# NaCaGraph: Visualizing Event Graph for Natural Catastrophes from News Articles

Cypher Language Command and Results
(Phase 3: Access)

**Li Lin, Chong Shen, Xiru Tong**

**Institute for Natural Language Processing**

**University of Stuttgart**

July 30, 2023

## Contents

# 1 Introduction

## 1.1 Neo4j

Neo4j is a native graph database that is built to store and retrieve connected data. It is designed to leverage not only data itself, but also data relationships. Neo4j is optimized such that each data record (i.e. *node*) contains direct pointers to all the nodes that it is connected to. These direct pointers are called *relationships*. All the information needed to find the next node is available in the node itself. The native storage layer is a connected graph. Due to this "principle of nativeness", Neo4j does not need to compute relationships between data records at the query time. Instead, the relationships are already stored in the database explicitly. Thus, queries of densely connected data are much faster. In contrast, databases which do not save direct pointers between records need to compute relationships between each two nodes by searching through the index repeatedly. This makes the querying significantly slower.

In this project, we deploy our database with the default name (*neo4j*) on the Neo4j Database Management System (DBMS) (version: *neo4j-community-4.4.21*).

## 1.2 Cypher

Cypher is a declarative graph query language designed for Neo4j. Cypher provides a visual way of matching patterns and relationships. It relies on a ASCII-art type of syntax, i.e. (Nodes)-[:CONNECT_TO]→(OtherNodes). Rounded brackets are used for circular nodes, and -[:ARROWS]→ for relationships.

Similar to SQL, Cypher allows users to focus on the content of the data retrieved from the database, rather than how to access to it. However, Cypher is a rather schema-flexible language. Users are not required to use a fixed schema to represent data. They can add new attributes and relationships as the graph evolve.

Cypher differs from SQL also in its syntax. While an SQL query starts with the entry to be returned, Cypher requires the return clause to be at the end of a query.

As a result, Cypher queries are often more concise than SQL queries.

## 2 Command and Results

### 2.1 Create the graph from the .xml file

#### 2.1.1 Xml tree structure

**.xml content can be understood as the following tree structure:**

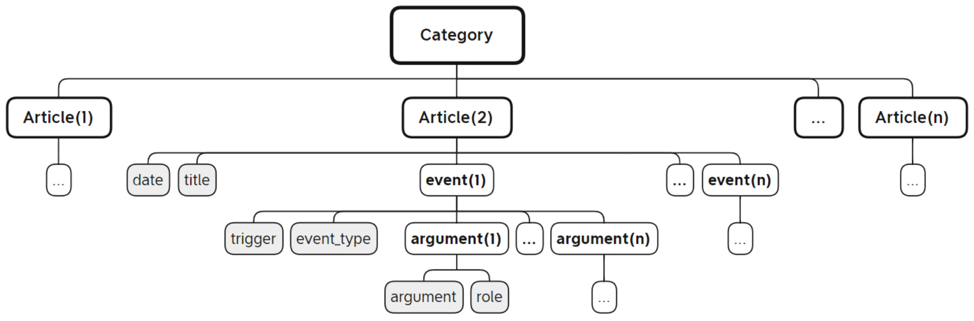


Figure 1: .xml content

**Query requirements:**

1. Local search:

can query out the relationships of topic - article - event_types - triggers - role_labels - arguments for each article.

2. Global search: can query out

1). all related articles of the same topic;

2). all related articles and triggers of the same event_type;

3). all related event_types and role_labels of the same trigger;

4). all related triggers and arguments of the same role_label.

#### 2.1.2 Cypher command of create graph

**Command:**

Code execution file: Create_Graph.py

```
CALL apoc.load.xml ("file: [Path]/[file_name].xml")
YIELD value
WITH value, value.topic AS topic_text
MERGE (t: Topic {text: topic_text, type:"topic"} )

WITH value, t
UNWIND value._children AS article
```

```
WITH t, article.aid AS articleId, article.url as articleURL,
    [item in article._children WHERE item._type = "title"][0] AS title,
    [item in article._children WHERE item._type = "date"][0] AS date,
    [item in article._children WHERE item._type = "event"] AS events
MERGE ( a:Article {text:title._text, type:"article", aurl:articleURL})
MERGE ( dat:Year {text: right(date._text,4), type:'year'})

MERGE (t)-[:PUBLISHED_TIME {text: date._text}]->(a)
MERGE (dat)-[:PUBLISHED_ARTICLE ]->(a)

WITH events, a,articleId,articleURL
UNWIND events AS event
WITH a,articleId,articleURL,
    [item in event._children WHERE item._type = "trigger"][0] AS trigger,
    [item in event._children WHERE item._type = "type"][0] AS event_type,
    [item in event._children WHERE item._type = "argument"] AS arguments

WHERE NOT trigger._text IS NULL
MERGE (et:EventType {text:"type:"+event_type._text, type:'event_type'})
MERGE (tr: Trigger {text: trigger._text, type:"trigger"})
MERGE (idtr: IdTrigger {text: trigger._text, type:"idtrigger", aid:
   articleId, aurl:articleURL, eventType:event_type._text } )

MERGE (et)-[:EVENT_FOR]->(tr)
MERGE (et)-[:EVENT_FOR_IDTR]->(idtr)
MERGE (a)-[:HAS_EVENT_TYPE { aid:articleId, aurl:articleURL }]->(et)
MERGE (a)-[:HAS_TRIGGER {text:"type:"+event_type._text, aid:articleId, aurl
   :articleURL}]->(idtr)

WITH arguments, tr,idtr,articleId,articleURL, event_type._text as et_text,
   trigger._text as tr_text
UNWIND arguments AS argument
WITH tr,idtr,articleId,articleURL, et_text,tr_text,
    [item in argument._children WHERE item._type = "mention"][0] AS
   argument,
    [item in argument._children WHERE item._type = "role"][0] AS
   argument_role

WHERE NOT argument._text IS NULL
MERGE (ro:RoleLabel {text: "role:"+argument_role._text, type:"role_label"})
MERGE (arg:Argument {text: argument._text, type:"argument"})
```

```
44 MERGE (idarg: IdArgument {text: argument._text, type:"idargument", aid:
       articleId, aurl:articleURL,eventType:et_text, trigger:tr_text,roleLabel:
       argument_role._text })
45
46 MERGE (tr)-[: TR_OF ]->(ro)
47 MERGE (ro)-[: ROLE_FOR]->(arg)
48 MERGE (ro)-[: ROLE_FOR_IDARG]->(idarg)
49 MERGE (idtr)-[:HAS_ROLE {aid:articleId, aurl:articleURL} ]->(ro)
50 MERGE (idtr)-[: HAS_ARGUMENT {text: "role:"+argument_role._text, aid:
       articleId, aurl:articleURL} ]->(idarg)
```

**Explain:**

Line 1-2: use tool apoc.load.xml to import .xml file, yield return value;

Line 3-4: get the <category> tag and its attribute as the topic_text, merge node t with label "Topic" and properties "text" and "type";

Line 6-11: get and unwind the <article> tags of <category> as each article, get the <article> tag attributs as articleId and articleUrl, get <title>, <date> and <event> tags of <article>;

Line 12-16: merge node a with label "Article" and properties "text", "type" and "aurl", merge node dat with label "Year" and properties "text" and "type"; merge relationship of type "PUBLISHED_TIME" between node t and node dat with property "text", merge relationship of type "PUBLISHED_ARTICLE" between node dat and node a without property;

Line 18-23: get and unwind the <event> tags of <article> as each event, get <trigger>, <type> and <argument> tags of <event>;

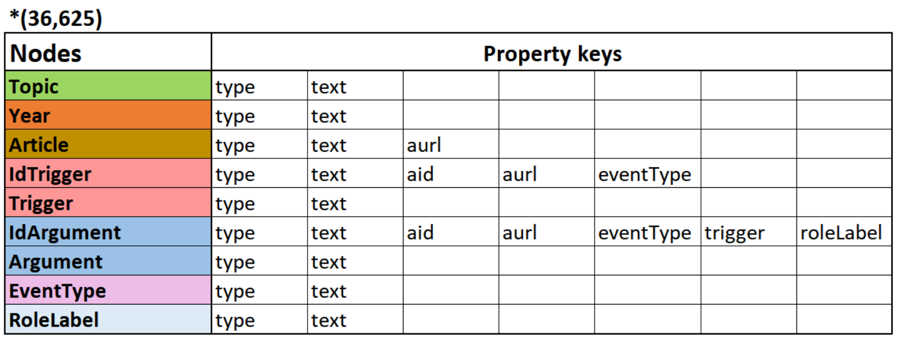
Line 25-33: if the <event> tag has no <trigger> content, it is skipped; merge node et with label "EventType" and properties "text" and "type", merge node tr with label "Trigger" and properties "text" and "type", merge node idtr with label "IdTrigger" and properties "text", "type", "aid", "aurl" and "eventType "; merge relationship of type " EVENT_FOR" between node et and node tr without property, merge relationship of type "EVENT_FOR_IDTR" between node et and node idtr without property, merge relationship of type "HAS_EVENT_TYPE" between node a and node et with property "aid" and "aurl", merge relationship of type "HAS_TRIGGER" between node a and node idtr with property "text", "aid" and "aurl";

Line 35-39: similar to line 18-23;

Line 41-50: similar to line 25-33.

### 2.1.3 Graph structure/Data model/Schema

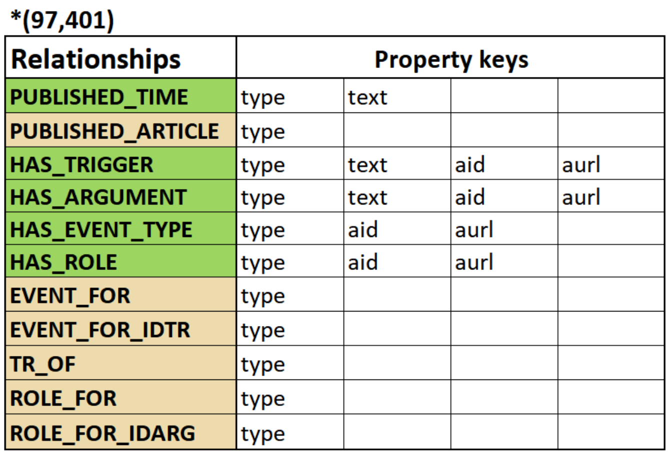
**Nodes and Edges:**



*(36,625)

| Nodes | Property keys | | | | | | |
|---|---|---|---|---|---|---|---|
| Topic | type | text | | | | | |
| Year | type | text | | | | | |
| Article | type | text | aurl | | | | |
| IdTrigger | type | text | aid | aurl | eventType | | |
| Trigger | type | text | | | | | |
| IdArgument | type | text | aid | aurl | eventType | trigger | roleLabel |
| Argument | type | text | | | | | |
| EventType | type | text | | | | | |
| RoleLabel | type | text | | | | | |

Figure 2: Nodes



*(97,401)

| Relationships | Property keys | | | |
|---|---|---|---|---|
| PUBLISHED_TIME | type | text | | |
| PUBLISHED_ARTICLE | type | | | |
| HAS_TRIGGER | type | text | aid | aurl |
| HAS_ARGUMENT | type | text | aid | aurl |
| HAS_EVENT_TYPE | type | aid | aurl | |
| HAS_ROLE | type | aid | aurl | |
| EVENT_FOR | type | | | |
| EVENT_FOR_IDTR | type | | | |
| TR_OF | type | | | |
| ROLE_FOR | type | | | |
| ROLE_FOR_IDARG | type | | | |

Figure 3: Edges

Graph visualization for schema:

**Command:**

```
CALL db.schema.visualization()
```

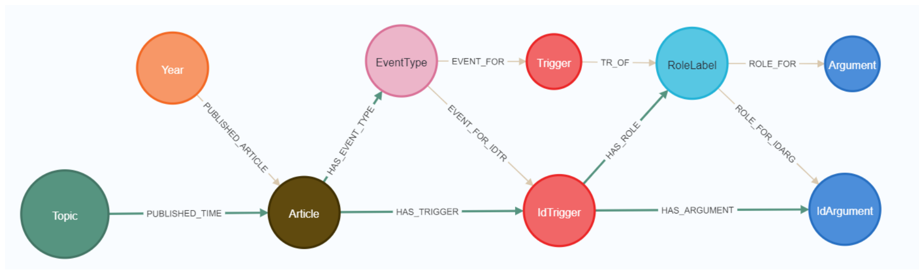


Figure 4: Data model

## 2.2 Query and Results

Execution file: query_scripts.cypher.

### 2.2.1 Example 1: Local search

**Query:**

Query the distribution of triggers, articles and topics with event_type is ”catastrophe” after 2020.

**Command:**

```
MATCH p=(:Topic )-[r1:PUBLISHED_TIME]->(arti:Article)-[r2:HAS_TRIGGER {text
    :"type:catastrophe"} ]->(n:IdTrigger) where toInteger(right(r1.text,4))
    >2020
return p
```

**Result:**

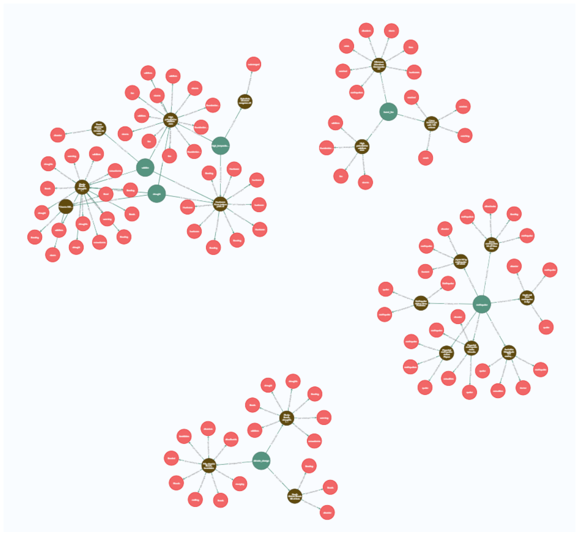
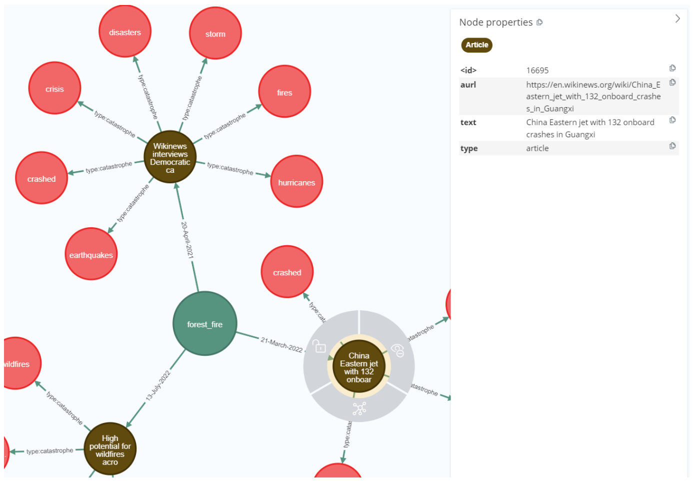


Figure 5: Example 1: result



Node properties

Article

| | |
|---|---|
| <id> | 16695 |
| aurl | https://en.wikinews.org/wiki/China_Eastern_jet_with_132_onboard_crashes_in_Guangxi |
| text | China Eastern jet with 132 onboard crashes in Guangxi |
| type | article |

Figure 6: Example 1: localised result

### 2.2.2 Example 2: Global search: Query about trigger ”fire”

**Query 2.1:**

Visual map of all relevant event_types, role_labels and arguments of trigger ”fire”.

**Command:**

```
match (:IdTrigger{text:"fire"})-->(idarg:IdArgument)
match p=(:EventType)-->(:Trigger{text:"fire"})-->(:RoleLabel)-->(:Argument{
    text:idarg.text})
return p
```

**Result:**

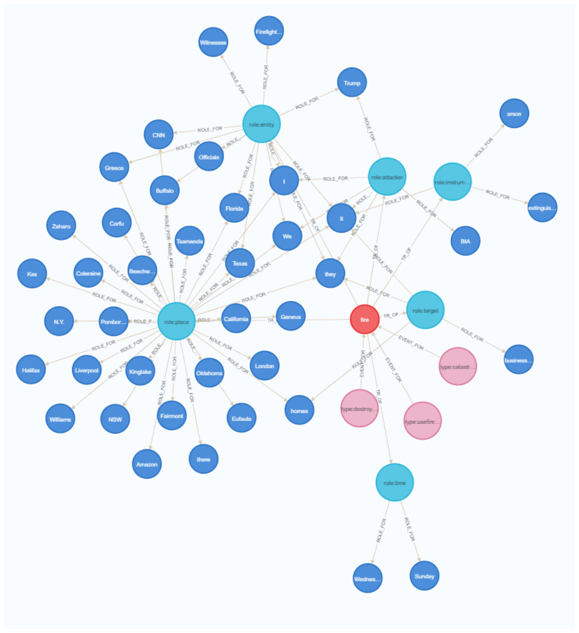
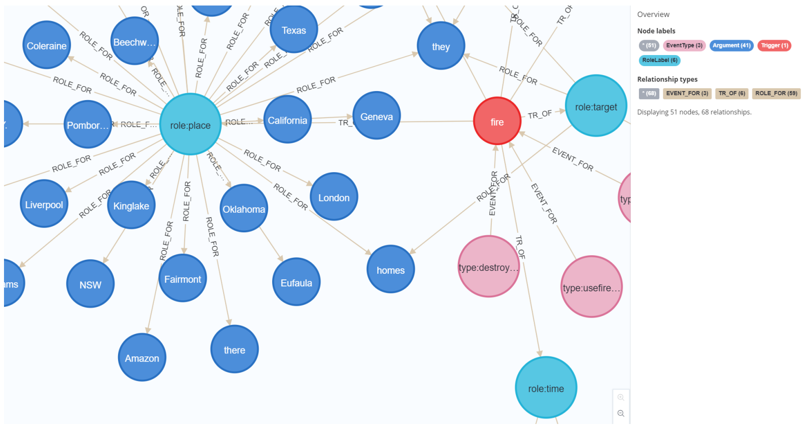


Figure 7: Example 2.1: result



Overview

**Node labels**

* (51) EventType (3) Argument (41) Trigger (1) RoleLabel (6)

**Relationship types**

* (68) EVENT_FOR (3) TR_OF (6) ROLE_FOR (59)

Displaying 51 nodes, 68 relationships.

Figure 8: Example 2.1: localised result

**Query 2.2:**

Visual map of the distribution topic, article and argument and published year of trigger "fire".

**Command:**

```
match  p=(t:Topic)-->(arti:Article)-[et:HAS_TRIGGER]->(idtr:IdTrigger{text
    :"fire"})-[rl:HAS_ARGUMENT]->(:IdArgument )
match (y:Year)-->(arti)
return p,y
```

**Result:**

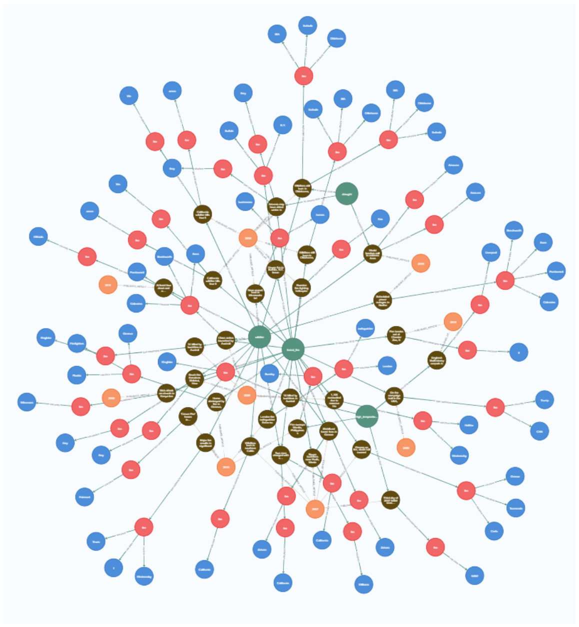
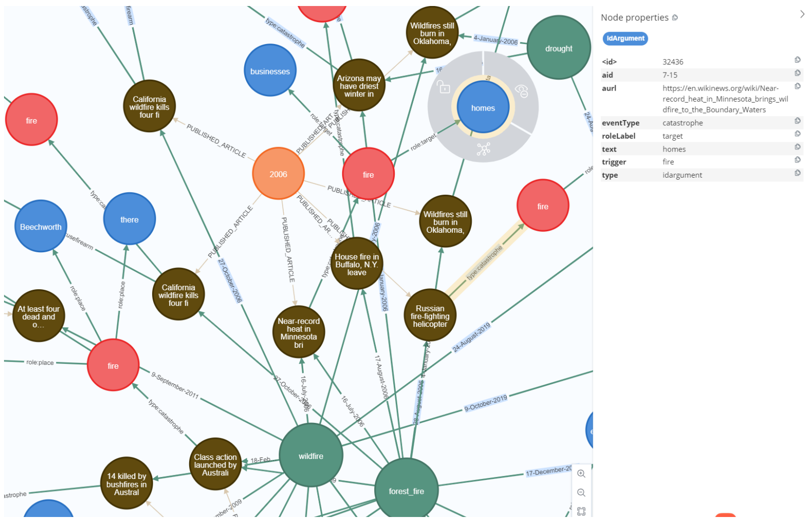


Figure 9: Example 2.2: result



fire
California wildfire kills four fi
businesses
Arizona may have driest winter in
Wildfires still burn in Oklahoma,
drought
homes
2006
fire
PUBLISHED_ARTICLE
there
Beechworth
fire
California wildfire kills four fi
House fire in Buffalo, N.Y. leave
Wildfires still burn in Oklahoma,
Russian fire-fighting helicopter
At least four dead and o...
fire
Near-record heat in Minnesota bri
role:place
role:target
type:catastrophe
wildfire
forest_fire
Class action launched by Australi
14 killed by bushfires in Austral

Node properties

IdArgument

| | |
|---|---|
| <id> | 32436 |
| aid | 7-15 |
| aurl | https://en.wikinews.org/wiki/Near-record_heat_in_Minnesota_brings_wildfire_to_the_Boundary_Waters |
| eventType | catastrophe |
| roleLabel | target |
| text | homes |
| trigger | fire |
| type | idargument |

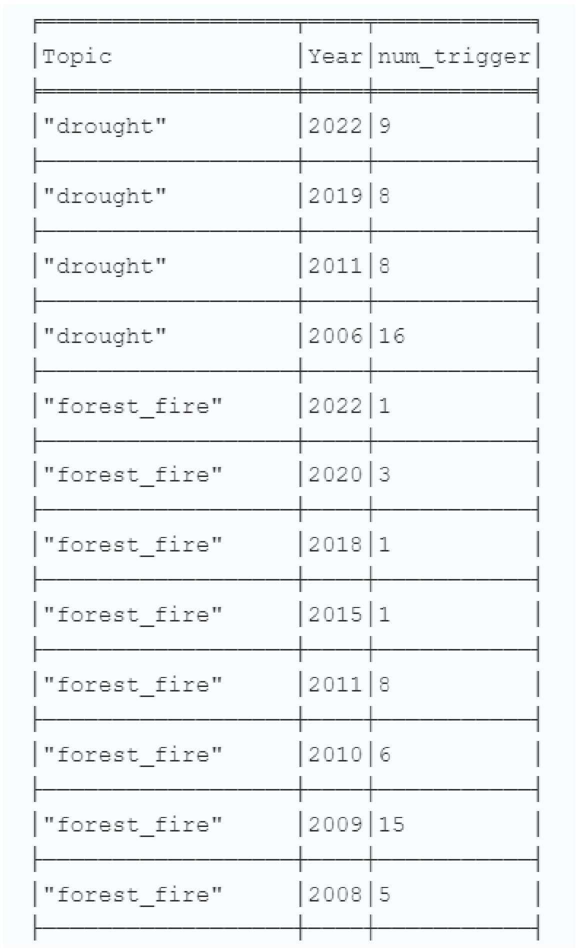
Figure 10: Example 2.2: localised result

**Query 2.3:**

Count the frequency of trigger ”fire” under each topic in different year after 2003. **Command:**

```
match p=(topic:Topic)-[r1:PUBLISHED_TIME]->(arti:Article)-->(idtr:IdTrigger
    {text:"fire"})
match (topic)-[r1]->(arti)-->(:EventType)-->(tr:Trigger{text:idtr.text})
match (year:Year)-->(arti)
where toInteger(right(r1.text,4))>2003
with topic,year,count(tr) as num_tr
return topic.text as Topic, toInteger(year.text) as Year, num_tr as
    num_trigger
order by Topic,Year desc
```

**Result:**



```
╒═══════════════════╤════╤═══════════╕
│Topic              │Year│num_trigger│
╞═══════════════════╪════╪═══════════╡
│"drought"          │2022│9          │
├───────────────────┼────┼───────────┤
│"drought"          │2019│8          │
├───────────────────┼────┼───────────┤
│"drought"          │2011│8          │
├───────────────────┼────┼───────────┤
│"drought"          │2006│16         │
├───────────────────┼────┼───────────┤
│"forest_fire"      │2022│1          │
├───────────────────┼────┼───────────┤
│"forest_fire"      │2020│3          │
├───────────────────┼────┼───────────┤
│"forest_fire"      │2018│1          │
├───────────────────┼────┼───────────┤
│"forest_fire"      │2015│1          │
├───────────────────┼────┼───────────┤
│"forest_fire"      │2011│8          │
├───────────────────┼────┼───────────┤
│"forest_fire"      │2010│6          │
├───────────────────┼────┼───────────┤
│"forest_fire"      │2009│15         │
├───────────────────┼────┼───────────┤
│"forest_fire"      │2008│5          │
├───────────────────┼────┼───────────┤
```

Figure 11: Example 2.3: localised result

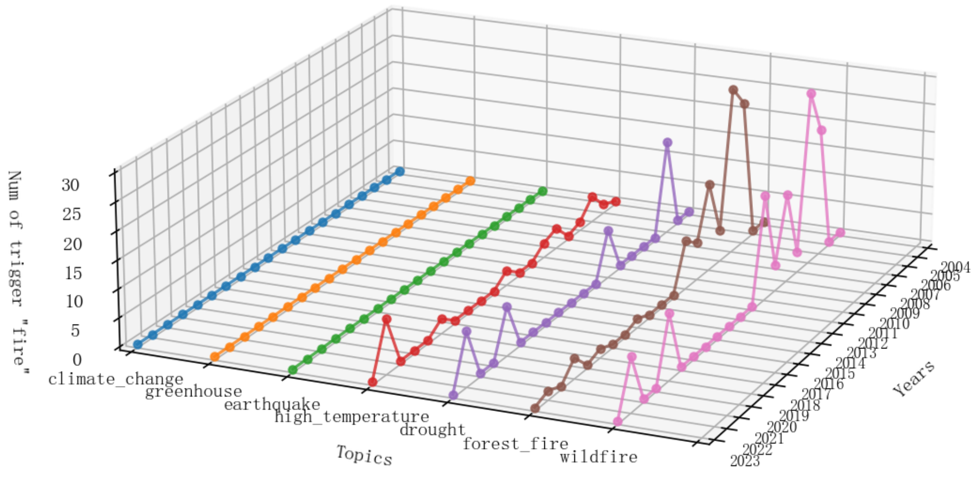


Figure 12: Example 2.3: after python processing

### 2.2.3 Example 3: Similarity between topics

**Question:**

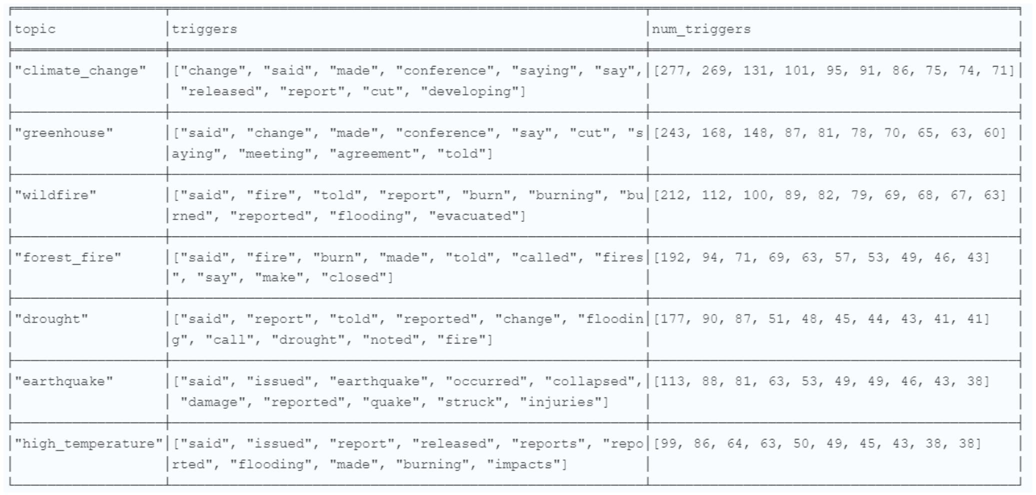
Count the first ten event types and triggers per topic, return topics, ten event_tpyes/triggers, ten frequency numbers.

**Command:**

```
match p=(topic:Topic)-[r1:PUBLISHED_TIME]->(arti:Article)-->(idtr:IdTrigger
    )
match (topic)-[r1]->(arti)-->(:EventType)-->(tr:Trigger{text:idtr.text})
match (year:Year)-->(arti)
with topic,tr,count(tr) as num_tr
order by num_tr desc
WITH topic, collect(tr.text)[0..10] AS triggers, collect(num_tr)[0..10] AS
    num_triggers
RETURN topic.text as topic, triggers,num_triggers
```

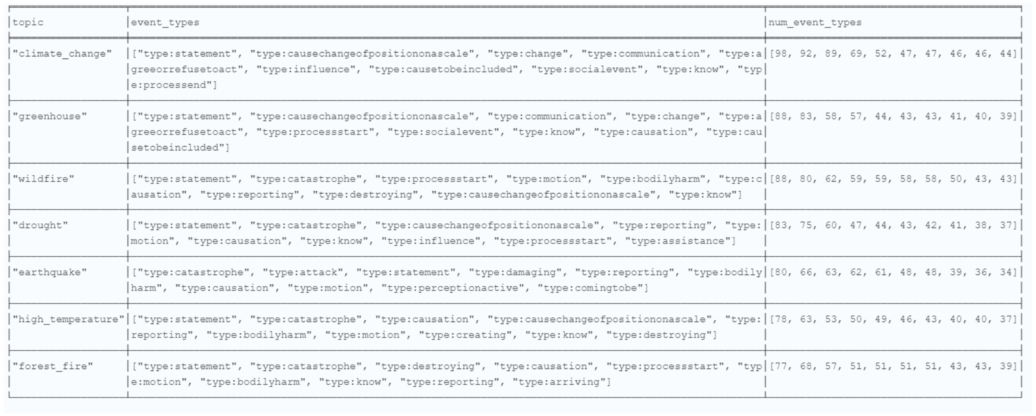
```
match p=(topic:Topic)-[r1:PUBLISHED_TIME]->(arti:Article)-->(et:EventType)
with topic,et,count(et) as num_et
order by num_et desc
WITH topic, collect(et.text)[0..10] AS event_types, collect( num_et )
    [0..10] AS num_event_types
RETURN topic.text as topic, event_types, num_event_types
```

**Result:**



```
|topic             |triggers                                                    |num_triggers                                |
|"climate_change"  |["change", "said", "made", "conference", "saying", "say",  |[277, 269, 131, 101, 95, 91, 86, 75, 74, 71]|
|                  | "released", "report", "cut", "developing"]                 |                                            |
|"greenhouse"      |["said", "change", "made", "conference", "say", "cut", "s  |[243, 168, 148, 87, 81, 78, 70, 65, 63, 60] |
|                  |aying", "meeting", "agreement", "told"]                     |                                            |
|"wildfire"        |["said", "fire", "told", "report", "burn", "burning", "bu  |[212, 112, 100, 89, 82, 79, 69, 68, 67, 63] |
|                  |rned", "reported", "flooding", "evacuated"]                 |                                            |
|"forest_fire"     |["said", "fire", "burn", "made", "told", "called", "fires  |[192, 94, 71, 69, 63, 57, 53, 49, 46, 43]   |
|                  |", "say", "make", "closed"]                                 |                                            |
|"drought"         |["said", "report", "told", "reported", "change", "floodin  |[177, 90, 87, 51, 48, 45, 44, 43, 41, 41]   |
|                  |g", "call", "drought", "noted", "fire"]                     |                                            |
|"earthquake"      |["said", "issued", "earthquake", "occurred", "collapsed",  |[113, 88, 81, 63, 53, 49, 49, 46, 43, 38]   |
|                  | "damage", "reported", "quake", "struck", "injuries"]       |                                            |
|"high_temperature"|["said", "issued", "report", "released", "reports", "repo  |[99, 86, 64, 63, 50, 49, 45, 43, 38, 38]    |
|                  |rted", "flooding", "made", "burning", "impacts"]            |                                            |
```

Figure 13: Example 3.1: ten triggers



```
|topic             |event_types                                                                                               |num_event_types                           |
|"climate_change"  |["type:statement", "type:causechangeofpositiononascale", "type:change", "type:communication", "type:a   |[98, 92, 89, 69, 52, 47, 47, 46, 46, 44]|
|                  |greeorrefusetoact", "type:influence", "type:causetobeincluded", "type:socialevent", "type:know", "typ   |                                          |
|                  |e:processend"]                                                                                            |                                          |
|"greenhouse"      |["type:statement", "type:causechangeofpositiononascale", "type:communication", "type:change", "type:a   |[88, 83, 58, 57, 44, 43, 43, 41, 40, 39]|
|                  |greeorrefusetoact", "type:processstart", "type:socialevent", "type:know", "type:causation", "type:cau   |                                          |
|                  |setobeincluded"]                                                                                          |                                          |
|"wildfire"        |["type:statement", "type:catastrophe", "type:processstart", "type:motion", "type:bodilyharm", "type:c   |[88, 80, 62, 59, 59, 58, 58, 50, 43, 43]|
|                  |ausation", "type:reporting", "type:destroying", "type:causechangeofpositiononascale", "type:know"]      |                                          |
|"drought"         |["type:statement", "type:catastrophe", "type:causechangeofpositiononascale", "type:reporting", "type:   |[83, 75, 60, 47, 44, 43, 42, 41, 38, 37]|
|                  |motion", "type:causation", "type:know", "type:influence", "type:processstart", "type:assistance"]       |                                          |
|"earthquake"      |["type:catastrophe", "type:attack", "type:statement", "type:damaging", "type:reporting", "type:bodily   |[80, 66, 63, 62, 61, 48, 48, 39, 36, 34]|
|                  |harm", "type:causation", "type:motion", "type:perceptionactive", "type:comingtobe"]                     |                                          |
|"high_temperature"|["type:statement", "type:catastrophe", "type:causation", "type:causechangeofpositiononascale", "type:   |[78, 63, 53, 50, 49, 46, 43, 40, 40, 37]|
|                  |reporting", "type:bodilyharm", "type:motion", "type:creating", "type:know", "type:destroying"]          |                                          |
|"forest_fire"     |["type:statement", "type:catastrophe", "type:destroying", "type:causation", "type:processstart", "typ   |[77, 68, 57, 51, 51, 51, 51, 43, 43, 39]|
|                  |e:motion", "type:bodilyharm", "type:know", "type:reporting", "type:arriving"]                           |                                          |
```

Figure 14: Example 3.2: ten event_tpyes

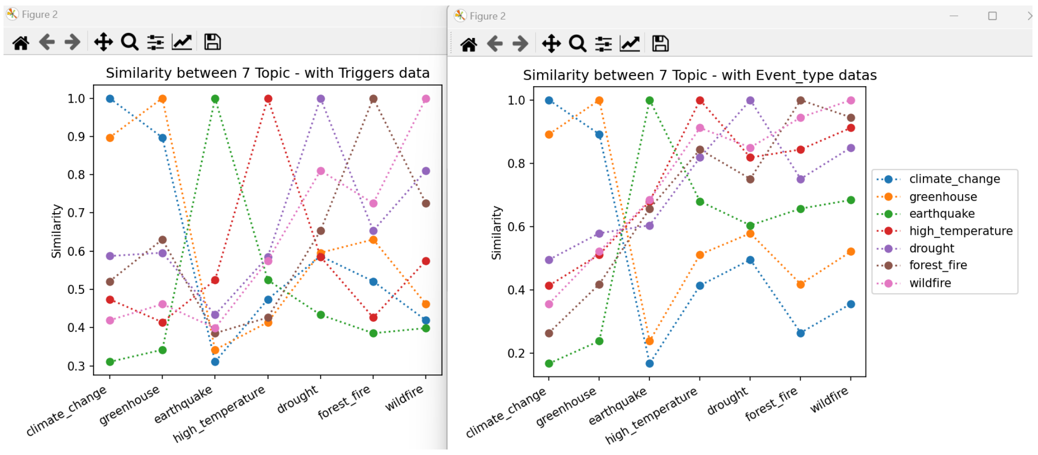


Figure 15: Example 3: after python processing

### 2.3 Download Graph style

**Command:**

```
:style
```

**Result:**
See stylesheets.grass file.

### 2.4 Export database

**Command:**
Go to the "/neo4j-community-4.4.21/bin" directory and execute the command in the terminal.

```
neo4j-admin dump --database "neo4j" --to "[PATH]\Access\NaCaGraph.db.dump"
```

**Result:**
See NaCaGraph.db.dump.