No-Code Queries Can Accelerate AI and Data Analytics

By Dr. Jans Aasman, CEO

The low-code, no-code methodology is becoming highly sought-after throughout the modern IT ecosystem—and with good reason. Options that minimize manually writing code capitalize on the self-service, automation idiom that’s imperative in a world in which working remotely and doing more with less keeps organizations in business.

Most codeless or low-code approaches avoid the need for writing language-specific code and replace it with a visual approach in which users simply manipulate on-screen objects via a drag-and-drop, point-and-click interface to automate code generation. The intuitive ease of this approach – which is responsible for new standards of efficiency and democratization of no-code development – has now extended to no-code query writing.

No-code querying provides two unassailable advantages to the enterprise. First, it considerably expedites what is otherwise a time-consuming ordeal, thereby accelerating data analytics and AI-driven applications and second, it can help organizations overcome the talent shortage of developers and knowledge engineers. Moreover, it does so by furnishing all the above benefits that make codeless and low-code options mandatory for success.

Read the full article at DZone.

NLP: Unlock the Hidden Business Value in Voice Communications

By Dr. Jans Aasman, CEO, Franz Inc.

Today organizations capture an enormous amount of information in spoken conversations, from routine customer service calls to sophisticated claims processing interactions in finance and healthcare. But most of this information remains hidden and unused due to the difficulty of turning these conversations into meaningful data that can be effectively analyzed through Natural Language Processing (NLP).

Simply applying speech recognition software to voice conversations often results in unreliable data. State-of-the-art speech recognition systems still have trouble distinguishing between homophones (words with the same pronunciation, but different meanings), as well as the difference between proper names (i.e. people, products) and separate words. In addition, there is also the challenge of identifying domain-specific words accurately. Thus, in most cases, using speech recognition software alone doesn’t produce accurate enough data for reliable NLP.

Domain-specific taxonomies are key to understanding conversations via speech recognition systems. With them, we can feed conversations to knowledge graphs that understand the conversation and make connections in the data. Knowledge graphs provide the ability to extract the correct meaning of text from conversations and connect concepts in order to add business value.

Knowledge graphs fed with NLP provide two prime opportunities for monetization. First, organizations can better understand their customers to improve products and services more to their liking, which in turn boosts marketing, sales and customer retention rates. Secondly, this analysis gives contact center agents real-time support for optimizing customer interactions to produce faster resolutions, better conversion rates, and cross-selling and up-selling opportunities. These approaches enable companies to capitalize on speech recognition knowledge graphs, accelerate their ROI, and expand their bottom lines.

Taxonomy Driven Speech Recognition

The story of taxonomy-driven speech recognition closely relates to knowledge graphs. The first wave of knowledge graphs was built from taking structured data and turning it into semantic graphs that support the linked open data movement. The next wave is all about unstructured data. People started doing Natural Language Processing on documents and textual conversations like emails and chats. Doing so accurately for a given domain requires a taxonomy to understand the words and concepts. Otherwise, downstream processes like entity extraction and event detection won’t work.

Read the full article at DZone.

Data-Centric Architecture Forum – DCAF 2021

Data and the subsequent knowledge derived from information are the most valuable strategic asset an organization possesses. Despite the abundance of sophisticated technology developments, most organizations don’t have disciplines or a plan to enable data-centric principles.

DCAF 2021 will help provide clarity.

Our overarching theme for this conference is to **make it REAL**. Real in the sense that others are becoming data-centric, it is achievable, and you are not alone in your efforts.

Join us in understanding how data as an open, centralized resource outlives any application. Once globally integrated by sharing a common meaning, internal and external data can be readily integrated, unlike the traditional “application-centric” mindset predominantly used in systems development.

The compounding problem is these application systems each have their own completely idiosyncratic data models. The net result is that after a few decades, hundreds or thousands of applications implemented have given origin to a segregated family of disparate data silos. Integration debt rises and unsustainable architectural complexity abounds with every application bought, developed, or rented (SaaS).

Becoming data-centric will improve data characteristics of findability, accessibility, interoperability, and re-usability (FAIR principles), thereby allowing data to be exported into any needed format with virtually free integration.\

Dr. Jans Aasman to present – Franz's approach to Entity Event Data Modeling for Enterprise Knowledge Fabrics

---

Text Analytics Forum 2020 – KMWorld Connect

Join us November 17, 2020 – Text Analytics has the ability to add depth, meaning, and intelligence to any organization's most under-utilized resource – text. Through text analytics, enterprises can unlock a wealth of information that would not otherwise be available. Join us as we explore the power of text analytics to provide relevant, valuable, and actionable data for enterprises of all kinds.

Jans Aasman to present – Analyzing Spoken Conversations for Real-Time Decision Support in Mission-Critical Applications

November 17, 2020 at 2PM Eastern

Knowledge Graphs: A Single Source of Truth for the Enterprise

Dr. Jans Aasman, CEO, Franz Inc.

The notion of a "single source of truth" for the enterprise has been the proverbial moving goalpost for generations of CIOs, only to be waylaid by brittle technology and unending legacy systems. Truth-seeking visions rebuffed by technological trends have continuously confounded business units trying to achieve growth and market penetration. But technology innovation has finally led us to a point where CIOs can now deliver that truth.

Graphing the Truth

Knowledge graphs possess the power to deliver a single source of truth by linking together any assortment of data sources required, standardizing their diversity of data elements, and eliminating silos. They support the most advanced analytics options and decentralized transactions, which is why they're now deployed as systems of records for some of the most significant, mission-critical use cases affecting our population.

Because they scale to include almost any number of applications – and link to other knowledge graphs as well – these repositories are the ideal solution for real-time information necessary to inform business users' performances with concrete, data-supported facts. Most importantly, users can get an exhaustive array of touchpoints pertaining to any customer, product, or interaction with an organization from the knowledge graph, making it a single source of truth.

Read the full article at Dataversity.

Using Microsoft Power BI with AllegroGraph

There are multiple methods to integrate AllegroGraph SPARQL results into Microsoft Power BI. In this document we describe two best practices to automate queries and refresh results if you have a production AllegroGraph database with new streaming data:

The first method uses Python scripts to feed Power BI. The second method issues SPARQL queries directly from Power BI using POST requests.

Method 1: Python Script:

Assuming you know Python and have it installed locally, this is definitely the easiest way to incorporate SPARQL results into Power BI. The basic idea of the method is as follows: First, the Python script enables a connection to your desired AllegroGraph repository. Then we utilize AllegroGraph's Python API within our script to run a SPARQL query and return it as a Pandas dataframe. When running this script within Power BI Desktop, the Python scripting service recognizes all unique dataframes created, and allows you to import the dataframe into Power BI as a table, which can then be used to create visualizations.

Requirements:

1. You must have the AllegroGraph Python API installed. If

you do not, installation instructions are here: https://franz.com/agraph/support/documentation/current/python/install.html

2. Python scripting must be enabled in Power BI Desktop. Instructions to do so are here: https://docs.microsoft.com/en-us/power-bi/connect-data/desktop-python-scripts

   a) As mentioned in the article, pandas and matplotlib must be installed. This can be done with ‘pip install pandas’ and ‘pip install matplotlib’ in your terminal.

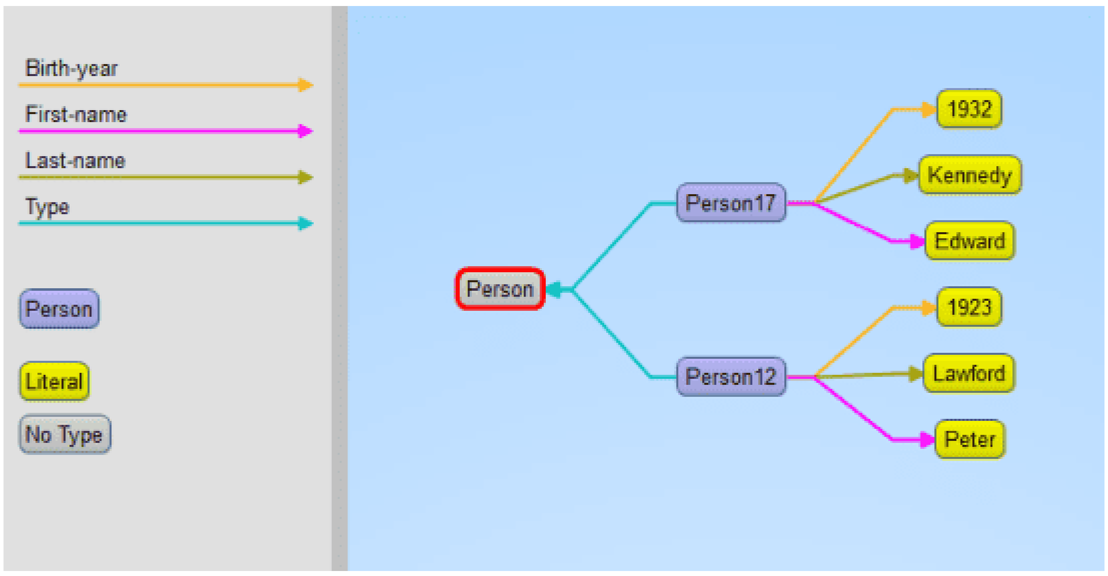
The Process:

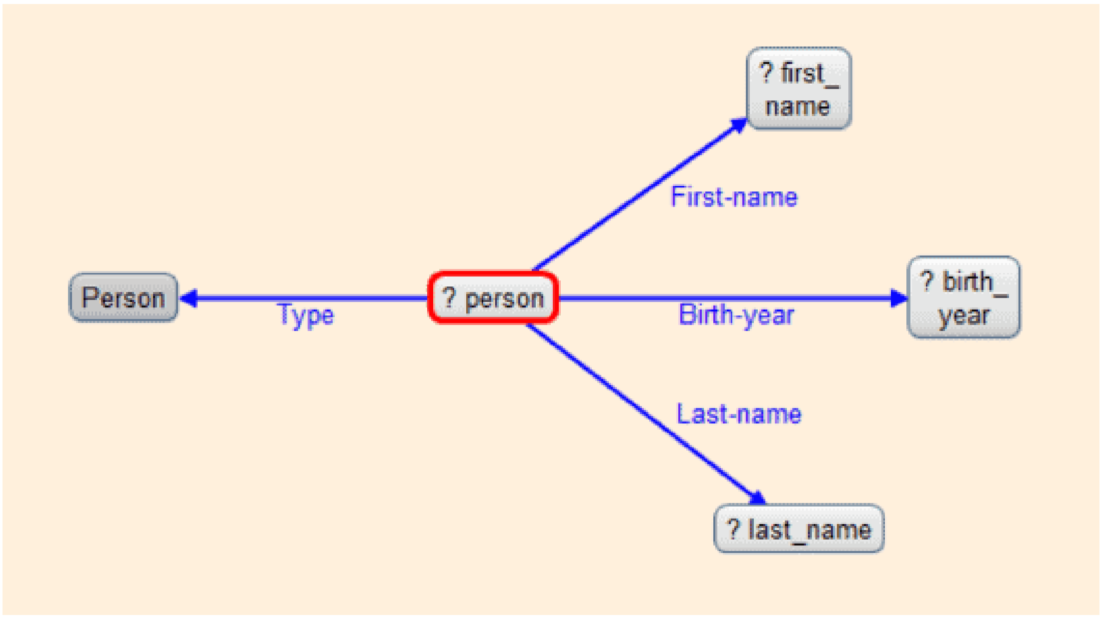
Once these requirements have been met, create a Python file with whatever script editor you usually use. The following code will create a connection to your desired repository. For this example, we will be using the Kennedy dataset that is available with the AllegroGraph distribution (See the ‘Tutorial’ directory). Load the Kennedy.ntriples file into your running AllegroGraph. (Replace the ‘****’ in the code with your corresponding username and password.)

```
#the necessary imports

import os

from franz.openrdf.connect import ag_connect

from franz.openrdf.query.query import QueryLanguage

import pandas as pd


#connect to your agraph repository

def setup_env_var(var_name, value, description):

os.environ[var_name] = value
```

```
print("{}: {}".format(description, value))

setup_env_var('AGRAPH_HOST', 'localhost', 'Hostname')

setup_env_var('AGRAPH_PORT', '10035', 'Port')

setup_env_var('AGRAPH_USER', '****', 'Username')

setup_env_var('AGRAPH_PASSWORD', '****', 'Password')

conn = ag_connect('kennedy', create=False, clear=False)
```

2. We then want to create a query. For this example, we will first show what our data looks like, what the visual query of the information is, and what the written query looks like. With the following query we want every person’s first and last names, as well as their birth years. Here is a small portion of the data visualized in Gruff, and then the visualization of the query:

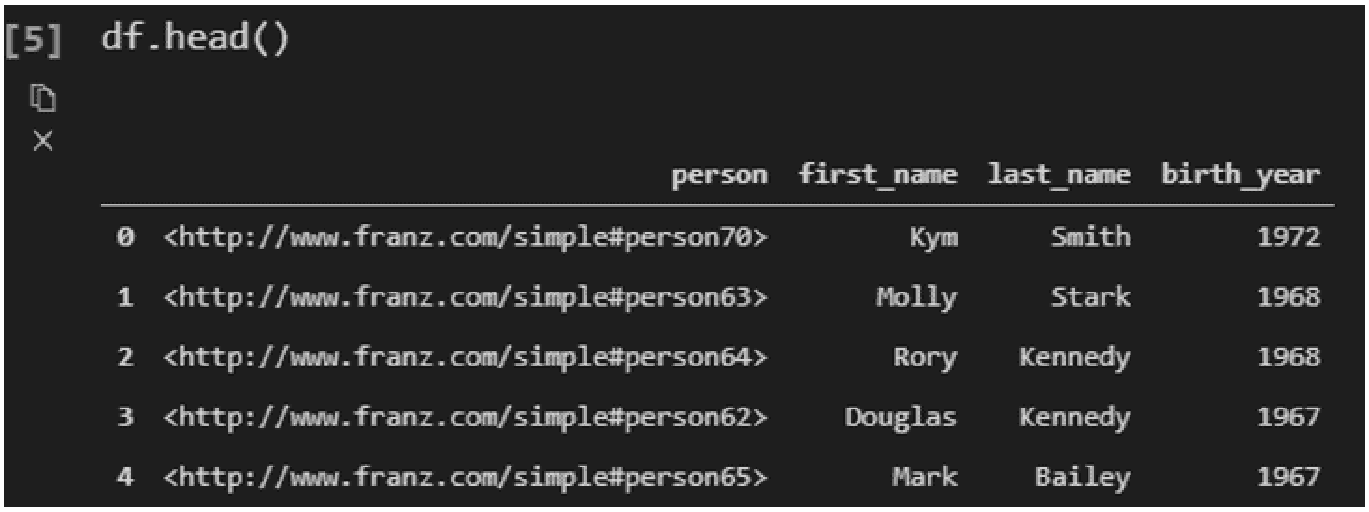




3. Then add the written query to the python script as a variable string (we added an additional line to the query to sort on birth year). Next use the API functionality to simply execute the query and turn the results into a pandas dataframe.

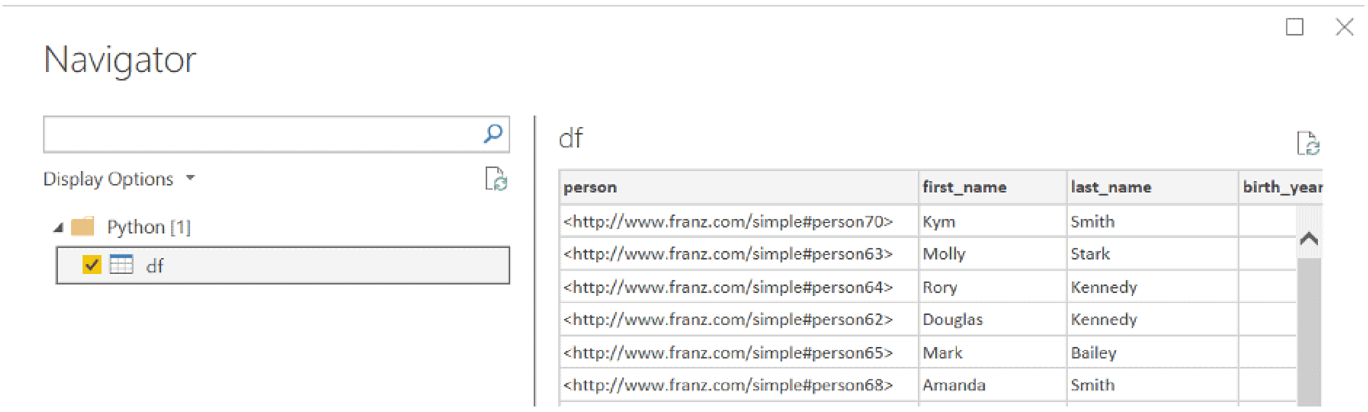
```
query = """select ?person ?first_name ?last_name ?birth_year
where
{ ?person <http://www.franz.com/simple#first-name> ?first_name
;
          <http://www.franz.com/simple#birth-year> ?birth_year
;
          rdf:type <http://www.franz.com/simple#person> ;
          <http://www.franz.com/simple#last-name> ?last_name .
}
order by desc(?birth_year)"""

with conn.executeTupleQuery(query) as result:
   df = result.toPandas()
```

When looking at the result, we see that we have a DataFrame!



```
[5]  df.head()

                                    person  first_name  last_name  birth_year
0  <http://www.franz.com/simple#person70>         Kym      Smith        1972
1  <http://www.franz.com/simple#person63>       Molly      Stark        1968
2  <http://www.franz.com/simple#person64>        Rory    Kennedy        1968
3  <http://www.franz.com/simple#person62>     Douglas    Kennedy        1967
4  <http://www.franz.com/simple#person65>        Mark     Bailey        1967
```

4. Now we will use this script in Power BI. When in Power BI Desktop, go to ‘Get Data’ and look for the python script option. Then simply copy and paste your entire script into the text box, and run the script. In this case, our output looks like this:



df

person	first_name	last_name	birth_year
<http://www.franz.com/simple#person70>	Kym	Smith	
<http://www.franz.com/simple#person63>	Molly	Stark	
<http://www.franz.com/simple#person64>	Rory	Kennedy	
<http://www.franz.com/simple#person62>	Douglas	Kennedy	
<http://www.franz.com/simple#person65>	Mark	Bailey	
<http://www.franz.com/simple#person68>	Amanda	Smith	

5. Next simply ‘Load’ the data, and then you can use the Power BI Desktop interface to create whatever visualizations you want! If you do have a lot of additional operations to perform on your dataframe, we recommend doing these in your python script.

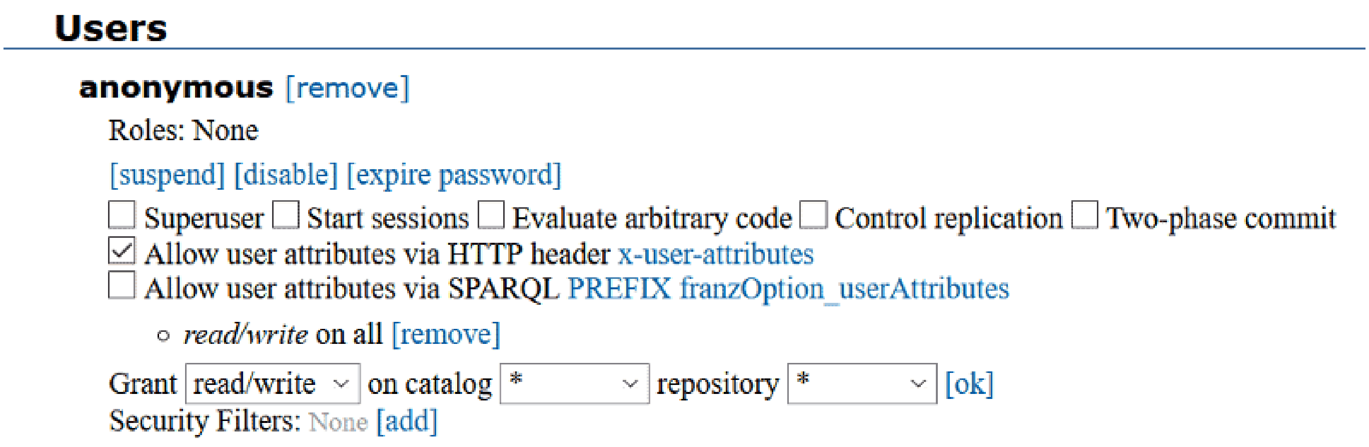
Method 2: POST Request:

For the SPARQL query via POST requests to work you need to url-encode the query. Every modern programming language will support that, but in our example we will be using Python again. This method is better for when you do not have python locally installed or prefer a different programming language.

It is possible to send a GET request from Power BI, but once the results from the query reach a certain size, a POST request is required, which is confusing to do within the Power BI Desktop interface. The following steps will show you how to do SPARQL Queries using POST requests. It looks a bit odd but it works well.

The Process:

1. In your AG WebView create an 'anonymous' user. (Go to admin -> Users -> [add a user] -> and add 'anonymous' as username without adding a password). You can use these settings:



Users

anonymous [remove]

Roles: None

[suspend] [disable] [expire password]

☐ Superuser ☐ Start sessions ☐ Evaluate arbitrary code ☐ Control replication ☐ Two-phase commit
☑ Allow user attributes via HTTP header x-user-attributes
☐ Allow user attributes via SPARQL PREFIX franzOption_userAttributes

- *read/write* on all [remove]

Grant read/write on catalog * repository * [ok]
Security Filters: None [add]

2. Go to your desired repository in WebView and Click on 'Queries' -> 'New'

3. Write a simple SPARQL query, and run it to make sure you get the correct response back.

4. In python create the following script: (Assuming your AllegroGraph is on your localhost port 10035 and your repo is called 'kennedy')

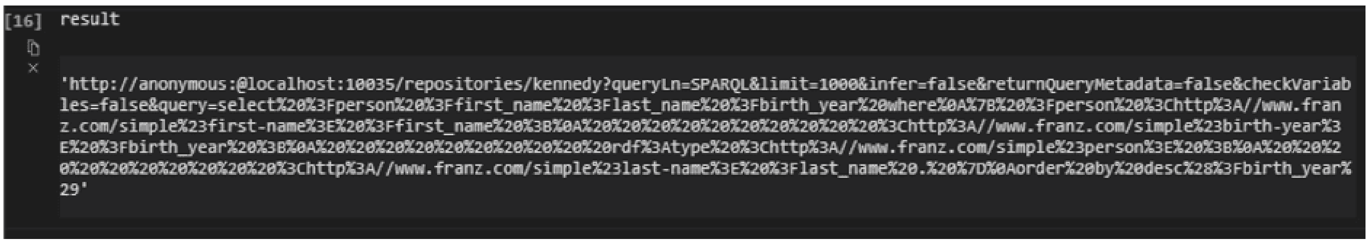
```
import urllib
```

```
def CreatePOSTquery(query):
    start = "http://anonymous:@localhost:10035/repositories/kennedy?queryLn=SPARQL&limit=1000&infer=false&returnQueryMetadata=false&checkVariables=false&query="
    response = start + urllib.parse.quote(query)
    return response
```

This function url-encodes the query and attaches it to the POST request. Replace the ‘localhost:10035’ and ‘kennedy’ strings in the start variable with your corresponding data. Then, using the same query as our previous example, we create our url-encoded POST query:

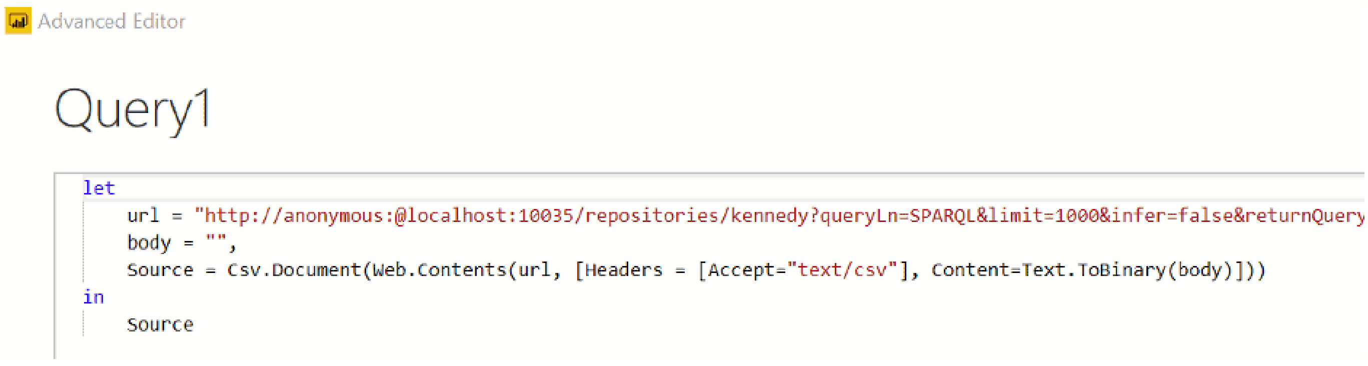
```
query = """select ?person ?first_name ?last_name ?birth_year where
{ ?person <http://www.franz.com/simple#first-name> ?first_name ;
          <http://www.franz.com/simple#birth-year> ?birth_year ;
          rdf:type <http://www.franz.com/simple#person> ;
          <http://www.franz.com/simple#last-name> ?last_name .
}
order by desc(?birth_year)"""

result = CreatePOSTquery(query)
print(result)
```

This gives us the following result:



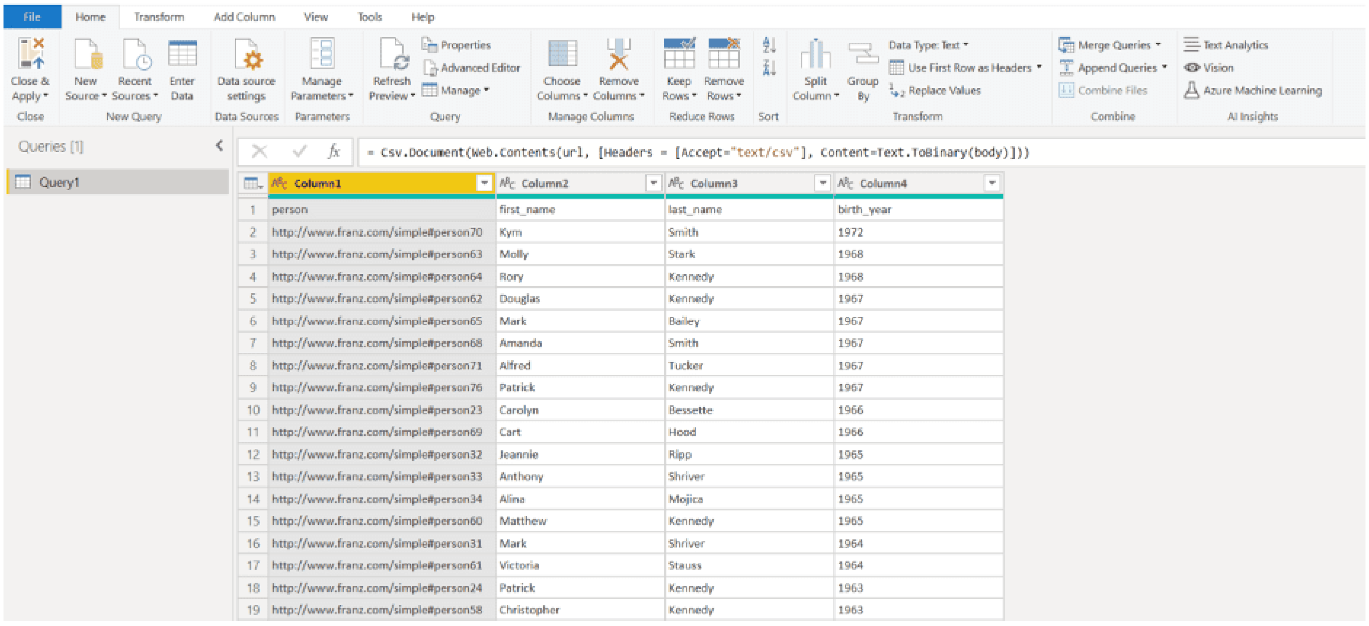
```
[16] result

'http://anonymous:@localhost:10035/repositories/kennedy?queryLn=SPARQL&limit=1000&infer=false&returnQueryMetadata=false&checkVariab
les=false&query=select%20%3Fperson%20%3Ffirst_name%20%3Flast_name%20%3Fbirth_year%20where%0A%7B%20%3Fperson%20%3Chttp%3A//www.fran
z.com/simple%23first-name%3E%20%3Ffirst_name%20%3B%0A%20%20%20%20%20%20%20%20%20%20%3Chttp%3A//www.franz.com/simple%23birth-year%3
E%20%3Fbirth_year%20%3B%0A%20%20%20%20%20%20%20%20%20%20rdf%3Atype%20%3Chttp%3A//www.franz.com/simple%23person%3E%20%3B%0A%20%20%2
0%20%20%20%20%20%20%20%3Chttp%3A//www.franz.com/simple%23last-name%3E%20%3Flast_name%20.%20%7D%0Aorder%20by%20desc%28%3Fbirth_year%
29'
```

5. Within Power BI Desktop we go to ‘Get data’ and create a ‘Blank query’ and go into the ‘Advanced Editor’ window. Using the following format we will get our desired results (please note that due to the length of the url-encoded request, it did not all fit in the image. Copy and pasting into the url field works fine. The ‘url’ variable needs to be in quotes and have a comma at the end):



```
let
    url = "http://anonymous:@localhost:10035/repositories/kennedy?queryLn=SPARQL&limit=1000&infer=false&returnQuery
    body = "",
    Source = Csv.Document(Web.Contents(url, [Headers = [Accept="text/csv"], Content=Text.ToBinary(body)]))
in
    Source
```

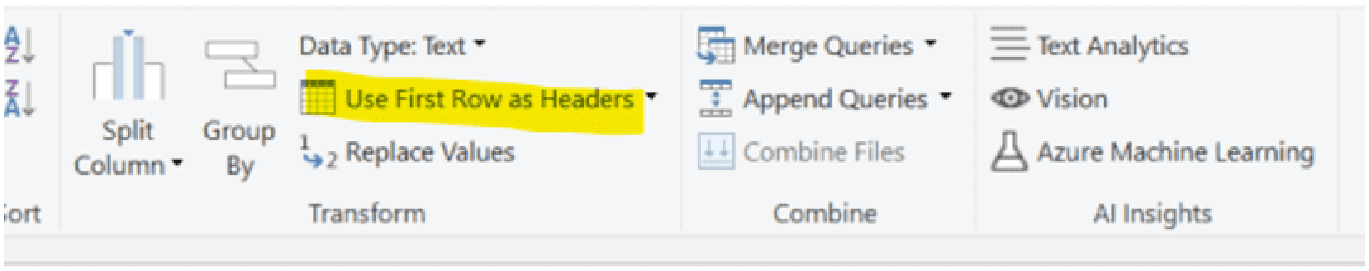
We see the following results:



```
= Csv.Document(Web.Contents(url, [Headers = [Accept="text/csv"], Content=Text.ToBinary(body)]))
```

	Column1	Column2	Column3	Column4
1	person	first_name	last_name	birth_year
2	http://www.franz.com/simple#person70	Kym	Smith	1972
3	http://www.franz.com/simple#person63	Molly	Stark	1968
4	http://www.franz.com/simple#person64	Rory	Kennedy	1968
5	http://www.franz.com/simple#person62	Douglas	Kennedy	1967
6	http://www.franz.com/simple#person65	Mark	Bailey	1967
7	http://www.franz.com/simple#person68	Amanda	Smith	1967
8	http://www.franz.com/simple#person71	Alfred	Tucker	1967
9	http://www.franz.com/simple#person76	Patrick	Kennedy	1967
10	http://www.franz.com/simple#person23	Carolyn	Bessette	1966
11	http://www.franz.com/simple#person69	Cart	Hood	1966
12	http://www.franz.com/simple#person32	Jeannie	Ripp	1965
13	http://www.franz.com/simple#person33	Anthony	Shriver	1965
14	http://www.franz.com/simple#person34	Alina	Mojica	1965
15	http://www.franz.com/simple#person60	Matthew	Kennedy	1965
16	http://www.franz.com/simple#person31	Mark	Shriver	1964
17	http://www.franz.com/simple#person61	Victoria	Stauss	1964
18	http://www.franz.com/simple#person24	Patrick	Kennedy	1963
19	http://www.franz.com/simple#person58	Christopher	Kennedy	1963

6. One last step is to turn the top row into the column names, which can be achieved by pressing the ‘Use first row as headers’:



The best part about both of these methods is that once the query has been created, Power BI can refresh the visuals using the same queries if your data changed. This can be achieved by scheduling refreshes within the Power BI Desktop interface (https://docs.microsoft.com/en-us/power-bi/connect-data/refresh-data#configure-scheduled-refresh)

Please send any questions or issues to: support@franz.com

Knowledge graphs enhance customer experience through speed and accuracy

KMWorld's recent article covers AllegroGraph and Franz's customer N3 Solutions.

The Full Article – KMWorld

KMWorld

Knowledge graphs are a way to model enterprise knowledge and represent complex interrelationships in data. Information stored in a graph database can enable rapid retrieval of well-targeted results and provide insights into customers' interests and needs. Gartner predicts a 100% per-year growth in applications for graph analytics and databases for the next several years. Although knowledge graphs have been deployed by major companies such as Google, Amazon, and LinkedIn due to their ability to incorporate relationships in their analyses as well as their speed, only in the last 5 years has their use become more widespread.

N3 is an outsourced sales company for major organizations that sell complex B2B software, hardware, and tech solutions. It supports businesses in 92 countries, provides services in 25 languages, and holds thousands of hours of conversations every month with customers and prospects. "In today's world of complex products, it takes a well-educated team to tell the story about how this technology can help a company become more competitive," said Shannon Copeland, COO of N3. "The sales team needs to be able to instantly access the information they need to do their job."

Faster insights

The company has been operating for 16 years, and in the last few years began an initiative to manage its knowledge in a more intentional way. "We generate a great deal of data," noted Copeland, "and we wanted to make more effective use of it to understand our customers. And because of the speed at which business is transacted now, we needed to get insights right away, not a month later in a report."

N3 built a data model to reflect the essential data elements and the associations among them and decided that a knowledge graph was the best way to represent the information. After looking into partner options, N3 chose to work with Franz, Inc., which provides a semantic graph database called AllegroGraph. "We decided to work with Franz because of its extensive experience and the fact that it had worked with a variety of industries,” Copeland said.

The system built by N3 allows sales teams to organize signals from the market in a way that allows them to better explain the products to prospective buyers. “We build relationships with tech buyers on behalf of our clients,” continued Copeland. “Our employees are typically college graduates who would like to begin their careers in sales and marketing in tech solutions. They take ownership of their territory and we help them be as sophisticated as a future CMO would be.” The resources supplied by the knowledge graph provide the support the sales team needs to tailor information to each prospective customer.

The specific expertise required by the team varies depending on the products being sold, the geographic region, and other factors, and the knowledge graph supports these needs. For example, if a team in southern Portugal needs to know the preferences of that market, the associations built into the graph database can provide the information that is essential for them. “The information we can access helps customers understand the answers to their questions very quickly,” Copeland commented. “We believe the experience that the customers have helps them scope out what they need and what the road map might be.”

The strength of graph databases

A graph databases is a type of NoSQL database that stores data according to associations among data elements rather than in the rows and columns of a relational database. Because graph databases use a dynamic schema rather than a fixed, table-based one, adding new data types and categories is much easier. And because they are semantics-based, graph databases have strengths in inferring intent, producing answers to questions, and making recommendations. They can also make inferences about possible associations from existing associations.

A graph database also provides much more context than a relational database and therefore can return more relevant results when a user is searching; they also integrate data from multiple sources. “At one telecom company we worked with, customer service reps might have [had] to open 15 databases to find out what went wrong and what the solution was,” said Jans Aasman, CEO of Franz. “We took their core customer data, billing information, every CRM call, and every action and put them into AllegroGraph, and the customer service reps were finally able to respond in a meaningful way, whether that was to make an offer to the customer or provide appropriate technical support.” The capability of graph databases to overcome silos and provide an integrated view of the customer is one of its strengths.

In order to create the graph database on which the knowledge graph is built, the relationships among entities need to be mapped. In the case of a hospital patient, the patient is the core entity, and the events are medical encounters or lab results, which may come out of different databases or a data warehouse. “The mapping is a major project, but it only needs to be done once,” Aasman pointed out. “After that, the relationships do not need to be regenerated during the search because they are indexed in AllegroGraph, which makes retrieval very rapid.”

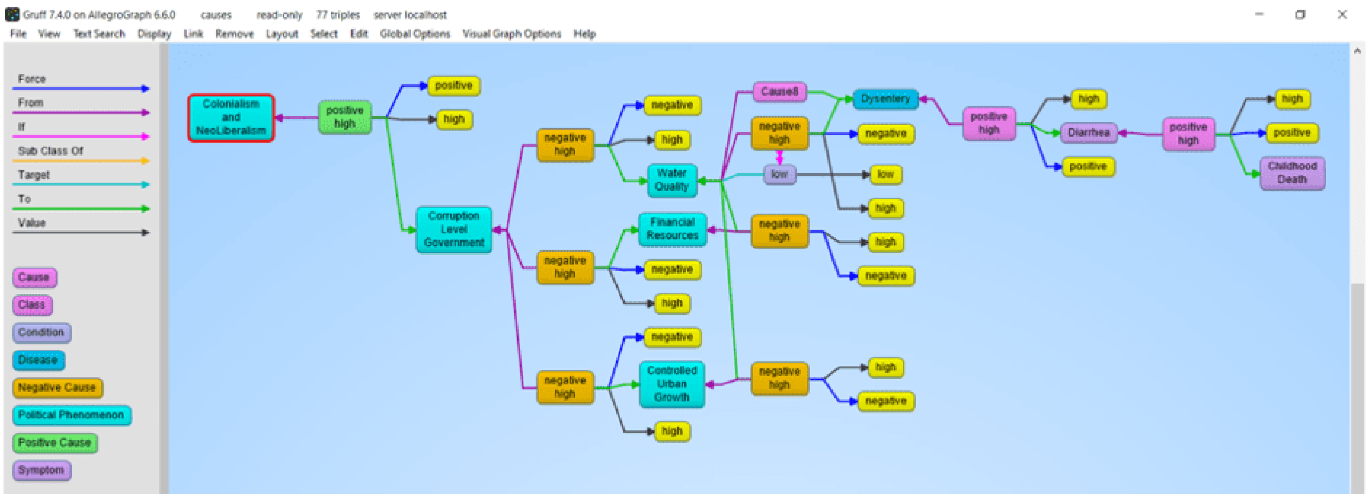
Document Knowledge Graphs with NLP and ML

A core competency for Franz Inc is turning text and documents into Knowledge Graphs (KG) using Natural Language Processing (NLP) and Machine Learning (ML) techniques in combination with AllegroGraph. In this document we discuss how the techniques described in [NLP and ML components of AllegroGraph] can be combined with popular software tools to create a robust Document Knowledge Graph pipeline.

We have applied these techniques for several Knowledge Graphs but in this document we will primarily focus on three completely different examples that we summarize below. First is the Chomsky Legacy Project where we have a large set of very dense documents and very different knowledge sources, Second is a knowledge graph for an intelligent call center where we have to deal with high volume dynamic data and real-time decision support and finally, a large government organization where it is very important that people can do a semantic search against documents and policies that steadily change over time and where it is important that you can see the history of documents and policies.

Example [1] Chomsky Knowledge Graph
The Chomsky Legacy Project is a project run by a group of admirers of Noam Chomsky with the primary goal to preserve all his written work, including all his books, papers and interviews but also everything written about him. Ultimately students, researchers, journalists, lobbyists, people from the AI community, and linguists can all use this knowledge graph for their particular goals and questions.

The biggest challenges for this project are finding causal relationships in his work using event and relationship extraction. A simple example we extracted from an author quoting Chomsky is that neoliberalism ultimately causes childhood death.



Example 2: N3 Results and the Intelligent Call Center

This is a completely different use case (See a recent KMWorld Articlehttps://allegrograph.com/knowledge-graphs-enhance-customer-experience-through-speed-and-accuracy/). Whereas the previous use case was very static, this one is highly dynamic. We analyze in real-time the text chats and spoken conversations between call center agents and customers. Our knowledge graph software provides real-time decision support to make the call center agents more efficient. N3 Results helps big tech companies to sell their high tech solutions, mostly cloud-based products and services but also helps their clients sell many other technologies and services.

The main challenge we tackle is to really deeply understand what the customer and agent are talking about. None of this can be solved by only simple entity extraction but requires elaborate rule-based and machine learning techniques. Just to give a few examples. We want to know if the agent talked about their most important talking points: that is, did the agent ask if the customer has a budget, or the authority to make a decision or a timeline about when they need the new technology or whether they actually have expressed their need. But also whether the agent reached the right person, and whether the agent talked about the follow-up. In addition, if the customer talks about competing technology we need to recognize that and provide the agent in real-time with a battle card specific to the competing technology. And in order to be able to do the latter, we also analyzed the complicated marketing materials of the clients of N3.

Example 3: Complex Government Documents
Imagine a regulatory body with tens of thousands of documents. Where nearly every paragraph has reference to other paragraphs in the same document or other documents and the documents change over time. The goal here is to provide the end-users in the government with the right document given their current task at hand. The second goal is to keep track of all the changes in the documents (and the relationship between documents) over time.

The Document to Knowledge Graph Pipeline

Process Name	Input	Output
1. Custom Taxonomy Creation	Corpus Analytics, Taxonomy tool	A SKOS taxonomy containing concepts, concept hierarchy, prefLabels, altLabels.
2. Document Preparation	Documents (pdf, word, ppt, xlsx), Apache Tika, Spacy for XML cleanup	An XML version of each document
3. Extract Document Meta Data	Document + Apache Tika	JSON dictionary of the Document MetaData
4. XML-to-Triples	XML+JSON dictionary, XMLToTriples.py	Graph-based document tree with chapters, sections, and paragraphs as triples. Also includes meta data as triples
5. Entity-Extraction	Paragraphs + taxonomies + AllegroGraph Entity extract or external extractors	Concepts, persons, places, currencies. Connected to paragraphs
6. LOD Enrichment	Paragraphs + IBM Natural Language Understanding.	Concept categories and links to DBpedia and GeoNames, etc.
7. Complex Relationship and Event extraction.	Paragraphs + Taxonomy + Rules in Spacy or AllegroGraph	Complex events and relationships, References to other document sections.
8. NLP and ML	Chapters and paragraphs + all the tools described [here], but also using Spacy, Gensim, BERT, SciKit Learn.	Similarities, sentiment, query answering, smart search, text classification, word embeddings, abstracts
9. Versioning and Document tracking	Old + New document, compare.py	Old document in historic repository, new document in current, changed graph.
10. Statistical Relationships	Concepts + OddRatio.py or OddsRatio.cl	Statistical relationships between concepts.

Let us first give a quick summary in words of how we turn documents into a Knowledge Graph.

[1] Taxonomy Creation

Taxonomy of all the concepts important to the business using open source or commercial taxonomy builders. An available industry taxonomy is a good starting point for additional customizations.

[2] Document Preparation

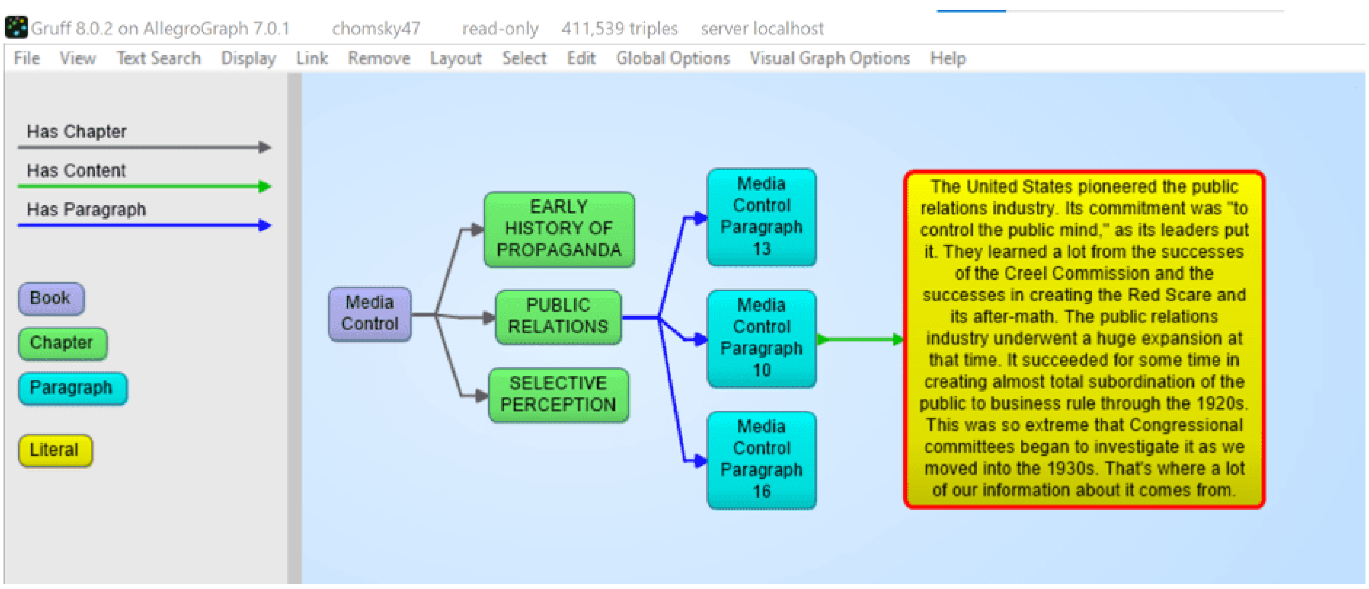
We then take a document and turn it into an intermediate XML using Apache Tika. Apache Tika supports more than 1000 document types and although Apache Tika is a fantastic tool, the output is still usually not clean enough to create a graph from, so we use Spacy rules to clean up the XML to make it as uniform as possible.

[3] Extract Document MetaData

Most documents also contain document metadata (author, date, version, title, etc) and Apache Tika will also deliver the metadata for a document as a JSON object.

[4] XML to Triples

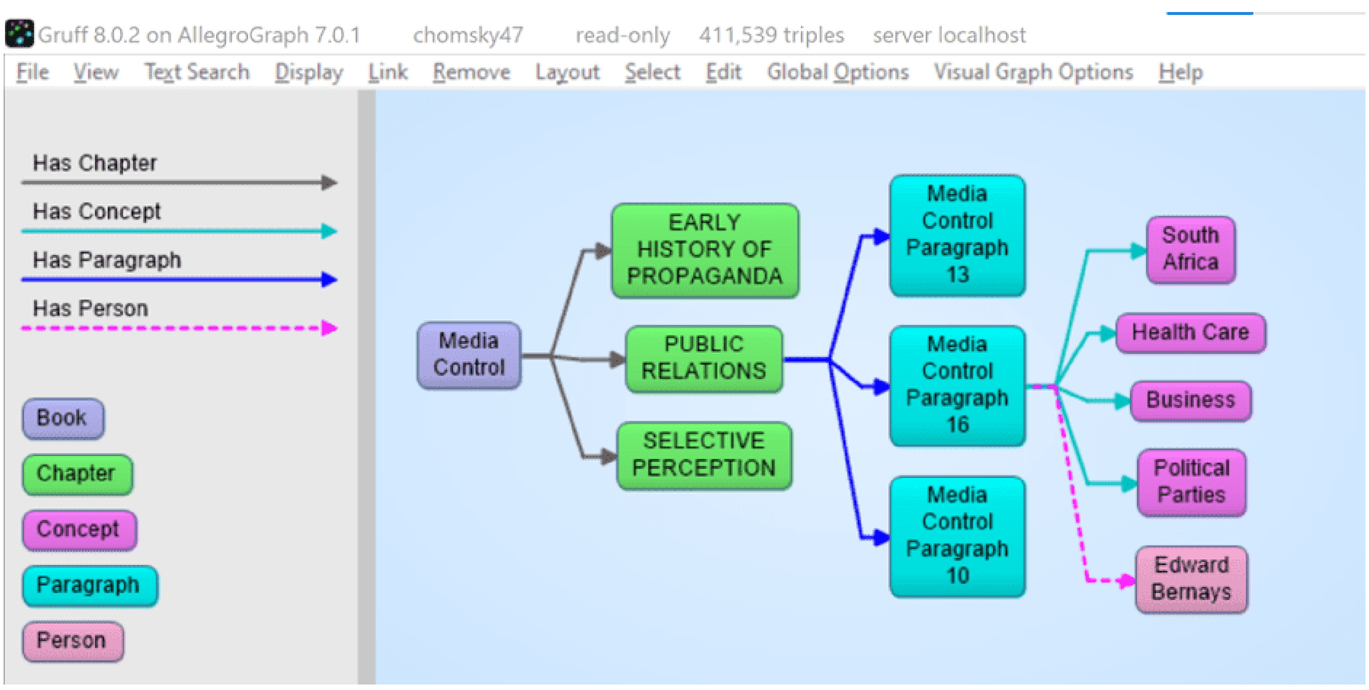
Our tools ingest the XML and metadata and transform that into a graph-based document tree. The document is the root and from that, it branches out into chapters, optionally sections, all the way down to paragraphs. The ultimate text content is in the paragraphs. In the following example we took the XML version of Noam Chomsky’s book Media Control and turned that into a tree. The following shows a tiny part of that tree. We start with the Media Control node, then we show three (of the 11) chapters, for one chapter we show three (of the 6) paragraphs, and then we show the actual text in that paragraph. We sometimes can go even deeper to the level of sentences and tokens but for most projects that is overkill.

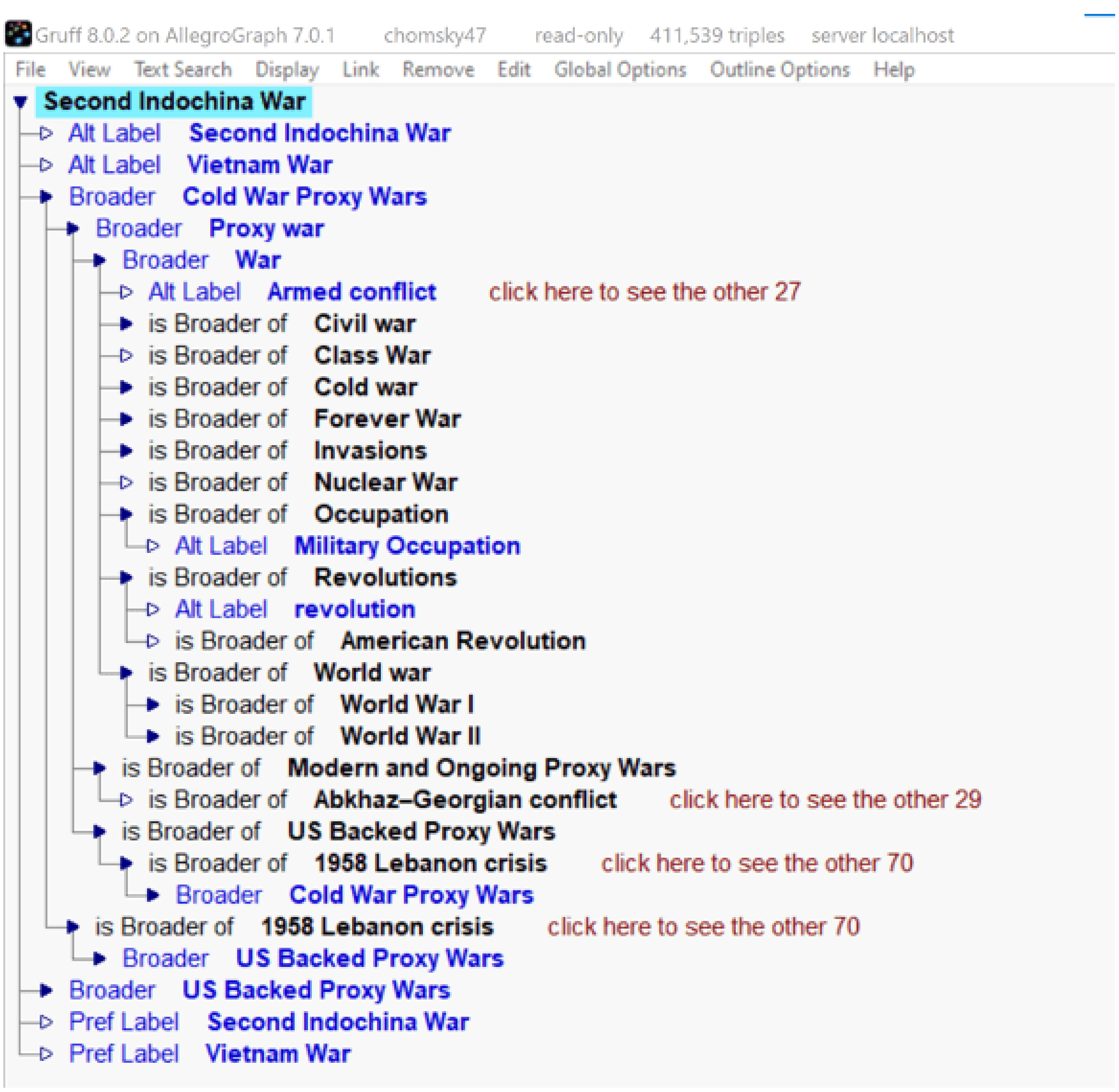


The United States pioneered the public relations industry. Its commitment was "to control the public mind," as its leaders put it. They learned a lot from the successes of the Creel Commission and the successes in creating the Red Scare and its after-math. The public relations industry underwent a huge expansion at that time. It succeeded for some time in creating almost total subordination of the public to business rule through the 1920s. This was so extreme that Congressional committees began to investigate it as we moved into the 1930s. That's where a lot of our information about it comes from.

[5] Entity Extractor

AllegroGraph's entity extractor takes as input the text of each paragraph in the document tree and one or more of the taxonomies and returns recognized SKOS concepts based on prefLabels and altLabels. AllegroGraph's entity extractor is state of the art and especially powerful when it comes to complex terms like product names. We find that in our call center a technical product name can sometimes have up to six synonyms or very specific jargon. For example the Cisco product Catalyst 9000 will also be abbreviated as the cat 9k. Instead of developing altLabels for every possible permutation that human beings *will* use, we have specialized heuristics to optimize the yield from the entity extractor. The following picture shows 4 (of the 14) concepts discovered in paragraph 16. Plus one person that was extracted by IBM's NLU.

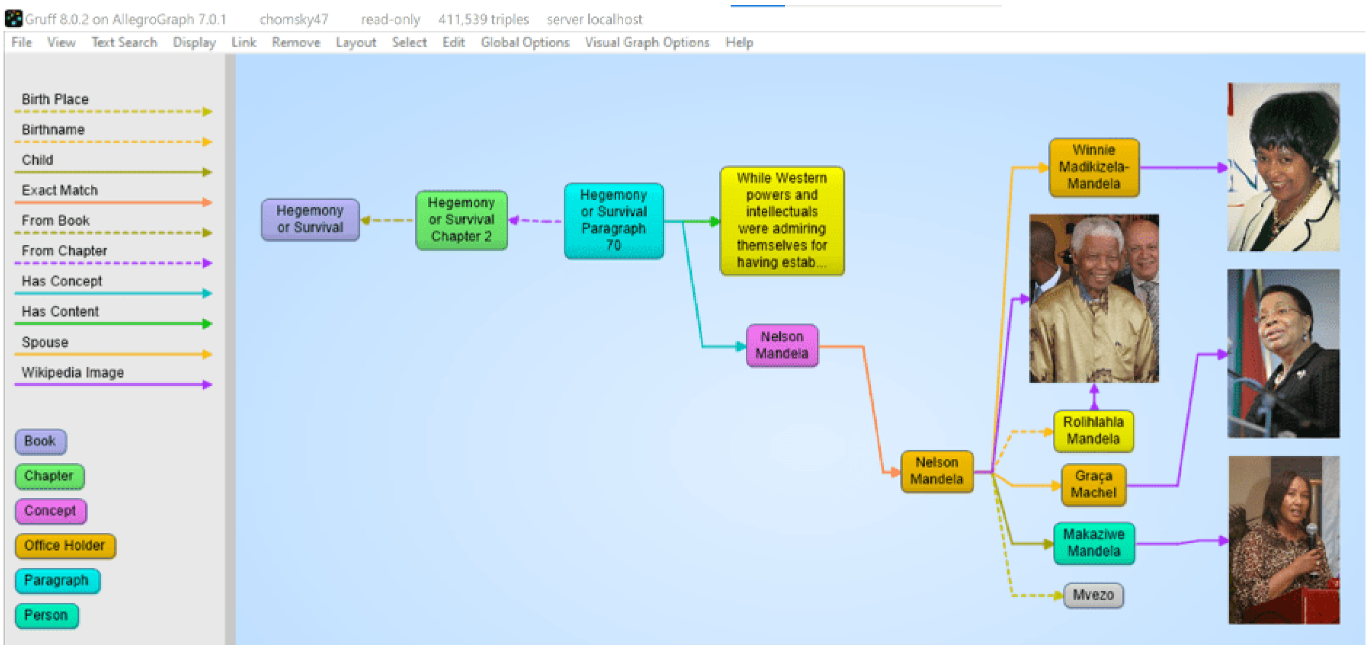

Gruff 8.0.2 on AllegroGraph 7.0.1 chomsky47 read-only 411,539 triples server localhost
File View Text Search Display Link Remove Layout Select Edit Global Options Visual Graph Options Help
Has Chapter
Has Concept
Has Paragraph
Has Person
Book
Chapter
Concept
Paragraph
Person
Media Control
EARLY HISTORY OF PROPAGANDA
PUBLIC RELATIONS
SELECTIVE PERCEPTION
Media Control Paragraph 13
Media Control Paragraph 16
Media Control Paragraph 10
South Africa
Health Care
Business
Political Parties
Edward Bernays


Gruff 8.0.2 on AllegroGraph 7.0.1 chomsky47 read-only 411,539 triples server localhost
File View Text Search Display Link Remove Edit Global Options Outline Options Help

```
▼ Second Indochina War
  ▷ Alt Label   Second Indochina War
  ▷ Alt Label   Vietnam War
  ▶ Broader   Cold War Proxy Wars
    ▶ Broader   Proxy war
      ▶ Broader   War
        ▷ Alt Label   Armed conflict      click here to see the other 27
        ▶ is Broader of   Civil war
        ▷ is Broader of   Class War
        ▶ is Broader of   Cold war
        ▶ is Broader of   Forever War
        ▶ is Broader of   Invasions
        ▷ is Broader of   Nuclear War
        ▶ is Broader of   Occupation
          ▷ Alt Label   Military Occupation
        ▶ is Broader of   Revolutions
          ▷ Alt Label   revolution
          ▷ is Broader of   American Revolution
        ▶ is Broader of   World war
          ▶ is Broader of   World War I
          ▶ is Broader of   World War II
      ▶ is Broader of   Modern and Ongoing Proxy Wars
        ▷ is Broader of   Abkhaz–Georgian conflict      click here to see the other 29
      ▶ is Broader of   US Backed Proxy Wars
        ▶ is Broader of   1958 Lebanon crisis      click here to see the other 70
          ▶ Broader   Cold War Proxy Wars
    ▶ is Broader of   1958 Lebanon crisis      click here to see the other 70
      ▶ Broader   US Backed Proxy Wars
  ▶ Broader   US Backed Proxy Wars
  ▷ Pref Label   Second Indochina War
  ▷ Pref Label   Vietnam War
```

[6] Linked Data Enrichment

In many use cases, AllegroGraph can link extracted entities to concepts in the linked data cloud. The most prominent being DBpedia, wikidata, the census database, GeoNames, but also many Linked Open Data repositories. One tool that is very useful for this is IBM's Natural Language Understanding program but there are others available. In the following image we see that the Nelson Mandela entity (Red) is linked to the dbpedia entity for Nelson Mandela and that then links to the DBpedia itself. We extracted some of his spouses and a child with their pictures.



[7] Complex Relationship and Event Extraction

Entity extraction is a first good step to 'see' what is in your documents but it is just the first step. For example: how do you find in a text whether company C1 merged with company C2. There are many different ways to express the fact that a company fired a CEO. For example: Uber got rid of Kalanick, Uber and Kalanick parted ways, the board of Uber kicked out the CEO, etc. We need to write explicit symbolic rules for this or we need a lot of training data to feed a machine learning algorithm.

[8] NLP and Machine Learning

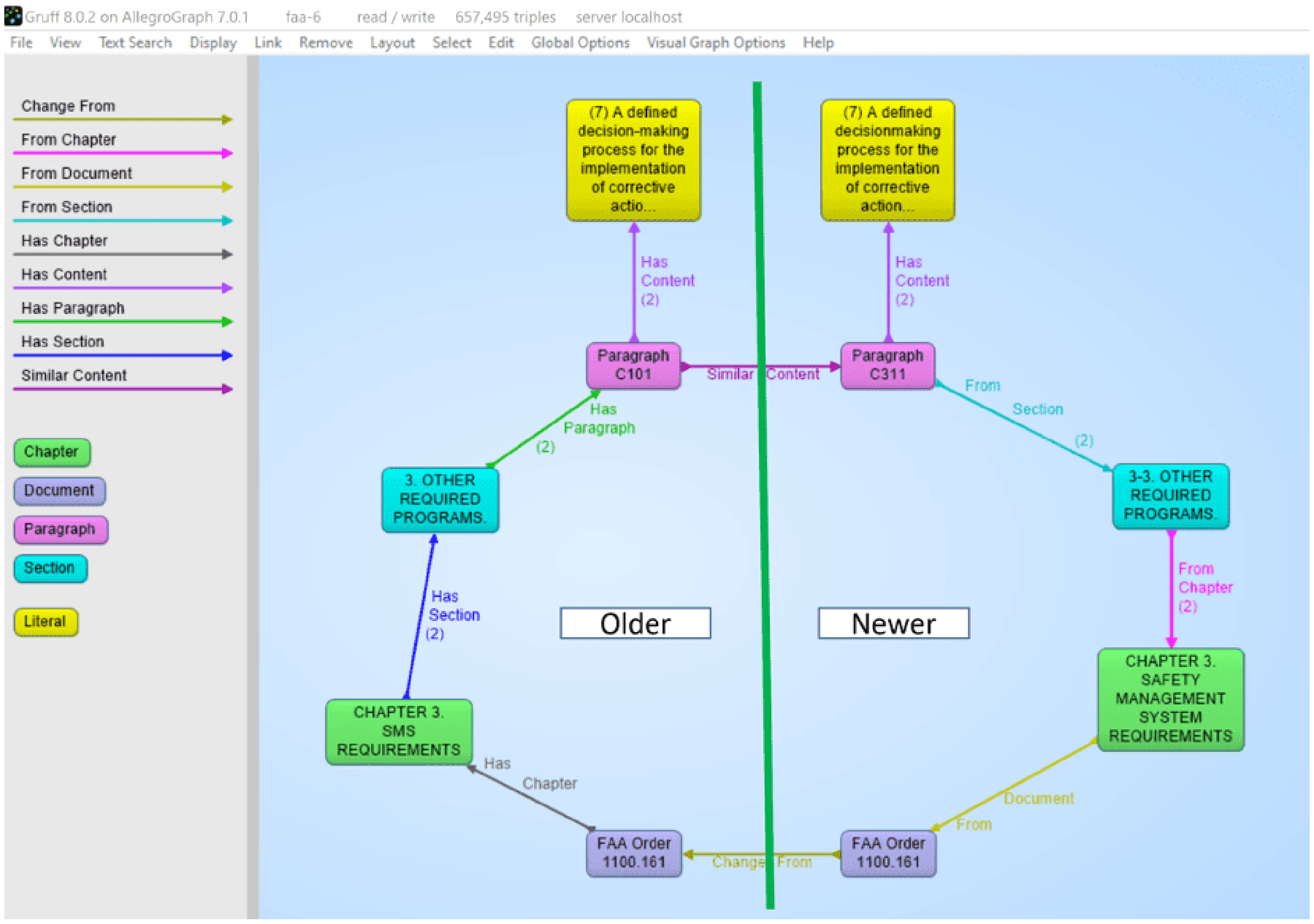
There are many many AI algorithms that can be applied in Document Knowledge Graphs. We provide best practices for topics like:

[a] Sentiment Analysis, using good/bad word lists or training data.
[b] Paragraph or Chapter similarity using statistical techniques like Gensim similarity or symbolic techniques where we just the overlap of recognized entities as a function of the size of a text.
[c] Query answering using word2vec or more advanced techniques like BERT
[d] Semantic search using the hierarchy in SKOS taxonomies.
[e] Summarization techniques for Abstractive or Extractive abstracts using Gensim or Spacy.

[9] Versioning and Document tracking

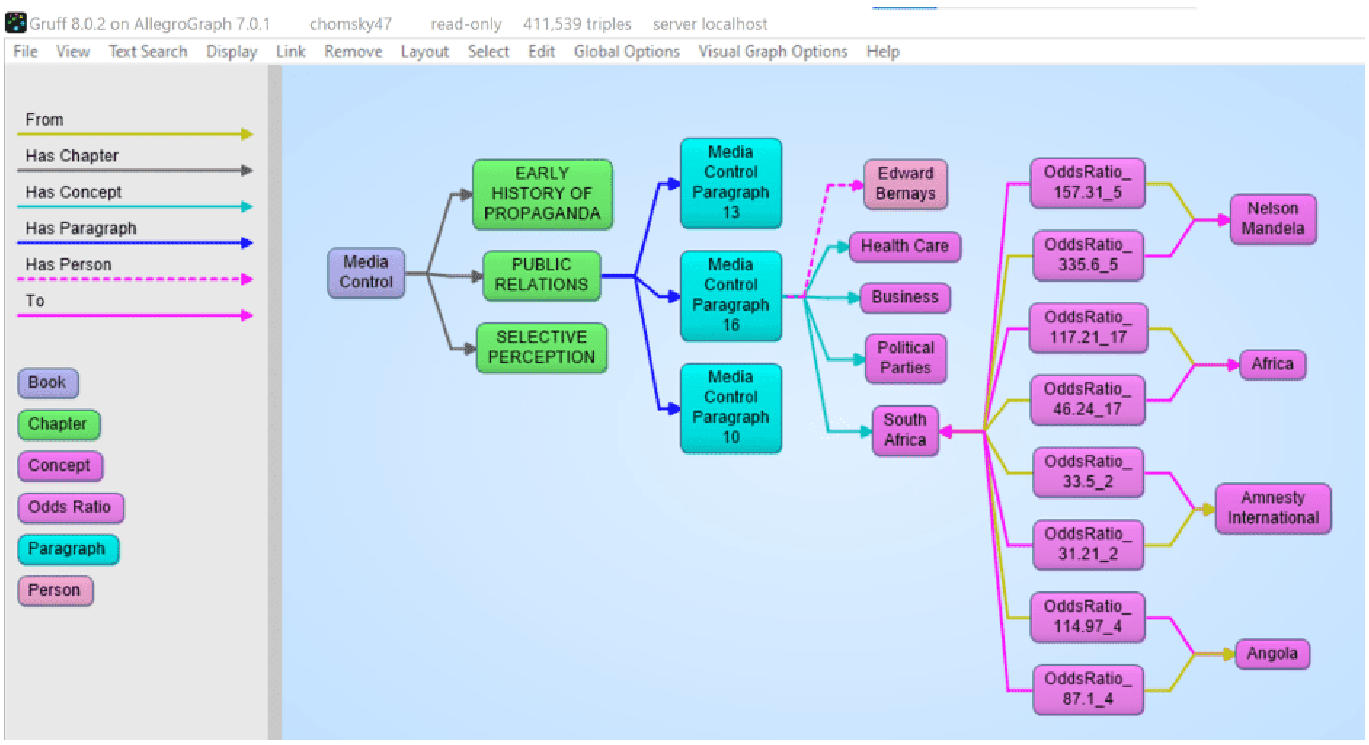
Several of our customers with Document Knowledge Graphs have noted the one constant in all of these KGs is that documents change over time. As part of our solution, we have created best practices where we deal with these changes. A crucial first step is to put each document in its own graph (i.e. the fourth element of every triple in the document tree is the document id itself). When we get a new version of a document the document ID changes but the new document will point back to the old version. We then compute which paragraphs stayed the same within a certain margin (there are always changes in whitespace) and we materialize what paragraphs disappeared in the new version and what new paragraphs appeared compared to the previous version. Part of the best practice is to put the old version of a document in a historical database that at all times can be federated with the ‘current’ set of documents.

Note that in the following picture we see the progression of a document. On the right hand side we have a newer version of a document 1100.161 with a chapter -> section -> paragraph -> contents where the content is almost the same as the one in the older version. But note that the newer one spells ‘decision making’ as one word whereas the older version said ‘decision-making’. Note that also the chapter titles and the section titles are almost the same but not entirely. Also, note that the new version has a back-pointer (changed-from) to the older version.


Gruff 8.0.2 on AllegroGraph 7.0.1 faa-6 read / write 657,495 triples server localhost
File View Text Search Display Link Remove Layout Select Edit Global Options Visual Graph Options Help
Change From
From Chapter
From Document
From Section
Has Chapter
Has Content
Has Paragraph
Has Section
Similar Content
Chapter
Document
Paragraph
Section
Literal
(7) A defined decision-making process for the implementation of corrective actio...
(7) A defined decisionmaking process for the implementation of corrective action...
Has Content (2)
Has Content (2)
Paragraph C101
Similar Content
Paragraph C311
Has Paragraph (2)
From Section (2)
3. OTHER REQUIRED PROGRAMS.
3-3. OTHER REQUIRED PROGRAMS.
From Chapter (2)
Has Section (2)
Older
Newer
CHAPTER 3. SMS REQUIREMENTS
CHAPTER 3. SAFETY MANAGEMENT SYSTEM REQUIREMENTS
Has Chapter
From Document
FAA Order 1100.161
Change From
FAA Order 1100.161

[10] Statistical Relationships

One important analytic one can do on documents is to look at the co-occurrence of terms. Although, given that certain words might occur more frequently in text, we have to correct the co-occurrence between words for the frequency of the two terms in a co-occurrence to get a better idea of the ‘surprisingness’ of a co-occurrence. The platform offers several techniques in Python and Lisp to compute these co-occurrences. Note that in the following picture we computed the odds ratios between recognized entities and so we see in the following gruff picture that if Noam Chomsky talks about South Africa then the chances are very high he will also talk about Nelson Mandela.


Gruff 8.0.2 on AllegroGraph 7.0.1 chomsky47 read-only 411,539 triples server localhost
File View Text Search Display Link Remove Layout Select Edit Global Options Visual Graph Options Help
From
Has Chapter
Has Concept
Has Paragraph
Has Person
To
Book
Chapter
Concept
Odds Ratio
Paragraph
Person
Media Control
EARLY HISTORY OF PROPAGANDA
PUBLIC RELATIONS
SELECTIVE PERCEPTION
Media Control Paragraph 13
Media Control Paragraph 16
Media Control Paragraph 10
Edward Bernays
Health Care
Business
Political Parties
South Africa
OddsRatio_157.31_5
OddsRatio_335.6_5
OddsRatio_117.21_17
OddsRatio_46.24_17
OddsRatio_33.5_2
OddsRatio_31.21_2
OddsRatio_114.97_4
OddsRatio_87.1_4
Nelson Mandela
Africa
Amnesty International
Angola