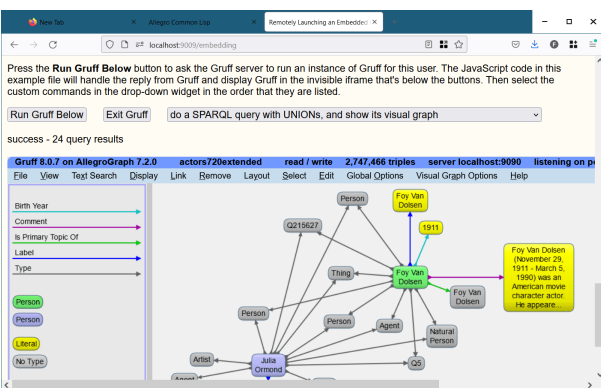


# Embedding Gruff In a Web Page

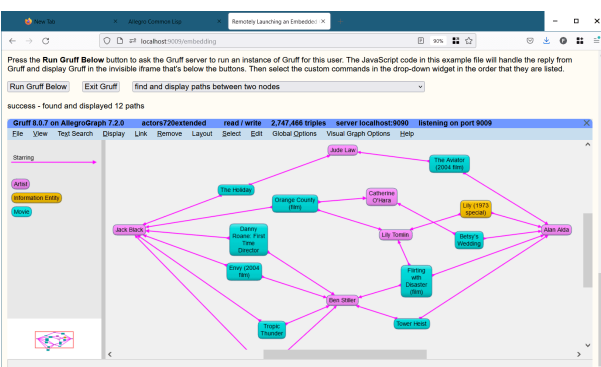
Gruff can be embedded in any web page to use Gruff inside your own web site or web application. The file `embedding.html` in the Gruff installation folder provides a complete example of this. Display that file in a web browser to see the complete instructions for setting everything up. That same file serves as an example web page, with buttons at the bottom for embedding Gruff into that web page after doing the setup.



A Gruff feature allows a single launcher instance of Gruff to be running as a server and listening for requests from web browsers. It will launch a separate instance of Gruff for each web page that requests one, up to a specified limit. It can

optionally use the launcher instance itself for one client, to minimize the number of Gruff executables that are running. Running Gruff as a server uses various command line options that are described under Running Gruff in a Web Browser.

Your web page needs to include an HTML iframe where Gruff will be placed, plus a link or button that asks a remote Gruff server to launch an instance of Gruff for the reader to use in that iframe. You will need to adapt the JavaScript code that's in `embedding.html` to make your link or button handle the reply from the Gruff server.



Simply embedding Gruff in an HTML iframe allows a reader to use Gruff by itself as usual inside your web page. A more advanced feature is that your web application can also send custom commands to Gruff. For

example, your application could derive a set of triples that it wants Gruff to display, and then send those triples to Gruff. The code in `embedding.html` also demonstrates this ability. The complete documentation for sending commands to Gruff is at [The HTTP Interface to Gruff](#).

---

## **KM Global Network Conference – Visualizing Knowledge (Recording)**

The Knowledge Management Global Network (KMGN) is a not for profit community founded in 2014. It is composed of a network of national communities for Knowledge Management practitioners. KMGN is the formalization of a relationship between KM partner association to share resources and work collaboratively.

**Dr. Jans Aasman presented – Visualizing Knowledge**

---

## **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')

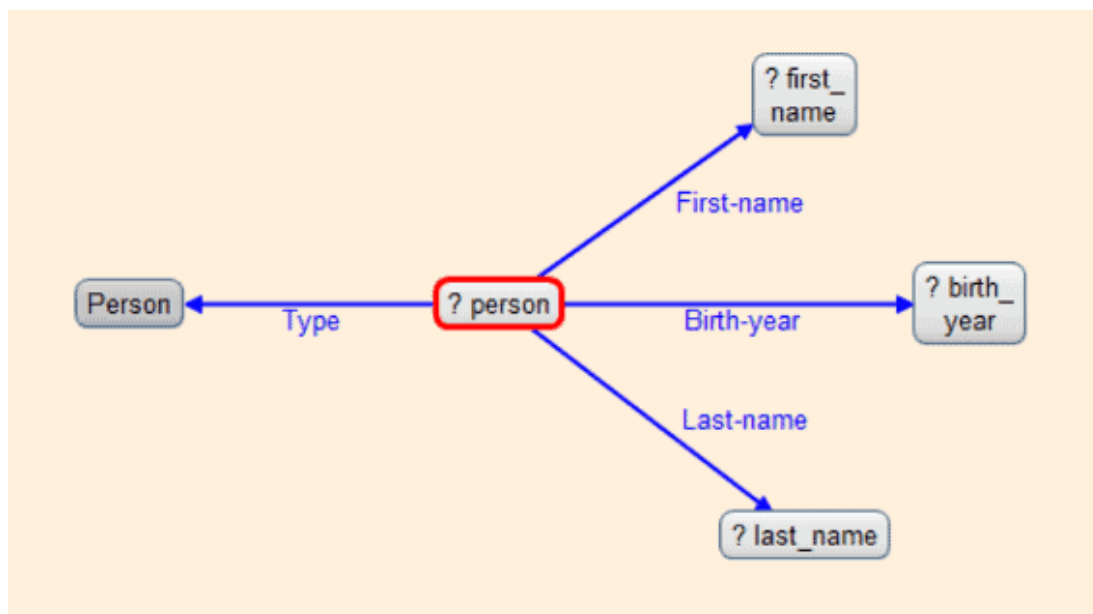
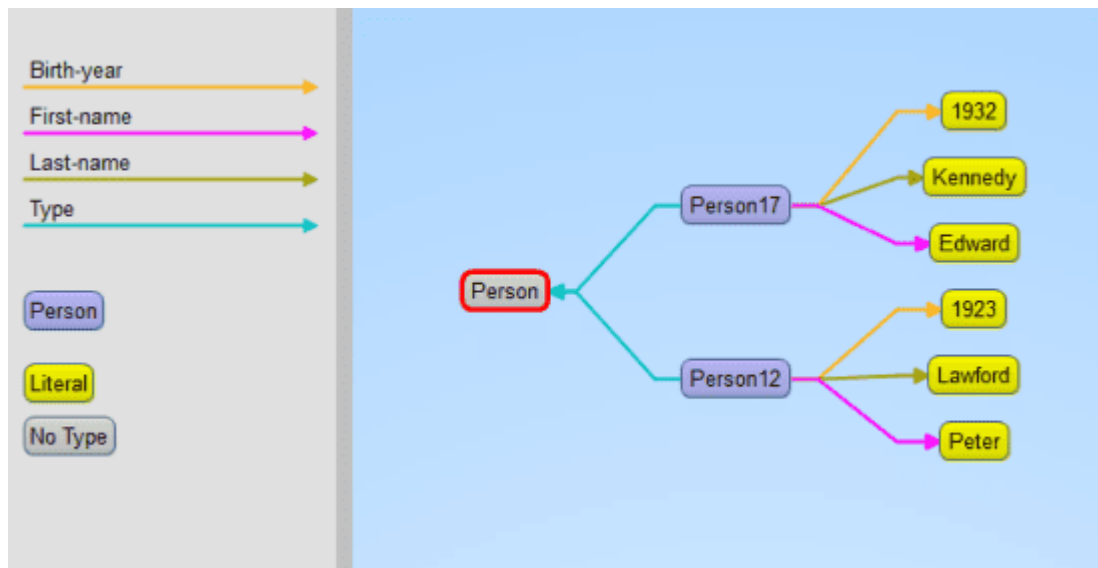
setup_env_var('AGRAPH_USER', '****', 'Username')

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:



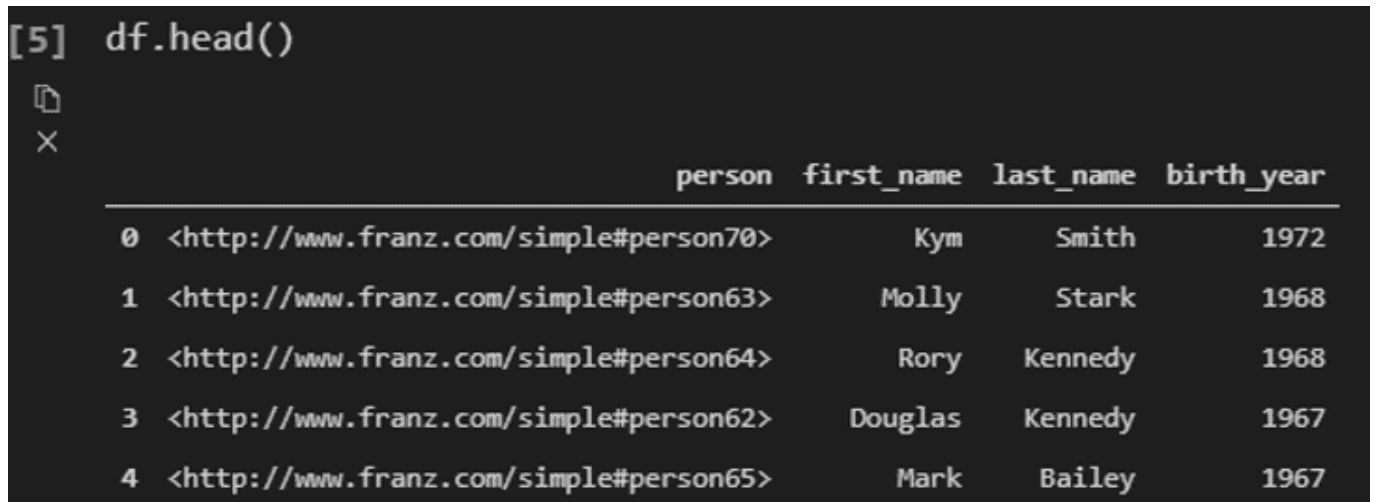
3. Then add the written query to the python script as a variable string (we added an additional line to the query to sort on birth year). Next use the API functionality to simply execute the query and turn the results into a pandas

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()
```

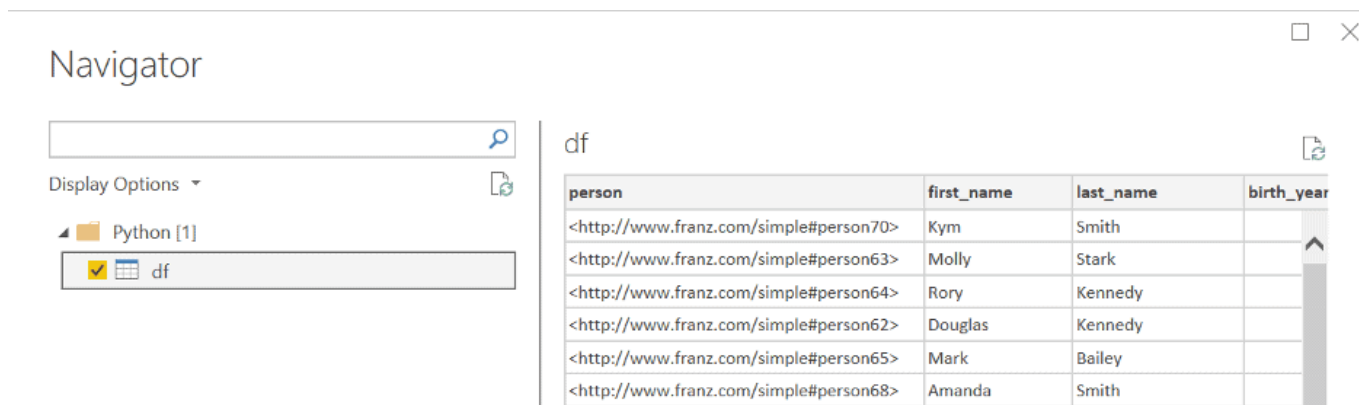
When looking at the result, we see that we have a DataFrame!



```
[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:



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  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?queryL  
n=SPARQL&limit=1000&infer=false&returnQueryMetadata=false&chec  
kVariables=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:





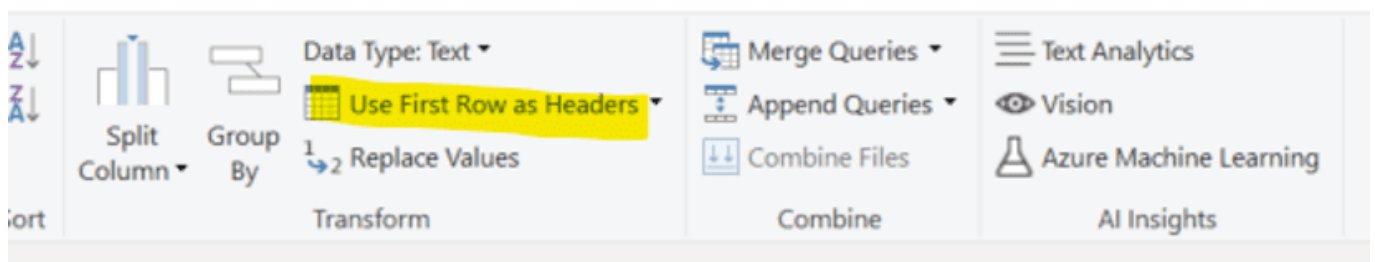
## Query1

```
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:

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](mailto:support@franz.com)

---

# Advanced Knowledge Graph Visualization with New Gruff v8

***High Performance Data Visualizations Accelerate Graph Search and Query Building – Driving Data Discoveries for Banks, Healthcare Providers and Enterprises Globally***

**OAKLAND, Calif., May 12, 2020** – Franz Inc., an early innovator in Artificial Intelligence (AI) and leading supplier of Semantic Graph Database technology for Knowledge Graph Solutions, today announced Gruff 8, a browser-based graph visualization software tool for exploring and discovering connections within enterprise Knowledge Graphs. Gruff 8, which has been integrated into AllegroGraph 7, enables users to visually build queries and visualize connections between data without writing code, which speeds discoveries and enhances the ability to uncover hidden connections within data.

“By augmenting Knowledge Graphs with visualizations, users can determine insights that would otherwise elude them,” said Jans Aasman, CEO of Franz Inc. “Gruff’s dynamic data visualizations increase users’ understanding of data by instantly illustrating relevant relationships, hidden patterns and

data's significance to outcomes. Gruff also helps make data actionable by displaying it in a way that decision-makers can see the significance of data relative to a business problem or solution."

"Few tools exist that can quickly turn arbitrary RDF graph pattern matches into clear visualizable results," said Michael Pool, Global Head of Semantic Modeling and Engineering, Senior Director at BNY Mellon Bank. "Gruff is invaluable in turning our knowledge graph data into useful and actionable analytic insights."

Gruff enables users to create visual Knowledge Graphs that display data relationships in views that are driven by the user. Ad hoc and exploratory analysis can be performed by simply clicking on different graph nodes to answer questions. Gruff's unique 'Time Machine' feature provides the capability to explore temporal context and connections within data. The visual query builder within Gruff empowers both novice and expert users to create simple to highly complex queries without writing any code.

**Browser-based Graph Visualization** – Gruff 8 is a browser-based application that does not require an additional download or application installation once AllegroGraph is installed. All AllegroGraph users need is a web browser and internet connection to login. This approach gives users the convenience to access Gruff from anywhere on any type of system, while also simplifying deployment and streamlining updates within enterprise environments.

Louis Rumanes at UnitedHealth Group Research and Development recognizes the value of using Gruff as a browser-based app and commented, "Nice job on Gruff in a browser and I think this will be a gamechanger."

**Accelerated Visual Graph Rendering** – Visual renderings within Gruff are now up to 3X faster. Users can dynamically lay out

cyclical graphs, display tables of properties and build SPARQL or Prolog queries as visual diagrams.

**Dynamic Graph Visualizations within AllegroGraph** – Gruff is fully integrated with AllegroGraph 7, Franz's leading semantic knowledge graph solution, which seamlessly leverages Gruff's advanced graph visualizations and graphical query builder to reveal hidden connections in knowledge graph data. AllegroGraph 7, with FedShard™, is a breakthrough Knowledge Graph solution that allows infinite data integration through a patented approach that unifies all data and knowledge base silos into an Entity-Event Knowledge Graph solution that can support massive big data analytics. AllegroGraph 7 utilizes unique federated sharding capabilities that drive 360-degree insights and enable complex reasoning across distributed Knowledge Graphs.

To support ubiquitous AI, a Knowledge Graph system needs to fuse and integrate data, not just in representation, but in context (ontologies, metadata, domain knowledge, terminology systems), and time (temporal relationships between components of data). The rich functional and contextual integration of multi-modal, predictive modeling, artificial intelligence suitable for large scale analytics is what distinguishes AllegroGraph 7 as a modern, scalable enterprise analytic platform.

AllegroGraph 7 is the first big temporal Knowledge Graph technology that encapsulates a novel entity-event model natively integrated with domain ontologies and metadata with dynamic ways of setting the analytics lens on all entities in the system (patient, person, devices, transactions, events, and operations) as prime objects that can be the focus of an analytic (AI, ML, DL) process.

"AllegroGraph 7's support of Entity-Event Data Modeling is the most welcome innovation and addition to our arsenal in reimagining healthcare and implementing Precision Medicine,"

said Dr. Parsa Mirhaji, Director of Center for Health Data Innovations at the Albert Einstein College of Medicine and Montefiore Health System, NY. “Precision Medicine is about moving away from statistical averages and broad-based patterns. It is about connecting many dots, from different contexts and throughout time, to support precision diagnosis and to recommend the precision care that can take into account all the subtle differences and nuisances of individuals and their personal experiences throughout their life. This technology is about saving lives, by leveraging data, context and analytics and is what Franz’s Entity-Event Data Modeling brings to the table.”

### **Gruff 8 Availability and Pricing**

Gruff 8 is immediately available as a free download from AllegroGraph.com and is integrated as part of AllegroGraph’s cloud offering on the Amazon Marketplace.

### **Gruff Webinar**

Join Franz’s webcast discussing Gruff 8 entitled “Visualizing and Exploring Knowledge Graphs with the New Browser based Gruff” – by registering for the May 14<sup>th</sup> Webinar.

### **About Franz Inc.**

Franz Inc. is an early innovator in Artificial Intelligence (AI) and leading supplier of Semantic Graph Database technology with expert knowledge in developing and deploying Knowledge Graph solutions. The foundation for Knowledge Graphs and AI lies in the facets of semantic technology provided by AllegroGraph and Allegro CL. AllegroGraph is a database technology that enables businesses to extract sophisticated decision insights and predictive analytics from highly complex, distributed data that cannot be uncovered with conventional databases. Unlike traditional relational databases or other NoSQL databases, AllegroGraph employs semantic graph technologies that process data with contextual

and conceptual intelligence. AllegroGraph is able run queries of unprecedented complexity to support predictive analytics that help organizations make more informed, real-time decisions. AllegroGraph is utilized by dozens of the top F500 companies worldwide. To learn more about Franz and AllegroGraph, go to [www.franz.com](http://www.franz.com).

---

## **Document Knowledge Graphs with NLP and ML**

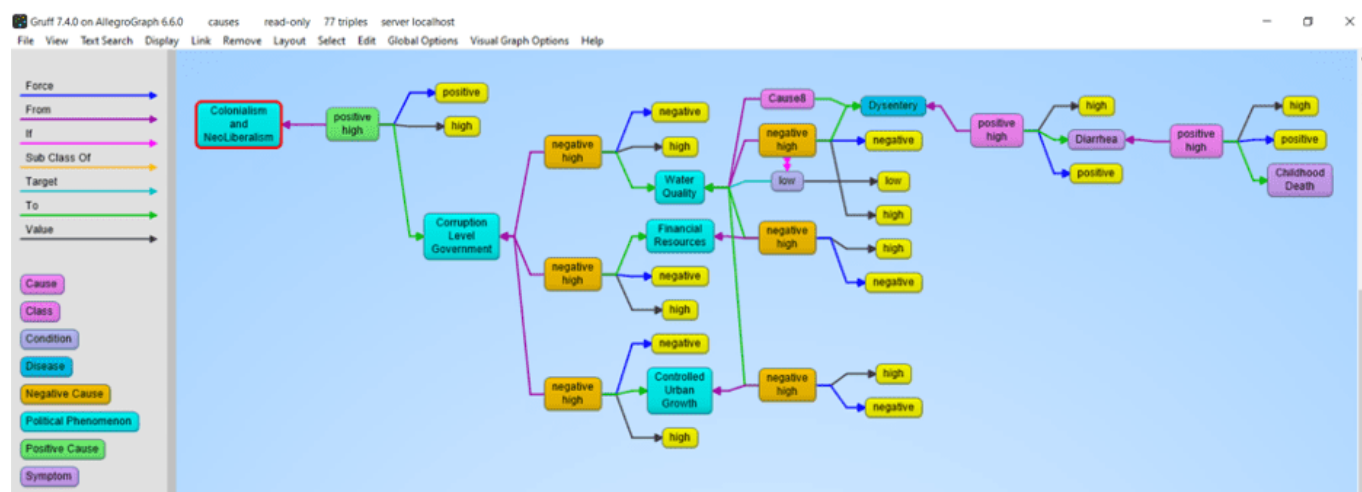
A core competency for Franz Inc is turning text and documents into Knowledge Graphs (KG) using Natural Language Processing (NLP) and Machine Learning (ML) techniques in combination with AllegroGraph. In this document we discuss how the techniques described in [NLP and ML components of AllegroGraph] can 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 Article <https://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**

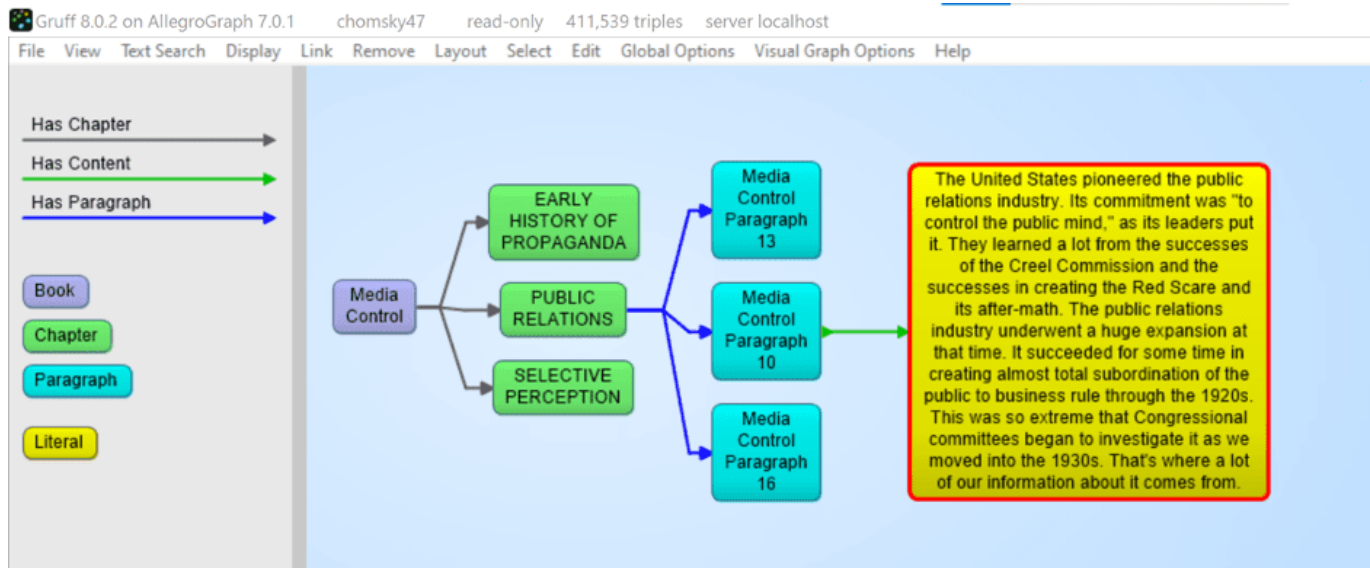
We then take a document and turn it into an intermediate XML using Apache Tika. Apache Tika supports more than 1000 document types and although Apache Tika is a fantastic tool, the output is still usually not clean enough to create a graph from, so we use Spacy rules to clean up the XML to make it as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

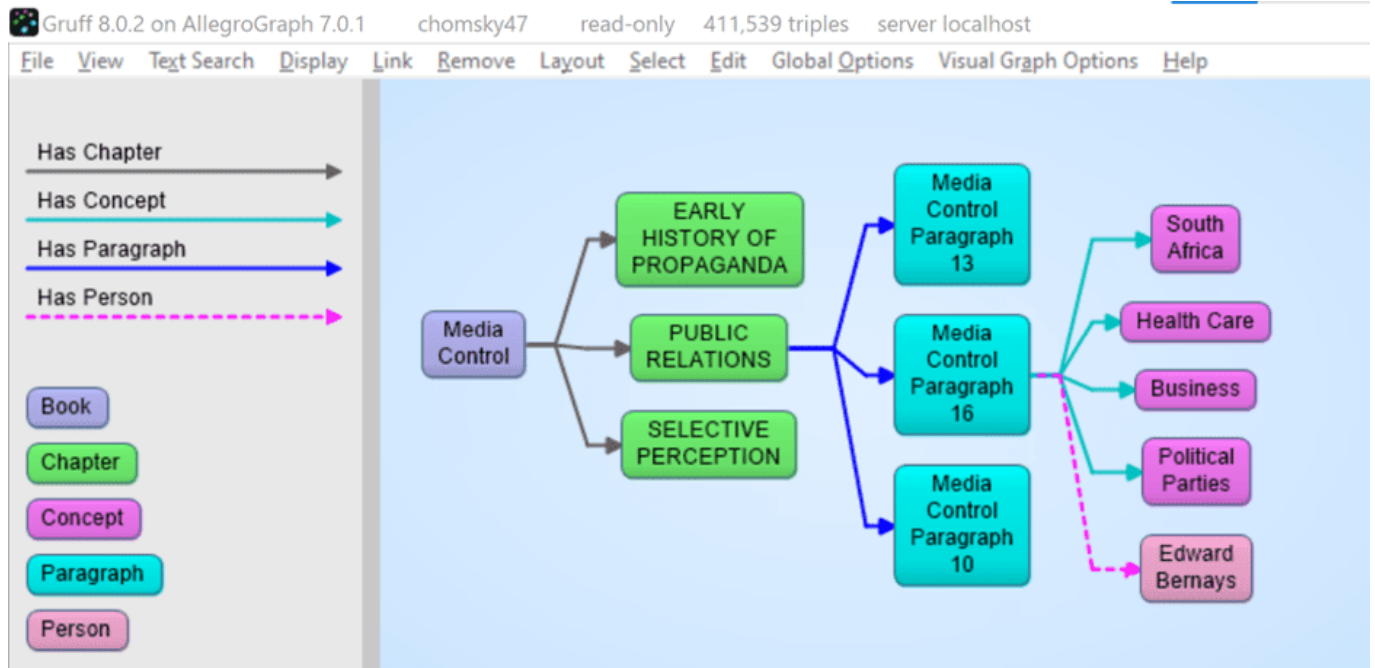
## **[4] XML to Triples**

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.



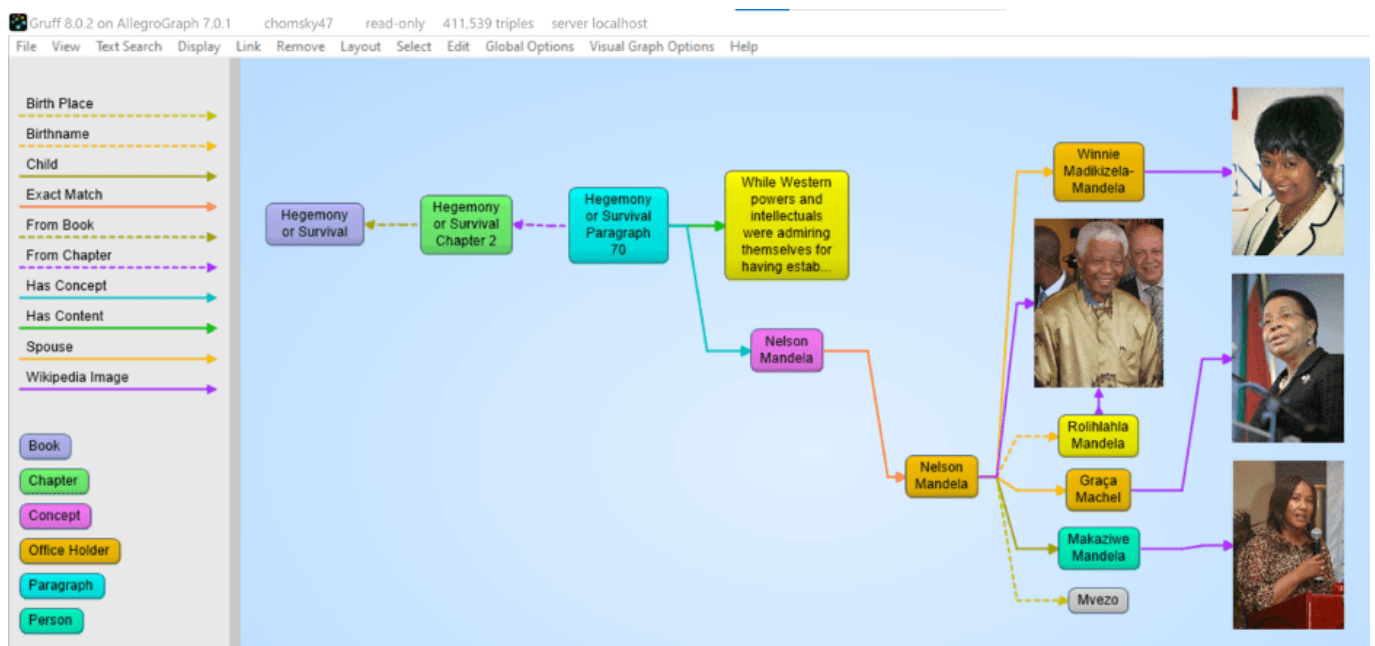
## [5] Entity Extractor

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.



## [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.

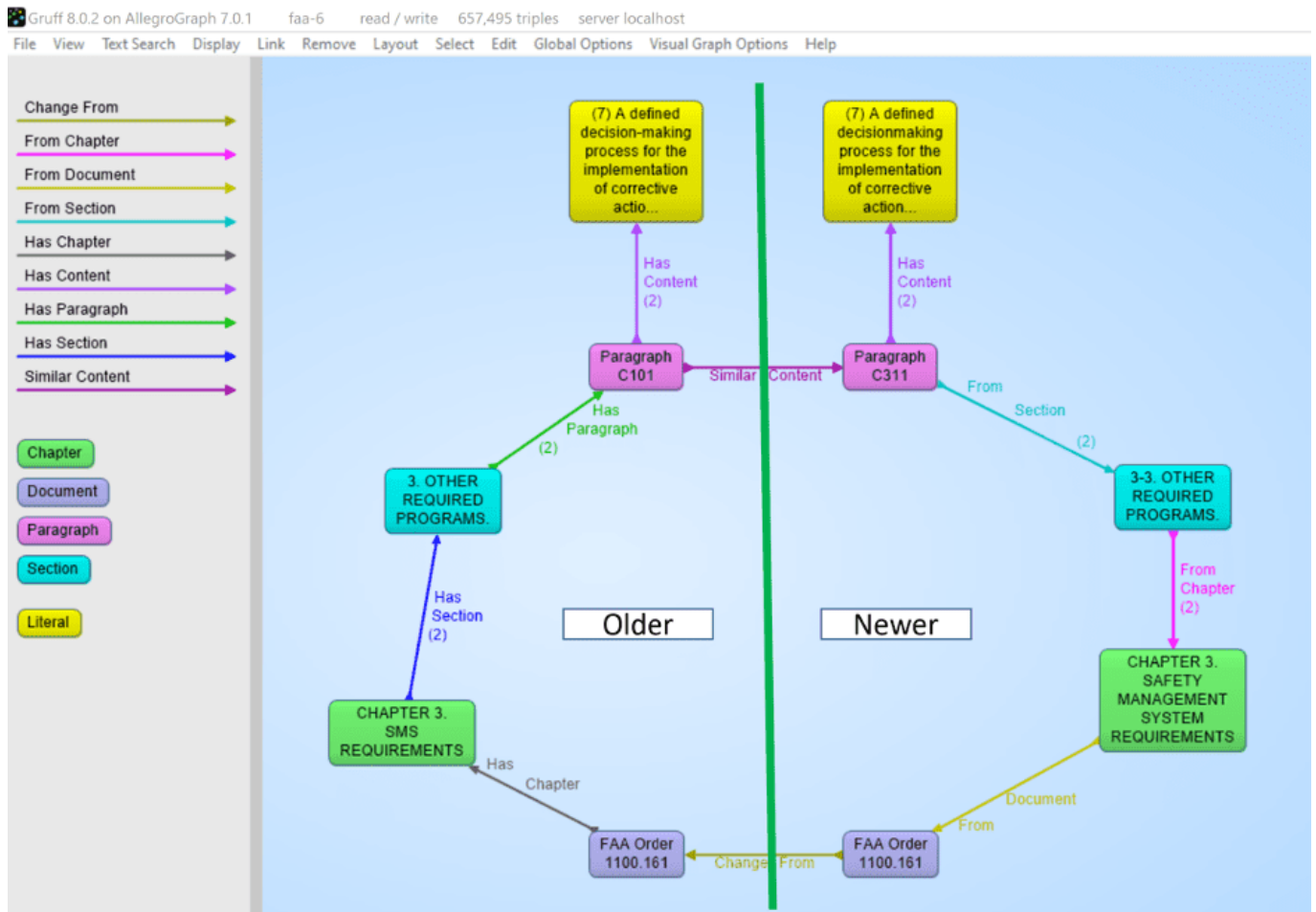
## **[9] Versioning and Document tracking**

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.

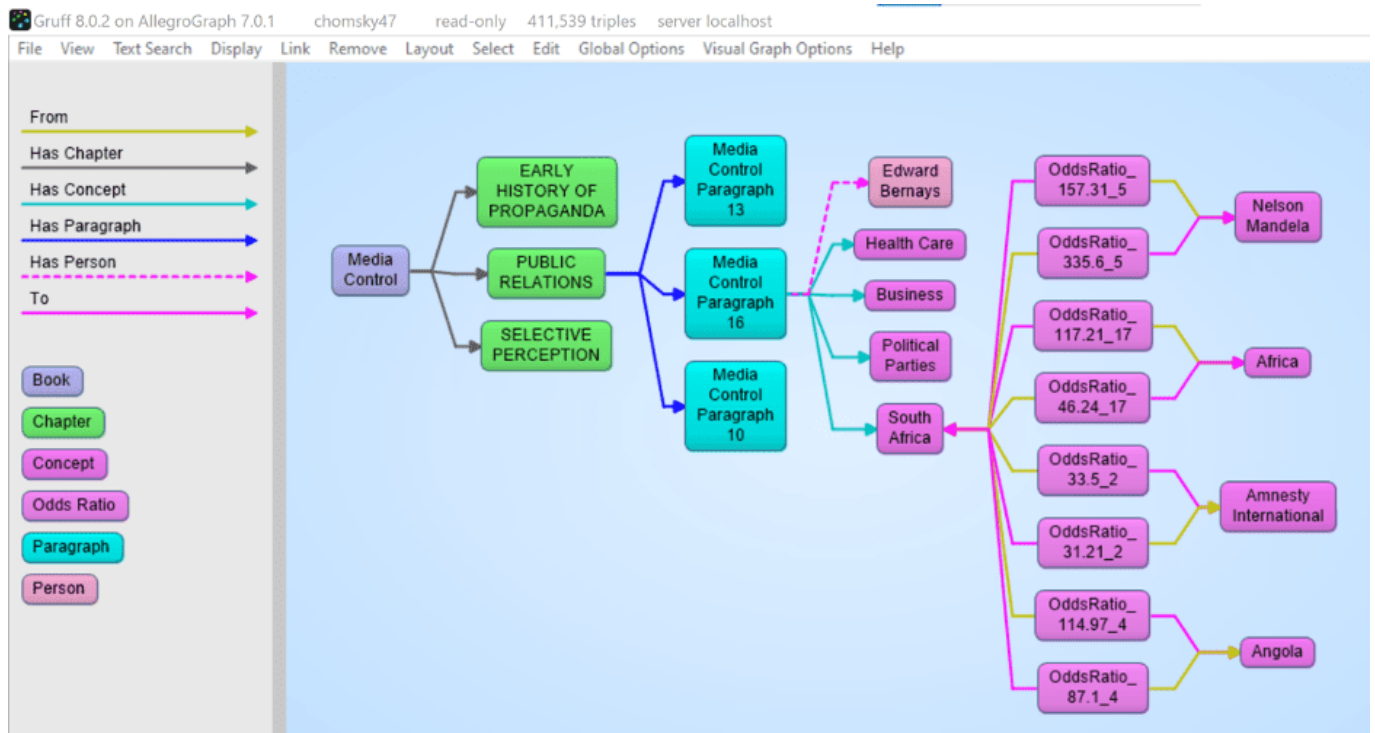


## [10] Statistical Relationships

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.



# Graphorum – Dr. Aasman Presenting

**Graph-Driven Event Processing for Intelligent Customer Operations**

Wednesday, October 16, 2019

10:15 AM – 11:15 AM

Level: Case Study



In the typical organization, the contents of the actual chat or voice conversation between agent and customer is a black hole. In the modern Intelligent Customer Operations center, the interactions between agent and customer are a source of rich information that helps agents to improve the quality

of the interaction in real time, creates more sales, and provides far better analytics for management. The Intelligent Customer Operations center is enabled by a taxonomy of the products and services sold, speech recognition to turn conversations into text, a taxonomy-driven entity extractor to take the important concepts out of conversations, and machine learning to classify chats in various ways. All of this is stored in a real-time Knowledge Graph that also knows (and stores) everything about customers and agents and provides the raw data for machine learning to improve the agent/customer interaction.

In this presentation, we describe a real-world Intelligent Customer Organization that uses graph-based technology for taxonomy-driven entity extraction, speech recognition, machine learning, and predictive analytics to improve quality of conversations, increase sales, and improve business visibility.

<https://graphorum2019.dataversity.net/sessionPop.cfm?confid=132&proposalid=11010>

---

# Ontology Summit 2020 – Knowledge Graphs

The Ontology Summit is an annual series of events that involves the ontology community and communities related to each year's theme chosen for the summit. The Ontology Summit was started by Ontolog and NIST, and the program has been co-organized by Ontolog, NIST, NCOR, NCB0, IA0A, NCO\_NITRD along with the co-sponsorship of other organizations that are supportive of the Summit goals and objectives.

Knowledge graphs, closely related to ontologies and semantic networks, have emerged in the last few years to be an important semantic technology and research area. As structured representations of semantic knowledge that are stored in a graph, KGs are lightweight versions of semantic networks that scale to massive datasets such as the entire World Wide Web. Industry has devoted a great deal of effort to the development of knowledge graphs, and they are now critical to the functions of intelligent virtual assistants such as Siri and Alexa. Some of the research communities where KGs are relevant are Ontologies, Big Data, Linked Data, Open Knowledge Network, Artificial Intelligence, Deep Learning, and many others.

Dr. Jans Aasman presented – ***“Why Knowledge Graphs Hit the Hype Cycle and What they have in common”***

[Presentation Page](#)

[Presentation Slides](#)



---

# Gruff Time Machine Tutorial



Here is an example for trying out the new time slider in Gruff's graph view. It uses triples from crunchbase.com that contain a history of corporate acquisitions and funding events over several years. Gruff's time bar allows you to examine those events chronologically, and also to display only the nodes that have events within a specified date range.

- Download the Crunchbase triples from the bottom of the Gruff download page at <https://allegrograph.com/products/gruff/>
- Create a new triple-store and used "File | Load Triples | Load N-Triples" to load that triples file into the new triple-store. Use "File | Commit" to ensure that the loaded triples get saved.
- Select "Visual Graph Options | Time Bar | Momentary Time Predicates" and paste the following five predicate IRIs into the dialog that appears. The time bar will then work with the date properties that are provided by these predicates, whenever you are browsing this

particular triple-store.

[http://www.franz.com/hasfunded\\_at](http://www.franz.com/hasfunded_at)

[http://www.franz.com/hasfirst\\_funding\\_at](http://www.franz.com/hasfirst_funding_at)

[http://www.franz.com/hasfounded\\_at](http://www.franz.com/hasfounded_at)

[http://www.franz.com/haslast\\_funding\\_at](http://www.franz.com/haslast_funding_at)

[http://www.franz.com/hasacquired\\_at](http://www.franz.com/hasacquired_at)

- Select “View | Optional Graph View Panes | Show Time Bar” to reveal the time bar at the bottom of the graph view. The keyboard shortcut for this command is Shift+A to allow quickly toggling the time bar on and off.
- Select “Display | Display Some Sample Triples” to do just that. The time bar will now display a vertical line for each of the requested date properties of the displayed nodes. Moving the mouse cursor over these “date property markers” will display more information about those events.
- Click down on the yellow-orange rectangle at the right end of the time bar and drag it to the left. This will make the “time filter range” smaller, and nodes that have date properties that are no longer in this range will temporarily disappear from the display. They will reappear if you drag the slider back to the

right or  
toggle the time bar back off.

For more information, the full time bar introduction is in the Gruff documentation under the command “View | Optional Graph View Panes | Show Time Bar”.

Check out the “Chart Widget” for showing date properties of the visible nodes.



---

# New Gruff v7.4 – Now Available!

## DOWNLOAD – Gruff

Gruff is the Knowledge Graph industry’s leading Graph Visualization software for exploring and discovering connections within data. Gruff provides novice users and graph experts the ability to visually build queries and explore connections as they developed over time.

Gruff produces dynamic data visualizations that organize connections between data in views that are driven by the user. This visual flexibility can instantly unveil new discoveries and knowledge that turn complex data into actionable business insights. Gruff was developed by Franz to address Graph Search in large data sets and empower users to intelligently explore graphs in multiple views including:

- **Graphical View with “Time Machine” feature** – See the

shape and density of graph data evolve over time

- **Tabular view** – Understand objects as a whole
- **Outline view** – Explore the often hierarchical nature of graphs
- **Query view** – Write Prolog or SPARQL queries
- **Graphical Query Builder** – Create queries visually via drag and drop

Gruff's 'Time Machine' feature provides users an important capability to explore temporal connections in your data. Users can see how relationships are created over time and are able to replay the evolving graph for new temporal based insights.



**Key New Features and Updates in Gruff v7.4** – To see the full list – Release Notes.

- The new command "File | Connect to Gruff Demo Server" lets you try out Gruff on the "extended actors" database at a public AllegroGraph server that's provided by Franz, when you don't have an AllegroGraph server yourself. See the Example button in the query view and in the graphical query view for a few example queries. "Help | Animated Demo" also works there.
- The graphical query view has new grouper boxes for graph group graph patterns, either for a particular graph or for a graph variable.
- The graphical query view now has node filters for the SPARQL operators IN and NOT IN (for limiting a node variable to a particular set of values), for langMatches (for selecting only literals of a particular language), and for CONTAINS, STRSTARTS, and STRENDS (for finding literals that contain specified text). Also, the "bound" and "not bound" filters were broken, and the LIMIT and

OFFSET values will now be included when saving a graphical query.

- Gruff can now connect to AllegroGraph servers through an HTTP proxy (as was possible with SPARQL endpoints already). See Global Options | Communications | HTTP Proxy.
- Additional triple file formats can now be loaded with the new commands “File | Load Triples | Load JSON-LD”, “Load TriG”, and “Load N-Quads Extended”. Corresponding new commands are also on the “File | Export Displayed Data As” child menu. Also, the new command “Global Options | Miscellaneous | Commit Frequency When Loading Triples” lets you control whether and how often commits will happen during loading.
- The query view’s “Create Visual Graph” button will now create link lines for additional SPARQL property path operators, namely InversePath ( ^ ) and AlternativePath ( | ). And it will draw the correct character for ZeroOrOnePath ( ? ). (See “Query Options | Show Links for Property Paths in Visual Graphs” for turning this off.)
- If the triple store defines label properties for predicates, then Gruff will now display those labels for the predicate objects as it has always done for nodes, as long as “Global Options | Node Label Predicates | Use Label Predicates for Node Labels” is on.
- When “Visual Graph Options | Node Labels | Show Full URIs on Nodes” is on, full URIs will be also displayed for the predicates in link labels. And full URIs will be shown in the legend as well.

Gruff Documentation





---

# **Webcast – Speech Recognition, Knowledge Graphs, and AI for Intelligent Customer Operations – April 3, 2019**

**Presenters – Burt Smith, N3 Results and Jans Aasman, Franz Inc.**

In the typical sales organization the contents of the actual chat or voice conversation between agent and customer is a black hole. In the modern Intelligent Customer Operations center (e.g. N3 Results – [www.n3results.com](http://www.n3results.com)) the interactions between agent and customer are a source of rich information that helps agents to improve the quality of the interaction in real time, creates more sales, and provides far better analytics for management.

Join us for this Webinar where we describe a real world Intelligent Customer Operations center that uses graph based technology for taxonomy driven entity extraction, speech recognition, machine learning and predictive analytics to improve quality of conversations, increase sales and improve business visibility.

View the recorded webinar.