

Graph Database Generator Manual

Table of Contents

**Getting started3**

Program Overview3

Glossary4

Installation4

**Running the program5**

Flow chart5

Major functions6

**Interpreting Output**8

Folders8

JSON Outputs8

**Future Reference**10

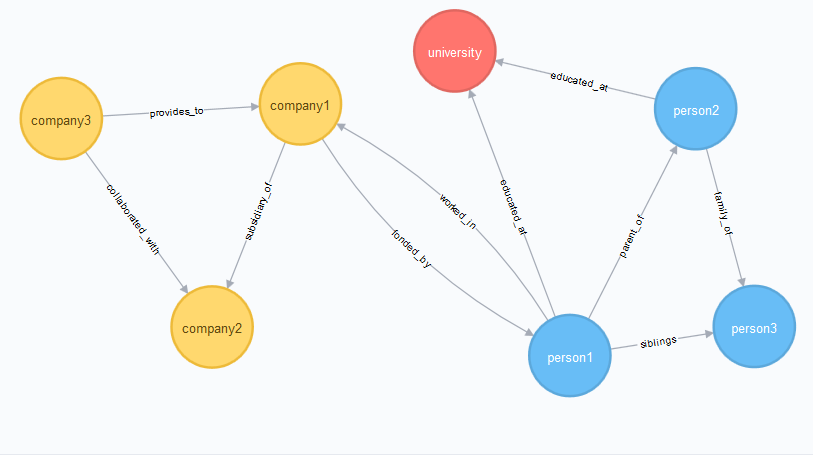
Challenges and improvements10

# Getting Started

Project overview

Director and Officer (D&O) liability insurance is a liability insurance payable as a reimbursement for loses on a company brought by the wrongful acts of directors or officers in this company[[1]](#footnote-1). To estimate the risk of insuring a company more accurately, we want to identify risky directors and their current companies based on the assumption that companies with riskier directors have bigger chances of suffering from a loss.

To achieve this goal, we want represent the relationships between D&O in specified companies with a graph database, where vertices are entities and edges are relationships. The relationship model is illustrated in figure 1.



*Figure 1. Relationships Graph Model*

In the long term, this graph database can be a valuable data asset contributing for different business project rather than merely D&O insurance.

Glossary

* SEC https://www.sec.gov/

U.S. Securities and Exchange Commission. The SEC holds primary responsibility for enforcing the federal [securities](https://en.wikipedia.org/wiki/Security_%28finance%29) laws, proposing securities rules, and regulating the securities industry, the nation's stock and options exchanges, and other activities and organizations, including the electronic securities markets in the United States

* EDGAR

EDGAR, the Electronic Data Gathering, Analysis, and Retrieval system, performs automated collection, validation, indexing, acceptance, and forwarding of submissions by companies and others who are required by law to file forms with the SEC

* CIK

A CIK number is a number given to an individual or company by the SEC. The number is used to identify the filings of a company, person, or entity in several online databases.

* DEF 14A

Also called a “definitive proxy statement,” Form DEF 14A is intended to furnish security holders with adequate information to be able to vote confidently at an upcoming shareholders' meeting

* Rosette Text Analytics https://www.rosette.com/

A tool that has capabilities of entities extraction, relations extraction, sentiment analysis and so on. In this project, I explore its API for entities and relations.

* Neo4j and Cypher https://neo4j.com/

Neo4j is the platform for creating a graph database. Cypher is the name of the language being used in Neo4j. In this project, we only used the Neo4j browser, where it has a terminal that we can input Cypher queries

Installation

Note you have to be on nationwide network to clone the repository.

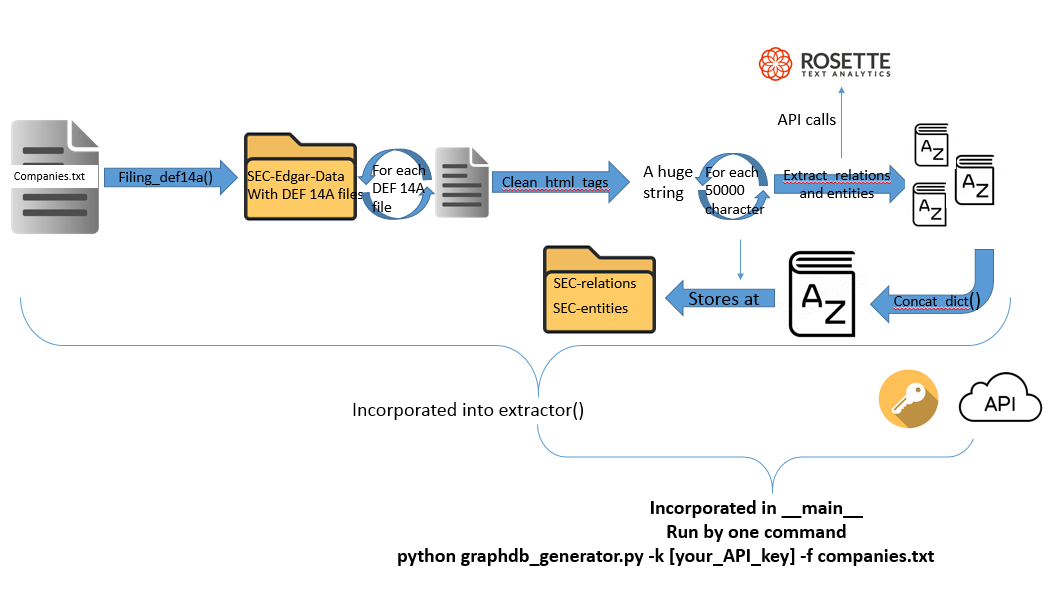
In your terminal:

$ git clone https://github.nwie.net/dengy1/EDO-Intern-Project.git  
$ cd Yuxi/sec-crawler  
$ python setup.py install

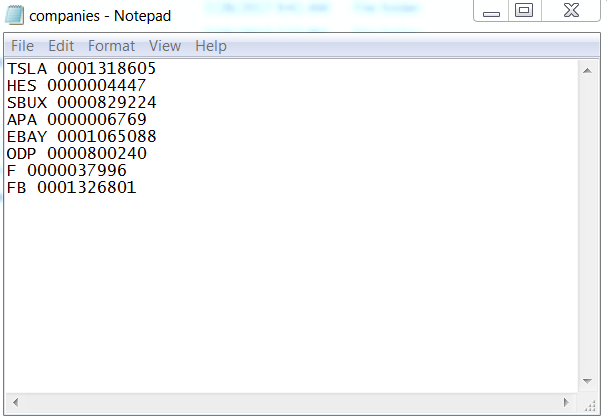
Now you have installed the program! Let’s see how to execute the program.

# Running the Program

Flowchart of the main program graphdb\_generator.py



As illustrated in the above graph, first type the companies’ ticker and CIK code in the companies.txt in a format: ticker 0010034554. For example:



After having the input of the program, simply run the command:

**python graphdb\_generator.py -k [Your\_API\_Key] -f companies.txt**

Where you\_API\_key can be obtained here https://developer.rosette.com/

Major functions

* filling\_DEF14A()
  + Input:(String company\_ticker, String company\_cik, String date,String count)
  + Output: None

Download all DEF 14A files from EDGAR automatically and store all the files in corresponding folders with path SEC-Edgar-Data/company\_ticker/company\_cik/DEF 14A/index\_year.txt, where index\_year is generated by EDGAR. Note that the file downloaded(index\_year.txt) is a raw html file with html tags

* clean\_html\_tags (filepath) returns a clean string without html tags
  + input: String filepath
  + output: String ascii\_str

The input filepath is the path to a raw html file. In this function, I utilize the package BeautifulSoup to clean up the html file so that it is more readable and more efficient to perform relations/entities extraction on.

* extract\_relations(string,API\_Key,URL)
  + Input: (String relations\_text\_data, string key,string altUrl=**'https://api.rosette.com/rest/v1/'**)
  + Output dict re\_result

performs relations extraction on the string relations\_text\_data. API key should be obtained from Rosette and specified in the terminal command, URL is set to default, so users need not worry about it. In this project, the string is the asci\_str returned by clean\_html\_tags()

* extract\_entities(string,API\_Key,URL)
  + Input: (String entities\_text\_data, string key, string altUrl=**'https://api.rosette.com/rest/v1/'**)
  + Output dict en\_result

performs entities extraction on the string entities\_text\_data. API key should be obtained from Rosette and specified in the terminal command, URL is set to default, so users need not worry about it. In this project, the string is the asci\_str returned by clean\_html\_tags()

* extractor(path,key,url)
  + input: String path,string key,string url

parse the companies.txt file

* concat\_dict( total\_str,entity\_dict,relation\_dict,key,url)
  + Input: String total\_str, dict entity\_dict, dict relation\_dict,string key,string url
  + Output:

Due to the maximum length of text Rosette can take, which is 50000, I break up the text and concatenate the results together into one JSON file by using concat\_dict()

# Interpreting Output

structure of folders

sec-crawler: the folder that as everything, including output and script

    SEC-Edgar-Data: the raw html files classified by folders named after companies tickl

         AAPL: All raw html files of APPLE

         DEF 14A: all DEF 14A html files

     SEC-relations:JSON files that document relationships object in a format ticker\_year\_relation

    SEC-entities:JSON files that document entities object in a format ticker\_year\_entities

Relationships and entities output explained

 relationships

Predicate: The main action or verb acting on the first argument, or the action connecting multiple arguments.

PredicateID: (optional) Label of the identified relationship type.

Argument(s): One or more subjects in the relationship.

Argument ID(s): (optional) The entity ID returned for arguments that can be linked to Wikidata.

Example:

"relationships": [

{

"arg1": "PayPal",

"arg1Id": "Q483959",

"arg2": "eBay",

"arg2Id": "Q58024",

"predicate": "Organization Acquired By",

"predicateId": "ORG-ACQUIRED-BY"

}]

This means Paypal is acquired by eBay

Entities

Mention:entity literal mention

Normalized: the real entity

Entity ID: If the entity exists in Wikidata, then Rosette returns the Wikidata QID. Otherwise, it creates a placeholder temporary (“T”) entity ID to link mentions of the same entity in the document. However, the TID may be different across documents for the same entity

Example

“Entities”:[{

"count": 16,

"entityId": "Q317521",

"mention": "Musk",

"normalized": "Messrs. Elon Musk",

"type": "PERSON"

}]

“Musk” appears 16 times in the text, where is actually linked to the entity “Messrs. Elon Musk”. The entity is a person and the corresponding QID in Wikidata is Q317521

# Future reference

Challenges

Network issue: to clone the GitHub, your device has to be connected with nationwide network. However, to run the program, your device has to be off nationwide network

Future Improvement

1. Deploy in the cloud to overcome the firewall in Nationwide
2. Scale up with threads or some services in the cloud

1. https://en.wikipedia.org/wiki/Directors\_and\_officers\_liability\_insurance#Corporate\_indemnification [↑](#footnote-ref-1)