INSTITUTE FOR DEFENSE ANALYSES

Generating the CausX TA1 Assessment Data

Peter A. Kind, *Project Leader*
Susan K. Numrich
Steven P. Wartik

November 2020

IDA Document
D-15381
Copy

INSTITUTE FOR DEFENSE ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

The Institute for Defense Analyses is a nonprofit corporation that operates three Federally Funded Research and Development Centers. Its mission is to answer the most challenging U.S. security and science policy questions with objective analysis, leveraging extraordinary scientific, technical, and analytic expertise.

About This Publication

This work was conducted by the IDA Systems and Analyses Center under contract HQ0034-14-D-0001, Project DA-5-4320, "Causal Exploration of Complex Operational Environments," for the DARPA. The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

Acknowledgements

Keith L. Green, Dale Visser

For More Information

Peter A. Kind, Project Leader
pkind@ida.org, 703-845-6657

Margaret E. Myers, Director, Information Technology and Systems Division
mmyers@ida.org, 703-578-2782

Rigorous Analysis | Trusted Expertise | Service to the Nation

INSTITUTE FOR DEFENSE ANALYSES

IDA Document D-15381

Generating the CausX TA1 Assessment Data

Peter A. Kind, *Project Leader*

Susan K. Numrich

Steven P. Wartik

Executive Summary

DARPA's Causal Exploration in Complex Environments (CausX) program involves extracting information on events and causal relationships between events from natural-language documents. Four companies participate in this extraction process: BBN, ISI, LCC, and Lum.AI.[1] They are termed *TA1 performers*, with the numeral "1" indicating their early role in the processing pipeline.

The Institute for Defense Analyses (IDA) team's 2020 assessment of TA1 performer product quality required selecting causal assertions and events from the data sets created by the performers' extraction applications. The IDA team divided the assessment into four comparison categories:

1. Causal assertions found by exactly one TA1 performer (BBN, ISI, LCC, or Lum.AI).
2. Causal assertions found by exactly two performers (BBN, ISI, and LCC). (Lum.AI, a relative latecomer, was not included.)
3. Causal assertions found by BBN, ISI, and LCC. (Lum.AI was not included.)
4. Events, both from and not from causal assertions. (Lum.AI extracts no events in the latter category.)

To ensure that performers would be evaluated by their best work, the IDA team developed an approach to select the highest-quality causal assertions and events. The team based quality on the degree to which TA2s (the set of performers in the processing pipeline after TA1s) find events and causal assertions useful. So-called generic events—those at the highest level of the CausX event ontology's event class hierarchy—were considered less desirable than events lower in the hierarchy. An event with factors, actors, or locations was considered higher quality than an event without them, and an event with all was considered better still. A causal assertion's quality was judged based on the quality of its events.

The IDA team obtained the TA1 data sets in the form of SQLite databases. The process of selecting causal assertions and events for assessment was complex and time consuming. The team wanted other parties to be able to review selection quality and therefore wanted to make the process repeatable. This document describes the process,

[1] See, respectively, https://www.raytheon.com/ourcompany/bbn, https://www.isi.edu/, http://www.languagecomputer.com/, and https://lum.ai/.

which is presented as a sequence of Linux commands. Each command is presented and explained. Executing these commands will re-generate the assessment data to the extent possible. (There is some use of random numbers, so re-generating the exact data set the IDA team used is unlikely.) The commands require a Linux environment running Python 3.6 or higher and the mysql database server.

Contents

Figures and Tables

1. Introduction

This document describes how the Institute for Defense Analyses (IDA) team generated the data used to assess the TA1 performers: BBN, ISI, LCC, and (to a lesser extent) Lum.AI. It discusses how the IDA team converted the data from the format in which it was delivered to them into spreadsheets used by the IDA team's coders.

The process, such as it was, often arose in response to immediate needs. Thus, it was never optimized and is sometimes redundant. It includes code that, like most code, has documentation that could be improved. However, it is effective, in the sense that it produced the necessary outputs and is repeatable.

A. Objective

The IDA team's objective was to assess the quality of causal assertions and events extracted from a sample corpus of 8,000 documents. By comparing and contrasting the causal assertions, IDA hoped to discover each performer's strengths and weaknesses.

B. Inputs

The IDA team received databases from Two Six Labs in the SQLite format. Two Six Labs supplied one database for each performer. Each database contained the performer's results of processing the sample corpus.

All databases used a schema established by Two Six Labs. BBN, ISI, and LCC submitted data in an established JavaScript Object Notation (JSON) format; Two Six Labs processed this data and converted it to the SQLite format (among others). The data bears some resemblance to the format used in previous triple store representations.

- Assertions and events are identified by URLs.
- Class hierarchies used in the triple store are used in the database. For example, as in the triple store, an event has a type, and this type is one of the subclasses of class `EventOrFactor`.

The significant structural difference, of course, is the absence of predicates. In the triple store, an attribute of an individual is specified through a triple with the individual as the subject, the attribute value as the object, and a predicate defining which attribute the triple is specifying. In the database, each row has primary key columns and one or more additional columns. The primary key columns correspond to the subject. Additional columns correspond to attributes. Each column name corresponds to a predicate. Two

tables may also be related, either explicitly or implicitly; this is analogous to triples whose subject and object are both individuals rather than literal values.

The SQLite databases also lack the provenance information that was present in the triple store, where every causal assertion, event, actor, and location individual was part of a triple that indicated the sentence from which the individual was extracted. In the SQLite databases, this information is represented in the table for events by including the complete sentence, with the text of the event enclosed in double square brackets. Everything has a relationship to an event. An assertion has an antecedent event and a consequent event. Events have actors and locations, as well as factors. The SQLite database only locates where in a sentence an extraction occurred for assertions and events. If a sentence contains multiple instances of "Russia" (and, for one performer, 1,267 of them do), it is not possible to determine which one triggered the extraction. This information was not judged necessary for the assessment, and the IDA team did not ask Two Six Labs to include it.

C. Environment

The IDA team generated the assessment data on a Windows 10 platform and used some free, open-source tools to generate the data:

- We used MySQL version 8.0.20 (community edition) as the database management system.
- We created a Python application to convert the SQLite databases to MySQL. We used Python version 3.6, and the application used the sqlite3 package built into the standard library.
- Many of the steps were performed using command-line tools. We used the bash shell, running in the Cygwin environment.[2]

Examples in this document are presented using bash. This is an arbitrary decision—the cmd shell in Windows could also have been used. For that matter, MySQL and Python both run on Linux, macOS, and Windows, so the steps can be carried out on any of those platforms. The process has also been executed on an Ubuntu Linux 18.04 platform. Some of the scripts contain absolute file names; these may need to be modified even if run on Windows (see Appendix A).

[2] See https://cygwin.com/.

2. Process

This section provides a step-by-step description of the process the IDA team followed to generate assessment data from the inputs supplied by Two Six Labs. In a nod to rational design processes,[3] we admit that we did not strictly adhere to the process below. What is presented is a repeatable process that, in hindsight, is better organized than what we actually did and produces the same results.

A. Step 1: Obtain Data

Two Six Labs made the data available on the U.S. Air Force's Virtual Distributed Laboratory (VDL)[4] in the following directory:

```
/programs/CauseEx/CauseEx Performers/TA5_SI/TA1 Experiments/
```

Download the following files:

- bbn_8k_20200518.db.gz
- isi_8k_20200601.db.gz
- lcc_8k_20200716.db.gz
- lum_8k_20200608.db.gz

B. Step 2: Uncompress Data

The database files Two Six Labs delivers are large and often exceed 1GB, so Two Six Labs compressed them prior to upload. In the directory to which you downloaded the files, execute the following commands:

```
$ gunzip bbn_8k_20200518.db.gz
$ gunzip isi_8k_20200601.db.gz
$ gunzip lcc_8k_20200716.db.gz
$ gunzip lum_8k_20200608.db.gz
```

Your directory should contain these four files with the ".gz" suffix removed. These files are interpretable as SQLite data.

[3] D. Parnas and S. Clements, "A Rational Design Process: How and Why to Fake It." *IEEE Transactions on Software Engineering* Vol. SE-12, Issue 2, February 1986.

[4] https://restricted.vdl.afrl.af.mil/

C. Step 3: Convert Databases from SQLite to MySQL

Python's sqlite3 package provides the capability to dump the contents of an SQLite database into a text file containing a series of SQL `INSERT INTO` statements. These statements are not standard SQL format (or, at any rate, are not accepted by MySQL).

The IDA team developed a simple Python application to convert the four databases from SQLite to MySQL. This script, dump-db.py, is executed as follows:[5]

```
$ python3 dump-db.py sqlite-database script-file
```

The `sqlite-database` parameter is one of the databases provided by Two Six Labs, and `script-file` is the file into which the output is written.

The commands to execute are as follows:

```
$ python3 dump-db.py bbn_8k_20200518.db bbn-data.sql
$ python3 dump-db.py isi_8k_20200601.db isi-data.sql
$ python3 dump-db.py lcc_8k_20200716.db lcc-data.sql
$ python3 dump-db.py lum_8k_20200608.db lum-data.sql
```

D. Step 4: Create a MySQL Database for Each Performer

Data for each performer is placed in a separate MySQL database. Execute the following commands:[6]

```
$ mysqladmin --user=user --password create bbn8k create isi8k \
      create lcc8k create lum8k
```

Here, *`user`* must be the name of a user with the right to create databases on your MySQL server. The command will prompt for a password, which will have been set by the MySQL administrator (i.e., whoever installed MySQL).

This and subsequent MySQL commands assume you are running a MySQL server on the same platform as these shell commands. If you are not, you need to use the `--host` flag. Furthermore, this and subsequent MySQL commands assume the specified user has adequate access rights.

E. Step 5: Create Schemas

Each database created in Step 4 should contain the set of tables into which data will be uploaded. These tables are structurally similar to those in the SQLite databases, but they are optimized for MySQL.

Execute the following commands:

---

[5] As mentioned in Section 1.C, we used Python version 3.6. Elsewhere, we used 3.8. On some systems, there is no minor version number; "python3" accesses the latest version of Python 3. In this section, "python3" means version 3.6 or higher.

[6] The backslash at the end of the first line means the command continues on the next line.

```
$ mysql --user=user --password bbn8k < schema-myisam.sql
$ mysql --user=user --password isi8k < schema-myisam.sql
$ mysql --user=user --password lcc8k < schema-myisam.sql
$ mysql --user=user --password lum8k < schema-myisam.sql
```

The IDA team used MySQL's MyISAM table format rather than the newer InnoDB. We found we could execute queries faster with MyISAM. MyISAM does not support transactions or foreign keys and is not well suited to situations in which multiple individuals are simultaneously querying and modifying tables. If the latter describes your work environment, you should consider switching to InnoDB.

F. Step 6: Populate Schemas

Load the data created in Step 3 into your MySQL databases:

```
$ mysql --user=user --password bbn8k < bbn-data.sql
$ mysql --user=user --password isi8k < isi-data.sql
$ mysql --user=user --password lcc8k < lcc-data.sql
$ mysql --user=user --password lum8k < lum-data.sql
```

Do not be surprised if some of these commands take time to complete. The IDA team observed them requiring the better part of two days.

G. Step 7: Create Supporting Databases

Although it is possible to create queries that retrieve data in a single step, it would be highly inefficient. Storing intermediate results greatly reduce both query complexity and processing time.

Create two databases using the following command:

```
$ mysqladmin --user=user --password create analysis create sentences
```

The sentences database contains information on causal assertions, events, and the sentences from which they are extracted. The analysis database partitions these causal assertions and events into categories useful for generating assessments. It also contains weighting tables useful in judging causal assertion and event quality.

Placement of sentence-related tables in the analysis database is, frankly, somewhat arbitrary. The sentences database was created with the intent to populate its tables early in the assessment process and to ensure its contents would not change thereafter. The analysis database was intended to be more mutable, although many of its sentence-related tables remained unchanged as well.

H. Step 8: Create Schema for the Sentences Database

As in Step 5, you need to create the tables in the sentence database. These tables are highly indexed in an attempt to shorten query times. Execute the following command:

```
$ mysql --user=user --password sentences < sentences-myisam.sql
```

I. Step 9: Populate the Sentences Database

Execute the following commands:

```
$ mysql --user=user --password < pt-01a-populate-ca-sentences.sql
$ mysql --user=user --password < pt-01b-populate-event-sentences.sql
$ mysql --user=user --password < \
      pt-01c-populate-ca-sentences-lum.sql
$ mysql --user=user --password < \
      pt-01d-populate-event-sentences-lum.sql
```

These commands aggregate the causal assertion tables in all four performer databases into a single table and do likewise for the event tables. Furthermore, although it is useful to create an index on sentence text, some sentences exceed MySQL's maximum permitted index length. These queries work around this by generating an SHA1 hash of the sentence text and indexing that.

J. Step 10: Identify 30 Sentences for Coder Calibration

The IDA team began its coding activities with a short guidebook of coding standards. We assumed these standards, which had never been exercised, were a prototype that needed to be tried and adjusted prior to the full assessment. We had six coders analyze the same set of 30 sentences. These sentences were chosen by executing the following command:

```
$ mysql --user=user --password < pt-02-30-sentences.sql
```

This creates and populates a table, `thirty_sentences_with_ca_s`, in the analysis database. Each row of the table identifies a sentence and the document containing it. Each of these sentences has the following characteristics:

1. BBN, ISI, and LCC all extracted at least one causal assertion from it. (At the time we performed this step, we had not been directed to include Lum.AI in our assessment.)
2. None of the causal assertions have generic antecedent or consequent events.
3. The sentence occurred in exactly one document.

Characteristic 2 was determined by testing whether the value in the `event_type` column was `http://ontology.causeex.com/ontology/odps/Event#Event`, the most general kind of event and one that TA2 performers generally consider of little or no value. The databases allow an event to have an optional second type, specified in the event_type2 column. The three performers all used this second type (although not often), and sometimes the value of this column is the general event. BBN and LCC provide a more specific kind of event in `event_type`, and the reason for using `event_type2` is unclear. ISI only uses the general event in `event_type2` when `event_type` is a collection of events. ISI, then, is sometimes able to identify an event as a collection without being able to determine the nature of elements in the collection.

The sentences were selected randomly from the set that met these criteria. That is, 187 sentences met the criteria, and a random number generator was used to select 30 of them. This step is therefore not entirely repeatable. Executing the query always yields 30 sentences, but it is unlikely to yield the same set. This randomization is necessary because MySQL retrieves rows in the order in which they were entered, and the data given to the IDA team had the rows grouped by document. Without randomization, the sentences would have been from a small set of documents, not randomly chosen from the entire set.

When this query was written, we knew that some sentences occur in multiple documents, but we had not realized that some documents contain multiple instances of the same sentence. Fortunately, the sample set did not include any duplicate sentences.

K. Step 11: Identify 30 Causal Assertions from the 30 Sentences

The next step is to choose causal assertions from the 30 sentences identified in Step 10. A performer may extract more than one causal assertion from a single sentence; we wanted to assess exactly one. Execute the following command:

```
$ mysql --user=user --password < pt-03-cas-from-30-sentences.sql
```

This creates a table, `thirty_cas_from_thirty_sentences`, containing all the sentences from the table in Step 10, plus three randomly chosen causal assertion IDs (one for each performer).

L. Step 12: Produce Causal Assertion and Event Data for Coders

IDA coders were presented with Excel spreadsheets containing causal assertions and events to assess. Creating those spreadsheets was a multi-step process. The first step was to extract information on the causal assertions and events. Execute the following commands:

```
$ mysql --user=user --password < pt-04a-assertions-data-bbn.sql
$ mysql --user=user --password < pt-04b-assertions-data-isi.sql
$ mysql --user=user --password < pt-04c-assertions-data-lcc.sql
```

Each of these commands causes the MySQL server to produce a comma-separated value (CSV) file containing results for a single performer. Each query file specifies the name and location of that file. The value given is server-specific. In the query files distributed, the files are placed in folder `C:\ProgramData\MySQL\MySQL Server 8.0\Uploads`.

Different versions of MySQL, or versions installed on different platforms, may place the files in different folders. These three query files, and other such files, may need editing to run on other computers.

After executing these queries, the folder in which the files are placed should contain three files whose names begin with 04.

M. Step 13: Produce Event Factor and Location Data for Coders

Some events have associated locations, and associated causal factors. To generate this data, execute the following commands:

```
$ mysql --user=user --password < pt-05a-factors-locations-bbn.sql
$ mysql --user=user --password < pt-05b-factors-locations-isi.sql
$ mysql --user=user --password < pt-05c-factors-locations-lcc.sql
```

These commands cause the MySQL server to produce CSV files whose names begin with 05.

When the queries in Steps 12–14 were written, we were still making sense of the data and were keeping queries simple. It is possible to combine these three steps; in fact, we did so later.

N. Step 14: Produce Event Actor Data for Coders

Some events have associated actors. To generate this data, execute the following commands:

```
$ mysql --user=user --password < pt-06a-actors-bbn.sql
$ mysql --user=user --password < pt-06b-actors-isi.sql
$ mysql --user=user --password < pt-06c-actors-lcc.sql
```

These commands produce CSV files whose names begin with 06. The files will be created in directory `C:\Program Data\MySQL\MySQL Server 8.0\Uploads`. The three query files (the ones whose names begin with `pt-06`) each specify this location; if it is not valid on your computer, change the query files.

O. Step 15: Stitch Together Data from Steps 12–14

Performing Steps 12–14 yields nine files, three each for BBN, ISI, and LCC. In a folder containing these files (either the folder from the previous step or one into which you have copied those files), execute the following commands:

```
$ python3 merge-csvs.py 0[456]a* > bbn-30-calib-sents.csv
$ python3 merge-csvs.py 0[456]b* > isi-30-calib-sents.csv
$ python3 merge-csvs.py 0[456]c* > lcc-30-calib-sents.csv
```

Files `bbn-30-calib-sents.csv`, `isi-30-calib-sents.csv`, and `lcc-30-calib-sents.csv` now contain the assertion, event, factor, location, and actor data for BBN, ISI, and LCC, respectively.

Be careful about opening these files in Excel. The files use the UTF-8 character set encoding, whereas Excel, especially older versions, uses Microsoft's Windows-1252 encoding. You may find that text is not translated properly and contains nonsensical characters. (Be forewarned that there are some nonsensical characters anyway, but you will encounter more if you use an encoding other than UTF-8.) To use the files, follow these steps:

1. Open Excel and either create a new workbook or open an existing one.
2. On the Data ribbon, in the Get & Transform Data section, click From Text/CSV. A window will pop up.
3. In the File Origin menu, select 65001: Unicode (UTF-8).
4. Click the Load button.

Excel will load data in your file into a new tab with the characters correctly encoded.

P. Step 16: Find Sentences that Occur in Only One Document

This step generates a table identifying sentences that occur in only one document. It was defined because of the realization of the prevalence of such sentences. The more often a sentence occurs in multiple documents, the more likely it is boilerplate like the following statement:

> In particular, the content of this site may not be disseminated, copied, made available to third parties, saved, used or altered without prior consent from dpa.

Execute the following command:

```
$ mysql --user=user --password < pt-07-sentences-not-shared.sql
```

This yields a table called `unshared_sentences`. The table identifies sentences not in any other document, the document they are in, and the sentence's offset within the document. If a sentence occurs more than once in a document, the first occurrence is chosen. Sentences used in calibration (Step 10) are not considered.

Q. Step 17: Find Sentences from Which BBN, ISI, and LCC Have Extracted Causal Assertions

This step creates a table containing a subset of the sentences identified in Step 16, those from which BBN, ISI, and LCC have all extracted causal assertions. These causal assertions are unique to a document and are not from the set of sentences identified in Step 10. Execute the following command:

```
$ mysql --user=user --password \
      < pt-08-sentences-with-CAs-from-3-performers.sql
```

R. Step 18: Produce Assessment Data for Sentences in Which BBN, ISI, and LCC All Found Causal Assertions

IDA coders were assigned 70 sentences in which all three performers identified causal assertions. This step generates those sentences. Execute the following commands:

```
$ mysql --user=user --password < pt-09a-nongeneric-CAs-bbn.sql
$ mysql --user=user --password < pt-09b-nongeneric-CAs-isi.sql
```

```
$ mysql --user=user --password < pt-09c-nongeneric-CAs-lcc.sql
```

The MySQL server will create three CSV files, one for each performer, whose names begin with 09.

Like Step 10, the queries in this step discriminate against general events, but they use a more sophisticated approach. The IDA team discovered that there were not enough causal assertions to eliminate general events entirely. Moreover, it was agreed that events with associated factors were preferred to those without. The causal assertions in the sentences were ranked as follows:

- The event types were scored as follows:
  - A causal assertion whose antecedent and consequent events were both general had a score of 0.
  - A causal assertion whose antecedent or consequent event (but not both) was general had a score of 2.
  - A causal assertion whose antecedent and consequent events were not general had a score of 4.
- If a causal assertion's antecedent event had one or more factors, it received an additional point.
- If a causal assertion's consequent event had one or more factors, it received an additional point.

Causal assertions were thus ranked on a scale of 0 to 6. Within groups of identical scores, causal assertions were selected randomly.

S. Step 19: Identify Shared Sentences and Select a Representative

Queries began to overlap. They all had to account for sentences that occurred in multiple documents and multiple times within the same document. We solved this problem by identifying for each sentence a single document and a single place in text (an integer offset of characters from the start) within that document and then creating a table containing the sentence, a representative document, and a representative place in text. To create and populate this table, execute the following command:

```
$ mysql --user=user --password \
      < pt-10-representative-shared-sentences.sql
```

This command creates a table called `representative_shared_sentences`. It also creates a view, `candidate_sentences`, that retrieves the union of `unshared_sentences` (Step 16) and `representative_shared_sentences`. Together, these tables supply all the sentences used in subsequent analysis.

T. Step 20: Identify Sentences with Causal Assertions from Two Performers

Coders assessed sentences with causal assertions from two of the three performers in an attempt to understand whether the third performer was more discriminating or insufficiently selective. This step generates a table of those sentences. Execute the following command:

```
$ mysql --user=user --password \
      < pt-11-sentences-with-CAs-from-2-performers.sql
```

The result is a table named `sentences_with_ca_s_from_two` in the analysis database. This table contains all sentences from which exactly two of the performers (BBN, ISI, and LCC) extracted causal assertions. Each row identifies the two performers.

U. Step 21: Identify Two-Performer Sentences to Assess

Step 20 identified all sentences from which two performers extracted causal assertions. Step 21 selects 100 sentences for each performer combination. Execute the following commands:

```
$ mysql --user=user --password \
      < pt-12a-create-selected-two-performer-sentences-table.sql
$ mysql --user=user --password \
      < pt-12b-selected-two-performer-sentences-bbn-isi.sql
$ mysql --user=user --password \
      < pt-12c-selected-two-performer-sentences-isi-lcc.sql
$ mysql --user=user --password \
      < pt-12d-selected-two-performer-sentences-bbn-lcc.sql
```

These commands create a table, `selected_two_performer_sentences`, and populate it with 300 rows, 100 for each of commands 2, 3, and 4. Sentences are selected by weighting each performer combination based on the number of general events in all assertions extracted from a sentence divided by the total number of assertions extracted from the sentence.

V. Step 22: Produce Assessment Data for Two-Performer Sentences

Each IDA coder was assigned 100 sentences from which exactly two performers had extracted causal assertions. This step uses the results from Step 21 to produce CSVs containing those sentences. Execute the following commands:

```
$ mysql --user=user --password \
      < pt-13a-create-assertion-functions.sql
$ mysql --user=user --password \
      < pt-13b-2-performer-CAs-bbn-isi.sql
$ mysql --user=user --password \
      < pt-13c-2-performer-CAs-isi-lcc.sql
$ mysql --user=user --password \
      < pt-13d-2-performer-CAs-bbn-lcc.sql
```

The upload directory will contain three CSV files whose names begin with 13.

Regarding the first command, the queries developed so far, particularly those used to download data, had some patterns that would reoccur in yet-to-be-developed queries. They were encapsulated in MySQL user-defined functions. The first command adds 11 functions to the analysis database. Nine of these functions encapsulate how to extract assertion-related text: cues, antecedents, and consequents. Each performer used their own variations on the standard. We needed separate functions for BBN, ISI, and LCC.

W. Step 23: Create Table of Sentences from Which Only One Performer Extracted Causal Assertions

IDA coders assessed sentences from which exactly one of the performers (BBN, ISI, or LCC) extracted one or more causal assertions. The intent was to determine if one of the performers has a comparative advantage in identifying extractable information. This step creates and populates a table of such sentences. Execute the following command:

```
$ mysql --user=user --password \
      < pt-14-sentences-with-CAs-from-1-performer.sql
```

The analysis database now contains table `sentences_with_ca_s_from_one`. Each row identifies a sentence and the performer who extracted causal assertions from it.

X. Step 24: Produce Assessment Data for One-Performer Sentences

Each IDA coder was assigned 100 sentences from which exactly one performer had extracted causal assertions. This step uses the table from Step 23 to produce CSVs containing those sentences. Execute the following commands:

```
$ mysql --user=user --password < pt-15a-1-performer-CAs-bbn.sql
$ mysql --user=user --password < pt-15b-1-performer-CAs-isi.sql
$ mysql --user=user --password < pt-15c-1-performer-CAs-lcc.sql
```

The upload directory will contain three files whose names begin with 15. These files contain all sentences from which only one performer extracted causal assertions, as well as information about the causal assertions extracted.

Each row of the CSV describes a single causal assertion. If a performer extracts multiple causal assertions from a single sentence, all causal assertions will appear in separate rows. Some IDA coders noted that they were repeatedly asked to analyze extractions from the same sentence. The queries return the rows in random order, and (in practice) the number of times a coder had to assess results from the same sentence was small.

Y. Step 25: Load Class Depth Data

This step loads data needed to improve the quality of event and causal assertion assessment. Execute the following command:

```
$ mysql --user=user --password < pt-16-load-depth-data.sql
```

Previous steps assessed event quality based on whether or not the event was generic—that is, whether column `event_type` in an `event` table had the value `http://ontology.causeex.com/ontology/odps/Event#Event`. There are two problems with this approach. First, it is incomplete. Class `Event` is a subtype of class `EventOrFactor`, and the value of the `event_type` column can be any subtype of `EventOrFactor`. `Event` is not the only generic class to consider.

Second, the approach is coarse. The hierarchy of possible event types is six layers deep. It is better for a performer to assign a type that is a leaf of the hierarchy tree rather than the leaf's parent or grandparent.

The IDA team ultimately used the following approach in subsequent steps. Figure 2-1 shows a portion of the hierarchy of types that may be assigned to an event. Each class in the picture has a depth from the root, `EventOrFactor`, measured by the length of the path (number of arrows) from the root to that class. `EventOrFactor` has depth 0. `Event`, `Factor`, and `Loan` have depth 1. `Accident` and classes in its row have depth 2. `Collision` and classes in its row have depth 3.

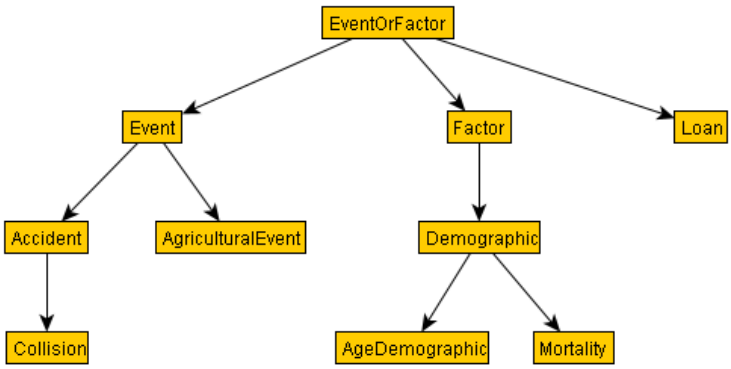

Figure 2-1. Event-or-Factor Hierarchy (Simplified)

Each class receives a weight, which is its depth divided by the maximum path length that passes through it. In Figure 2-1, `Event` has weight 1/3, `Accident` 2/3, and `Collision` 1. `AgriculturalEvent` also has weight 1, because it is a leaf class, and the maximum path length through it is also its depth. `Loan`, despite being rather high in the hierarchy, has weight 1. The rationale is that a TA1 performer that assigns type `Loan` has found the best possible match, as the ontology does not further categorize loans.

This step creates a table, `class_depth_weighting`, in the `analysis` database. Each row of the table lists a class (a URL) and its weight.

Z. Step 26: Produce Assessment Data for Lum.AI

Each IDA coder was assigned three kinds of data to assess from Lum.AI:

1. Causal assertions from the 30 sentences used for calibration (Steps 10–15).
2. Causal assertions from the sentences in which BBN, ISI, and LCC all found causal assertions (Steps 17 and 18).
3. Causal assertions from sentences in which only Lum.AI found causal assertions.

Execute the following commands to produce this data:

```
$ mysql --user=user --password \
      < pt-17a-lum-assertions-from-30-sentences.sql
$ mysql --user=user --password \
      < pt-17b-lum-assertions-from-sentences-with-CAs-from-3.sql
$ mysql --user=user --password \
      < pt-17c-lum-assertions-not-found-elsewhere.sql
```

The upload directory will contain three CSV files whose names start with 17.

As it happened, Lum.AI found causal assertions in only 13 of the 30 calibration sentences. Lum.AI found 17 causal assertions in those sentences. The query IDA used to produce data for those sentences, `pt-17a-lum-assertions-from-30-sentences.sql`, extracts only one causal assertion per sentence.

AA. Step 27: Select Sentences for Event Assessment (Related to Causal Assertions)

Each IDA coder assessed 200 events. Half of these events were drawn from sentences containing causal assertions—more specifically, half of these events were associated with causal assertions. Execute the following commands to select the sentences:

```
$ mysql --user=user --password \
      < pt-18a-generate-assertion-scores.sql
$ mysql --user=user --password \
      < pt-18b-sentences-with-cas-from-4.sql
$ mysql --user=user --password \
      < pt-18c-event-weighting.sql
```

This produces a table, `event_analysis_sentences`, containing 100 sentences from which BBN, ISI, LCC, and Lum.AI all extracted causal assertions. The final command defines a function used in subsequent steps.

The first query does something that, in retrospect, should have been done earlier. It creates a table containing every causal assertion along with a numeric score of that causal assertion's quality. Score is based on the considerations described earlier: the absence of general events, the presence of factors, and the presence of actors and locations. Absence of general events is weighted much higher than the others, which offsets one performer having up to nine factors for an event. The weights make a causal assertion with two non-generic events better than a causal assertion with one generic event and nine factors.

BB. Step 28: Produce Assessment Data for Events

This step produces CSVs containing all the information needed to assess events from the sentences in Step 26. Execute the following commands:

```
$ mysql --user=user --password \
      < pt-19a-events-from-100-sentences-bbn.sql
$ mysql --user=user --password \
      < pt-19b-events-from-100-sentences-isi.sql
$ mysql --user=user --password \
      < pt-19c-events-from-100-sentences-lcc.sql
$ mysql --user=user --password \
      < pt-19d-events-from-100-sentences-lum.sql
$ mysql --user=user --password \
      < pt-19e-80-more-lum-events.sql
```

The upload directory will contain five CSV files whose names begin with 19.

These events are associated with causal assertions. The events are evenly divided between antecedent and consequent events. The antecedent events allow for the possibility of a causal assertion having antecedents other than causes. The queries do not attempt to represent all kinds of antecedent. In practice, antecedents other than causes have appeared in the results.

Step 29 yields events that are not associated with causal assertions. Lum.AI does not extract an event unless it is associated with a causal assertion. The query in `pt-19e-80-more-lum-events.sql` downloads extra Lum.AI-extracted events for coding. These events are not any of the ones extracted by the query in `pt-19d-events-from-100-sentences-lum.sql`.

CC. Step 29: Produce Assessment Data for Events Not Associated with Causal Assertions

Coders also assessed 25 events that were not associated with causal assertions. This step creates the necessary data. Execute the following commands:

```
$ mysql --user=user --password < pt-20a-unassociated-events-bbn.sql
$ mysql --user=user --password < pt-20b-unassociated-events-isi.sql
$ mysql --user=user --password < pt-20c-unassociated-events-lcc.sql
```

The upload directory will contain three CSV files whose names begin with 20.

3. Script-Based Execution

Appendix B summarizes the commands in Section 2. The commands have also been collected into a single script, `generate-coder-data.sh`, which is included with this distribution. It has been tested in a Linux environment. Invoke it as follows:

```
$ generate-coder-data.sh data-directory admin-user user
```

where:

- `data-directory` is a directory containing the four uncompressed SQLite databases downloaded from VDL. That is, the script does not perform Steps 1 and 2.
- `admin-user` is the name of a MySQL user with administrative privileges. This user must be able to create databases.
- `user` is the name of a MySQL user with privileges to create tables, views, and functions, and select and update data. The user must also have the FILE and SUPER privileges. It can be the same as admin-user, although many MySQL installations often grant administrative privileges to specific users. If you do not have these privileges, ask your administrator to execute the `mysqladmin` commands.

The script does not perform Step 15, which requires copying files from a platform-specific location.

The script repeatedly invokes the `mysql` command. Each time it does, it asks for a password. Using the script requires entering a password 56 times. There are ways around this (e.g., MySQL's `mysql_config_editor` command), but they are not portable. The IDA team is not delivering professional-quality software for CausX and does not have the time or resources to develop a general-purpose password management approach. The technique in the script, though painful to use, is secure. Users are welcome to customize the script.

To use this script, you may have to modify the query files that specify the path in which the MySQL server exports data. For example, `pt-04a-assertions-data-bbn.sql` contains the path:

```
C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/04a-30-sentences-bbn.csv
```

Modify this path to the location where your MySQL server exports files. See Appendix A for the full list of files to modify. To determine the path, run the `mysql` client and execute the following command:

```
mysql> SHOW VARIABLES LIKE 'secure_file_priv';
```

The output from this command will show the directory into which files are exported.

Appendix A.
Files Included in This Distribution

Table A-1 lists all the files you should have received with this document. You can use these files to follow the process in Section 2 and to run the shell script described in Section 3.

If an asterisk appears in the Path column, the files in that row have hardwired file paths. You may need to modify them to make them work on your platform.

Table A-1. Files Included in Distribution

File	Description	Path?
databases.txt	The SQLite databases used as inputs.	
generate-coder-data.sh	Shell script to perform most of the commands in Section 2.	
pt-01a-populate-ca-sentences.sql pt-01b-populate-event-sentences.sql pt-01c-populate-ca-sentences-lum.sql pt-01d-populate-event-sentences-lum.sql	Populate tables in the sentences database with information on sentences and the events and causal assertions extracted from them.	
pt-02-30-sentences.sql	Create a table containing 30 sentences from which all performers have extracted causal assertions.	
pt-03-cas-from-30-sentences.sql	Select, for each performer, 30 causal assertions from the 30 sentences.	
pt-04a-assertions-data-bbn.sql pt-04b-assertions-data-isi.sql pt-04c-assertions-data-lcc.sql	Download the selected causal assertions CSVs.	*
pt-05a-factors-locations-bbn.sql pt-05b-factors-locations-isi.sql pt-05c-factors-locations-lcc.sql	Download the factors and locations for the selected causal assertions.	*
pt-06a-actors-bbn.sql pt-06b-actors-isi.sql pt-06c-actors-lcc.sql	Download the actors for the selected causal assertions.	*
pt-07-sentences-not-shared.sql	Identify causal-assertion-containing sentences that occur exactly once in the corpus.	
pt-08-sentences-with-CAs-from-3-performers.sql	Find all sentences from which all performers have extracted causal assertions.	

pt-09a-nongeneric-CAs-bbn.sql pt-09b-nongeneric-CAs-isi.sql pt-09c-nongeneric-CAs-lcc.sql	Download data on sentences from which all performers have extracted causal assertions.	*
pt-10-representative-shared-sentences.sql	Identify sentences that appear in multiple documents, or more than once in the same document, and select one instance of each of the sentences for subsequent analysis.	
pt-11-sentences-with-CAs-from-2-performers.sql	Identify sentences from which exactly two of the three TA1 performers have extracted causal assertions.	
pt-12a-create-selected-two-performer-sentences-table.sql pt-12b-selected-two-performer-sentences-bbn-isi.sql pt-12c-selected-two-performer-sentences-isi-lcc.sql pt-12d-selected-two-performer-sentences-bbn-lcc.sql	Select causal assertions from the sentences from which exactly two of the three TA1 performers have extracted causal assertions.	
pt-13a-create-assertion-functions.sql pt-13b-2-performer-CAs-bbn-isi.sql pt-13c-2-performer-CAs-isi-lcc.sql pt-13d-2-performer-CAs-bbn-lcc.sql	Download data on sentences from which two performers have extracted causal assertions.	*
pt-14-sentences-with-CAs-from-1-performer.sql	Identify sentences from which exactly one performer has extracted causal assertions.	
pt-15a-1-performer-CAs-bbn.sql pt-15b-1-performer-CAs.isi.sql pt-15c-1-performer-CAs-lcc.sql	Download data on sentences from which exactly one performer has extracted causal assertions.	*
pt-16-load-depth-data.sql	Upload data used to compute event quality.	
pt-17a-lum-assertions-from-30-sentences.sql pt-17b-lum-assertions-from-sentences-with-CAs-from-3.sql pt-17c-lum-assertions-not-found-elsewhere.sql	Identify causal assertions from Lum.AI to be used in assessment.	
pt-18a-generate-assertion-scores.sql pt-18b-sentences-with-cas-from-4.sql pt-18c-event-weighting.sql	Select 100 sentences from which all four performers have extracted causal assertions.	
pt-19a-events-from-100-sentences-bbn.sql pt-19b-events-from-100-sentences-isi.sql pt-19c-events-from-100-sentences-lcc.sql pt-19d-events-from-100-sentences-lum.sql	Download events from the 100 sentences.	*

Pt-19e-80-more-lum-events.sql	Download 80 more Lum.AI events, compensating for the absence of events not associated with a causal assertion.	*
pt-20a-unassociated-events-bbn.sql pt-20b-unassociated-events-isi.sql pt-20c-unassociated-events-lcc.sql	Download data on 250 events not associated with a causal assertion.	
required-files.txt	A list of all files in this table. The generate-coder-data.sh script uses it for error checking.	
schema-myisam.sql	Defines a MySQL schema for TA1 performer databases.	
sentences-myisam.sql	Defines a MySQL schema for information on sentences.	

Appendix B. Command Summary

This appendix summarizes the commands used to generate the assessment data. Execute these commands in the order they appear. Be aware that (1) the first command corresponds to Step 3—downloading data from VDL and uncompressing it is not automated, and (2) Step 15, stitching together CSVs, is not included, as the location of CSV files generated by the MySQL server is not portable.

```
$ python3 dump-db.py bbn_8k_20200518.db bbn-data.sql
$ python3 dump-db.py isi_8k_20200601.db isi-data.sql
$ python3 dump-db.py lcc_8k_20200716.db lcc-data.sql
$ python3 dump-db.py lum_8k_20200608.db lum-data.sql
$ mysqladmin --user=admin-user --password \
      create bbn8k create isi8k create lcc8k create lum8k
$ mysql --user=user --password bbn8k < schema-myisam.sql
$ mysql --user=user --password isi8k < schema-myisam.sql
$ mysql --user=user --password lcc8k < schema-myisam.sql
$ mysql --user=user --password lum8k < schema-myisam.sql
$ mysql --user=user --password bbn8k < bbn-data.sql
$ mysql --user=user --password isi8k < isi-data.sql
$ mysql --user=user --password lcc8k < lcc-data.sql
$ mysql --user=user --password lum8k < lum-data.sql
$ mysqladmin --user=admin-user --password create analysis create sentences
$ mysql --user=user --password sentences < sentences-myisam.sql
$ mysql --user=user --password < pt-01a-populate-ca-sentences.sql
$ mysql --user=user --password < pt-01b-populate-event-sentences.sql
$ mysql --user=user --password < pt-01c-populate-ca-sentences-lum.sql
$ mysql --user=user --password < pt-02-30-sentences.sql
$ mysql --user=user --password < pt-03-cas-from-30-sentences.sql
$ mysql --user=user --password < pt-04a-assertions-data-bbn.sql
$ mysql --user=user --password < pt-04b-assertions-data-isi.sql
$ mysql --user=user --password < pt-04c-assertions-data-lcc.sql
$ mysql --user=user --password < pt-05a-factors-locations-bbn.sql
$ mysql --user=user --password < pt-05b-factors-locations-isi.sql
$ mysql --user=user --password < pt-05c-factors-locations-lcc.sql
$ mysql --user=user --password < pt-06a-actors-bbn.sql
$ mysql --user=user --password < pt-06b-actors-isi.sql
$ mysql --user=user --password < pt-06c-actors-lcc.sql
$ mysql --user=user --password < pt-07-sentences-not-shared.sql
$ mysql --user=user --password \
      < pt-08-sentences-with-CAs-from-3-performers.sql
$ mysql --user=user --password < pt-09a-nongeneric-CAs-bbn.sql
```

```
$ mysql --user=user --password < pt-09b-nongeneric-CAs-isi.sql
$ mysql --user=user --password < pt-09c-nongeneric-CAs-lcc.sql
$ mysql --user=user --password < pt-10-representative-shared-sentences.sql
$ mysql --user=user --password \
      < pt-11-sentences-with-CAs-from-2-performers.sql
$ mysql --user=user --password \
      < pt-12a-create-selected-two-performer-sentences-table.sql
$ mysql --user=user --password \
      < pt-12b-selected-two-performer-sentences-bbn-isi.sql
$ mysql --user=user --password \
      < pt-12c-selected-two-performer-sentences-isi-lcc.sql
$ mysql --user=user --password \
      < pt-12d-selected-two-performer-sentences-bbn-lcc.sql
$ mysql --user=user --password < pt-13a-create-assertion-functions.sql
$ mysql --user=user --password < pt-13b-2-performer-CAs-bbn-isi.sql
$ mysql --user=user --password < pt-13c-2-performer-CAs-isi-lcc.sql
$ mysql --user=user --password < pt-13d-2-performer-CAs-bbn-lcc.sql
$ mysql --user=user --password \
      < pt-14-sentences-with-CAs-from-1-performer.sql
$ mysql --user=user --password < pt-15a-1-performer-CAs-bbn.sql
$ mysql --user=user --password < pt-15b-1-performer-CAs-isi.sql
$ mysql --user=user --password < pt-15c-1-performer-CAs-lcc.sql
$ mysql --user=user --password < pt-16-load-depth-data.sql
$ mysql --user=user --password \
      < pt-17a-lum-assertions-from-30-sentences.sql
$ mysql --user=user --password \
      < pt-17b-lum-assertions-from-sentences-with-CAs-from-3.sql
$ mysql --user=user --password \
      < pt-17c-lum-assertions-not-found-elsewhere.sql
$ mysql --user=user --password < pt-18a-generate-assertion-scores.sql
$ mysql --user=user --password < pt-18b-sentences-with-cas-from-4.sql
$ mysql --user=user --password < pt-18c-event-weighting.sql
$ mysql --user=user --password < pt-19a-events-from-100-sentences-bbn.sql
$ mysql --user=user --password < pt-19b-events-from-100-sentences-isi.sql
$ mysql --user=user --password < pt-19c-events-from-100-sentences-lcc.sql
$ mysql --user=user --password < pt-19d-events-from-100-sentences-lum.sql
$ mysql --user=user --password < pt-19e-80-more-lum-events.sql
$ mysql --user=user --password < pt-20a-unassociated-events-bbn.sql
$ mysql --user=user --password < pt-20b-unassociated-events-isi.sql
$ mysql --user=user --password < pt-20c-unassociated-events-lcc.sql
```

Appendix C. Ranking Criteria

Selecting the set of causal assertions and events to assess proved to be one of the most significant and challenging tasks. Table C-1 shows the number of causal assertions and events each performer extracted. The IDA team lacked the resources to assess everything in the limited time available to meet the DARPA requirement. Choosing which events to evaluate was vital.

Table C-1. Selected Extraction Quantities

Performer	# of Causal Assertions	# of Events
BBN	401,087	4,614,493
ISI	46,255	1,545,711
LCC	59,737	2,062,786
Lum.AI	39,862	73,423

Early on, the IDA team made the decision to select high-quality causal assertions and events for coding. The team wanted to give each TA1 performer the opportunity to showcase their best work.

TA2 performers were clear that certain kinds of events and causal assertions were more useful than others. In particular, they could not use causal assertions in which both the antecedent and consequent were generic.[7] They preferred events with associated factors, locations, and times.

The IDA team initially used the confidence scores each performer assigned. Each causal assertion, event, and location has an associated confidence—a real number between 0 and 1 that expresses "the probability that the referenced individual is correct."[8] This approach did not yield useful results. There was little uniformity in how TA1 performers assigned confidence (Table C-2). Furthermore, high confidence values did not correspond to the TA2 needs stated above. The IDA team decided not to base selection on confidence.

[7] A generic event is one whose primary type is `http://ontology.causex.com/ontology/odps/Event#Event`. If the secondary type has this value, the event is not considered generic.

[8] Taken from the definition of the `numeric_confidence` property in the CausX ontology.

Table C-2. Causal Assertion Confidence Ranges

Performer	Minimum	Average	Maximum
BBN	0.65	0.80	1
ISI	0.2	0.59	1
LCC	0	0.37	0.80
Lum.AI	0.23	0.51	1

The IDA team therefore devised its own selection approach. The objective of this approach was to provide coders with each TA1 performer's high-quality events and causal assertions, where quality is judged based on perceived usefulness to TA2 performers. Coders could then assess the accuracy of information used by TA2s.

The nature of quality changed as coding continued. The details of judging quality differed for each comparison. The differences reflected the IDA team's growing knowledge of database content and the relative strengths and weaknesses of each TA1 performer. This appendix describes each comparison quality in a separate section. There is one quality criterion shared among all comparisons, and it is considered the most significant: the presence or absence of generic events.

1. Comparison 1 Quality Criteria

In Comparison 1, quality q of a performer's causal assertion is determined by the following formula:

$$q = 2 \times e + f$$

where:

$$e = \begin{cases} 2 & \text{if neither antecedent nor consequent event is generic} \\ 1 & \text{if antecedent or consequent event is generic} \\ 0 & \text{if both antecedent and consequent event are generic} \end{cases}$$

$$f = \begin{cases} 2 & \text{if both antecedent and consequent events have factors} \\ 1 & \text{if either antecedent or consequent event has factors} \\ 0 & \text{if neither antecedent nor consequent event has factors} \end{cases}$$

Quality is therefore an integer between 0 and 6. A causal assertion with non-generic antecedent and consequent events that both have factors is ranked most highly. A causal assertion with non-generic antecedent and consequent events without factors is ranked equally to a causal assertion in which one of the antecedent or consequent is generic and both have factors.

This ranking system yields many more highly ranked causal assertions than necessary. Coders were asked to assess 70 causal assertions per TA1 performer. BBN had over 5,000 causal assertions with $q = 6$. Causal assertions were selected from the candidate set using a random number generator.

By the time the IDA team was asked to assess Lum.AI extractions, we had devised the more sophisticated ranking system, discussed in Section 2.Y, based on distance along the path from root to leaf. We opted to determine the quality of Lum.AI causal assertions using the simple formula:

$$q = w_a + w_c$$

where w_a is the weighting of the antecedent event and w_c is the weighting of the consequent event. We did not use the number of factors. We planned to do a separate evaluation of events and decided that, with this improved event-type ranking scheme, class depth weight would suffice.

2. Comparison 2 Quality Criteria

In Comparison 2, quality was based solely on event genericity. However, the evaluation function differed. Comparison 2 concerns sentences from which exactly two of the performers (BBN, ISI, and LCC) have extracted causal assertions. TA1 performers, and especially BBN, often extract many causal assertions from a single sentence (as many as 72). These do not tend to be meaningful causal assertions, so we limit candidate sentences to those with a maximum of five causal assertions. For each performer, we then judge sentence quality according to the genericity of all causal assertion events in the sentence:

$$s = \sum_{i=1}^{n} e_i / n$$

where n is the number of causal assertions in a sentence, and e_i is the genericity of the i^{th} causal assertion. The assertion score is then the sum of the two performers' scores. Under this weighting system, a sentence with multiple non-generic causal assertions is preferred to a sentence with a single non-generic causal assertion. The approach favors these sentences under the assumption that identifying specific kinds of events likely means being able to identify more factors.

3. Comparison 3 Quality Criteria

Comparison 3 used the same formula as Comparison 1. In Comparison 3, every sentence was unique and could be assessed independently. There was no need to balance one performer's quality against another's, as in Comparison 2. This might seem counterintuitive. Comparison 1 considers sentences in which BBN, ISI, and LCC all found causal assertions, but the formula above does not consider multiple performers' results. The explanation is that sentences used for Comparison 1 were not (necessarily) the same for all performers. The starting criterion for sentence selection was that each performer had extracted a causal assertion from the sentence, but a given performer's selection from that

set was independent of other performers' results. The sentences selected for BBN, then, were largely different from those selected for ISI and LCC. Any overlap was coincidental.

As in Comparison 1, quality for Lum.AI causal assertions was based on class depth.

4. Comparison 4 Quality Criteria

Comparison 4 assessed events, not causal assertions. To qualify, an event could not have more than five factors, actors, locations, or topics. The IDA team imposed this restriction in consideration of the assessment time constraints.

Coders were given two categories of events: those associated with causal assertions (i.e., the antecedent or consequent of a causal assertion), and those not associated.

a. Criteria for Events Associated with Causal Assertions

To select the events associated with causal assertions, the IDA team found all sentences from which BBN, ISI, LCC, and Lum.AI had each extracted a single causal assertion. These sentences were then scored and ranked as follows. A score was computed for each causal assertion based on the completeness of the antecedent and consequent events, including whether or not the events were generic. Furthermore, if the event had actors, the team assessed whether the performers had adhered to the definition of an actor: a person, group of persons, or organization. Some performers included monetary amounts, physical locations, and measurements as actors. These types of actors lowered a causal assertion's score.

The sentences were then sorted according to the sum of each performer's assertion score. This favored the sentences from which every performer had extracted something reasonable, and penalized those in which one performer had done exceptionally well and others had done poorly. The IDA team wanted to select sentences in which every performer has done good, if not their best, work.

From these sentences, the IDA team extracted the better of the two events associated with the sentence's causal assertion. (Remember that each sentence had one causal assertion per performer.) Event selection was again based on completeness, with weights assigned: the more factors, actors, locations, and topics, the better.

The IDA team had planned to extract 100 events. Unfortunately, there were not 100 sentences across the entire corpus from which each performer had extracted only one causal assertion without at least one generic event. Results varied, but performers had about 70 non-generic events meeting this criterion. Furthermore, it was not possible to limit the set to complete events.

b. Criteria for Events Not Associated with Causal Assertions

The IDA team first tried selecting unassociated events using the same criteria as associated events (Section 4.a). This approach tended to produce groups of identically typed events. Apparently, event types correlate with the presence or absence of factors, actors, or locations; this holds true for all performers.

In an effort to increase the number of event types present for assessment, the IDA team opted for a different and simpler approach to select unassociated events. Each event is assigned a score from 0 to 4.5. If an event is generically typed, its score is always 0. Otherwise, its score is:

$$f + a + l + {}^{t}/_{2} + s$$

where:

f = 1 if the event has factors, 0 if not
a = 1 if the event has actors, 0 if not
l = 1 if the event has locations, 0 if not
t = 1 if the event has topics, 0 if not
s = 1 if the event has a start time, 0 if not

The presence of a topic is not as significant as the presence of other properties and is weighted accordingly.

Abbreviations and Acronyms

BBN	Bolt Beranek and Newman
CSV	Comma Separated Value
CausX	Causal Exploration
DARPA	Defense Advanced Research Projects Agency
IDA	Institute for Defense Analyses
ISI	Information Sciences Institute
JSON	JavaScript Object Notation
LCC	Language Computer Corporation
SHA	Secure Hash Algorithm
URL	Uniform Resource Locator
UTF-8	Unicode Transformation Format
VDL	Virtual Distribution Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE (DD-MM-YY)	2. REPORT TYPE	3. DATES COVERED (From – To)
00-11-20	Final	

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Generating the CausX TA1 Assessment Data	HQ0034-14-D-0001
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBERS

6. AUTHOR(S)	5d. PROJECT NUMBER
Peter A. Kind, Susan K. Numrich, Steven P. Wartik	DA-5-4320
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESSES	8. PERFORMING ORGANIZATION REPORT NUMBER
Institute for Defense Analyses 4850 Mark Center Drive Alexandria, VA 22311-1882	D-15381

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR'S / MONITOR'S ACRONYM
Dr. Joshua Elliott Defense Advanced Research Projects Agency 675 N. Randolph St, Arlington, VA 22203	DARPA
	11. SPONSOR'S / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

Project Leader: Steven P. Wartik

14. ABSTRACT

In Spring 2020, IDA performed an assessment of DARPA's Causal Exploration (CausX) program. The CausX program involves extracting events, causal assertions, and related elements from natural language documents. IDA's assessment compared the extraction quality of four performers participating in CausX. The IDA team developed an approach to select each performer's highest-quality work. To ensure "highest-quality work" was well-defined, the team created an automatable, repeatable process for selecting events and causal assertions. This document describes the process and how to perform it. It uses open source software and can be performed on a variety of modern computing platforms; IDA has executed it on Windows and Linux. All files necessary to execute the process are included in this deliverable. An analyst who wants to reproduce IDA's results can use these files and follow the process steps to obtain the same spreadsheet data (allowing for some variation due to randomization; steps involving randomization note the fact) IDA used in its assessment.

15. SUBJECT TERMS

Causal Exploration, Event, Causal Assertion, Natural Language Processing, SQLite, MySQL, Python, Repeatable Process

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Joshua Elliott
Unclassified	Unclassified	Unclassified	Unlimited	34	19b. TELEPHONE NUMBER (Include Area Code) 703-526-4754