  - ⋆ We present attribute aware representation, and we show that based on our studies, they accommodate for 2 points of the F1 score.

- Focus on the task of information extraction over COVID19-related publications from domains such as bio-medicine, sociology and computer science. Specifically we:

  - ⋆ define **coarse-grained relations** which assist in expediting the task of data annotation through interdisciplinary domains.
  - ⋆ curate a **sample of carefully annotated relations** and release a dataset containing 2K functional relations.
  - ⋆ train a state-of-the-art relation extraction model, which was able to extract over 2M relations from 160K publications.

1.2 Thesis Outline

Chapter 2 lays out the previous work regarding the tasks of mathematical word problem solving, procedural reading comprehension, and scientific information extraction.

In chapter 3, we present our research toward improving the quality of algebra word problem solving. We introduce a new formalism for representing mathematical solutions and a dataset (called MathQA) annotated with that formalism. Furthermore, we introduce our modeling approach toward solving higher level algebra word problems.

In chapter 4, we showcase DYNAPRO, our model designed for the task of reading comprehension over procedural text. Our model leverages pre-trained language models to construct entity-aware and attribute aware representations and jointly predict for the attribute values and transitions.

In chapter 5, we explore interdisciplinary information extraction in order to create a large scale relation based search interface. First, we introduce a new schema that holds a balance between informativeness and coverage. Then we present our data collection approach, and train a model. Finally we showcase our interface enabling users to perform relation based queries and we present our evaluations for our information extraction model and the utility of our search interface.

Lastly, chapter 6 presents a summary of the topics discussed in the dissertation and outlines future research directions for further improving the understanding and analysis of scientific literature.

1.3 Publication

Portions of the thesis have appeared in the following publications:

- Amini, A., Gabriel, S., Lin, S., Koncel-Kedziorski, R., Choi, Y. and Hajishirzi, H., 2019, June. MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long

and Short Papers) (pp. 2357-2367).

- Amini, A., Bosselut, A., Mishra, B.D., Choi, Y. and Hajishirzi, H., 2020, February. Procedural Reading Comprehension with Attribute-Aware Context Flow. Best paper honorable mention in Automated Knowledge Base Construction (AKBC) conference.

- Amini, A., Hope, T., Wadden, D., van Zuylen, M., Parasa, S., Horvitz, E., Weld, D.S., Schwartz, R. and Hajishirzi, H., 2021, June. Extracting a Knowledge Base of Mechanisms from COVID-19 Papers. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 4489-4503).

Chapter 2

Related Work

This chapter discusses relevant background and related work.

2.1 Algebra word problem solving

Solving with Handcrafted Features: Due to sparsity of suitable data, early work on math word problem solving used pattern-matching to map word problems to mathematical expressions (Bobrow [1964]; Charniak [1968, 1969]), as well as non-neural statistical modeling and semantic parsing approaches (Liguda and Pfeiffer [2012]).

Some effort has been made on parsing the problems to extract salient entities (Hosseini et al. [2017]). This approach views entities as containers, which can be composed into an equation tree representation. The equation tree representation is changed over time by operations implied by the problem text.

Many early works focused on solving addition and subtraction problems (Briars and Larkin [1984]; Dellarosa [1986]; Bakman [2007]). As word problems become more diverse and complex, we require models capable of solving simultaneous equation systems. This has led to an increasing focus on finding semantic alignment of math word problems and mentions of numbers (Roy and

Roth [2018]). The main idea behind those work is to find all possible patterns of equations and rank them based on the problem.

Additional Datasets: Several smaller datasets have been compiled in recent years. Most of these works have focused on algebra word problems, including MaWPS (Koncel-Kedziorski et al. [2016a]), Alg514 (Kushman et al. [2014a]), and DRAW-1K (Upadhyay and Chang [2017]). Many of these datasets have sought to align underlying equations or systems of equations with word problem text. While recent works like (Liang et al. [2018]; Locascio et al. [2016]) have explored representing math word problems with logical formalisms and regular expressions, our work is the first to provide well-defined formalisms for representing intermediate problem-solving steps that are shown to be generalizable beyond algebra problems (Section 3.2).

Large-Scale Datasets: Several large-scale math word problem datasets have been released in recent years. These include Dolphin18K (Huang et al. [2016]), Math23K (Wang et al. [2017]) and AQuA(Ling et al. [2017]). We choose the 2017 AQuA dataset to demonstrate use of our formalism on an existing large-scale math word problem solving dataset. The AQuA provides over 100K GRE- and GMAT-level math word problems. The problems are multiple choice and come from a wide range of domains.

The scale and diversity of this dataset makes it particularly suited for use in training deep-learning models to solve word problems. However there is a significant amount of unwanted noise in the dataset, including problems with incorrect solutions, problems that are unsolvable without brute-force enumeration of solutions, and rationales that contain few or none of the steps required to solve the corresponding problem. Using the AQuA dataset in addition to the formalism for representing sequential steps to solve a problem, we introduce a new dataset called MathQA (Section 3.3). The motivation for our dataset comes from the fact we want to maintain the challenging nature of the problems included in the AQuA dataset, while removing noise that hinders the ability of neuralized models to learn the types of signal neccessary for problem-solving by logical reasoning.

Neural Word Problem Solvers: Following the increasing availability of large-scale datasets like

AQuA, several recent works have explored deep neural approaches to math word problem solving (Wang et al. [2017]). Our representation language is motivated by exploration of using intermediate formalisms in the training of deep neural problem-solving networks, as is done in the work of (Huang et al. [2018b]) to solve problems with sequence to sequence models. While this work focused on single-variable arithmetic problems, our work introduces a formal language of operations for covering more complex multivariate problems and systems of equations.

Interpretability of Solvers: While the statistical models with handcrafted features introduced by prior work are arguably "interpretable" due to the relative sparsity of features as well as the clear alignments between inputs and outputs, new neuralized approaches present new challenges to model interpretability of math word problem solvers (Huang et al. [2018a]). While this area is relatively unexplored, a prior approach to increasing robustness and interpretability of math word problem-solving models looks at using an adversarial dataset to determine if models are learning logical reasoning or exploiting dataset biases through pattern-matching (Liang et al. [2018]).

2.2 Procedural reading comprehension

Most previous work in reading comprehension (Rajpurkar et al. [2016]) focuses on identifying a span of text that answers a given question about a static paragraph. In contrast, this dissertation focuses on procedural reading comprehension, which inquires about how the states of entities change over time. There are several previous works that focus on understanding procedural text in multiple domains. Mishra et al. [2018] introduced the PROPARA dataset, a collection of procedural texts that describe how entities change during scientific processes (e.g., photosynthesis), which is paired with questions about several aspects of these processes, such as the entity attributes or state transitions. Bosselut et al. [2018] focus on cooking recipes, which describe instructions on how to change ingredients. Math word problems (Kushman et al. [2014b]; Hosseini et al. [2017]; Koncel-Kedziorski et al. [2016b]) describe how the states of entities change throughout mathematical

procedures. Narrative question answering (Weston et al. [2015]; Kočiskỳ et al. [2018]; Lin et al. [2019]) inquires about the state of a story over time.

These resources have influenced many models (e.g., EntNet (Henaff et al. [2017]), QRN (Seo et al. [2017]), MemNet (Weston et al. [2014])) that track entities in narratives. The closest of these works to ours, however, is the line of work focusing on the PROPARA and NPN-COOKING datasets. Bosselut et al. [2018] use an attention-based neural network, the neural process network (NPN), to identify ingredient state transitions in cooking recipes. To track state in scientific processes, models such as Pro-local and Pro-Global (Mishra et al. [2018]) first identify locations of entities using an entity recognition approach and use manual rules or global structure of the procedural text to consistently track entities. Tandon et al. [2018] leverage manually defined and KB-driven commonsense constraints to avoid nonsensical transitions when predicting entity states (e.g., a tree doesn't move to a new location). KG-MRC (Das et al. [2019]) maintains a knowledge graph of entities over time and identifies entity states by predicting the location spans with respect to each entity using a reading comprehension model. NCET (Gupta and Durrett [2019b]) introduces a neural conditional random field model to maintain the consistency of state predictions. Most recently, ET_{BERT} (Gupta and Durrett [2019a]) uses transformers to construct representations of each entity and predict the state transitions from a set of pre-defined classes.

In this dissertation, we integrate these prior observations, and develop a model that jointly identifies entities, attributes, and transitions over time. Unlike previous work that is designed to address either attributes or transitions, our model benefits from the clues that are implicitly and explicitly mentioned for both entity attributes and transitions. Leveraging both aspects of procedural reading comprehension leads us to a general and adaptive framework for the task, and an accompanying model that achieves state of art on several datasets across different domains.

Schema	Entity types	Relations	Example
SciERC	CS methods/tasks (free-form spans)	used-for	Use **GNNs** for **relation extraction.**
SemRep	Clinical (drugs, diseases, anatomy . . .)	causes, affects, treats, inhibits, interacts, used . . .	. . . intratympanic **dexamethasone injections** for patients with intractable **Meniere's disease.**
ChemProt	Chemicals, proteins	direct/indirect regulator, inhibitor, activator . . .	**Captopril** inhibited **MMP-9** expressions in right ventricles.
DDI	Drugs	interacts	**Quinolones** may enhance the effect of **Warfarin.**
GENIA	Proteins, cellular entities	binding, modification, regulation . . .	**BMP-6** induced phosphorylation of **Smad1/5/8.**
PICO	Clinical	Interventions, outcomes	The **bestatin** group achieved **longer remission.**
Ours: MECHANIC	Medicine, epidemiology, genetics, molecular bio., CS, math, ecology, economics . . . (free-form)	direct (activities, functions) / indirect (influences, associations)	· **RL** can be used to learn **mitigation policies in epidemiological models.** · **Histophilus-somni** causes **respiratory, reproductive, cardiac and neuronal diseases in cattle.**

Table 2.1: Our broad concept of mechanisms covers many relations within existing science-IE schemas. The table shows examples of representative schemas, and the types of entities and relations they capture.

2.3 Interdisciplinary information extraction

Mechanisms in science: The concept of mechanisms, also referred to as *functional relations*, is fundamental across the sciences. For example mechanisms are described in biomedical ontologies (Burek et al. [2006]; Röhl [2012]; Keeling et al. [2019]), engineering (Hirtz et al. [2002]), and across science. Mechanisms can be natural (e.g., the mechanism by which amylase in saliva breaks down starch into sugar), artificial (electronic devices), non-physical constructs (algorithms, economic policies), and very often a blend (a pacemaker regulating the beating of a heart through electricity and AI algorithms).

Although seemingly intuitive, exact definitions of mechanisms are subject to debate in the philosophy of science (Röhl [2012]; Keeling et al. [2019]). An Oxford dictionary definition of

mechanisms refers to *a natural or established process by which something takes place or is brought about*. More intricate definitions discuss "complex systems producing a behavior", "entities and activities productive of regular changes", "a structure performing a function in virtue of its parts and operations", or the distinction between "correlative property changes" and "activity determining how a correlative change is achieved" (Röhl [2012]).

Abstract definitions can help with generalization across many important types of mechanisms. The schema we propose (Section 5.2) is inspired by such definitions, operationalizing them and making them more concrete, and also simple enough for models and human annotators to identify.

Information extraction from scientific texts: There is a large body of literature on extracting information from scientific papers, primarily in the biomedical sphere. This information often corresponds to very *specific* types of mechanisms, as shown in Table 2.1. Examples include ChemProt (Li et al. [2016]) with mechanisms of chemical-protein regulation, drug interactions in the DDI dataset (Segura Bedmar et al. [2013]), genetic and cellular activities/functions in GENIA (Kim et al. [2013]), semantic roles of clinical entities (Kilicoglu et al. [2011]), PICO interventions and outcomes (Wallace et al. [2016]; Nye et al. [2018]), and computer science methods/tasks in SciERC (Luan et al. [2018]). Such schemas have been used, for example, to extract genomic KBs (Poon et al. [2014]) and automate systematic reviews (Nye et al. [2020]). Our schema draws on these approaches, but with a much broader reach across concepts seen in COVID-19 papers (Table 2.1, Figure 5.2).

An important area in information extraction focuses on *open* concepts, with prominent approaches being Open IE (Etzioni et al. [2008]) and Semantic Role Labeling (SRL; (Carreras and Màrquez, 2005)), which share similar properties and predictions (Stanovsky et al. [2018]). While such methods are intended to be domain independent, they perform significantly worse in the scientific domain (Groth et al. [2018]). Kruiper et al. [2020] developed a multi-stage process to post-process Open IE outputs, involving trained models and humans to find a balance between generic and fine-grained clusters of relation arguments and omitting noisy clusters.

In contrast, we introduce a unified schema that enables annotating a dataset of mechanism relations between free-form spans and training IE models to automatically generalize across diverse relation types.

Our schema is also related broadly to the task of training reading comprehension models on procedural texts describing scientific processes (such as short paragraphs written by crowd workers to explain photosynthesis in simple language; (Dalvi et al., 2018)). Our representation of scientific texts in terms of a graph of causal relations can potentially help infer processes across science.

COVID-19 IE: Recent work (Verspoor et al. [2020a]) has focused on extracting information from the CORD-19 corpus (Wang et al. [2020b]). PICO concepts are extracted and visualized in an exploratory interface in the COVID-SEE system (Verspoor et al. [2020b]). In (Wang et al. [2020a]), genes, diseases, chemicals and organisms are extracted and linked to existing biomedical KBs with information such as gene-disease relations. Additional relations based on the GENIA schema are extracted from the text. To address the novel COVID-19 domain, the schema is enriched with new entity types such as viral proteins and immune responses.

In this dissertation, we focus on a more general schema that captures diverse concepts appearing in literature related to COVID-19, an emerging domain with novel concepts coming from many fields and subfields. The mechanism KG we construct includes—as a subset —diverse biomolecular and clinical information (such as chemical-disease relations) as part of a general mechanism schema.

Chapter 3

Math word problem solving

3.1 Overview

Answering math word problems poses unique challenges for logical reasoning over implicit or explicit quantities expressed in text. Math word-problem solving requires extraction of salient information from natural language narratives. Automatic solvers must transform the textual narratives into executable meaning representations, a process that requires both high precision and, in the case of story problems, significant world knowledge.

As shown by the geometry question in Figure 3.1, math word problems are generally narratives describing the progress of actions and relations over some entities and quantities. The operation program underlying the problem in Figure 3.1 highlights the complexity of the problem-solving task.

Here, we need the ability to deduce implied constants (π) and knowledge of domain-specific formulas (area of the square).

We introduce a new operation-based representation language for solving math word problems. We use this representation language to construct MathQA[1], a new large-scale, diverse dataset of

[1]The dataset is available at: `https://math-qa.github.io/math-QA/`

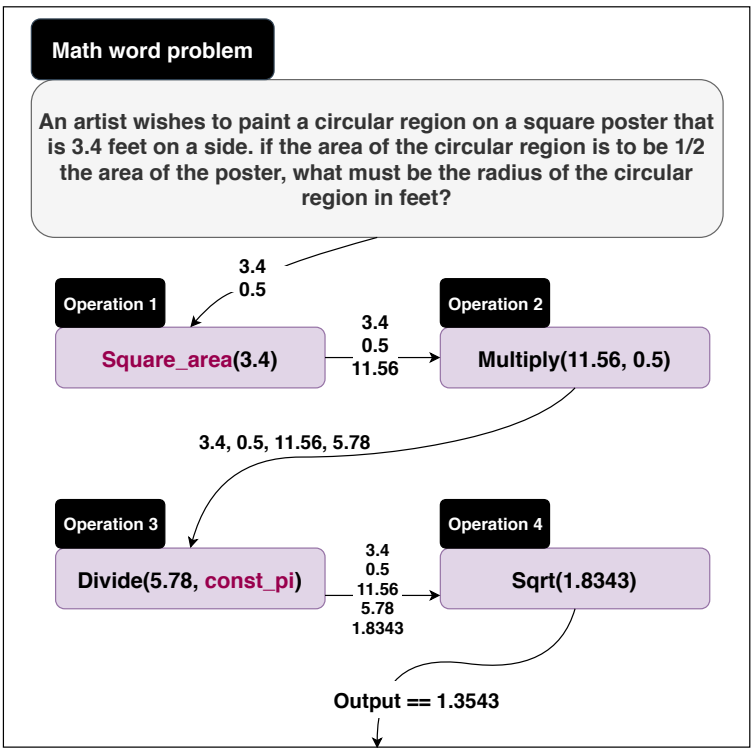


Figure 3.1: Example of a math word problem aligned with representation language by crowd-sourced annotation

37k English multiple-choice math word problems covering multiple math domain categories by modeling operation programs corresponding to word problems in the AQuA dataset (Ling et al. [2017]). We introduce a neural model for mapping problems to operation programs with domain categorization.

Most current datasets in this domain are small in scale (Kushman et al. [2014a]) or do not offer precise operational annotations over diverse problem types (Ling et al. [2017]). This is mainly due to the fact that annotating math word problems precisely across diverse problem categories is challenging even for humans, requiring background math knowledge for annotators. Our representation language facilitates the annotation task for crowd-sourcing and increases the interpretability of the proposed model.

Our sequence-to-program model with categorization trained on our MathQA dataset outperforms previous state-of-the-art model on the AQuA test set in spite of the smaller training size.

These results indicate the superiority of our representation language and the quality of the formal annotations in our dataset. Our model achieves competitive results on MathQA, but is still lower than human performance indicating that the dataset poses new challenges for future research. Our contributions are as follows:

- We introduce a large-scale dataset of math word problems that are densely annotated with operation programs
- We introduce a new representation language to model operation programs corresponding to each math problem that aim to improve both the performance and the interpretability of the learned models.
- We introduce a neural architecture leveraging a sequence-to-program model with automatic problem categorization, achieving competitive results on our dataset as well as the AQuA dataset.

3.2 Representing Math Word Problems

A math word problem consists of a narrative that grounds mathematical formalisms in real-world concepts. Solving these problems is a challenge for both humans and automatic methods like neural network-based solvers, since it requires logical reasoning about implied actions and relations between entities. For example, in Figure 3.2, operations like addition and division are not explicitly mentioned in the word problem text, but they are implied by the question. As we examine the context of a math word problem, we have to select arguments for operations based on which values are unimportant for solving the problem and which are salient. In Figure 3.2, the numeric value "100" appears in the context but does not appear in the underlying equation.

By selecting implied operations and arguments, we can generate a program of intermediate steps for solving a math word problem. Each step involves a mathematical operation and its related arguments. In Figure 3.2, there are three addition operations and one division. As illustrated in the figure, operations can be dependant to the previous ones by the values they use as arguments.

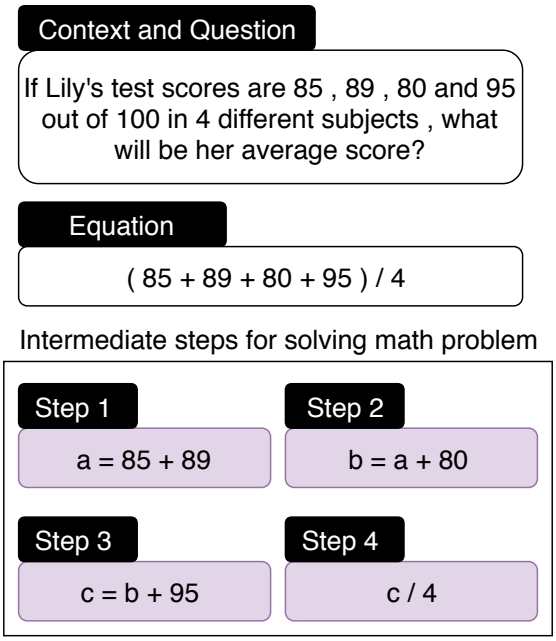


Figure 3.2: Example of a math word problem with its underlying equation and intermediate steps for problem-solving

Every math word problem can be solved by sequentially executing these programs of dependent operations and arguments.

We define formalisms for expressing these sequential operation programs with a domain-aware representation language. An operation program in our representation language is a sequence with n operations. The general form is shown below. Each operation o_i takes in a list of arguments $\mathbf{a}$ of length i:

$$o_1(\mathbf{a}_1)o_2(\mathbf{a}_2)...o_n(\mathbf{a}_n) \tag{3.1}$$

Given this general definition, the problem in Figure 3.2 has the following representation[2]:

$$\begin{aligned} &\text{add}_1(85, 89)\text{add}_2(174, 80) \\ &\text{add}_3(254, 95)\text{divide}_4(349, 4) \end{aligned} \tag{3.2}$$

Our representation language consists of 58 operations and is designed considering the following

[2]Here the arguments 174, 254 and 349 are the outputs of operations 1, 2 and 3 respectively.

objectives:

- Correctness → Operation programs should result in the correct solution when all operations are executed.

- Domain-awareness → Operation problems should make use of both math knowledge and domain knowledge associated with subfields like geometry and probability to determine which operations and arguments to use.

- Human interpretability → Each operation and argument used to obtain the correct solution should relate to part of the input word problem context or a previous step in the operation program.

Learning logical forms has led to success in other areas of semantic parsing (Cheng et al. [2017]; Zelle and Mooney [1996]; Zettlemoyer and Collins [2007, 2005]) and is a natural representation for math word problem-solving steps. By augmenting our dataset with these formalisms, we are able to cover most types of math word problems[3]. In contrast to other representations like simultaneous equations, our formalisms ensure that every problem-solving step is aligned to a previous one. There are three advantages to this approach. First, we use this representation language to provide human annotators with clear steps for how a particular problem should be solved with math and domain knowledge. Second, our formalisms provide neural models with a continuous path to execute operations for problems with systems of equations, instead of forcing models to align equations before problem solving. This reduces the possibility of intermediate errors being propagated and leading to a incorrect solution. Finally, by having neural models generate a solution path in our representation language before computing the final solution, we are able to reconstruct the logical hops inferred by the model output, increasing model interpretability.

[3]We omit high-order polynomials and problems where the solutions are entirely nonnumeric.

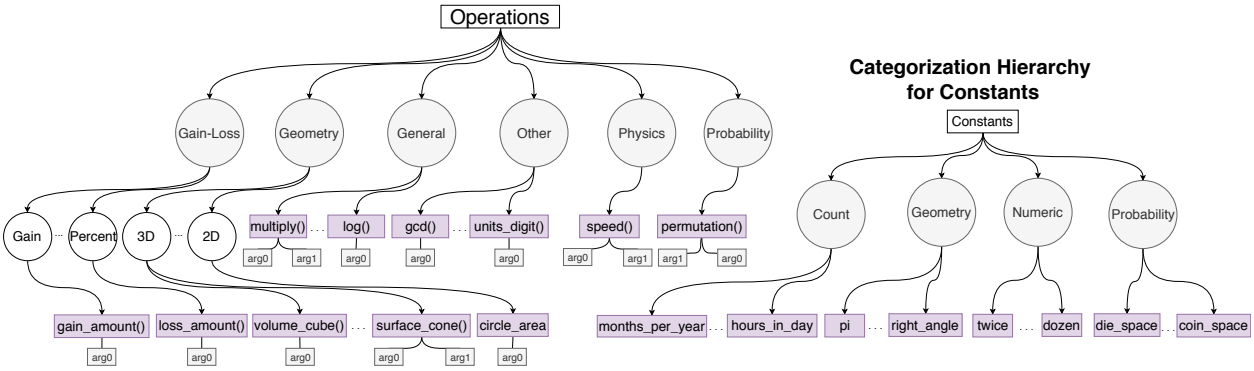


Figure 3.3: Category-based Hierarchies for Operation Formalisms

3.3 Dataset

Our dataset (called MathQA) consists of 37,200 math word problems, corresponding lists of multiple-choice options and aligned operation programs. We use problems in the AQuA dataset and carefully annotate those problems with formal operation programs.

Math problems are first categorized into math domains using term frequencies (more details in Section 3.4.2). These domains are used to prune the search space of possible operations to align with the word problem text. Figure 3.3 shows the category-based hierarchies for operation formalisms.

We use crowd-sourcing to carefully align problems with operation programs (Section 3.3.1). Table 3.1 shows overall statistics of the dataset.[4]

3.3.1 Annotation using Crowd Workers

Annotating GRE level math problems can be a challenging and time consuming task for humans. We design a dynamic annotation platform to annotate math word problems with formal operation programs. Our annotation platform has the following properties: (a) it provides basic math knowl-

[4]We also experimented with an automatic dynamic programming approach to annotation that generates operation programs for problems using numbers in the AQuA rationales. Due to the noise in the rationales, only 61% of those problems pass our human validation. This is mainly due to the fact that the rationales are not complete programs and fail to explicitly describe all important numbers and operations required to solve the problem. To maintain interpretability of operation paths, we did not include automatic annotations from our dataset and focus on operation programs derived by crowdsourcing.

Category	#Prob.	Avg #words	#Vocab	Avg #ops
Geometry	3,316	34.3	1,839	4.8
Physics	9,830	37.3	3,340	5.0
Probability	663	38.9	937	5.0
Gain-Loss	4,377	34.3	1,533	5.7
General	17,796	38.6	6,912	5.1
Other	1,277	31.3	1,425	4.7
All	37,259	37.9	6,664	5.3

Table 3.1: Statistics for our dataset; the total number of operations in the dataset is 58.

edge to annotators, (b) it is dynamic by iteratively calculating intermediate results after an operation submission, and (c) it employs quality control strategies.

Dynamic Annotation Platform: The annotators are provided with a problem description, a list of operations related to the problem category, and a list of valid arguments. They iteratively select operations and arguments until the problem is solved.

- **Operation Selection:** The annotators are instructed to sequentially select an operation from the list of operations in the problem category. Annotators are provided with math knowledge by hovering over every operation and getting the related hint that consists of arguments, formula and a short explanation of the operation.

- **Argument Selection:** After selecting the operation, the list of valid arguments are presented to the annotators to choose from. Valid arguments consist of numbers in the problem, constants in the problem category, and the previous calculations. The annotators are restricted to select only from these valid arguments to prevent having noisy and dangling numbers. After submission of an operation and the corresponding arguments, the result of the operation is automatically calculated and added as a new valid argument to the argument list.

- **Program Submission:** To prevent annotators from submitting arbitrary programs, we enforce restrictions to the final submission. Our platform only accepts programs which include some

numbers from the problem, and whose final calculation is very close to the correct numeric solution.

High Quality Crowd Workers: We dynamically evaluate and employ high-quality annotators through a collection of quality-control questions. We take advantage of the annotation platform in *Figure Eight*.[5] The annotators are randomly evaluated through a pre-defined set of test questions, and they have to maintain an accuracy threshold to be able to continue their annotations. If an annotator's accuracy drops below a threshold, their previous annotations are labeled as untrusted and will be added to the pool of annotations again.

Alignment Validation: To further evaluate the quality of the annotated programs, we leverage a validation strategy to check whether the problems and annotated programs are aligned or not. According to this strategy, at least 2 out of 3 validators should rank the operation program as valid for it to be selected. The validation accuracy is 94.64% across MathQA categories. (Please refer to Appendix A.1.2 for further examples and the setup details.)

3.4 Models

We develop encoder-decoder neural models to map word problems to a set of feasible operation programs. We match the result of the executed operation program against the list of multiple-choice options given for a particular problem. The matching solution is the final model output.

We frame the problem of aligning an operation program with a math word problem as a neural machine translation (NMT) task, where the word problem $\mathbf{x}$ and gold operation program $\mathbf{y}$ form a parallel text pair. The vocabulary of $\mathbf{y}$ includes all possible operations and arguments in our representation language.

[5] https://www.figure-eight.com

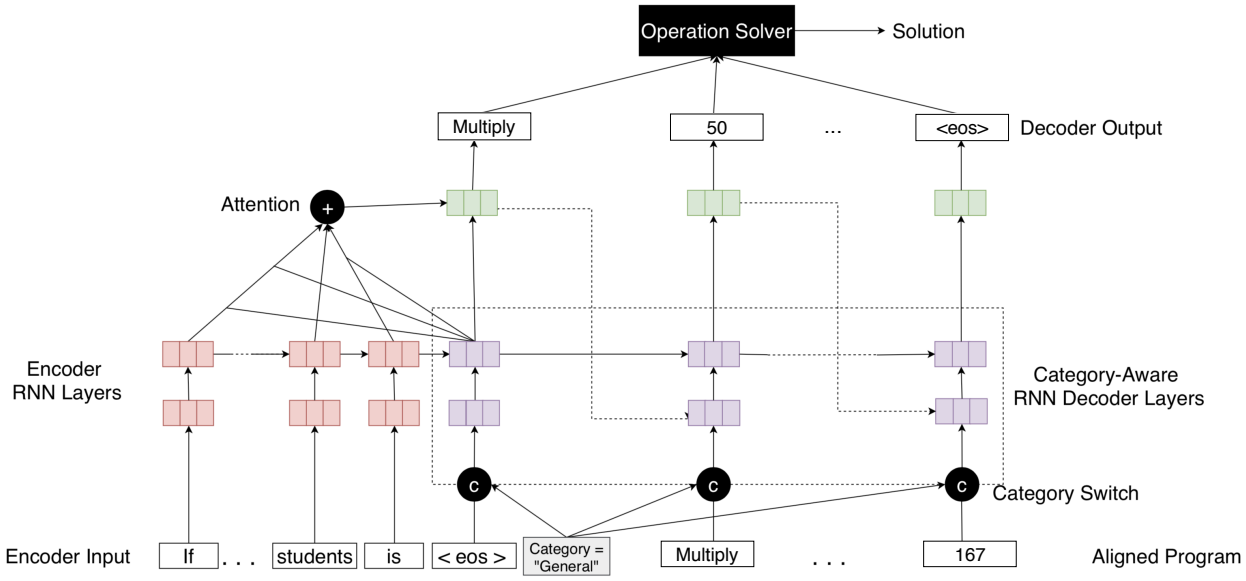


Figure 3.4: Architecture of sequence-to-program model with categorization. Shown with example problem "If the average marks of three batches of 62 , 60 and 45 students respectively is 50 , 55 , 60 , then the average marks of all the students is."

3.4.1 Sequence-to-Program

For our initial sequence-to-program model, we follow the attention-based NMT paradigm of (Bahdanau et al. [2015]; Cho et al. [2014]). We encode the source word problem text $\mathbf{x} = (x_1, x_2, ..., x_M)$ using a bidirectional RNN encoder θ^{enc}. The decoder θ^{dec} predicts a distribution over the vocabulary and input tokens to generate each operation or argument in the target operation program. For our sequence-to-program model vocabulary, we use informed generation, in which the program tokens are generated separately from the vocabulary of operations O or arguments A.

The encoded text is represented by $\mathbf{h}^{enc} = (h_1^{enc}, h_2^{enc}, .., h_M^{enc})$, a sequence of d-dimensional hidden states, where M is the length of the input text. A context vector a_i is computed by taking the weighted sum of the attention model weights $\alpha_{t,i}$ for each timestep $t \in (1, 2, ..., T)$ and each encoder hidden state h_i^{enc}:

$$a_i = \sum_{i=1}^{M} \alpha_{t,i} h_i^{enc}.$$

We compute the d-dimensional decoder hidden state h_i^{dec} using a LSTM recurrent layer:

$$h_i^{dec} = LSTM(h_{i-1}^{dec}, y_{i-1}, a_i) \tag{3.3}$$

At each timestep, we make a prediction for an operator op_i or argument arg_{ik}, where k corresponds to the index of the argument in operator i's argument list. This prediction is conditioned on the previous tokens $(y_1, ..., y_{i-1})$ and the input $\mathbf{x}$ to decode an entire operation program $\mathbf{y} = (y_1, y_2, ..., y_N)$ of length N:

$$P(\mathbf{y}|x) = \prod_{i=1}^{N} P(y_i|y_{<i}, \mathbf{x}) \tag{3.4}$$

$$P(y_i|y_{<i}, \mathbf{x}) = g(f(h_i^{dec}, y_i, a_i)) \tag{3.5}$$

Here f is a 1-layer feed-forward neural network and g is the softmax function. During training time, we minimize the negative log-likelihood (NLL) using the following objective:

$$\mathcal{L}(\theta^{enc}, \theta^{dec}) = -logP(\mathbf{y}|\mathbf{x}; \theta^{enc}, \theta^{dec}) \tag{3.6}$$

At test time, we only observe the input text when predicting operation programs:

$$\hat{\mathbf{y}} = \texttt{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \tag{3.7}$$

3.4.2 Categorized Sequence-to-Program Model

We extend our base sequence-to-program model to integrate knowledge of math word problem domain categories. We modify the RNN decoder layers that compute the decoder hidden state to be category-aware. Here, the category label c is deterministically computed by the category extractor (explained below). It functions as a hard decision switch that determines which set of parameters to

use for the hidden state computation:

$$h_i^{dec} = LSTM_c(h_{i-1}^{dec}, y_{i-1}, a_i) \tag{3.8}$$

The updated objective function from equation (3.6) is shown below:

$$\mathcal{L}(\theta^{enc}, \theta_c^{dec}) = -logP(\mathbf{y}|\mathbf{x}; \theta^{enc}, \theta_c^{dec}) \tag{3.9}$$

The full model architecture is shown in Figure 3.4.

Domain-Specific Category Extraction: We first construct a lexicon of n-grams relating to a specific domain. The lexicon is a list consisting of domain-specific categories and associated n-grams. For each domain category c in the lexicon, we select associated n-grams $\mathbf{n}_c$ that occur frequently in word problems belonging to domain category c, but rarely appear in other domain categories. We compute n-gram frequency f_{pc} as the number of n-grams associated with a category c appearing in the text of a word problem p. We obtain a list of potential categories for p by choosing all categories for which $f_{pc} > 0$, and then assign a category label to p based on which category has the highest n-gram frequency.

3.4.3 Solving Operation Programs

Once a complete operation program has been decoded, each operator in the program is executed sequentially along with its predicted set of arguments to obtain a possible solution. For each word problem p and options o, we generate a beam of the top n decoded operation programs. We execute each decoded program g to find the solution from the list of options $\mathbf{o}$ of the problem. We first choose options that are within a threshold of the executed value of g. We select g as the predicted solution by checking the number of selected options and the minimum distance between the executed value of g and a possible option for p. For the problems in AQuA that do not belong in any category of

MathQA, we randomly choose an option.

3.5 Experimental Setup

3.5.1 Datasets

Our dataset consists of $37k$ problems which are randomly split in $(80/12/8)\%$ training/dev/test problems. Our dataset significantly enhances the AQuA dataset by fully annotating a portion of *solvable* problems in the AQuA dataset into formal operation programs.

Subset	Train	Valid
Unsolvable - No Words	37	0
Unsolvable - Sequence	1,991	4
Unsolvable - Requires Options	6,643	8
Unsolvable - Non-numeric	10,227	14
Duplicates	17,294	0
Solvable	65,991	229
Total	97,467	254

Table 3.2: Full original AQuA solvability statistics.

We carefully study the AQuA dataset. Many of the problems are near-duplicates with slight changes to the math word problem stories or numerical values since they are expanded from a set of 30,000 seed problems through crowdsourcing (Ling et al. [2017]). These changes are not always reflected in the rationales, leading to incorrect solutions. There are also some problems that are not solvable given current math word problem solving frameworks because they require a level of reasoning not yet modeled by neural networks. Sequence problems, for example, require understanding of patterns that are difficult to intuit without domain knowledge like sequence formulas, and can only be solved automatically through brute-force or guessing. Table 3.2 shows a full breakdown of the AQuA dataset by solvability.[6]

[6]There is overlap between unsolvable subsets. For example, a sequence problem may also be a duplicate of another problem in the AQuA dataset.

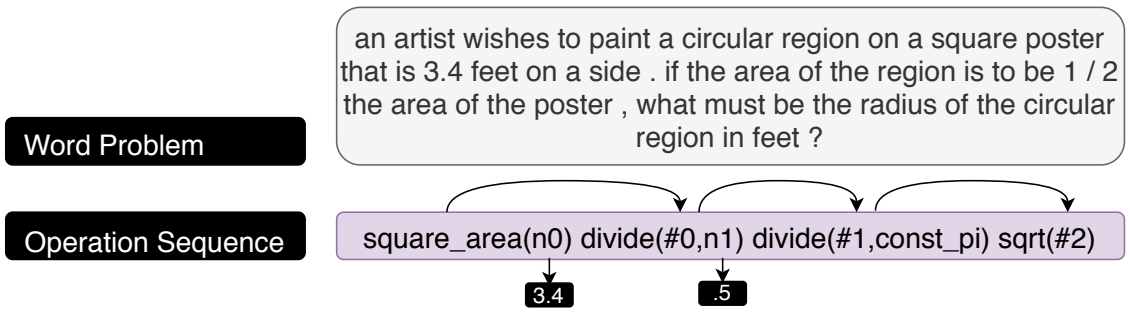


Figure 3.5: Example of an operation program generated by our Seq2prog model with categorization

3.5.2 Annotation Details

We follow the annotation strategy described in Section 3.3 to formally annotate problems with operation programs.[7]

Annotator Agreements and Evaluations: Our expert evaluation of the annotation procedure for a collection of 500 problems shows that 92% of the annotations are valid. Additionally, it has 87% agreement between the expert validation and the crowd sourcing validation task.

Annotation Expansion: The AQuA dataset consists of a group of problems which share similar characteristics. These problems can be solved with similar operation programs. We find closely similar problems, replace numeric values with generic numbers, and expand annotations to cover more problems from the AQuA dataset. For similarity, we use Levenshtein distance with a threshold of 4 words in edit distance.

3.5.3 Model and Training Details

We use the official python implementation of OpenNMT (Klein et al. [2017]). We choose a LSTM-based encoder-decoder architecture. We use Adam optimizer (Kingma and Ba [2015]), and the

[7]We tried two other strategies of showing extra information (rationales or end solutions) to annotators to facilitate solving problems. However, our manual validation showed that annotators mostly used those extra information to artificially build an operation program without reading the problem.

learning rate for training is 0.001. The hidden size for the encoder and decoder is set to $d = 100$. Both the encoder and decoder have 2 layers. The word embedding vectors are randomly initialized. At inference time, we implemented a beam search with beam size of 200 for AQuA and 100 for MathQA.

The program vocabulary consists of the operations O in our representation language and valid arguments A. For valid arguments, we do not use their actual values since the space is very large. Instead, we keep a list of numbers according to their source. Constants are predefined numbers that are available to all problems. Problem numbers are added to the list according to their order in the problem text. Calculated numbers in the intermediate steps are added to the list according to the operation order.

3.6 Experimental Results

3.6.1 Results

Table 3.3 compares the performance of our sequence-to-program models trained on MathQA with baselines on MathQA and AQuA test sets. The base model is referred to as "Seq2prog," while our model with categorization is "Seq2prog + cat." For accuracy, the performance was measured in terms of how well the model would perform on an actual math test.

We observe improvement for our "Seq2prog + cat" model despite the fact that our training data is proportionally smaller than the AQuA dataset, and our model is much simpler than the state-of-the-art model on this dataset. This indicates the effectiveness of our formal representation language to incorporate domain knowledge as well as the quality of the annotations in our dataset.

Model	MathQA	AQuA
Random	20.0	20.0
AQuA Model	-	36.4
Seq2prog	51.9	33.0
Seq2prog + cat	**54.2**	**37.9**

Table 3.3: Experimental results for accuracy on our MathQA and AQuA test sets

3.6.2 Analysis

Qualitative Analysis: Table 3.4 and Figure 3.5 show some examples of problems solved by our method. We analyzed 50 problems that are solved wrongly by our system on the MathQA dataset. Table 3.5 summarizes four major categories of errors.

The most common type of errors are problems that need complicated or long chain of mathematical reasoning. For example, the first problem in Table 3.5 requires reasoning that goes beyond one sentence. Other errors are due to limitations in our representation language. For example, the second problem in Table 3.5 requires the *factorization* operation which is not defined in our representation language. Future work can investigate more domains of mathematics such as logic, number factors, etc. Some errors are due to the slightly noisy nature of our categorization strategy. For example, the third problem in Table 3.5 is mistakenly categorized as belonging to *physics* domain due to the presence of words *m, cm, liter* in the problem text, while the correct category for the problem is *geometry*. The final category of errors are due to problems that do not have enough textual context or erroneous problems (e.g., fourth problem in Table 3.5).

Impact of Categorization: Table 3.3 indicates that our category-aware model outperforms the base model on both AQuA and MathQA datasets. The gain is relatively small because the current model only uses categorization decisions as hard constraints at decoding time. Moreover, the problem categorization might be noisy due to our use of only one mathematical interpretation for each domain-specific n-gram. For example, the presence of the words “square” or “cube” in the text of a math word problem indicate that the word problem is related to the geometry domain, but these

Problem : A rectangular field is to be fenced on three sides leaving a side of 20 feet uncovered. if the area of the field is 10 sq. feet, how many feet of fencing will be required? **Operations :**`divide(10,20), multiply(#0, const_2), add(20, #1)`
Problem : How long does a train 110m long running at the speed of 72 km/hr takes to cross a bridge 132m length? **Operations :**`add(110, 132), multiply(72, const_0.2778), divide(#0, #1), floor(#2)`

Table 3.4: Problems solved correctly by Seq2prog+cat model.

Error type	**Problem**
Hard problems (45%)	Jane and Ashley take 8 days and 40 days respectively to complete a project when they work on it alone. They thought if they worked on the project together, they would take fewer days to complete it. During the period that they were working together, Jane took an eight day leave from work. This led to Jane' s working for four extra days on her own to complete the project. How long did it take to finish the project?
Limitation in representation language (25%)	How many different positive integers are factors of 25?
Categorization errors (12.5%)	A cistern of capacity 8000 litres measures externally 3.3 m by 2.6 m by 1.3 m and its walls are 5 cm thick. The thickness of the bottom is:
Incorrect or insufficient problem text) (17.5%)	45 x ? = 25 % of 900

Table 3.5: Examples of mistakes made by our system. The reason of the errors are underlined.

unigrams can also refer to an exponential operation (n^2 or n^3).

To measure the effectiveness of our categorization strategy, we used human annotation over 100 problems. The agreement between human annotators is 84% and their agreement with our model is 74.5%. As a future extension of this work, we would like to also consider the context in which domain-specific n-grams appear.

Discussions: As we mentioned in section 3.2, the continuous nature of our formalism allows us to solve problems requiring systems of equations. However, there are other types of word problems that are currently unsolvable or have multiple interpretations leading to multiple correct solutions. While problems that can only be solved by brute-force instead of logical reasoning and non-narrative problems that do not fit the definition of a math word problem (in Table 3.2 these appear as "no word") are removed from consideration, there are other problems that are beyond the

scope of current models but could pose an interesting challenge for future work. One example is the domain of sequence problems. Unlike past word problem-solving models, our models incorporate domain-specific math knowledge, which is potentially extensible to common sequence and series formulas.

3.7 Summary

In this work, we introduced a representation language and annotation system for large-scale math word problem-solving datasets that addresses unwanted noise in these datasets and lack of formal operation-based representations. We demonstrated the effectiveness of our representation language by transforming solvable AQuA word problems into operation formalisms. Experimental results show that both our base and category-aware sequence-to-program models outperform baselines and previous results on the AQuA dataset when trained on data aligned with our representation language. Our representation language provides an extra layer of supervision that can be used to reduce the influence of statistical bias in datasets like AQuA. Additionally, generated operation programs like the examples in figure 3.5 demonstrate the effectiveness of these operation formalisms for representing math word problems in a human interpretable form.

The gap between the performance of our models and human performance indicates that our MathQA still maintains the challenging nature of AQuA problems. As a future work, we suggest an extension to the representation language and models to cover unsolvable problems, including sequence and high-order polynomial problems.

Chapter 4

Procedural Reading Comprehension

4.1 Overview

Procedural text describes how entities (e.g., `fuel`, `engine`) and their attributes (e.g., `locations`) change throughout a process (e.g., a scientific process or cooking recipe). Procedural reading comprehension is the task of answering questions about the underlying process described in the text (Figure 4.1). This task, in turn, requires inferring attributed of the entities in the process, and their transitions, which might only be implicitly mentioned. For instance, in Figure 4.1, the `creation` of the `mechanical energy` in the `alternator` can be inferred from the second and third sentences.

Full understanding of a procedural text requires capturing the interplay between all the components of the process: the affected entities, their attributes and their transitions. Recent works in understanding procedural text develop domain-specific models for tracking entities in scientific processes (Mishra et al. [2018]) or cooking recipes (Bosselut et al. [2018]). More recently, Gupta and Durrett [2019a] leverage pre-trained language models to obtain general entity-aware representations of procedural text, and predict entity transitions from a set of pre-defined classes independent of entity attributes. Pre-defining the set of entity states limits the general applicability of the model, however, as entity attribute values might be arbitrary spans of text. Moreover, entity attributes can

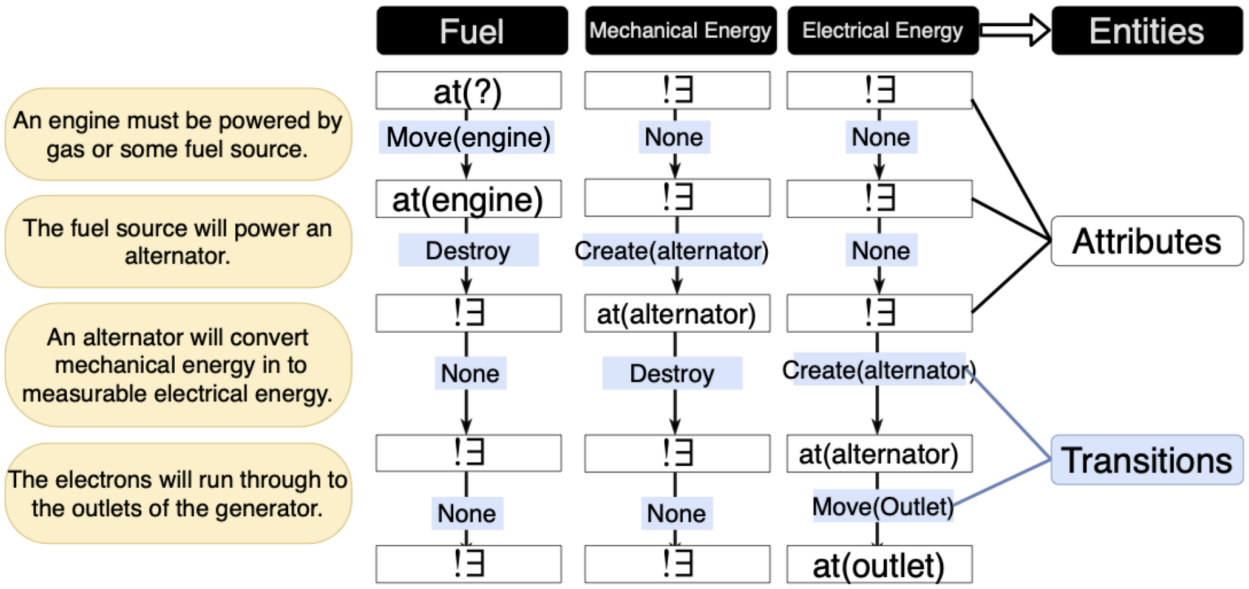


Figure 4.1: Example of a procedural text and the predicted attributes and transitions for each entity. Procedural reading comprehension is the task of answering questions about the underlying process. Sample questions from ProPara include: 'What is the process's input?', 'What is the process's output?', 'What is the location of the entity?'.

be used for tracking entity state transitions. For example, in Figure 4.1, the location of `fuel` can be effectively inferred from text as `engine` without the explicit mention of the `movement` transition in the first sentence. In addition, the phrase `converted` in the third sentence gives rise to predicting two transition actions of `destruction` of one type of `energy` and `creating` the other type.

In this work, we introduce a general formalism to represent procedural text, and develop an end-to-end neural procedural reading comprehension model. Our formalism represents entities, their attributes, and their transitions across time, and the model jointly identifies these attributes and transitions using a dynamic contextual encoding of the text. Our model computes an attribute-aware representation of the procedural text at a certain timestep by leveraging distributions over predicted attribute values, either as a span of text or from a pre-defined set of classes. Then, it pairs this attribute-aware encoding with an entity-aware representation to predict state transitions for entities, thereby capturing the dynamic nature of the entities in the contextual encoding of the process.

Our experiments show that our method achieves state of the art results across multiple tasks

evaluated in the PROPARA dataset (Mishra et al. [2018]) to track entity attributes and their transitions in scientific processes. Additionally, a simple variant of our model achieves state of the art results in the NPN-COOKING (Bosselut et al. [2018]) dataset.

Our contributions are three-fold:

- We present a general formalism to model procedural text, which can be adapted to different domains.
- We develop DYNAPRO, an end to end neural model that jointly and consistently predicts entity attributes and their state transitions, leveraging pre-trained language models.
- We show that our model can be adapted to several procedural reading comprehension tasks using the entity-aware and attribute-aware representations, achieving state of art results on several diverse tasks.

4.2 Procedural Text Representation

Procedural text consists of a sequence of sentences describing how entities and their attributes change throughout a process. We introduce a general formalism to represent a procedural text:

$$p = (E, A, T), \tag{4.1}$$

where E is the list of entities participating in the process, A is the list of entity attributes, and T is the list of transitions. We formulate the model's input to be a combination of a query about an entity attribute and a partial context of the procedural text.

Entities are the participants in a process. For example, in the scientific processes in PROPARA, entities might include elements such as `energy`, `fuel`, etc. In the cooking recipe domain, the

entities could be ingredients such as `milk`, `flour`, etc. Entities could be given a priori based on the task (e.g., PROPARA) or they could be inferred from the context (e.g., math word problems).

Attributes are entity properties that can change over time. We model attributes as functions $\texttt{Attribute(e)} = \texttt{val}$, that assign a value `val` to an attribute of the entity e. The entity state at each time is derived by combining all the attribute values of that entity. Attribute values can be either spans of text or can be derived from a pre-defined set of classes. For example, in PROPARA, an important attribute of an entity is its `location`, which can be a span of text. The NPN-COOKING dataset introduces several attributes (such as `shape` and `cookedness`) for each ingredient. Example attributes addressing the entities in PROPARA are modeled as follows:

$$\texttt{exists(e)} = \{\texttt{nowhere}, \texttt{unknown}, \texttt{span\_of\_text}\}$$

$$\texttt{at\_loc(e)} = \texttt{l} \rightarrow \texttt{ Assigns the location l to entity e}$$

Transitions capture changes in the entity states. More specifically, transitions indicate how entity attributes change over time. We model each transition with an action name and a list of arguments that include the entity and some attribute values. For example, PROPARA consists of four transition types : $\texttt{Create(e, loc)}$, $\texttt{Destroy(e)}$, $\texttt{None(e)}$ and $\texttt{Move(e, loc)}$.

4.3 Model

We introduce DYNAPRO, depicted in Figure 4.2, an end-to-end neural architecture that jointly predicts entity attributes and their transitions. DYNAPRO first obtains the representation of the procedural text corresponding to an entity at each time step (Section 4.3.2). It then identifies entity attributes for current and previous time steps (Section 4.3.3) and uses them to develop an attribute-aware representation of the procedural context (Section 4.3.4). Finally, DYNAPRO uses the entity-aware and attribute-aware representations to predict transitions that happen at that time step

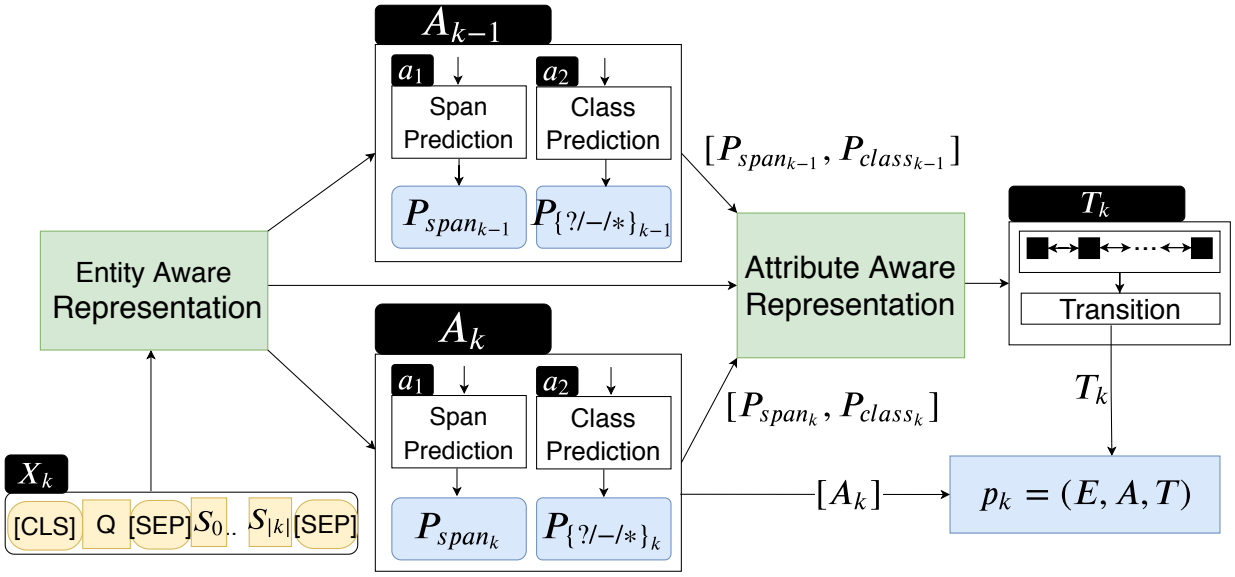


Figure 4.2: DYNAPRO takes the procedural context X_k as input and predicts attributes A_{k-1}, A_k and transitions T_k at each time step k. $P_{\{?,-,*\}}$ indicates the probability of the location type among `nowhere`, `unknown`, and `span_of_text` respectively.. The model uses the changes in attribute values from time steps $k-1$ to k to predict transitions.

(Section 4.3.5).

4.3.1 Task Reformulation

Given a procedural text $\langle S_0 \dots S_k \dots S_T \rangle$ and an entity e, DYNAPRO encodes procedural context X_k at each time step k and obtains the entity-aware representation vector $R_k(e)$. The procedural context is formed by concatenating the query containing the entity name, and a fragment of the procedural text. The entity name and the query are included in the procedural context to construct an entity-aware representation of the context. Since entity attributes are changing throughout the process, we form the context at each step k by truncating the procedural text up to the k^{th} sentence. More formally, the procedural context is defined as:

$$X_k(e) = [cls]Q_e[sep][C_i]S_0 \dots S_k[sep], \tag{4.2}$$

where $\langle S_0 \ldots S_k \rangle$ is the fragment of the procedural text up to the k^{th} sentence, Q_e is the entity-aware query (e.g., "Where is e?"), $[C_i]$ includes tokens that are reserved for attribute value classes (e.g., `nowhere`, `unknown`), and $[cls]$ and $[sep]$ are special tokens to capture sentence representations and separators. Note that the input samples for any paragraph are constructed per sentence and entity. So the complexity of the input data is $|S| * |E|$ per procedural text where $|S|$ and $|E|$ are the average number of sentences and entities in each procedural text respectively.

4.3.2 Entity-aware Representation

DYNAPRO then uses a pre-trained language model to encode the procedural context $X_k(e)$ from Equation 4.2 and returns the entity-aware representation $R_k(e) = BERT(X_k(e))$, where $BERT(w_i)$ is the embedding of the i-th token from the transformer layer. Hereinafter, we will remove the argument e from equations for ease of notation.

4.3.3 Attribute Identification

DYNAPRO identifies attribute values for each entity from the entity-aware representation $R_k(e)$ by jointly predicting attribute values from a pre-defined set of classes or extracting them as a text span.

Class Prediction: Certain attribute values are predicted from a set of pre-defined classes. For instance, the `existence` attribute of an entity is one of $\{$`nowhere`, `unknown`, `span_of_text`$\}$. The distribution P_{class_k} over pre-defined attribute values is predicted from the entity-aware representation R_k:

$$P_{class_k} = \text{softmax}(f_{\theta_1}(g(R_k))), \tag{4.3}$$

where g is a non-linear function, f is a linear function and θ_1 are learnable parameters.

Span Prediction: Defining all attribute values a priori limits the general applicability of the procedural text understanding model. Some attribute values are only mentioned within a `span_of_text`.

For example, the full set of `locations` an entity can be in may be difficult to explicitly pre-define into classes, but may be easily searchable in text. For span prediction, we follow the standard procedure of phrase extraction in reading comprehension (Seo et al. [2016]) that predicts two probability distributions over start and end tokens of the span:

$$\begin{aligned} P_{span_k} &= [P_{start_k}, P_{end_k}] \\ P_{start_k} &= \text{softmax}(f_{\theta_2}(g(R_k))) \\ P_{end_k} &= \text{softmax}(f_{\theta_3}(g(R_k))), \end{aligned} \tag{4.4}$$

where g is a non-linear function, f is a linear function and θ_2 and θ_3 are learnable parameters used to compute the probability distributions over start and end tokens of the span.

In order to capture the transitions of entity attributes, our model captures attributes for time steps $k-1$ and k given a procedural context X_k. More specifically, we use Equations 4.3 and 4.4 to compute the probability distributions $P_{class_{k-1}}$, $P_{span_{k-1}}$, P_{class_k} and P_{span_k} for both time steps k and $k-1$ at given timestep k.

4.3.4 Attribute-aware Representation

For each entity e at each time step k, DYNAPRO computes attribute-aware representations R_{a_k} of the context by encoding entities and their attributes using the predicted distributions P_{span_k} and P_{class_k}. The intuition is to assign higher weight in the contextual representation to the tokens w corresponding to the attribute value of the entity at time step k.

$$R_{a_k} = \sum_{class} (R_k.P_{class_k} \cdot m_{class}) \cdot P_{span_k}(w), \tag{4.5}$$

where $class \in \{\texttt{nowhere}, \texttt{unknown}, \texttt{span}\}$ are the pre-defined classification of attributes, and P_{class_k} and P_{span_k} denote the probability distributions of attribute values over pre-defined classes and the span of text respectively (as calculated using Equations 4.3 and 4.4). m_{class} is a vector that

masks out the input tokens that do not correspond to a specific class.

Finally, we model the flow of the context by concatenating attribute-aware representations at time steps k and $k-1$ as:

$$R_{a_{k-1:k}} = [R_{a_k}, R_{a_{k-1}}]. \tag{4.6}$$

4.3.5 Transition Classification

DYNAPRO predicts attribute transitions from entity-aware and attribute-aware representations. In order to make smooth transition predictions and avoid redundant transitions we include a Bi-LSTM layer before the classification of the transition.

$$\begin{aligned} R_{seq_k} &= \text{LSTM}(h, [R_k, Ra_{k-1:k}]) \\ P_{transition_k} &= \text{softmax}(f_{\theta_4}(g([R_{seq_k}]))), \end{aligned} \tag{4.7}$$

where h is the hidden vector of sequential layer, θ_4 are learnable parameters and R_{seq_k} is the output of the sequential layer.

4.3.6 Inference and Training

Training: Our model is trained end-to-end by optimizing the loss function below:

$$loss_{total} = (loss_{span} + loss_{class})_{k-1} + (loss_{span} + loss_{class})_k + loss_{transition_k} \tag{4.8}$$

Each loss function is defined as a cross entropy loss. $(loss_{span}, loss_{class})_k$ and $loss_{transition_k}$ are the losses of attribute prediction and the transition prediction modules at time step k, respectively.

Inference: At each time step k, the attributes A_k and transitions T_k are predicted given P_{span_k}, P_{class_k}, and $P_{transition_k}$. The final output of the model consists of two sets of predictions, the

attributes $A_{0...K}$ and transitions $T_{0...K}$ which are combined to track entities throughout a process given a task-specific objective (see Section 4.4.3 for further details).

4.4 Experiments and Results

4.4.1 Datasets

We evaluate our model over the PROPARA dataset introduced by (Mishra et al. [2018]). This dataset contains over 400 manually-written paragraphs of scientific process descriptions. Each paragraph includes average of 4.17 entities and 6 sentences. The vocabulary size of is 2500. The entities are extracted by experts and the transitions are annotated by crowd-workers.

We also evaluate our model on the NPN-COOKING dataset introduced by (Bosselut et al. [2018]). This corpus contains ∼65k cooking recipes. Each recipe comes with a set of ingredients tracked during the process. Training samples are heuristically annotated for attributes and state transitions by string matching, and dev/test samples are annotated by crowd-workers. We randomly sample from the training recipes that have ingredients whose `location` attribute is changed.

4.4.2 Tasks and metrics

We evaluate DYNAPRO on three tasks in PROPARA and one task in NPN-COOKING.

Document-level Predictions: This task was introduced by Tandon et al. [2018] and evaluates four different questions per entity and process. These questions involve identifying whether the entity is an (1) input or (2) output of the process, and identifying the (3) moves and (4) conversions of the entity in the process. The final metrics reported for this evaluation are the average precision, recall and F1 score across all four questions.

Sentence-level Predictions: This task was introduced by Mishra et al. [2018] and considers three categories of questions. The first, $\mathbf{Cat-1}$, asks if a specific entity is `created`/`destroyed`/`moved` in the process. $\mathbf{Cat-2}$ asks the time step at which an entity is `created`/`destroyed`/`moved`. Finally, $\mathbf{Cat-3}$ asks about the location where an entity is `created`/`destroyed`/`moved`. The evaluation metric calculates the score of all transitions for each question and reports the macro- and micro-average of the scores among three question types.

Action Dependencies: This task was recently introduced by Mishra et al. [2019] to check whether the actions predicted by a model influence future events in the procedural paragraph. The metrics reported for this task are the precision, recall, and F1 scores of the dependency links between events averaged over all paragraphs.

Location Prediction in Recipes: The task is to identify the location of different entities in cooking recipes. In this domain, the list of attributes are fixed. We evaluate by measuring the change in location (Bosselut et al. [2018]) and computing F1 and accuracy in attribute prediction.

4.4.3 Implementation Details

We use the official implementation of $BERT_{base}$ from the `huggingface` library (Wolf et al. [2019]). The learning rate for training is $3e{-}5$ and the training batch size is 8. The hidden size of the sequential layer is set to 1000 and 200 for class prediction and transition prediction, respectively.

We use the predicted A_{k-1} to initialize the attribute of timestep 0, and at any other timestep, we use the A_k predictions for finding the value of an attribute at timestep k. In the sentence level evaluation task introduced in (Mishra et al. [2018]), the consistency is not required. The inference phase for this task only uses the attribute predictions. For the document-level predictions, we construct the final predictions by favoring the transition predictions. In case of inconsistency (i.e., there is no valid attribute prediction to support the transition), we refer to the attribute value to

	Sentence-Level					Document Level			Action Dependency		
Model	**Cat-1**	**Cat-2**	**Cat-3**	**Ma-Avg**	**Mi-Avg**	**P**	**R**	**F1**	**P**	**R**	**F1**
ProLocal	62.7	30.5	10.4	34.5	34.0	**77.4**	22.9	35.3	24.7	18.0	20.8
EntNet	51.6	18.8	7.8	26.1	26.0	50.2	33.5	40.2	32.8	38.6	35.5
QRN	52.4	15.5	10.9	26.3	26.5	55.5	31.3	40.0	32.6	30.3	31.4
ProGlobal	63.0	36.4	35.9	45.1	45.4	46.7	52.4	49.4	43.4	37.0	39.9
KG-MRC	62.9	40.0	38.2	47.0	46.6	64.5	50.7	56.8	46.5	39.5	42.7
NCET	70.6	44.6	41.3	52.2	52.3	64.2	53.9	58.6	-	-	-
NCET + ELMo	**73.7**	47.1	41.0	53.9	54.0	67.1	**58.5**	62.5	50.4	28.6	36.5
ET_{BERT}	73.6	**52.6**	-	-	-	-	-	-	-	-	-
XPAD	-	-	-	-	-	70.5	45.3	55.2	62.0	32.9	43.0
DYNAPRO	72.4	49.3	**44.5**	**55.4**	**55.5**	75.2	58.0	**65.5**	**64.9**	**32.9**	**43.7**

Table 4.1: Results comparing DYNAPRO to prior state of the art methods on sentence-level, document-level and Action Dependency tasks of PROPARA (test set).

Model	**F1**	**Accuracy**
NPN	35.1	51.3
KG-MRC	-	51.6
DYNAPRO	**36.3**	**62.9**

Table 4.2: F1 and accuracy on the location prediction task in NPN-COOKING.

deterministically infer the transition.

To adapt the results of DYNAPRO to identify action dependencies, we postprocess the results using similar heuristics described in the original task. To adapt DYNAPRO to the NPN-COOKING dataset, we use a 243-way classification to predict attributes because the attributes are known apriori.

4.4.4 Results and Analyses

Table 4.1 and Table 4.2 compare DYNAPRO with previous models (detailed in Section 2.2) designed for the PROPARA and NPN-COOKING datasets. As shown in the tables, DYNAPRO outperforms these models in most of the evaluations.

Document-level Task: We observe the most significant gain (3% absolute in F1) on the document-level task, indicating the model achieves a better global understanding of the procedural

Question	P	R	F1
Inputs	93.0	74.0	82.4
Outputs	81.2	91.5	86.0
Conversions	77.5	58.3	66.5
Move	53.7	47.8	50.6

Table 4.3: Precision, Recall and F1 of DYNAPRO on each question in PROPARA document-level predictions.

text by making joint predictions of entity attributes and transitions. Overall, in most document-level tasks, DYNAPRO predicts transitions with higher precision. In addition, Table 4.3 shows DYNAPRO's performance on each individual question. The `movement` question achieves the lowest score whereas the `input`/`output` questions are correctly answered more frequently. This pattern is likely due to the fact that `movements` require two span predictions while `create`/`destroy` transitions, which affect the `input`/`output` questions, are less complex.

Sentence-level Task: DYNAPRO outperforms the state-of-the-art models on the macro- and micro-average of the three question scores, and gives comparable results to previous work on each individual question type. We note that ET_{BERT} (Gupta and Durrett [2019a]) only predicts actions (`Create`, `Destroy`, `Move`), but fails to predict location attributes as spans. DYNAPRO obtains a good performance on the $\mathbf{Cat-1}$ and $\mathbf{Cat-2}$ questions while also learning to predict answers for questions with more complex structure. We carefully analyze the prediction scores of different transitions (`Create`, `Destroy`, `Move`) for each category. In $\mathbf{Cat-1}$, our model is better than ET_{BERT} for `destroy` transitions, but is worse for `movement` and `create` transitions. In $\mathbf{Cat-2}$, our model is better than ET_{BERT} for `destroy` and `create` transitions, but is worse than ET_{BERT} for `movement` transitions. Note that our model predicts transitions between time steps by evaluating changes in span predictions, while ET_{BERT} directly predicts the transition class. Directly predicting transitions results in more accurate predictions for `movement` transitions because many of them are explicitly mentioned in the text, but it does not identify the exact locations of `movements`. On the other hand, most `destroy` and `create` transitions are not explicitly mentioned in the text, and

using our system to identify changes in entity classes results in more accurate predictions.

Action Dependency: DYNAPRO outperforms all previous work with F1 score of 43.7. Note that XPAD (Mishra et al. [2019]) explicitly favors predicting state changes that result in dependencies across steps. In contrast, DYNAPRO is only optimized to track entities.

Location Prediction in Recipes: The NPN-COOKING dataset of (Bosselut et al. [2018]) contains a large number of cooking recipes whose ingredients can be mapped to various attributes. (e.g., `location`, `cookedness`, etc.) Table 4.2 shows that a simple variant of DYNAPRO achieves the best performance at predicting the `locations` of ingredients, showcasing the importance of procedural text encoding over time.

4.4.5 Ablation Studies and Analyses

In order to better understand the importance of DYNAPRO's components, we evaluate different variants of DYNAPRO on the document-level task of the PROPARA dataset:

(A.1) **No class prediction** – the model only uses span predictions.

(A.2) **No transition classification** – the model does not include transition classification.

(A.3) **No span prediction** – any token in the span can be predicted using a classifier over the document's words along with the pre-defined class values.

(A.4) **No attribute-aware representation** – the model only considers entity-aware representations in Equation 4.7 for transition predictions.

(A.5) **CLS instead of attribute-aware representation** – the model uses the `[CLS]` encoding of R_k instead of the attribute-aware representation R_{a_k}.

(A.6) **No sequential modeling** – the model removes the sequential smoothing of the predicted transitions by removing the LSTM from Equation 4.7.

(A.7) **Full procedural input** – uses the full text of the procedure instead of the truncated text X_k at time step k.

Ablation		F1
Full model (DYNAPRO)		**71.9**
(A.1)	No class prediction	53.8
(A.2)	No transition prediction	66.3
(A.3)	No span prediction	55.1
(A.4)	No attribute-aware representation	69.5
(A.5)	CLS instead of attribute-aware representation	70.9
(A.6)	No sequential modeling in transition module	68.8
(A.7)	Full procedural input	61.0

Table 4.4: Ablation study of different components in DYNAPRO by comparing F1 score on PROPARA **Document Level task** (development set).

Table 4.4 shows that removing each component from DYNAPRO hurts the performance, indicating that joint prediction of attribute spans with classes **(A.1)**, and transitions **(A.2)** are all important in procedural reading comprehension. Importantly, we see that removing span prediction by using a single classifier over a joint vocabulary of document tokens and pre-defined attribute classes **(A.3)** hurts the performance. We also see the importance of combining entity-aware representations with attribute-aware representations that incorporate the flow of context **(A.4)**, and constructing attribute-aware representations in comparison to using a [CLS] token **(A.5)**. The study also shows that using a sequential layer for transition modeling can improve final predictions **(A.6)**. Finally, we see the importance of truncating sentences up to a certain time step **(A.7)**, rather than considering the full document. When we increase the context size to include the full document, DYNAPRO does not capture the sequential nature of the procedure and cannot identify the correct attributes at each time step. This shortcoming is partially because the larger document-level context increases the scope of possible span candidates. Over this larger search space, span predictions affect attribute representations more significantly than other predictions (such as pre-defined classes and transitions).

#	Sentence	Label	Prediction
1.1	Blood enters the right side of your **heart**.	heart	right side of your heart
1.2	Blood travels to the **lungs**.	lungs	lungs
1.3	Carbon dioxide is removed from the blood	lungs	lungs
1.4	Blood returns to left side of your **heart**	heart	left side of your heart
2.1	**Blood** travels to the lungs	blood	blood
2.2	Carbon dioxide is removed from the blood.	-	?
3.1	Fuel converts to energy when air and petrol mix.	-	air and petrol
3.2	The car **engine** burns the mix of air and petrol.	engine	air and petrol
3.3	Hot gas from the burning pushes the **pistons**.	piston	air and petrol
3.4	The resulting energy powers the **crankshaft**.	crankshaft	crankshaft

Table 4.5: Examples of correct and incorrect predictions of DYNAPRO. Entities in the first, second, and third examples are `blood`, `carbon dioxide`, `energy`, respectively.

4.4.6 Error Analysis

Qualitative: Table 4.5 shows three types of common mistakes in the final predictions. In the first example, DYNAPRO successfully tracks the `blood` entity while it circulates in the `body`, yet there is a mismatch of what portion of the text it chooses as the span. In the second example, the model correctly predicts the location of `carbon dioxide` as `blood`, but there is not enough external knowledge provided for the model to predict that this entity gets `destroyed` after its removal. In the third example, the model mistakenly predicts the `air and petrol` as a container for the `energy`, but since the changes are explicitly happening to the container, they do not propagate to the `energy`. From these analysis we identified three common patterns in DYNAPRO's errors:

Incorrect Class Prediction: Our analyses show that most errors are due to incorrect predictions of `unknowns` vs. `spans` ($\sim 34\%$ of the times `unknown` location should be predicted, DYNAPRO predicts a `span`). Here, we present the confusion matrix of class predictions:

- Gold label as `nowhere`: our system predicts attribute as `nowhere` (80%), `unknown` (8%), and `span of text` (12%).

- Gold label as `unknown`: our system predicts attribute as `unknown` (56%), `nowhere` (10%), and `span of text` (34%).
- Gold label as `span of text`: our system predicts attribute as `span of text` (67%), `unknown` (18%), and `nowhere` (15%).

Incorrect Span Predictions: Among predicted spans, 55% of spans are identified correctly. For incorrect span predictions about 9% are due to incorrect span boundary predictions (Predicting `cd` instead of `cd and dvd`) and the rest are due to finding an incorrect phrase.

Inconsistent Transitions: We categorize possible inconsistencies in transition predictions into three categories. (The percentages show how many times that inconsistency was observed for all predictions):

- **Creation (2.0%):** When the supporting attribute is predicted to be $\texttt{non} - \texttt{existence}$ or the previous attribute shows that the entity already `exists`.
- **Move (1.5%):** When the predicted attribute is not changed from previous prediction or it refers to a $\texttt{non} - \texttt{existence}$ case.
- **Destroy (1.0%):** When the predicted attribute for the last timestep is $\texttt{non} - \texttt{existence}$.

4.5 Conclusion

We introduce an end-to-end model that benefits from both entity-aware representations and attribute-aware representations to jointly predict attribute values and their transitions for entities in a process. We present a general formalism to model procedural texts, and introduce a model to translate procedural text into that formalism. We show that entity-aware and temporal-aware construction of the input helps yield better entity-aware and attribute-aware representations of the procedural context. Finally, we show that our model can make inferences about state transitions by tracking transitions in

attribute values. Our model achieves state of the art results on various tasks over the PROPARA and NPN-COOKING datasets. Future work involves extending our method to automatically identifying entities and their attribute types and adapting to other domains.

Chapter 5

Interdisciplinary Information Extraction

5.1 Overview

> "Some experts are familiar with one field, such as AI or nanotechnology [...] no one is capable of connecting the dots and seeing how breakthroughs in AI might impact nanotechnology, or vice versa." *–Yuval Noah Harari, Homo Deus, 2016*

The effort to mitigate the COVID-19 pandemic is an interdisciplinary endeavor the world has rarely seen (Apuzzo and Kirkpatrick [2020]). As one recent example, expertise in virology, physics, epidemiology and engineering enabled a group of 200 scientists to understand and bring attention to the airborne transmissibility of the SARS-CoV-2 virus (Morawska et al. [2020]). The diverse and rapidly expanding body of past and present findings related to COVID-19 (Wang et al. [2020b]) makes it challenging to keep up, hindering scientists' pace in making new discoveries and connections.

Research in natural language processing (NLP) has provided important resources to extract *fine-grained* relations from scientific papers in specific areas, such as certain subfields of biomedicine (Kim et al. [2013]; Nye et al. [2018]) or computer science (Wadden et al. [2019]). However, these cover only a fraction of all concepts in the literature; in biomedicine alone, there are myriad

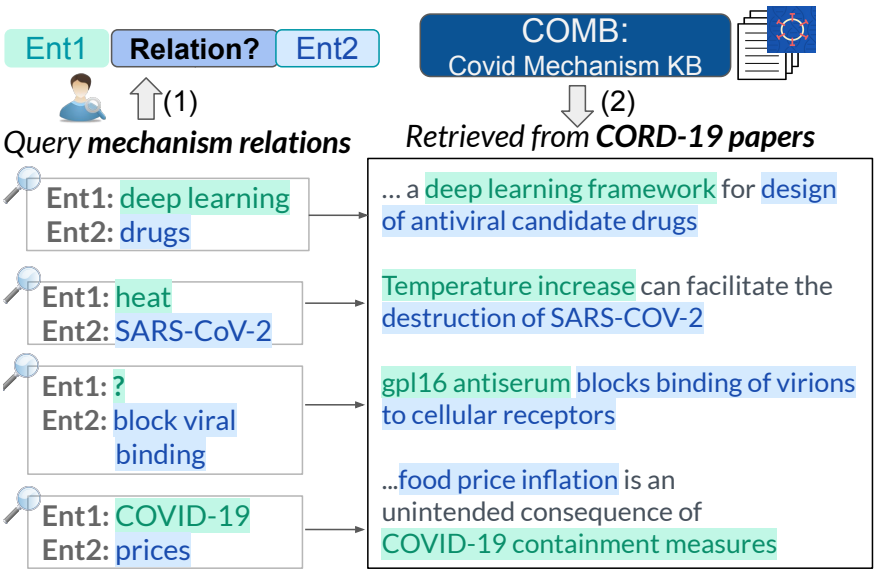


Figure 5.1: Our COVID-19 Mechanism KB (COMB) is extracted from scientific papers and can be searched for diverse activities, functions and influences (1), retrieving relations from the literature (2).

concepts (Salvadores et al. [2013]) not covered by NLP resources. For COVID-19 research, the challenge is especially pronounced due to diversity and emerging concepts; even reading just one paper may require background knowledge in multiple biomedical subfields, physics, chemistry, engineering, computer science, and the social sciences. For example, consider a paper studying the indoor dynamics of aerosolized SARS-CoV-2 and the effect of ventilation on transmission by using simulation models, or work on economic impacts of COVID-19 on prices and consumption.

To make progress in consolidating such diverse information, we introduce a unified schema of *mechanisms* as a *unified language* covering activities, functions and influences across the sciences. These can be proteins that block viral binding, algorithms to design drugs, the effect heat has on viruses, or COVID-19 has on food prices (Figure 5.1).

We build on the fact that mechanisms underlie much of the natural language of scientific papers (Röhl [2012]), and construct a unified schema with two coarse-grained mechanism relations:

- **Direct Mechanisms**: mechanistic *activities* (e.g., viral binding) or *functions* engendered by

natural or artificial entities (e.g., a protein used for binding or algorithm used for diagnosis).

- **Indirect Mechanisms**: *influences and associations* such as economic effects of COVID-19 or complications associated with medical procedures.

Our coarse-grained relation schema, over free-form text spans, strikes a balance between the granular information extracted by Closed-IE approaches (Freitag [1998]; Hoffmann et al. [2010]) and the schema-free breadth of Open IE approaches (Etzioni et al. [2008]; Stanovsky et al. [2018]), which often lead to generic and uninformative relations for scientific applications (Kruiper et al. [2020]).

Furthermore, our schema facilitates construction of a high-quality KB that synthesizes interdisciplinary knowledge. We construct precisely this, releasing MECHANIC (**Mech**anisms **AN**otated in **C**OVID-19 papers) – an annotated dataset of 2,400 mechanisms based on our schema. We train a state-of-the-art model to extract this information from scientific papers, and use it to build COMB (**C**OVID-19 **O**pen **M**echanism Knowledge **B**ase) – a broad-coverage KB of 1.5M mechanisms in COVID-19 papers. We analyze the characteristics of COMB, showing the distribution of relations across scientific subfields and comparing their quality to other IE approaches.

We demonstrate the utility of COMB in two studies with experts. In the first study, our system achieves high precision and recall in scientific search with structured queries on both diverse viral mechanisms and applications of AI in the literature. In the second study, we evaluate COMB in a usability study with MDs active in treating and researching COVID-19. Our system is rated higher than PubMed search by the clinical experts, in terms of utility and quality.

Our main contributions include:

- We introduce a unified schema for *mechanisms* that generalizes across many types of activities, functions and influences. We construct and distribute MECHANIC, an annotated dataset of papers related to COVID-19, with 2,400 instances of our mechanism relation.

- Using MECHANIC, we train an IE model and apply it to 160K abstracts in COVID-19 literature, constructing COMB, a KB of 1.5M mechanism instances. Manual evaluation of relations sampled from our KB shows them to have 88% accuracy. We also find a model trained on our data reaches roughly 80% accuracy on a sample of general biomedical papers from across the PubMed corpus, with no additional training, demonstrating the generalization of our approach.

- We showcase the utility of COMB in structured search for mechanisms in the literature. In a study with MDs working to combat COVID-19, our system is rated higher than PubMed search in terms of utility and quality.

5.2 Mechanism Relation Schema

We present a schema that builds upon and consolidates many of the types of mechanisms discussed in Section 2.3. Our defined schema has three key properties: (1) it uses a generalized concept of mechanism relations, capturing specific types of mechanisms in existing schema and extending them broadly; (2) it includes flexible, generic entities not limited to predefined types, and (3) it is simple enough for human annotators and models to identify in the natural language of scientific texts. This schema enables forming our KB by identifying a set of mechanism relations in a corpus of scientific documents (Section 5.3.6).

We formally define each mechanism as a relation $(E_1, E_2, \texttt{class})$ between entities E_1 and E_2, where each entity E is a text span and the `class` indicates the type of the mechanism relation. Entities all share a single common type and can be either natural (e.g., protein functions, viral mechanistic activities) or artificial (e.g., algorithms, devices), to capture the generality of the concepts in science (see Figure 5.2). We allow each entity to take part in multiple relations (tuples) within a given text, leading to a "mechanism graph". Mechanisms are categorized into two

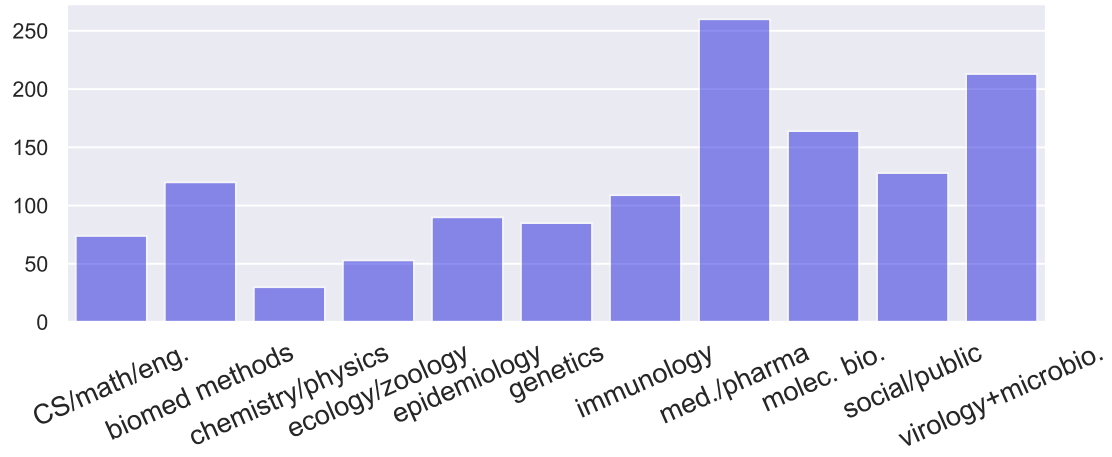


Figure 5.2: MECHANIC covers a diverse set of scientific fields. Histogram of domains in MECHANIC (sample of 350 relations). Manually labeled relation entities, based on a list of scientific disciplines from Wikipedia.

coarse-grained classes:[1]

Direct mechanisms include *activities* of a mechanistic nature – actions explicitly performed by an entity, such as descriptions of a virus binding to a cell, and explicit references to a function (e.g., a use of a drug for treatment, or the use of AI for drug design as in Figure 5.1).

Indirect mechanisms include influences or associations without explicit mechanistic information or mention of a function (such as describing observed effects, without the process involved). These relations correspond more to "input-output correlations" (Röhl [2012]), such as indicating that COVID-19 may lead to economic impacts but not *how* (Figure 5.1), as opposed to direct mechanisms describing "inner workings" – revealing more of the intermediate states that lead from initial conditions (COVID-19) to final states (price inflation) or explicitly describing a function. As an example for the utility of this distinction between direct and indirect relations, consider an MD looking to generate a structured list of all *uses* of a treatment (direct mechanism), but not include side effects or complications (indirect).

[1]We also provide a dataset and extraction model for more granular relations in the form of *(subject, object, predicate)*. We focus on the coarse-grained mechanism schema due its broader flexibility and coverage. See Appendix B.1.1 for details.

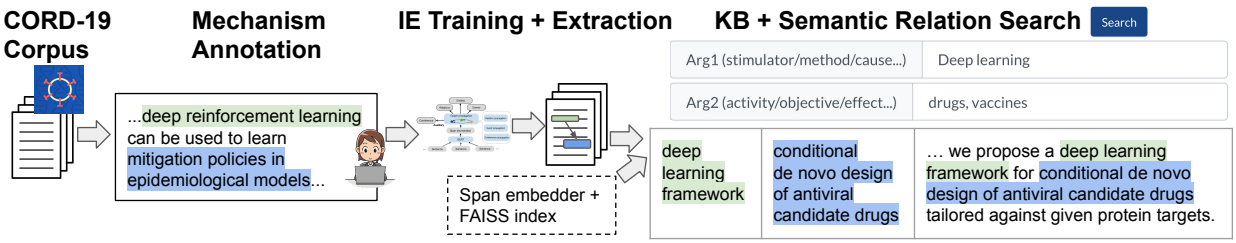


Figure 5.3: **Overview of our approach**. We collect annotations of mechanisms (textual relations) from the CORD-19 corpus, which are used to train an IE model. We apply the model to over 160K documents in the corpus, extracting over 1.5M relations that are fed into our KB. Entity mention spans are embedded with a language model tuned for semantic similarity, and indexed with FAISS for fast similarity search as part of our search interface.

5.3 KB Construction

We describe our approach (depicted in Figure 5.3) for extracting a knowledge base of mechanisms using our unified schema. We first curate MECHANIC, an annotated dataset of general mechanisms from a small collection of scientific papers (Section 5.3.1). We then train a model on our annotated data to extract mechanism relations from the entire CORD-19 corpus of scientific papers (Section 5.3.2); we provide our automated metric for evaluation and show that comparing to other baselines, such as more domain-specific schemas and SRL based model, the model trained on our schema achieve the best performance (Section 5.3.3,5.3.4). We use the model to build COMB, a knowledge base of mechanisms across the entire CORD-19 corpus of scientific papers (Section 5.3.5), which supports semantic search for relations (Section 5.3.6).

5.3.1 Collecting Mechanism Annotations

We construct a dataset of mechanism relations in texts randomly sampled from the CORD-19 corpus (Wang et al. [2020b]) that includes scientific papers connected to COVID-19. To circumvent annotation challenges in scientific datasets (Luan et al. [2018]) and ensure high-quality annotations, we follow a three-stage process of (1) annotating entities and relations using biomedical experts, (2) unifying span boundaries with an NLP expert, and (3) verifying annotations with a bio-NLP expert.

Our annotation process is a relatively low-resource and generalizable approach for a rapid response to the COVID-19 emergency.

In the first stage, five annotators with biomedical and engineering background annotate all mechanism relations as defined in Section 5.2 (full annotation guidelines are available in Appendix B.1.2). Relations are annotated as either direct/indirect. Entities are annotated as the longest span of text that is involved in a relation with another entity, while not including redundant or irrelevant tokens. As in related tasks (Luan et al. [2018]), annotators are guided to resolve doubt on span boundaries by selecting the longest relevant span.

Annotators had a one-hour training session. In the first part of the training session, annotation guidelines were reviewed. The guidelines included simple explanations of direct/indirect mechanisms along with introductory examples (e.g., "*the virus makes use of spike protein to bind to a cell*", "*A virus leads to respiratory infection*"). In the second part, annotators saw examples from papers in the annotation interface (see Figure B.2, Appendix B.1), and performed a few live training annotations.

We initially observed significant variation between annotators in identifying span boundaries for entity annotations, stemming from inherent subjectivity in such annotation tasks (Stanovsky et al. [2018]; Luan et al. [2018]) and from lack of NLP experience by some annotators. In the second stage, an NLP expert annotator conducted a round of style unification by viewing annotations and adjusting span boundaries to be more cohesive while preserving the original meaning, focusing on boundaries that capture essential but not redundant or generic information (e.g., adjusting the span *substantial* ***virus replication*** *by unknown mechanisms* to include only *virus replication*). Finally, in the third stage, a bio-NLP expert with experience in annotating scientific papers verified the annotations and corrected them as needed. The expert accepted 81% of the annotations from the second stage without modification, confirming the high quality of the stage-2 data. Relation label mismatches accounted for 5% of the remaining 19%. Other sources of disagreement were span mismatches and new relations added by the bio-NLP expert adjudicator.

The resulting dataset (MECHANIC : **Mech**anisms **AN**otated in **C**OVID-19 papers) contains 2,370 relation instances (1645 direct, 725 indirect) appearing in 1,000 sentences from 250 abstracts.[2] Average span length is 4 tokens, while the average distance between relation arguments is 11.40 tokens.

5.3.2 Task Definition and Model

Using MECHANIC, we train an IE model to extract mechanism relations from sentences in scientific documents. We train DyGIE++ (Wadden et al. [2019]), a state-of-the-art end-to-end IE model which extracts entities and relations jointly (without assuming to have entity spans given), classifying each relation as one of $\{\texttt{DIRECT}, \texttt{INDIRECT}\}$.[3] We evaluated the performance of the model within three tasks:

Entity detection: Given a boolean span matching function $\text{m}(s_1, s_2) = \mathbb{1}(s_1 \text{ matches } s_2)$, a predicted entity mention $\hat{e}$ is correctly *identified* if there exists some gold mention e^* in $\mathcal{D}$ such that $\text{m}(\hat{e}, e^*) = 1$ (since there is only one entity type, an entity mention is correctly classified as long as its span is correctly identified).

Following common practice in work on Open IE (Stanovsky et al. [2018]), we report results using a partial-matching similarity function, in this case based on the widely-used Rouge score: $\text{m}_{\text{rouge}}(s_1, s_2)$ is true if Rouge-L$(s_1, s_2) > 0.5$ (Lin [2004]).

Relation detection / classification: Given a boolean span matching function, a predicted coarse-grained relation $\hat{r} = (\hat{E}_1, \hat{E}_2, \hat{y})$ is correctly *identified* if there exists some gold relation $r^* = (E_1^*, E_2^*, y^*)$ in $\mathcal{D}$ such that $\text{m}(\hat{E}_1, E_1^*) = 1$ and $\text{m}(\hat{E}_2, E_2^*) = 1$. It is properly *classified* if, in

[2]The dataset is similar in size to related scientific IE datasets (Luan et al. [2018]) which share related challenges in collecting expert annotations of complex or ambiguous concepts over difficult texts.

[3]We use DyGIE++ with SciBERT (Beltagy et al. [2019]) embeddings fine-tuned on our task and perform hyperparameter grid search (for dropout and learning rate only) and select the best-performing model on the development set ($7e-4$ and 0.43, respectively). Full details are in Appendix B.1.3.

addition, $\hat{y} = y^*$.

Relation identification measures the model's ability to identify mechanisms of any type - direct or indirect - while relation classification aims to discriminate between direct and indirect types of mechanism mentions in the text.

5.3.3 Baselines

SemRep: The SemRep dataset (Kilicoglu et al. [2011]), consisting of 500 sentences from MEDLINE abstracts and annotated for semantic predication. Concepts and relations in this dataset relate to clinical medicine from the UMLS biomedical ontology (Bodenreider [2004]), with entities such as drugs and diseases. Some of the relations correspond to mechanisms (such as X TREATS Y or X CAUSES Y); By the lead of domain experts, we map these existing relations to our mechanism classes and use them to train DyGIE. Other relations are even broader, such as PART-OF or IS-A – we do not attempt to capture these categories as they often do not reflect a functional relation.

Scierc: SciERC dataset (Luan et al. [2018]), consisting of 500 abstracts from computer science papers that are annotated for a set of relations, including for USED-FOR relations between methods and tasks. We naturally map this relation to our `DIRECT` label and discard other relation types, and use this dataset to train DYGIE.

SRL: Finally, we also use a recent BERT-based SRL model (Shi and Lin [2019]). We select relations of the form (`Arg0, verb, Arg1`), and evaluate using our partial metrics applied to `Arg0` and `Arg1` respectively.

5.3.4 Best Performing Model over MECHANIC

We use the DYGIE package (Wadden et al. [2019]) to train models for entity and relation extraction over MECHANIC and we utilize SciBERT (Beltagy et al. [2019]) for our text embeddings and

Model	**RC**	**RD**	**ED**
OpenIE	-	15.5	25.6
SRL	-	24.5	27.7
DYGIE(SemRep)	6.8	8.3	32.5
DYGIE(SciERC)	18.6	20.4	39.2
DYGIE(MECHANIC)	**42.8**	**45.6**	**50.2**

Table 5.1: F1 scores. Relations from SRL and OpenIE do not map directly to `DIRECT MECHANISM` and `INDIRECT MECHANISM` classes, and do not have relation classification scores.

finetune upon that, with learning rate for finetuning set to $5e-5$ with weight decay of 0.01. The training was run for 100 epochs with the $slanted\_triangular$ (Howard and Ruder [2018]) learning rate scheduler. We used the AdamW (Loshchilov and Hutter [2017]) optimization algorithm. In our objective function we assign equal weights to relation and span loss terms. The maximum allowed length of spans is 12.

The hyperparameters achieving best performance over our development search are 0.43, $7e-4$ and 215 for dropout, learning rate, and hidden size respectively. All other parameters are kept to default values [4].

Table 5.1 compares the performance of our best model with the baselines introduced in Section 5.3.3. Figure 5.4 shows Precision@K results, with our model reaching high absolute numbers.

5.3.5 Extracting a KB of Mechanisms

To form our corpus-level KB, we apply the trained model to each document in our corpus (all 160K abstracts in the CORD-19 corpus) to extract mechanism relations and then integrate the extracted relations. We find that our trained model achieves high precision scores for high confidence predictions (precision $\geq 80\%$ within top-20 predicted relations (see Figure 5.4). Therefore, our corpus-level KB is constructed by filtering predictions with low confidence.

To integrate relations and entities across the corpus, we use standard surface-level string nor-

[4]Available in our code repository: https://github.com/AidaAmini/DyGIE-MECHANIC

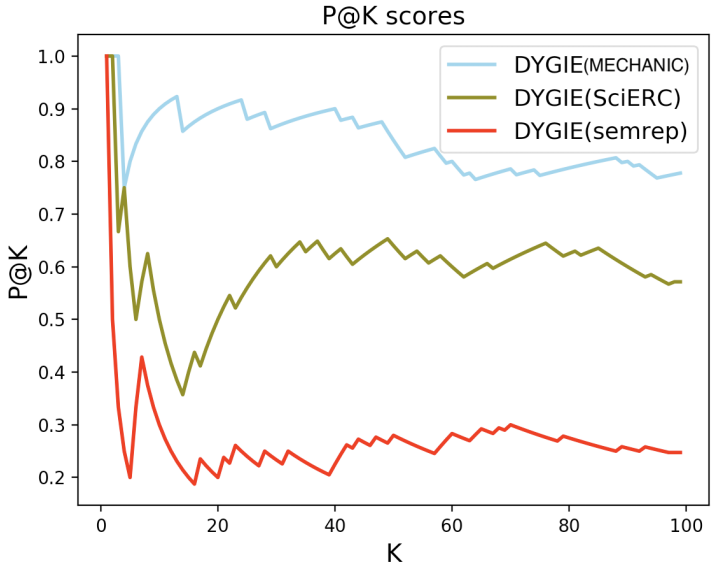


Figure 5.4: Precision@K of our model compared with pre-trained SciERC and SemRep baselines. P@K for our model is high in absolute numbers.

malization (such as removing punctuation, lemmatizing, and lowercasing) and unify and normalize entity mentions using coreference clusters of entities within a document.[5] Each coreference cluster is assigned a representative entity as the mention with the longest span of text, and all other entities in that cluster are replaced with the representative entity. This is particularly useful for normalizing pronouns such as *it* with the original mention they referred to (e.g., a specific virus or method *it* refers to).

Our final KB (COMB) consists of 1.5M relations in the form of $(E_1, E_2, \texttt{DIRECT/INDIRECT})$ filtered by high confidence score ($>= 90\%$), where entities E_i are standardized free-form spans of text.

5.3.6 Semantic Relation Search

The constructed KB enables applications for retrieving relations across concepts from many disciplines. For example, searching for all documents that include mechanisms to incorporate *AI* in

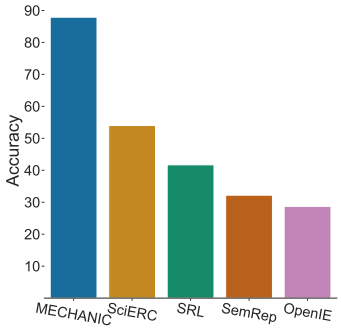
[5]We use a pre-trained DyGIE++ model trained on SciERC to obtain coreference clusters.

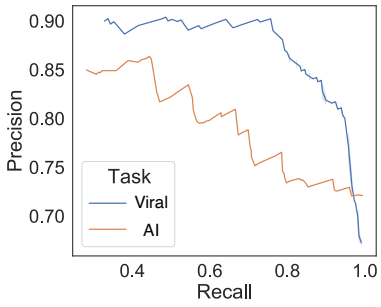
studies of *heart disease* ($E_1 =$ AI, $E_2 =$ heart disease, $\texttt{DIRECT}$) requires going beyond simply finding documents that mention *AI* and *heart disease*. Here, we describe our approach for searching over the KB by encoding entities and relations, capturing related concepts (such as *cardiac disease* and *heart conditions*), as well as simpler surface matches (*artificial intelligence* ***methods***, *artificial intelligence* ***models***).

Specifically, for a given query $\mathbf{q}(E_1^q, E_2^q, \texttt{class})$, our goal is to find mechanisms r_i in COMB whose entities are free-form texts similar to E_1^q, E_2^q in the query. The $\texttt{class}$ is used to filter for the type of relation—for example, when explicitly requiring $\texttt{DIRECT}$ mechanisms.

Entity encoding: We obtain an encoding function $f : E \mapsto \mathbb{R}^d$ to encode all unique spans (entities) in the KB to a d dimensional vector space. The encoding function is derived by fine-tuning a language model (LM) originally trained on PubMed papers (Gururangan et al. [2020]) on semantic similarity tasks. For fine-tuning, we use sentence pairs in STS (Cer et al. [2017]) and SNLI (Bowman et al. [2015]) following (Reimers and Gurevych [2019]), and add biomedical sentence pairs from the BIOSSES dataset (Soğancıoğlu et al. [2017]).

Relation similarity: Given a query $\mathbf{q}$, we rank the set of all COMB relations with the same $\texttt{class}$ as the query. For each candidate relation $r = (E_1, E_2, \texttt{class})$ in COMB, we compute its similarity to the query relation $\mathbf{q}$ as the minimum similarity between encodings of their corresponding entities: $\min_{j \in \{1,2\}} f(E_j) \cdot f(E_j^q)$. With this definition, a relation (E_1, E_2) with E_1 very similar to the first entity of the query E_1^q but E_2 distant from E_2^q will be ranked low. For example, with the query ($E_1^q =$ deep learning, $E_2^q =$ drugs), the relation ($E_1 =$ microscope, $E_2 =$ drugs) will be ranked low due to the pair (deep learning, microscope). For efficient search, we create an index of embeddings corresponding to the 900K unique surface forms in COMB and employ a system designed for fast similarity-based search (Johnson et al. [2017]).





Metric	PubMed	COMB
Search	71%	**90%**
Utility	69.5%	**92%**
Interface	78%	**90%**
Overall	74%	**91%**

s

Figure 5.5: Evaluating COMB in studies with experts. COMB is found to have high quality and utility, outperforming other approaches. **Left:** COMB outperforms external resources with either *specific* types of mechanisms or *open* relations, in human evaluation for correctness and usefulness of predictions, on a sample of 300 predicted relations. **Center:** Retrieved relations are ranked by query similarity (Section 5.3.6) and compared to human relevance labels to compute precision/recall. Higher-ranked results are overall judged as relevant by humans. **Right:** Results of COMB search study with five practicing MDs, using both our system and PubMed to search the literature. Experts were given a post-study questionnaire with questions grouped by subject (search, utility, interface). Our mechanism search system performed substantially better than PubMed.

5.4 Evaluating COMB

In this section, we evaluate the constructed KB of mechanisms in terms of correctness and informativeness (Section 5.4.1), and its utility in searching for mechanisms (Section 5.4.2). Our main goal is to ensure the mechanism relations have high quality to support our large-scale KB and search applications. We further show that our schema is useful as compared to other schema.

5.4.1 KB Correctness and Informativeness

We employ two annotators with biomedical and CS backgrounds to judge the quality of the predicted relations in COMB. In particular, following (Groth et al. [2018]), annotators are given a predicted relation together with the sentence from which it was extracted. We collapse all entities/relations into one generic type for this analysis. Annotators are asked to label the predicted relation as correct if (1) it accurately reflects a mechanistic relation mentioned in the sentence (*correctness*), and (2) the extracted entities and relation label are sufficient to convey the meaning of the relation,

Relation query	Example results from KB search interface
E1 Warm climate E2 Coronavirus	Experimental data showed that coronavirus survival was negatively impacted by ozone and high temperature.
E1 COVID-19 E2 Bilateral ground glass opacities	The typical features of COVID-19 in chest CT include bilateral, peripheral, and multifocal ground-glass opacities with or without superimposed consolidations.
E1 Aerosols E2 SARS-CoV-2 transmission	SARS-CoV-2 is transmitted efficiently via the air (via respiratory droplets and/or aerosols) between ferrets.

(a) **Viral mechanism search**. Queries for $(E1, E2)$ relations, and example retrieved results.

E1 Convolutional neural networks E2	3D patch - based convolutional neural networks were trained to predict conductivity maps from B1 transceive phase data.
E1 Computer vision E2	We present a computational pipeline using algorithms from computer vision to decompose ciliary motion into quantitative elemental components.
E1 Graph neural networks E2	We propose a new model called GraphDTA that represents drugs as graphs and uses graph neural networks to predict drug – target affinity.

(b) **AI search**. Queries consists of only E_1, to find all applications of AI approaches/areas.

Table 5.2: Example search queries and results for the viral mechanism and AI applications tasks.

without referring to the source sentence (*informativeness*). We collect human judgements for 300 predicted relations for our approach and baselines, sampled from 150 randomly selected sentences. Agreement is 71% by Cohen's Kappa and 73% by Matthew's Correlation Coefficient.

Comparing KB quality to other schemas: To showcase the benefit of our approach, we compare the relations extracted using a DyGIE model trained on MECHANIC, versus a DyGIE model trained on other resources that are most related to our mechanisms: SemRep (Kilicoglu et al. [2011]) captures a wide range of biomedical relations (such as drug-drug interactions), and SciERC (Luan et al. [2018]) contains relations relevant to computer science (such as "method-task" and "used-for" relations).[6] In addition, we compare with a Semantic Role Labeling (SRL) method (Shi and Lin [2019]) that captures broad relations between free-form spans that focus on agents and actions, and a neural OpenIE model (Stanovsky et al. [2018]).

[6]We use an expert annotator to align external resources to our `direct` or `indirect mechanism` annotations, e.g., USED-FOR is mapped to *direct* mechanism).

Figure 5.5 (left) shows that 88% of relations from COMB are marked as correct by human raters, demonstrating that our approach extracts mechanism relations with better quality than external resources. These results suggest that our predicted relations are of overall high quality and can be used to build our corpus-level KB and explore its utility.

Examining Generalization: COVID-19 papers are highly diverse both topically and chronologically. We conduct a small-scale preliminary experiment examining whether a model trained on MECHANIC can generalize to capture mechanism relations in the general biomedical papers, from a much larger corpus of open access papers on PubMed Central (PMC).[7] We randomly sample a set of 200 predicted relations from papers across the entire PMC corpus, and label them using the same criteria used above. As expected, we find that performance drops, but encouragingly is still considerably high: after filtering predictions with confidence lower than 90% in the same way we construct COMB, 76% of relations are considered correct. When filtering for confidence with a threshold of 95% (which captures 70% of the samples), the rate of correct predictions is 78%. In future work it would be interesting to fine-tune our model on a small set of labeled examples from the general PMC corpus to potentially improve these results.

5.4.2 COMB Utility

We design several search tasks and user studies to evaluate the utility of the constructed KB (Section 5.4.2.1) and compare it with the PubMed medical KB and search engine (Section 5.4.2.2), as judged by medical doctors working on the front lines of COVID-19 treatment and research. All tasks are designed to evaluate our framework's utility in helping researchers and clinicians looking to quickly search for mechanisms or cause-effect relations in the literature and retrieve a list of structured results.

[7] https://www.ncbi.nlm.nih.gov/pmc/

5.4.2.1 Search Quality

We form search queries based on a wide range of topics pertaining to (1) SARS-CoV-2 mechanisms (such as modes of transmission, drug effects, climatic influences, molecular-level properties) and (2) applications of AI in this area. Table. 5.2a and 5.2b show queries and example relations returned from COMB, along with the context sentences from which they were extracted.

Viral mechanism search: Queries are formed based on statements in recent scientific claim-verification work (Wadden et al., 2020; see full list in Appendix B.2.2). For example, for the statement *the coronavirus cannot thrive in warmer climates*, we form the query as $(E_1 =$ Warm climate, $E_2 =$ coronavirus$)$ (see Table 5.2a row 1). For statements reflecting an indirect association/influence, we filter for `INDIRECT` relations (Table 5.2a row 2). For statements that reflect an undirected mechanism relation (e.g., *Lymphopenia is associated with severe COVID-19 disease*), we query for both directions.

AI applications search: This task is designed to explore the uses of AI in COVID-19 papers (Table 5.2b). We use queries where the first entity E_1 is a leading subfield or method within AI (e.g., *deep reinforcement learning* or *text analysis*), and the second entity E_2 is left unspecified. Since all queries relate to *uses* of AI, we filter for `DIRECT` relations. These open-ended queries simulate an exploratory search scenario, and can potentially surface inspirations for new applications of AI against COVID-19 or help users discover where AI is being harnessed.

Evaluation: Expert annotators are instructed to judge if a relation is related to the query or not and if the sentence actually expresses the mechanism. These annotations are used as ground-truth labels to compute precision/recall scores of the relations extracted by our algorithm. Since it is not feasible to label every relation, annotators are shown a list of 20 relations for each query including high and low rank relations returned by our search algorithm.[8] In total, we use 5 annotators to obtain 1,700 relevance labels across both tasks. Inter-annotator agreement is high by several metrics,

[8]Specifically, for each query we retrieve the top-1000 similar relations from COMB, ranked as described in Section 5.3, and select the top and bottom 10 relations (20 per query, 200(=20x10) per task, 400(=200x2) in total), shuffle their order, and present to annotators together with the original sentence from which each relation was extracted.

ranging from 0.7–0.8 depending on the metric and task; (see Appendix B.2.1). Annotators have graduate/PhD-level background in medicine or biology (for the first task) and CS or biology (for the second task).

Results: Figure 5.5 (center) shows our results for both tasks. For biomedical search queries, we observe 90% precision that remains stable for recall values as high as 70%. For *AI applications* we observe a precision of 85% at a recall of 40% that drops more quickly. This lower precision is likely due to the fact that E_2 is unspecified, leading to a wider range of results with more variable quality.

Overall, these results showcase the effectiveness of our approach in searching for mechanisms between diverse concepts in COVID-19 papers.

5.4.2.2 Comparing COMB with PubMed

This experiment compares the utility of COMB in structured search for causal relationships of clinical relevance to COVID-19 with PubMed[9]—a prominent search engine for biomedical literature that clinicians and researchers frequently peruse as their go-to tool. PubMed allows users to control structure (e.g., with MeSH terms or pharmacological actions), is supported by a KB of biomedical entities used for automatic query expansion, and has many other functions.

Expert evaluation: We recruit five expert MDs—with a wide range of specialities including gastroenterology, cardiology, pulmonary and critical care—who are active in treating COVID-19 patients and in research. Each expert completed search randomly ordered tasks using both PubMed and our COMB UI, showing the full set of ranked relations, as well as the sentence snippet mentioning the relation, the paper title, and hyperlink to abstract. At the end of the study after all search tasks are completed for both our system and PubMed, experts are given a questionnaire of 21 7-point Likert-scale questions to judge system utility, interface, and search quality. The first 16 questions are taken from a Post Study System Usability Questionnaire (PSSUQ; (Lewis, 2002))

[9] https://pubmed.ncbi.nlm.nih.gov/

widely used in system quality research. The last 5 questions are designed by the authors to evaluate search quality such as overall result relevance and ranking (for the full question list, see Appendix B.2.2, Figure B.4). Each question is asked twice, once for PubMed and once for our system, leading to $21 \times 2 \times 5 = 210$ responses.

Search queries: We provide experts with seven search queries that were created by an expert medical researcher, relating to causal links (e.g., between COVID-19 and cardiac arrhythmias) and functions (e.g., Ivermectin as a treatment). See full set of queries in Appendix B.2.

Results: Figure 5.5 (right) shows the average Likert scores (normalized to [0%,100%]) across all questions and users for COMB and PubMed. The results show that the medical experts strongly prefer COMB to PubMed (overall average of 91% vs. 74%, with non-normalized scores of 6.6 vs. 5.2). On average across the 21 questions, the majority of the five experts assigned our interface a higher score than PubMed, at an average rate of 3.5/5. This rate increases further when considering ties—on average 4.75/5 of the experts assigned our system a score equal or higher than PubMed.

Overall, our system significantly outperforms PubMed in this task, with an average gap of roughly 20% for search and utility-related questions (Wilcoxon signed rank test p-value is significant at 4.77×10^{-7}). These results are particularly interesting and indicate the potential of COMB because of the experts' strong familiarity with PubMed and the simple nature of our UI.

Our system searches and retrieves *relations*—only texts explicitly mentioning relations that match the input query. This often more precisely reflects the query than results returned by PubMed, which do not have the additional layer of structured information in COMB. For example, for the query (E_1=cardiac arrhythmias, E_2=COVID-19), PubMed returns the following title of one paper: *Guidance for cardiac electrophysiology during the **COVID-19** pandemic [....] Electrocardiography and **Arrhythmias** Committee*—E_1 and E_2 are both mentioned, but not within a mechanism relation.

5.5 Summary

We introduced a unified schema for *mechanisms* that generalizes across many types of activities, functions, and influences. We constructed and distributed MECHANIC, a dataset of papers related to COVID-19 annotated with this schema. We trained an IE model and applied it to COVID-19 literature, constructing COMB, a KB of 1.5M mechanisms. We showcased the utility of COMB in structured search for mechanism relations in COVID-19 literature. In a study with MDs active in the fight against the disease, our system is rated higher than PubMed search for both utility and quality. Our unified view of mechanisms can help generalize and scale the study of COVID-19 and related areas. More broadly, we envision a KB of mechanisms that enables the transfer of ideas across the literature (Hope et al. [2017]), such as by finding relationships between mechanisms in SARS-CoV-2 and other viruses, and assists in literature-based discovery (Swanson and Smalheiser [1996]) by finding cross-document causal links.

Ethical considerations

Our knowledge-base and search system is primarily intended to be used by biomedical researchers working on COVID-19, and researchers from more general areas across science. Models trained and developed on our dataset are likely to serve researchers working on COVID-19 information extraction, and scientific NLP more broadly. We hope our system will be helpful for accelerating the pace of scientific discovery, in the race against COVID-19 and beyond.

Our knowledge-base can include incorrect information to the extent that scientific papers can have wrong information. Our KB includes metadata on the original paper from which the information was extracted, such as journal/venue and URL. Our KB can also miss information included in some papers.

Our data collection process respected intellectual property, using abstracts from CORD-19 (Wang et al. [2020b]), an open collection of COVID-19 papers. Our knowledge-base fully attributes

all information to the original papers. All annotators were given extensive background on our objectives, and told their annotations will help build and evaluate a knowledge-base and search engine over COVID-19 research. Graduate-student annotators were payed 25 USD per hour. MD experts helped evaluate the tool on a voluntary basis.

Chapter 6

Conclusion

In this thesis we explored the application of natural language processing tasks toward understanding and parsing scientific text. We show challenges exist within technical literature and we explored (a) data annotation, (b) modeling, and (c) large scale application within the boundaries of scientific and technical language.

Chapter 2 presents background endeavor of natural language processing toward scientific text understanding. Within this chapter, we provide more details on the models and datasets used toward solving algebra word problem, reading comprehension and scientifics information extraction tasks.

In chapter 3, we tackled the challenges of automatically solving algebra word problems. In order to overcome the shortcomings within this domain, we introduced a new mathematical representation language and a large-scale dataset carefully annotated with the representation. We experiment with the seq2seq modeling architecture and how including the category of each math word problem can help with the model's performance. Our model achieved the state-of-the-art results over a more complex model, which revealed the importance of latent representations.

In chapter 4, we tackled the task of reading comprehension and question answering over procedural text. We showed that by considering both attribute values as well as the transitions, our model achieves more precise conclusions over the PROPARA and NPN-COOKING datasets.

Finally, in chapter 5 we showed the pipeline for constructing a large scale application over scientific documents designed to help with navigating the relational queries over COVID-19 related entities. The pandemic resulted in many research efforts from various disciplines. We aim to enable the researcher to search for functional relations between entities from underlying interdisciplinary context. We defined a new representation schema that strikes a balance between informativeness and coverage. Using this schema, we collected a dataset carefully annotated and vetted by experts during multiple annotation rounds. Using the new dataset, we explored the interdisciplinary information extraction task and by training the state of the art model, we constructed a knowledgebase containing over 1.5M relations. We also provided various evaluations and analysis over the quality of our trained model and the utility of our knowledgebase and the search interface.

6.1 Future Research Directions

This dissertation shows the potential to enhance understanding and analysing scientific literature. Our contributions have improved the progress in terms of data annotation, modeling, and application. Below we propose the future research directions and applications toward further improvement of scientific text understanding.

6.1.1 Dynamic data collection

MathQA (introduced in Section 3.3) is collected via dynamic annotation platform. With this platform we enable the use of intermediate results. Similar data curation paradigm can be applied to many tasks to ensure the quality of future data collection tasks. Among those are data collection for the other theoretical domains such as logical problems solving, or procedural text such as cocking recipes.

6.1.2 Domain-aware multi discipline information extraction

In chapter 5, we developed a baseline based on DyGIE (Wadden et al. [2019]) to extract relations over MECHANIC dataset. Here, we propose to extend the relation prediction approach with information about underlying scientific domains in MECHANIC dataset. DyGIE is designed to use the shared language structures in every domain. For example, from general structure of the phrase "entity X `is used against` entity Y", the functional relation is inferred, independent of the what is the underlying domain for entities X and Y.

However, relation prediction can benefit from domain-specific information. For instance, in the sentence: "There are few studies of image analysis about face mask detection.", there is no explicit surface-level information that points to a functional relation, however, the meaning of individual phrases `image analysis` and `face mask detection` can help with extracting information.

In addition to the meaning, different aspects of each phrase (e.g. gender, count) affect the relation prediction task in each domain differently. For instance, in computer science domain, the functional relations are mostly defined for intelligent agents, which are gender neutral. In contrast, in bio-medical domain, the gender can play significant role in predicting the side effects on recipients of certain medicine. Therefore, we propose to develop a domain-aware IE approach that uses separate representation spaces for different domains. Each domain-aware representation is initialized with pre-trained language model of that specific domain and will be fine-tuned using training samples of that domain to account for domain specific information.

Model: Our proposed model consists of three components: (1) domain prediction unit, (2) representation unit, (3) relation decoding unit. The input to the model is a scientific document $\mathcal{D}$ represented as a sequence of input tokens $\{w_1, \ldots, w_n\}$. The model computes the embeddings from a pool of pre-trained language models mapped to be used for each domain.

Domain prediction module: This module ranks relevance of each input document $\mathcal{D}$ to each underlying domain. The list of domains is pre-determined by examining the documents included in original CORD-19 corpus and can include various examples such as biology, chemistry, sociology, and computer science. The output of this module is a vector of similarity scores between the document and each of the domains.

Domain-aware representation module: The model will use a separate representation unit for each domain. Each domain-specific representation is pre-trained over large scale corpus of documents relevant to the same domains and is fine-tuned by the relating documents in the CORD-19 dataset. Given a vector of domain relevance scores from previous layer, our model combines these weight vectors using max pooling or their average scores in order to calculate the contextualized representation vector of the input document.

Domain-aware relation prediction module: This unit uses the contextualized representation vector and domain similarity scores to recursively determine the entity spans and the relation labels. Here, the relation labels can be from predefined classes or from span of text.

Dataset: This task can be evaluated over MECHANIC dataset as it provides a rich resource of interdisciplinary and diverse publication related to coronaviruses. A clustering algorithm can be implemented for mapping each document in the corpus to a relevant pre-defined domain. The quality of automatically assigning a domain can be measures by crowd-sourcing the labels for a small sample of documents and comparing them against the authomatically generated labels.

Benchmarks and Evaluation: The performance of category-aware information extraction task can be compare with the performance of DYGIE(MECHANIC) model in Section 5.3.4. In order to automatically compare the quality of generated labels against the previous models, we propose to compare the performance with the three previously introduced tasks: named entity recognition,

relation identification, and relation classification (see Section 5.3.2).

Additionally, since the labels can be generated from the vocabulary, a human evaluation task can be employed to measure both *correctness* and *soundness* of each label assigned to each relation. These scores can be compared with the results in Figure 5.5 left.

6.1.3 Sequential link prediction

The knowledge graph over scientific findings might lack certain edges. This can be due to various reasons, spanning over infeasibility of the task, lack of promising results, and limited resources to pursue all research directions. Therefore, there remains a strong potential for the non-existing link to be a valid relations. Finding the valid relations and pruning the relations that do not have a potential is with great value, leading to saving time and resources on research directions with no promise. A potential application of this task is scoring system for scientific proposals.

Benchmark: One of the challenging aspects of pursuing this research direction is construction of the benchmark. Since the publications originally collected in CORD-19 dataset are temporal, the timestamp of each paper can be used in a heuristic approach to create training samples. Each cluster of similar nodes in the knowledge graph representing similar method or concept can be divided into previous work and hypothesis based on the timeline they are pursued.

Model: There are few potential directions to approach this problem. Similar to the task of procedural reading comprehension in chapter 4, we suggest creating sequential stories of tasks, methods and finding. Sequentially embedding the samples and following the results can reveal the potential future directions for similar tasks.

In addition, the model should consider that absence of a particular edge might be based on not pursuing those approaches due to limited budget, resources, and time. It might be beneficial to add more attributes such as computing resource needed, run-time, and number of times that approach is used to encode the cost and feasibility.

Bibliography

Matt Apuzzo and David D. Kirkpatrick. 2020. Covid-19 changed how the world does science, together. *New York Times*.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Machine translation by jointly learning to align and translate. In *ICLR*.

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. In *arXiv preprint math/0701393*.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In *EMNLP*.

Daniel G Bobrow. 1964. Natural language input for a computer problem solving system.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. In *ICLR*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Diane J Briars and Jill H Larkin. 1984. An integrated model of skill in solving elementary word problems. In *Cognition and instruction 1(3)*, pages 245–296.

Patryk Burek, Robert Hoehndorf, Frank Loebe, Johann Visagie, Heinrich Herre, and Janet Kelso. 2006. A top-level ontology of functions and its application in the open biomedical ontologies. *Bioinformatics*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *CoNLL)*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval-2017*.

Eugene Charniak. 1968. Calculus word problems. Ph.D. thesis, Massachusetts Institute of Technology.

Eugene Charniak. 1969. Computer solution of calculus word problems. In *Proceedings of the 1st international joint conference on Artificial intelligence*, pages 303–316. Morgan Kaufmann Publishers Inc.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning structured natural language representations for semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 44–55. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *NAACL*.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2019. Building dynamic knowledge graphs from text using machine reading comprehension. In *ICLR*.

Denise Dellarosa. 1986. A computer simulation of childrens arithmetic word-problem solving. In *Behavior Research Methods, Instruments, Computers 18(2)*, pages 147–154.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. In *EMNLP*.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. 2008. Open information extraction from the web. *Communications of the ACM*.

D. Freitag. 1998. Toward general-purpose learning for information extraction. In *COLING-ACL*.

Paul Groth, Mike Lauruhn, Antony Scerri, and Ron Daniel Jr. 2018. Open information extraction on scientific text: An evaluation. In *CoNLL*.

Aditya Gupta and Greg Durrett. 2019a. Effective use of transformer networks for entity tracking. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Hong Kong, China. Association for Computational Linguistics.

Aditya Gupta and Greg Durrett. 2019b. Tracking discrete and continuous entity state for process understanding. *CoRR*, abs/1904.03518.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *ACL*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. *CoRR*, abs/1612.03969.

Julie Hirtz, Robert B Stone, Daniel A McAdams, Simon Szykman, and Kristin L Wood. 2002. A functional basis for engineering design: reconciling and evolving previous efforts. *Research in engineering Design*.

R. Hoffmann, Congle Zhang, and Daniel S. Weld. 2010. Learning 5000 relational extractors. In *ACL*.

Tom Hope, Joel Chan, Aniket Kittur, and Dafna Shahaf. 2017. Accelerating innovation through analogy mining. In *KDD*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2017. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Danqing Huang, Jing Liu, Chin-Yew Lin, and Jian Yin. 2018a. Neural math word problem solver with reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 213–223.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Danqing Huang, Jin-Ge Yao, Chin-Yew Lin, Qingyu Zhou, and Jian Yin. 2018b. Using intermediate representations to solve math word problems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Diane Marie Keeling, Patricia Garza, Charisse Michelle Nartey, and Anne-Ruxandra Carvunis. 2019. Philosophy of biology: The meanings of'function'in biology and the problematic case of de novo gene emergence. *Elife*, 8:e47014.

Halil Kilicoglu, Graciela Rosemblat, Marcelo Fiszman, and Thomas C Rindflesch. 2011. Constructing a semantic predication gold standard from the biomedical literature. *BMC bioinformatics*.

Jin-Dong Kim, Yue Wang, and Yamamoto Yasunori. 2013. The genia event extraction shared task, 2013 edition-overview. In *BioNLP Shared Task Workshop*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *the 3rd International Conference for Learning Representations (ICLR)*.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *ArXiv e-prints*.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016a. Mawps: A math word problem repository. In *NAACL*.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016b. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157.

Ruben Kruiper, Julian Vincent, Jessica Chen-Burger, Marc Desmulliez, and Ioannis Konstas. 2020. In layman's terms: Semi-open relation extraction from scientific texts. In *ACL*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014a. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014b. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland. Association for Computational Linguistics.

James R Lewis. 2002. Psychometric evaluation of the pssuq using data from five years of usability studies. *International Journal of Human-Computer Interaction*.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database*.

Chao-Chun Liang, Yu-Shiang Wong, Yi-Chung Lin, and Keh-Yih Su. 2018. A meaning-based statistical english math word problem solver. In *Proceedings of NAACL-HLT 2018*.

Christian Liguda and Thies Pfeiffer. 2012. Modeling math word problems with augmented semantic networks. In *International Conference on Application of Natural Language to Information Systems*, pages 247–252. Springer.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*.

Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. 2019. Reasoning over paragraph effects in situations. *arXiv preprint arXiv:1908.05852*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of ACL*.

Nicholas Locascio, Karthik Narasimhan, Eduardo DeLeon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 247–252. Association for Computational Linguistics.

I. Loshchilov and F. Hutter. 2017. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *EMNLP*.

Bhavana Dalvi Mishra, Lifu Huang, Niket Tandon, Wen tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: A challenge dataset and models for process paragraph comprehension. In *NAACL-HLT*.

Bhavana Dalvi Mishra, Niket Tandon, Antoine Bosselut, Wen tau Yih, and Peter Clark. 2019. Everything happens for a reason: Discovering the purpose of actions in procedural text. *ArXiv*, abs/1909.04745.

Ines Montani and Matthew Honnibal. 2018. Prodigy: A new annotation tool for radically efficient machine teaching. *Artificial Intelligence*.

Lidia Morawska, Donald K Milton, et al. 2020. It is time to address airborne transmission of covid-19. *Clinical Infectious Diseases*.

Benjamin Nye, Junyi Jessy Li, Roma Patel, Yinfei Yang, Iain J Marshall, Ani Nenkova, and Byron C Wallace. 2018. A corpus with multi-level annotations of patients, interventions and outcomes to support language processing for medical literature. In *ACL*.

Benjamin E Nye, Jay DeYoung, Eric Lehman, Ani Nenkova, Iain J Marshall, and Byron C Wallace. 2020. Understanding clinical trial reports: Extracting medical entities and their relations. *arXiv preprint arXiv:2010.03550*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Hoifung Poon, Chris Quirk, Charlie DeZiel, and David Heckerman. 2014. Literome: Pubmed-scale genomic knowledge base in the cloud. *Bioinformatics*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*.

Johannes Röhl. 2012. Mechanisms in biomedical ontology. In *Journal of Biomedical Semantics*. BioMed Central.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. In *Proceedings of NAACL-HLT 2018*.

Manuel Salvadores, Paul R Alexander, Mark A Musen, and Natalya F Noy. 2013. Bioportal as a dataset of linked biomedical ontologies and terminologies in rdf. *Semantic web*.

Isabel Segura Bedmar, Paloma Martínez, and María Herrero Zazo. 2013. Semeval-2013 task 9: Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013). ACL.

Min Joon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-reduction networks for question answering. In *ICLR*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

Gizem Soğancıoğlu, Hakime Öztürk, and Arzucan Özgür. 2017. Biosses: a semantic sentence similarity estimation system for the biomedical domain. *Bioinformatics*.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *NAACL*.

D. Swanson and N. Smalheiser. 1996. Undiscovered public knowledge: A ten-year update. In *KDD*.

Niket Tandon, Bhavana Dalvi Mishra, Joel Grus, Wen tau Yih, Antoine Bosselut, and Peter Clark. 2018. Reasoning about actions and state changes by injecting commonsense knowledge. In *EMNLP*.

Shyam Upadhyay and Ming-Wei Chang. 2017. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Karin Verspoor, K Bretonnel Cohen, Michael Conway, Berry de Bruijn, Mark Dredze, Rada Mihalcea, and Byron C Wallace. 2020a. Proceedings of the 1st workshop on nlp for covid-19 (part 2) at emnlp 2020. In *Workshop on NLP for COVID-19 at EMNLP 2020*.

Karin Verspoor, Simon Šuster, Yulia Otmakhova, Shevon Mendis, Zenan Zhai, Biaoyan Fang, Jey Han Lau, Timothy Baldwin, Antonio Jimeno Yepes, and David Martinez. 2020b. Covid-see: Scientific evidence explorer for covid-19 related research. *arXiv preprint arXiv:2008.07880*.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *EMNLP*.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *EMNLP*.

Byron C Wallace, Joël Kuiper, Aakash Sharma, Mingxi Zhu, and Iain J Marshall. 2016. Extracting pico sentences from clinical trial reports using supervised distant supervision. *The Journal of Machine Learning Research*.

Jingqi Wang, Huy Anh Pham, Frank Manion, Masoud Rouhizadeh, and Yaoyun Zhang. 2020a. Covid-19 signsym: A fast adaptation of general clinical nlp tools to identify and normalize covid-19 signs and symptoms to omop common data model. *arXiv preprint arXiv:2007.10286*.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Kinney, Ziyang Liu, William Merrill, et al. 2020b. Cord-19: The covid-19 open research dataset. *arXiv preprint arXiv:2004.10706*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1050–1055. AAAI Press/MIT Press.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pages 658–666.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLPCoNLL)*, pages 678–687. Association for Computational Linguistics.

Appendix A

MathQA Collection and Training

A.1 Human Annotation Guidelines

A.1.1 Annotation Platform

Collecting labels for mathematical solutions is a hard and time-consuming task. Due to the underlying difficult nature of solving arithmetic problems, human-labeled data collection for math word problem solving is limited to small problem sets. We design our annotation task to reduce the time and background knowledge needed for finding the mathematical solutions.

First, the solvable problems are extracted from AQuA dataset. The problem is unsolvable with operation programs if it does not results in numeric values, are of higher order of polynomial, requires options for finding the plausible answer, and problems that are only solvable by brute-force. In addition, there existed duplicated problems in original AQuA. Figure A.4 shows the percentages of each unsolvable category within train/dev/test sets in original AQuA.

By examining the solvable problems, we identified six major categories within the problem set (Geometry, Physics, Probability, Gain-loss, General arithmetic and Other). Solving the problems in one category mainly requires the operations defined for that category. Therefore, in order to minimize the search space for crowd-workers, we designed separate annotation tasks per category.

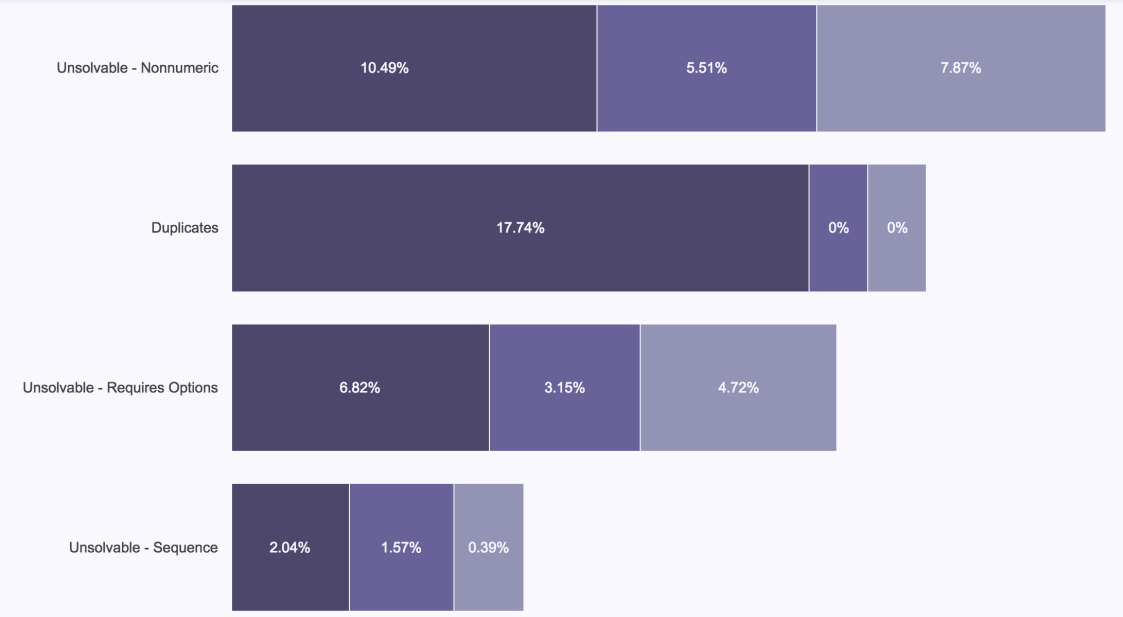


Figure A.1: Solvability of original AQuA dataset within train/dev/test sets.(in addition to these unsolvable problems, 35 problems in the train set did not contain words)

Each annotation task contained general arithmetic operations alongside the category based operations and constants. Figure A.2 shows the annotation platform. At each step the crowd-worker is enabled to select the most suitable operation by just a click. Afterward a list of plausible arguments for that operations are shown. In order to avoid dangling numbers, we only allow the arguments to be from: (1) the problem description, (2) the previous calculations, and (3) the common constants for each category (e.g. number π for geometry). Each time the operation is completed with its arguments the intermediate results are calculated and appeared as plausible argument in the next operations.

In addition the problem is allowed to be submitted only if the result of the last operation constructed matched the correct numeric value of the problem. We experiments with different setups on how much of the information should be revealed to the crowd-workers. Our experiments showed that hiding the final number resulted in the more accurate and logically correct solutions.

Category	#Annotated	#Validated	#Expanded
Geometry	2194	1816	3035
Physics	2287	2211	8136
Probability	689	655	3757
Gain-Loss	946	916	4213
General	3130	3056	17499
Other	643	636	1555

Table A.1: Annotation statistics

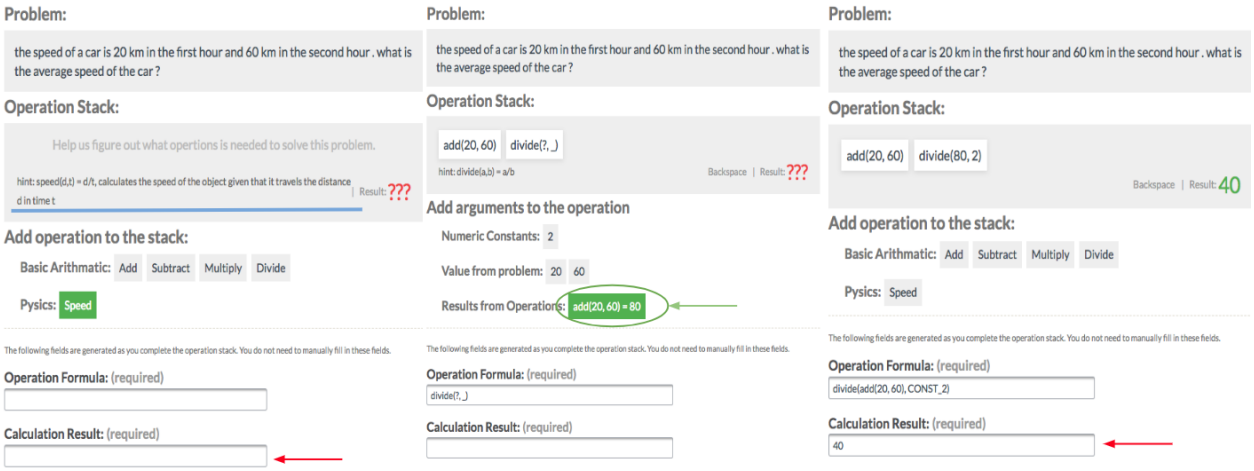


Figure A.2: A View of our human in the loop annotation task. Each view is designed based on the underlying mathematical domain (a). Since the operations in future steps might be dependant to the result of this step, after each calculation the result is added to the stack of possible arguments (b). Submission is only possible when they reach to close range of correct answer and we log their solution pattern in operation formula field (c)

A.1.2 Alignment Validation Case Study

Not all the mathematical solutions that results in correct numerical values are rationally correct. As illustrated in Figure A.3, the correct value representing the average of two numbers 20 and 60, can also be achieved by subtracting the two numbers. Though this operation sequence passes the checking mechanism employed at annotation task, such solutions are logically incorrect and are not expandable if the numeric values change. Therefore, another round of validation based on human judgement is necessary.

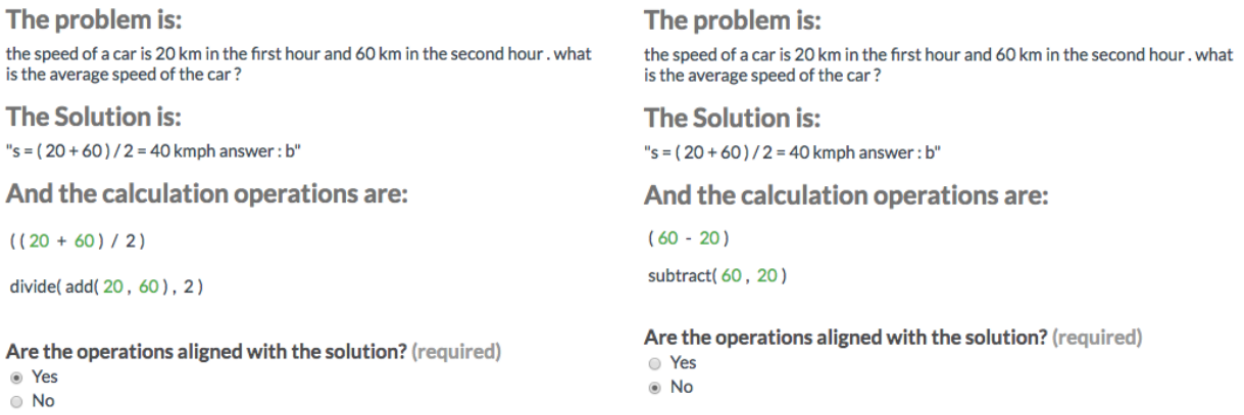


Figure A.3: A View of our human validation task. Two separate annotations are found for a problem where both result in correct numeric value. But the one on the right is logically wrong and should be mark as invalid.

We designed the annotation validation for identifying the wrong annotation that passed the annotations checking mechanism. In order to ensure the feasibility of the pipeline, we created the task with the goal of minimizing human labor. At each step the the labeled solution is presented to the human evaluator along with the problem description and the rationale. In addition, the intermediate results are shown by clicking at each section of the solution. (See Figure A.3). We repeated each validation task with three different annotators and kept the solutions that majority of the annotators marked as valid. Figure A.4 shows the statistics of the agreements between annotators in this phase.

Table A.1 shows the number of annotations collected for each category, the portion of those that are marked as valid in our pipeline, and the total number of the problems after expansion phase.

A.2 Automatic Annotation using Rationales

Each problem in the AQuA dataset is accompanied with a rationale, an explanation of the solution in natural language. Our automatic annotation strategy focuses on problems with rationales that are mostly accurate in their description of the problem solving process. Given a word problem x, we

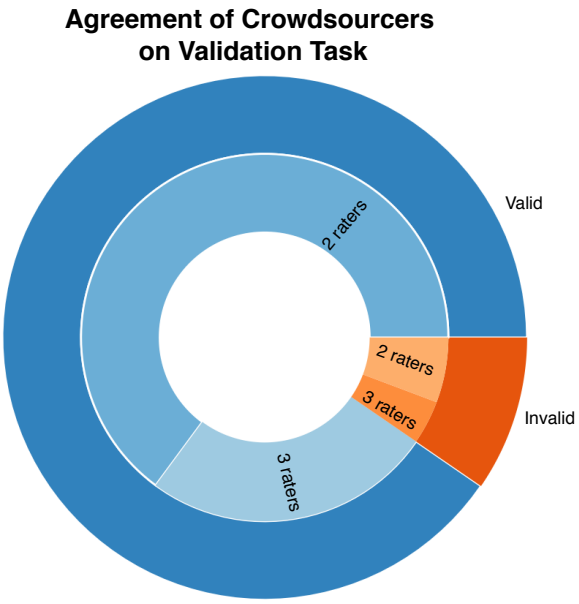


Figure A.4: Tracking of agreement among annotators on the validation task.

expect that the numbers mentioned in the rationale for x are of importance and can be used toward forming a corresponding operation program. In other words, we assume that each number in the rationale either directly appears in an operation program associated with x or can be calculated from previously mentioned numbers using operations in the program.

We design an algorithm that uses these assumptions to generate feasible operation programs from AQuA rationales. Our automatic annotation algorithm uses a dynamic programming approach to derive operation programs from the numbers in the rationale. Operations are selected sequentially. Some numbers in the rationale may be ignored, as long as the induced program leads to the correct final solution.

The algorithm takes in the numbers appearing in the problem, domain-specific constants and numbers calculated at previous timesteps as input. At each time step (t) we only search for combinations of operations and arguments that can result in the t^{th} number mentioned in the rationale. This process continues until the final answer is reached.

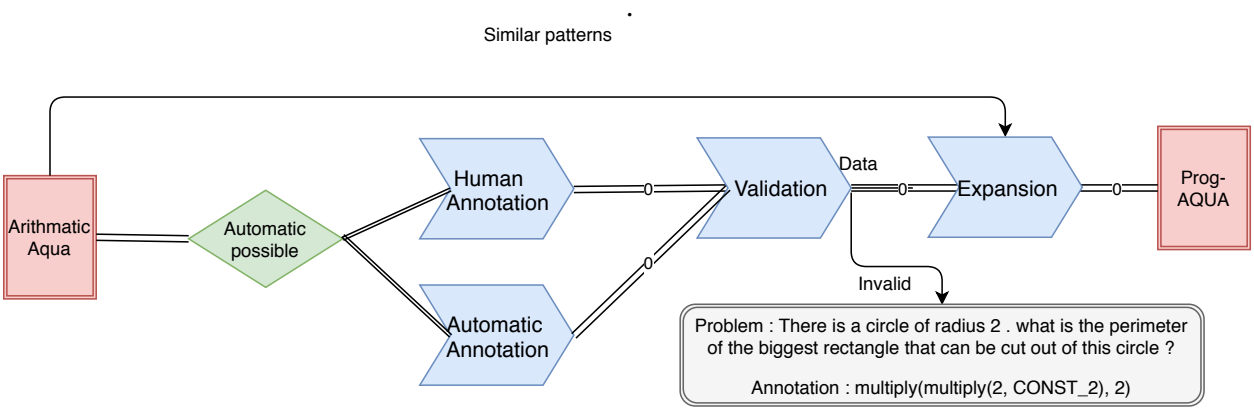


Figure A.5: Complete pipeline of data collection including the automatic solution generation step.

We keep the plausible paths that show how numbers in rationales are reproduced. We use the mathematical expressions described in the rationales to prune programs which do not use those expressions. Finally, we prune those paths that do not achieve the correct solution. We use automatic annotations for problems which lead to at most five operation programs.

Similar as the previous setup, we employ human annotators to validate automatically-annotated operation programs. The annotators check whether the problems and annotated programs are aligned or not. Due to the noise in the rationales, only 80.39% of those problems pass our human validation. This is mainly due to the fact that the rationales are not complete programs and fail to explicitly describe all important numbers and operations required to solve the problem. Figure A.5 shows the overall annotation pipeline including the automatic annotation generation step.

Appendix B

Mechanism Extraction and training

B.1 Data Annotation

B.1.1 Granular Relations

In addition to the two coarse-grained relation classes, we also experimented with *granular* relations, where the `class` represents a specific type of a mechanism relation explicitly mentioned in the text (we constrain the mention a single token for simplicity, e.g., *binds, causes, reduces*; see Figure B.1 for examples of granular relations). While more granular, these relations are also less general – as the natural language of scientific papers describing mechanisms often does not conform to this more rigid structure (e.g., long-range and implicit causal relations). We thus focus most of our work on coarse-grained relations. We release our dataset and a model for extraction of granular relations to support future research and applications, in our code repository.

B.1.2 Annotation Collection

We utilize the Prodigy (Montani and Honnibal [2018]) annotation platform which provides the ability to select span boundaries and relations with ease. Each annotator undergoes a training

Sentences	Granular relations
Viral infections probably **initiate** a large percentage of childhood and adult asthmatic attacks based on a history of preceding ' cold '.	(Viral infections, **initiate,** childhood and adult asthmatic attacks)
PUVA is useful to **treat** human platelet (PTL) concentrates in order to **eliminate** Leishmania spp.	(PUVA, **treat,** human platelet (PTL) concentrates) (PUVA, **eliminate,** Leishmania spp)

Figure B.1: Examples of granular relations.

session in which we cover the definitions of spans and relations as well as use of the platform. Each annotator is presented with a document consisting of definitions regarding the task. In each annotation task, each annotator is presented with a short text from a scientific paper. These texts discuss various entities of interest.

- **Naturally-occurring** entities include for example viruses, cells, organs. An entity can also be some activity or process for example the sequence of steps forming viral entry into a cell or cellular energy production.

- **Designed engineered** entities include for example devices and technologies (such as a hammer, an MRI machine, a software system, or a human-created chemical). It can also be a more abstract human-engineered construct such as methodologies and approaches (e.g., mathematical models, analysis techniques, or public health strategies).

Annotator's task is to find entities (call them "X and Y") that take part in one of the mechanism relations below.

Direct: explicit functional relations such as:

- X is “USED FOR / BY” Y
- X uses Y / X is used by/for Y, X is applied to Y, X needs / is based on Y

  These forms of relations may appear in examples such as:

  - ⋆ The virus makes use of spike protein to bind to a cell.
  - ⋆ Erlenmeyer flask designed to hold liquid.
  - ⋆ The algorithm requires a certain type of data.
  - ⋆ A data analysis method is applied to the data.
- X “DOES OPERATION / ACTION” Y
- X is an entity that is actively performing an operation/activity

  These forms of relations may appear in examples such as:

  - ⋆ The heart pumps blood.
  - ⋆ A virus binds to a cell.

Effect: Indirect/implicit, what general effects and observed correlations/associations entities have, not in the context of being explicitly a functional relation in used_for or do_operation. Where a greater “distance” is implied between *how* something causes/leads to something else; where the annotator’s understanding is that there may be an implicit functional relation. The examples may include:

- Smoking causes higher blood pressure.
- Smoking is associated with cancer.
- A virus leads to respiratory infection.

- Two genes/proteins appear to collocate often.

Additional remarks are also conveyed to the human annotators to improve the quality of labeled data:

Span Selection Remarks are as:

- Select spans that contain relevant information, do not leave important parts out, but do not include non-informative parts. When in doubt, select the longer span.
- Avoid including words like “the, this” in spans unless needed. Do mark the reference words if they refer to entities you previously annotated in the same text.
- Avoid annotating very generic terms on their own without context (e.g., “approach”, “method”, “result”).

Relation Selection Remarks are as:

- Imagine these relations as coming up in a search interface, would users be able to get sufficient information by viewing a relation?
- If X is RELATION TO ”Y and Z”, they may be marked as two separate relations. However, only break “Y and Z” down if the relation does not lose its meaning and can actually be decoupled into two separate entities.
- Note that the direction of the relations matter.

Each annotator attended an hour long session in which these definitions and remarks are reviewed and several examples are labeled interactively.

Table B.1 shows examples of differences between annotations, with disagreements in the span boundaries. This reflects the challenging nature of our task with relations between flexible, open entities.

Context	Annotator 1	Annotator 2
Predicted siRNAs should effectively silence the genes of SARS - CoV-2 during siRNA mediated treatment.	(predicted siRNAs, silence the genes of SARS - CoV-2, DIRECT)	(siRNAs, silence the genes of SARS - CoV-2 during siRNA mediated treatment, DIRECT)
Recent reports show that the inhibition of **NSP4** expression by small interfering RNAs leads to alteration of the production and distribution of other viral proteins and mRNA synthesis , suggesting that NSP4 also **affects virus replication** by unknown mechanisms.	(NSP4, affects virus replication, INDIRECT)	(NSP4, virus replication by unknown mechanisms, INDIRECT)

Table B.1: Examples of differences between two annotators. The core meaning of the relation is equivalent across both annotations.

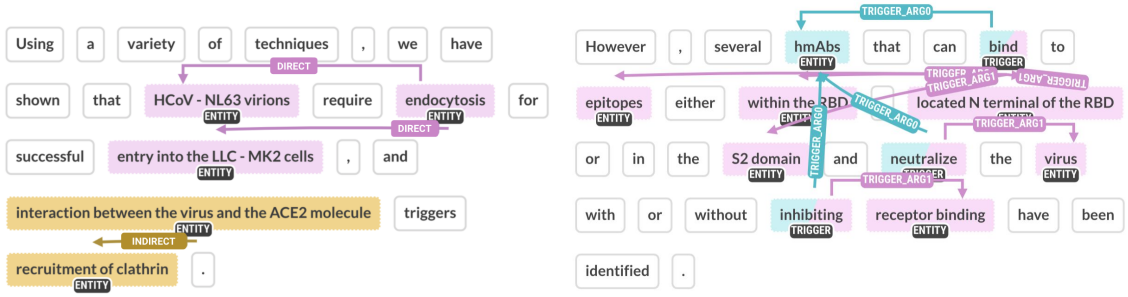


Figure B.2: Example of the annotation interface for coarse (left) and granular (right) mechanism relations.

B.1.3 Hyperparameter Search

We perform hyperparameter search over these sets of parameters:

- **Dropout** is randomly selected from intervals $[0, 0.5]$.
- **Learning rate** is randomly selected between $[1e-5, 1e-2]$.
- **Hidden Size** is randomly selected from interval $[64, 512]$.

Hyperparameter search is implemented using grid search with the Allentune library (Dodge et al. [2019]). For each experiment we set the search space to be among 30 total samples in hyperparameter space. We select the best-performing parameters using the development set.

B.1.4 Granular relation prediction

Granular relations are evaluated in the same fashion as coarse-grained relations, with the additional requirement that the predicted predicate token $\hat{p}$ must match the gold p^*. Our evaluation shows that the model trained to predict granular triples achieves $F1$ score of 44.0. When predicting relations without trigger labels (i.e., `(s, o)`), the model achieves $F1$ scores of 53.4. These results are not comparable to those for MECHANIC, which includes more documents and relations that did not directly conform to the `(s, o, p)` schema.

B.1.5 Best Performing Model over MECHANIC-G

Here too, we use the DYGIE package (Wadden et al. [2019]) with SciBERT (Beltagy et al. [2019]). Due to technical equivalence in the annotation schema of our granular relations and the event extraction task in (Wadden et al. [2019]), we make use of the event extraction functionality of DYGIE. For fine-tuning the embedding weights of SciBERT, we used the same learning rate weight as for MECHANIC, and the best hyperparameters found are 0.30, $11e-4$ and 372 for dropout, learning rate and hidden layer size respectively. All other parameters are kept to default values (available in our code repository).

B.2 Human evaluation guidelines

B.2.1 KB Correctness and Informativeness evaluation guideline:

Relation quality evaluations over various domains For the task involving the exploration of viral mechanisms, we used 10 recent scientific claims taken from (Wadden et al. [2020]). These 10 claims, and the queries constructed for them, are as follows:

- Remdesevir has exhibited favorable clinical responses when used as a treatment for coronavirus. X = [Remdesevir], Y = [SARS-CoV-2, coronavirus, COVID-19]

- Lopinavir / ritonavir have exhibited favorable clinical responses when used as a treatment for coronavirus. X = [Lopinavir, Ritonavir], Y = [SARS-CoV-2, coronavirus, COVID-19]
- Aerosolized SARS-CoV-2 viral particles can travel further than 6 feet. X = [Air, Aerosols, Droplets, Particles, Distance], Y = [SARS-CoV-2 transmission]
- Chloroquine has shown antiviral efficacy against SARS-CoV-2 in vitro through interference with the ACE2-receptor mediated endocytosis. X = [Chloroquine], Y = [ACE2-receptor, Endocytosis, interference with the ACE2-receptor mediated endocytosis]
- Lymphopenia is associated with severe COVID-19 disease. X = [Lymphopenia], Y = [severe COVID-19 disease, COVID-19]
- Bilateral ground glass opacities are often seen on chest imaging in COVID-19 patients. X = [Bilateral ground glass opacities], Y = [chest imaging in COVID-19 patients]
- Cardiac injury is common in critical cases of COVID-19. X = [COVID-19], Y = [Cardiac injury]
- Cats are carriers of SARS-CoV-2. X = [Cats], Y = [SARS-CoV-2]
- Diabetes is a common comorbidity seen in COVID-19 patients. X = [Diabetes], Y = [COVID-19]
- The coronavirus cannot thrive in warmer climates. X = [warmer climates], Y = [coronavirus]
- SARS-CoV-2 binds ACE2 receptor to gain entry into cells. X = [SARS-CoV-2], Y = [binds ACE2 receptor, binds ACE2 receptor to gain entry into cells]

For the AI open-ended search task, we used the following approaches/areas as queries (see guidelines and examples in our code repository): *artificial intelligence, machine learning, statistical models, predictive models, Graph Neural Network model, Convolutional Neural Network model,*

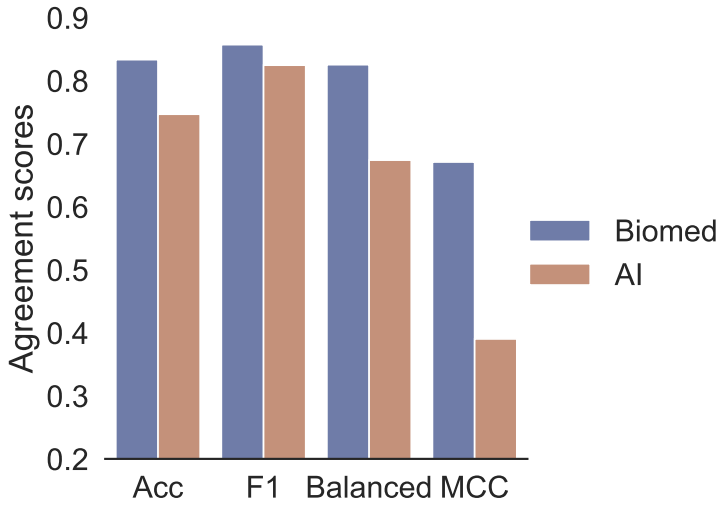


Figure B.3: Average pairwise annotator agreement by several metrics. In the AI task human labels were more diverse but with overall high precision / recall.

Recurrent Neural Network model, reinforcement learning, image analysis, text analysis, speech analysis.

For both tasks, we use the following metrics to measure pairwise agreement between annotators (Figure B.3): standard accuracy (proportion of matching rating labels), F1 (taking into account both precision and recall symmetrically), balanced accuracy (with class weights to down-weight the higher proportion of positive ratings), and finally the Matthew Correlation Coefficient (MCC) score, using the corresponding functions in the Scikit-Learn Python library (Pedregosa et al. [2011]).

B.2.2 KB Utility

MDs are instructed to search with our interface and with PubMed search engine, for the following 7 topics:

- Query 1: Cardiac arrhythmias caused by COVID 19
- Query 2: Hydroxychloroquine and its effect on COVID 19
- Query 3: Ivermectin and its role in management of COVID 19

- Query 4: Pulmonary embolism effect on complications related to COVID 19
- Query 5: Liver disease and COVID 19
- Query 6 : Inflammatory bowel disease and COVID -19
- Query 7 : Antibody therapy and its uses/effects on COVID-19

The full list of the post-study evaluation questions given to MDs is shown in Figure B.4.

Topic	Question
IR	The overall search results accurately matched the query.
IR	I was satisfied by the overall ranking of results.
IR	I found results that were both relevant and interesting or new to me, making this system useful for rapid explorations.
IR	I wanted to read the papers shown.
overall	**Overall, I am satisfied with this system.**
UI/UX	Overall, I am satisfied with how easy it is to use this system.
UI/UX	It was simple to use this system.
UI/UX	I felt comfortable using this system.
UI/UX	The information (such as on-screen messages and other documentation) provided with this system was clear.
UI/UX	The organization of information on the system screens was clear.
UI/UX	The system gave error messages that clearly told me how to fix problems.
UI/UX	Whenever I made a mistake using the system, I could recover easily and quickly.
UI/UX	The interface of this system was pleasant.
UI/UX	I liked using the interface of this system.
UI/UX	It was easy to learn to use this system.
utility	I was able to understand and judge each individual result quickly and without effort.
utility	I was able to complete the tasks and scenarios quickly using this system.
utility	The information was effective in helping me complete the tasks and scenarios.
utility	I believe I could become productive quickly using this system.
utility	It was easy to find the information I needed.
utility	This system has all the functions and capabilities I expect it to have.

Figure B.4: List of post-study questions given to MDs.