Predicting Future Scientific Research with Knowledge Graphs

Matheus Schmitz[1] – mschmitz@usc.edu
Rehan Ahmed[1] – rehanahm@usc.edu
[1]University of Southern California

1 Introduction

The rise of Data Science as an academic discipline hails at least in part from its efficiency as a leverage tool, able to power-up many pre-existing data-dependent approaches to problem solving, knowledge discovery & comprehension, and predictive systems, among others.

Looking to leverage this strength, this project was devised with the intent of combining data science techniques, primarily Knowledge Graphs and Artificial Intelligence, in the pursuit of uncovering trends and areas of interest for future scientific research.

Given the volume of academic papers published every year, a subject matter for the development of the predictive system had to be decided upon. The authors saw no more suitable choice than Artificial Intelligence itself.

2 Data Gathering

The data used throughout this project was obtained from four sources, all of which are well-regarded repositories of academic research, some wide-encompassing and others focused on topics pertinent to computer science. Those sources are: *Google Scholar*, *arXiv*, *Semantic Scholar*, and *dblp*.

Data gathering was accomplished with a combination of web scraping tools (*Scrapy*) and APIs (*Semantic Scholar, dblp*).

Multiple sources were necessary as each possesses complementary capabilities. *Google Scholar* allows for the mining of highly relevant papers, but not of their abstracts, while *arXiv* does not possess relevance but has abstracts and allows for crawling based on title, making it a perfect pairing for papers obtained from *Google Scholar*.

Semantic Scholar for its turn has an API that vastly simplifies data crawling in large volumes. Its API's json-formatted outputs enabled the creation of a "seed trail" crawler, which iteratively extracts papers based on the citations and references of previously crawled papers. This feature was critical for this project, as given the total size of the published data on the internet, if one were to randomly crawl papers, the resulting graph would be very sparsely connected. The "seed trail" crawling approach creates a strongly connected graph, enabling many of the further steps of this project.

dblp offers a simple DOI lookup tool which simplified the task of gathering complimentary information for the papers crawled from *Semantic Scholar*.

3 Entity Resolution

Given the various sources used, entity resolution was performed in three steps: Firstly, *Google Scholar* and *arXiv* were merged using entity linkage functions that considered title, author, and date. Second, *Semantic Scholar* and *dblp* were merged using DOI. Third, the two previous merges were combined into one, based on title, author and DOI. Performing blocking by author achieved a reduction ratio of 0.93, shrinking a total of 43 million pairs into 3 million pairs for comparison. Entity resolution resulted in 20534 unique papers.

Using a validation set the authors found the True Positive Rate (TPR) of the entity linkage to be 99.53%.

4 External Knowledge Graph Linkage

Papers available on the aforementioned sources have no additional information about the authors other than their names. In order to complement that data, the crawled dataset was linked to *DBPedia* through programmatic queries to its API, which, thanks to an existing property *dbo:academicDiscipline*, enable a high precision (albeit not high recall, with only 298 matches) extraction of information about the papers' authors. This data was appended to the dataset.

5 Triple Generation & Ontology

Using python's *RDFlib* the dataset was converted into a set of RDF triples in the Turtle format. All entity types, predicates and properties are derived from *schema.org, xml, xds, foaf* and *DBpedia.*

The resulting ontology contains 4 entity types: Scholarly Article, Person, Genre, and Publisher.

Connecting those entities are 6 predicate (relationship) types: creditText, reference, citation, author, genre, and publisher.

The ontology uses a total of 12 property types: name, birthdate, DBPedia URL, PageRank, headline, abstract, URL, DOI, influentialCitationCount, citationVelocity, dateModified, and datePublished.

The main entity in the resulting RDF ontology is ScholarlyArticle, which contains the 20.5k academic articles crawled from the web with their full data, and another 435k articles which were mentioned as references but not crawled. An URI was also created for each of the references in each of the crawled papers to enable easy extensions of the graph.

Simple statistics about the triple generation process are as follows: 453k paper URIs (including references), 51k author URIs, 207 genres, 86 publishers, 1.5M relationships, 505k nodes, 2.65M tiples.

The Turtle RDF was deployed to a Neo4J Graph Database and made available via an endpoint on a *Flask* user interface.

6 PageRank

Post deployment of the graph database, the PageRank algorithm was applied to it, allowing for the identification of the most relevant nodes in the graph. This addresses one of the identified challenges for this project which was providing novel functionality based on the gathered data, as a mere author/title query would not make the graph any more useful than the current relational databases from which the data was obtained.

7 Predicting Future Research

Combining a Knowledge Graph with Artificial Intelligence techniques to generate insights regarding potential new areas for scientific inquiry was the motivation for this project.

Having the graph data as a start point, an NLP pipeline was then built to generate novel research predictions. The model contains three main stages, the first consisting of priming the *GPT2* model, which means selecting a number of existing abstracts to serve as the references for a new abstract to be generated, then having the model parse through them.

The second step was text generation itself, in which the model wrote a long article (sizes vary, but an average length was 4000 words). This route was chosen as, were the research predictions to be generated directly from the source abstracts, they would be highly correlated, with the predictions being no more than a reshuffle and rewrite of the abstracts contained in its sources' papers, and thus useless at suggesting novel fields for exploration.

The third and final stage consisted of writing a brand-new abstract for the "paper" written during stage two. Here *HuggingFace's* summarization pipeline was employed, allowing the central topics of the large text to be condensed into a 500-word abstract whose manual evaluation showed it to be both novel and highly coherent in its grammar and structure.

The quality of the predictions was assessed by evaluating their correlation with their sources, whereby for the purposes and intents of this project a low correlation was targeted, as, to satisfy the project's goal of generating new and interesting insights for potential areas of research it is necessary for the predictions to be dissimilar from their sources, because were they to be similar they would then be useless in providing novel and unexpected ideas to be considered for research. The goal was not to predict what the abstract of a paper should be, but rather to generate yet-unseen concepts. For that metric it was found that only 7% of the papers had their sources among the top recommendations of similar papers – very good score.

8 Topic Modeling

A weakness identified during the triples generation step was the inconsistency in the topic structures among the sources, which totaled 207 distinct yet overlapping topics.

Aiming to overcome this issue, a new Topic Modeling architecture was developed for the gathered data. The approach chosen is based on Latent Dirichlet Allocation (LDA). Experimentation revealed 10 to be a good choice for the number of topics among the highly connected data that was crawled. This topic structure could then be applied to all the papers regardless of their origin, as the architecture was developed to find topic similarity based on the papers' abstracts, which was available for all papers.

The resulting topics allow for the clear identification of clusters of research, such as AI-Medicine, AI-Reinforcement Learning, AI-Transfer Learning, among others.

9 Recommender System

Furthering the goal of providing novel and differentiated utility based on Knowledge Graphs, the LDA-based topic model was leveraged to create a Recommender System, which allows the generation of paper recommendations based on latent similarity between the paper's abstracts and a user-provided textual query.

The backend technology for the Recommender System relies of Nearest-Neighbors Cosine Similarity over LDA vectorized strings. Graph nodes had their vectors precomputed to assure high performance on real-time incoming queries. Upon manual evaluation the system proved highly satisfactory.

10 User Interface

Aiming at providing users a single hub for accessing and exploring all the work hereto presented, a Front-End user interface was developed via python's *Flask API*.

The Front-End allows access to the Back-End Neo4J database through both Cypher queries as well as textual queries for author, title, DOI, and URIs existing in the graph.

Along with direct queries to the graph, the UI also allows queries to the Recommender System, which then returns the top 10 recommendations based on the query.

A secondary tab allows users to visually explore the LDA - derived topics, including their spatial distance in a condensed 2-dimensional space and the content of the words associated with each topic, whose LDA distribution can also be explored via an interactive TF-IDF weight manipulator.

PageRank was precomputed for all nodes in the graph and comes available with any query returns for papers.

The *DBPedia* linkage data was also extracted and appended to the graph. It comes available with any query returns for authors.

The future research predictions mentioned in item 7 were also added to the graph and can be found both via the content of their abstracts as well as via a unique predicate/relationship which identifies them: *creditText*, which points to the source papers that were used to generate a given research prediction.

Predicting Future Scientific Research with Knowledge Graphs

Matheus Schmitz
Rehan Ahmed

The Big Picture

Find Relevant Articles

Extract Important Data

Iterate and Store Results

Build Knowledge Graph

Create Training Data and Learn

Predict 2021 Papers from Recent Years

Knowledge Graph Construction Phase

Machine Learning Phase

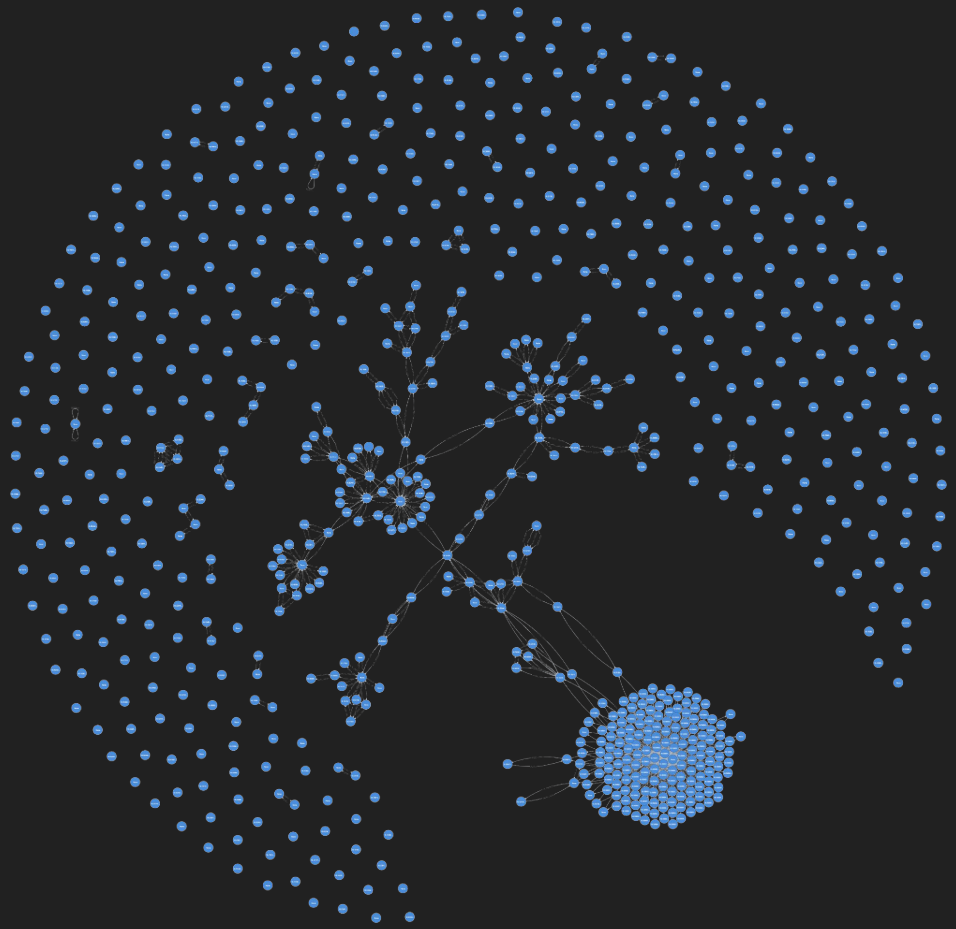
Challenges

(1) Building a connected graph:

Too much data out there, over 50k CS papers published per year on arXiv alone!

(2) Handling papers referenced in the KG but not crawled

Papers by one author:



Challenges

(3) Making plausible inferences about the future!

(And not generating predictions that are merely a reshuffling of their sources' abstracts)

(4) Making the graph useful

(You don't need a KG to look up papers by author or title)

Data Sources

dblp
computer science bibliography

Extract citations and references

Popular articles

arXiv

Abstracts for the popular articles

“Seed trail” crawling

Web Scraping

Entity Resolution

43 Million Pairs
Reduction Ratio: 0.93
3M pairs compared

Result: 20534 records

(1) Link Using
Title
Author
Date

Final Dataset

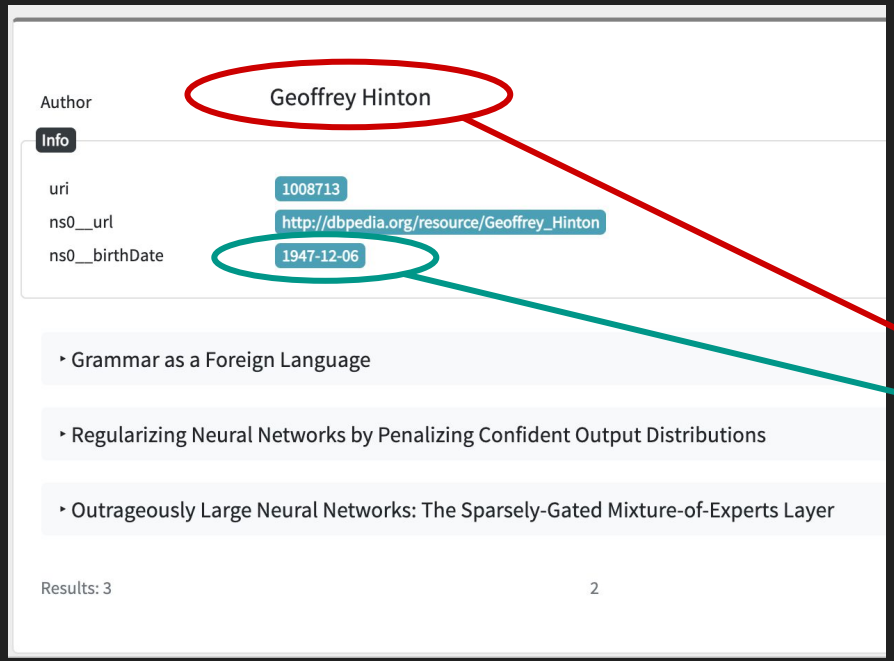
(2) Link Using
DOI

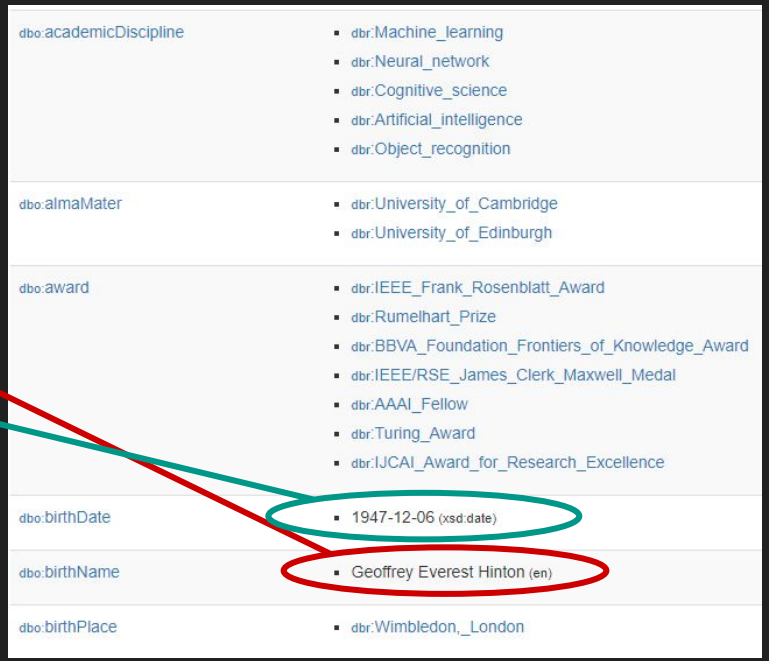
(3) Link Using
Title
Author
DOI

dblp
computer science bibliography

External KG Linkage

Our Knowledge Graph





Triple Generation

453k Paper URIs (incl. references)

51k Author URIs

298 DBpedia Matches

383 Predicted Papers

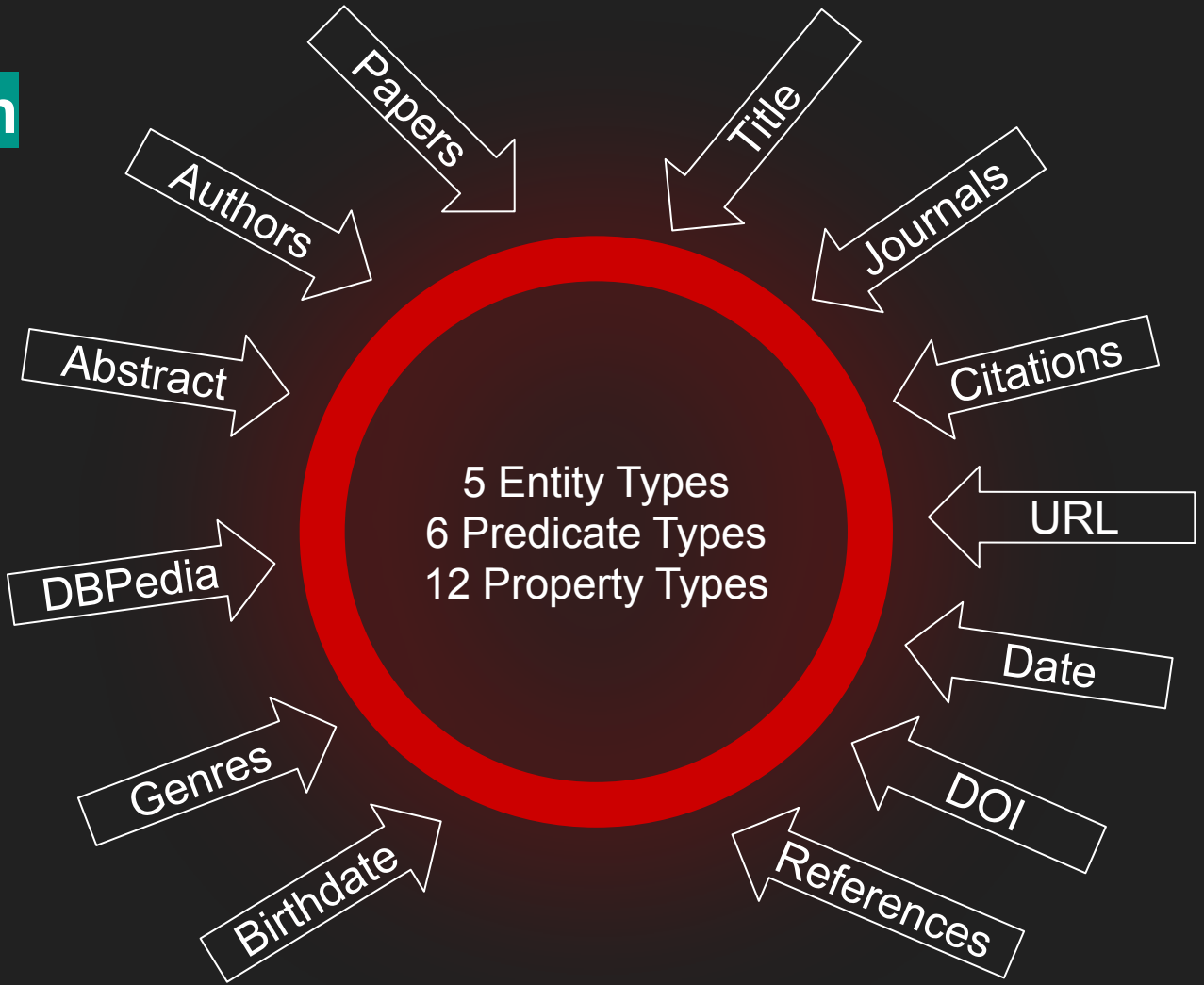
207 Genres

86 Publishers

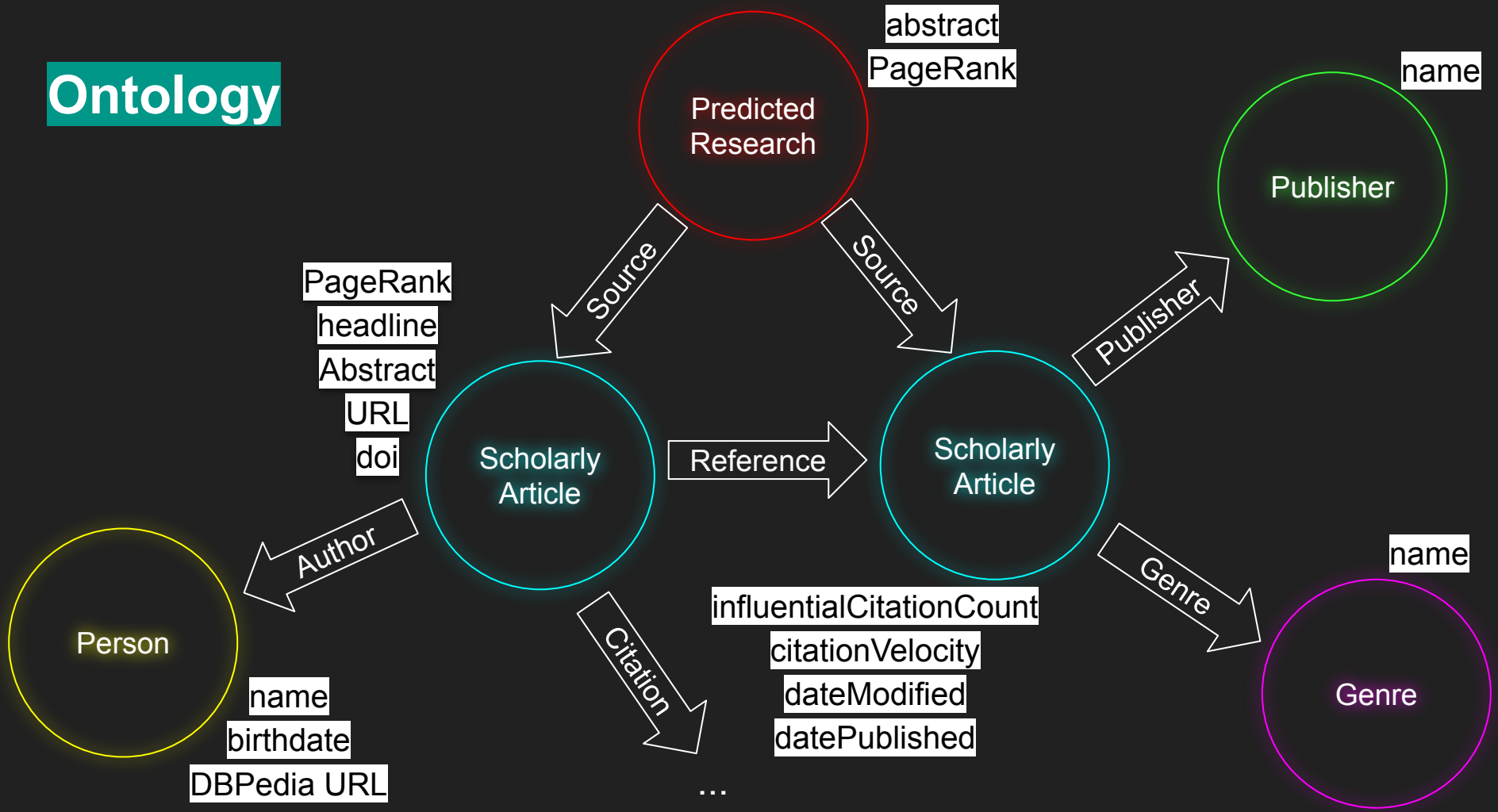
1.5M Relationships

505k Nodes

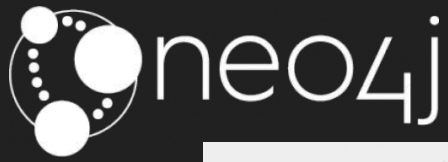
2.65M Triples!

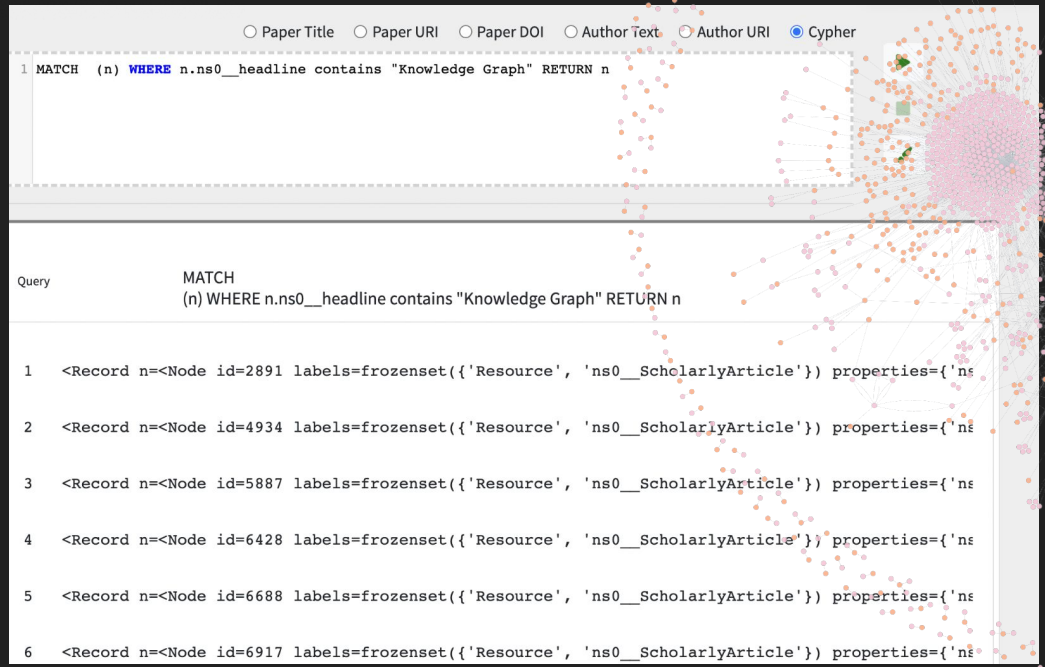


Ontology

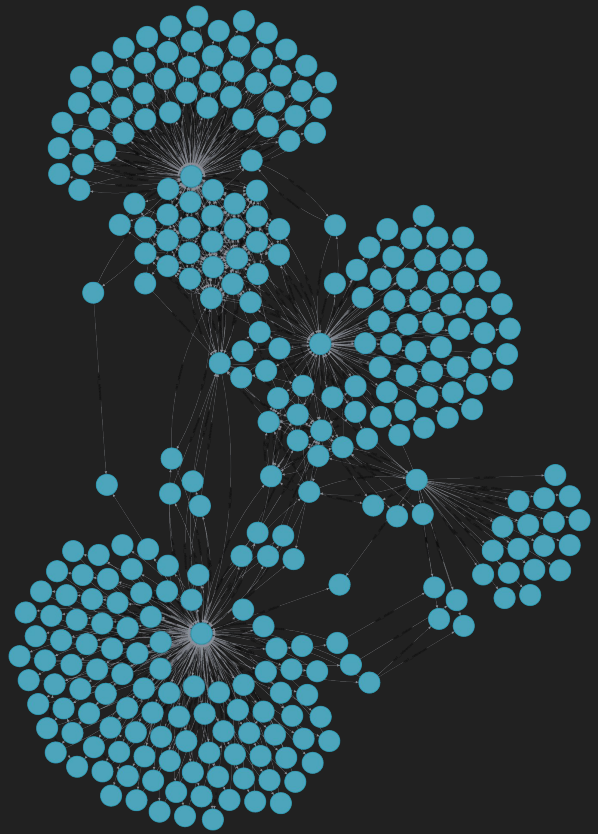

abstract
PageRank
Predicted Research
name
Publisher
Source
Source
Publisher
PageRank
headline
Abstract
URL
doi
Scholarly Article
Reference
Scholarly Article
Author
Person
name
birthdate
DBPedia URL
Citation
influentialCitationCount
citationVelocity
dateModified
datePublished
...
Genre
name
Genre

Graph Database





PageRank



Sample Endpoint Query:

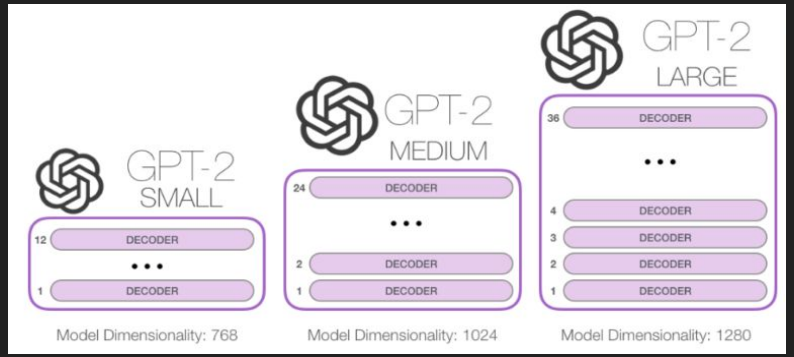
Title	Representation Learning: A Review and New Perspectives
URI	221
ns1__doi	10.1109/TPAMI.2013.50
ns0__datePublished	2013
ns0__url	https://www.semanticscholar.org/paper/184ac0766262312ba76bbdece4e7ffad0aa8180b
pagerank	21.52842592122827
ns1__citationVelocity	1384
ns1__b_influentialCitati	372
onCount	
Authors	Yoshua Bengio, P. Vincent, Aaron C. Courville
Abstract	The success of machine learning algorithms generally depends on data repres because different representations can entangle and hide more or less the diff behind the data. Although specific domain knowledge can be used to help de generic priors can also be used, and the quest for AI is motivating the design c algorithms implementing such priors. This paper reviews recent work in the a deep learning, covering advances in probabilistic models, autoencoders, mar motivates longer term unanswered questions about the appropriate objectiv computing representations (i.e., inference), and the geometrical connections estimation, and manifold learning.

Predicting Future Research

OpenAI

Transformers

build passing | license Apache-2.0 | website online | release v2.0.0



CO PRO

Predicting Future Research

(1) Reading the reference abstracts (priming the model)

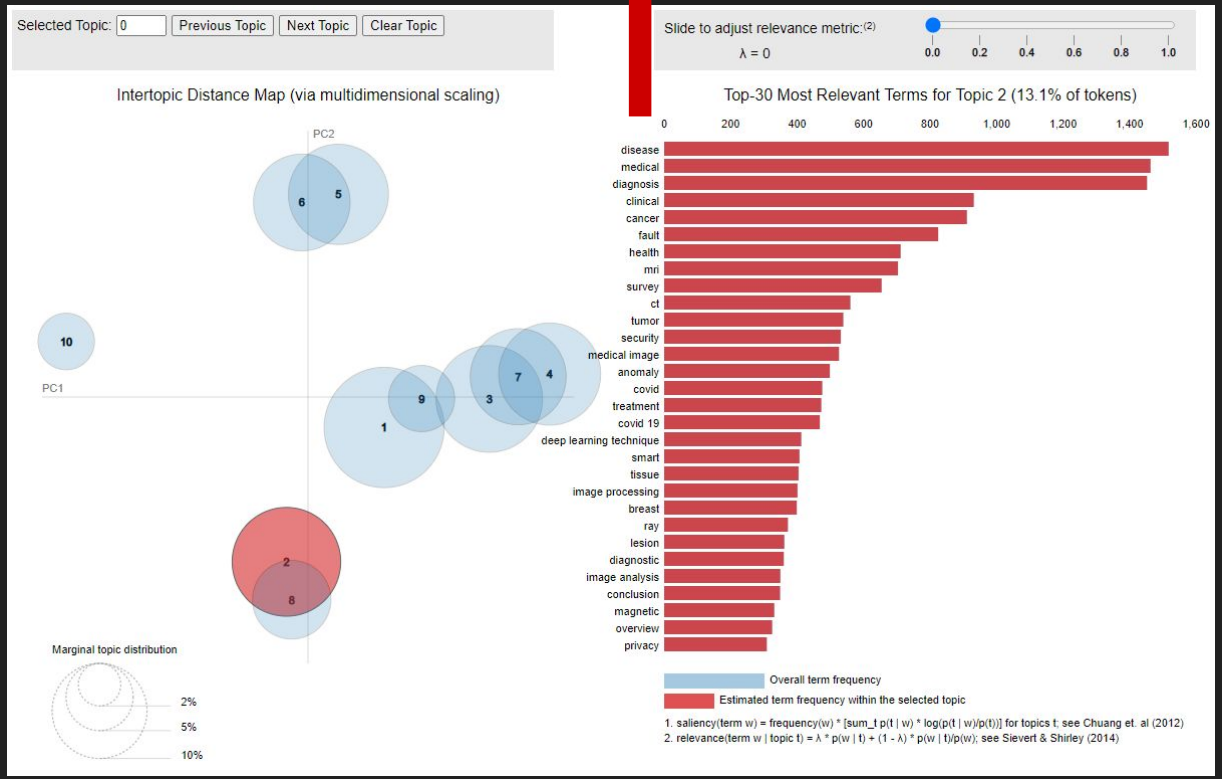
(2) Writing an entire new paper (text generation)

(3) Writing the paper's abstract (text summarization)

BONUS!

Topic Modeling

Medicine



Latent Dirichlet Allocation

BONUS!

Recommender System

```
%%time

INPUT_STRING = "Deep Support Vector Quantum Neural Graph Networks"

get_recommendation(INPUT_STRING,
                   NUM_RECOMMENDATIONS=10,
                   vectorizer = vectorizer,
                   LDA_obj = LDA_obj,
                   LDA_data = LDA_data,
                   topic_names = topic_names,
                   papers = papers)
```

```
Wall time: 1.07 s
```

	ID	Cosine Distance	Title	Modeled Topic	Citations
1	4751	0.011244	Underwater Target Classification Using Deep Ne...	feature; deep; classification; neural; neural ...	0.0
2	768	0.013166	Deep Learning for Monotonic Support Vector Mac...	feature; deep; classification; neural; neural ...	0.0
3	5567	0.013761	Stacked Sparse Autoencoder (SSAE) based framew...	feature; deep; classification; neural; neural ...	13.0
4	8586	0.020387	DeepGene: an advanced cancer type classifier b...	feature; deep; classification; neural; neural ...	13.0
5	6171	0.021423	Extract Features Using Stacked Denoised Autoen...	feature; deep; classification; neural; neural ...	0.0
6	4641	0.024708	Hybrid shallow and deep learned feature mixtur...	feature; deep; classification; neural; neural ...	0.0
7	4335	0.025183	DeepPPI: Boosting Prediction of Protein-Protei...	feature; deep; classification; neural; neural ...	20.0
8	3423	0.027645	Human emotion recognition using deep belief ne...	feature; deep; classification; neural; neural ...	36.0
9	7712	0.030773	A deep learning method for lincRNA detection u...	feature; deep; classification; neural; neural ...	0.0
10	1302	0.031278	ECG assessment based on neural networks with p...	feature; deep; classification; neural; neural ...	6.0

Trained on the entire graph

Paper recommendations using LDA similarity between input and abstracts

Not reliant on searching keywords in the database

Even hosted on a local machine can return query in 1 second

Validation

Validation Set: 98.4% TPR on DBpedia authors

Novelty in Generated Abstracts:

Only 7% of synthetic papers find their sources on the top 5 recommendations

Full dataset: 0.02% self-evident FPs (impossible 3-way matches)

Validation Set: 99.53% TPR on record linkage

Reduction Ratio: 0.93

Final Result

Recommender System

Text Query

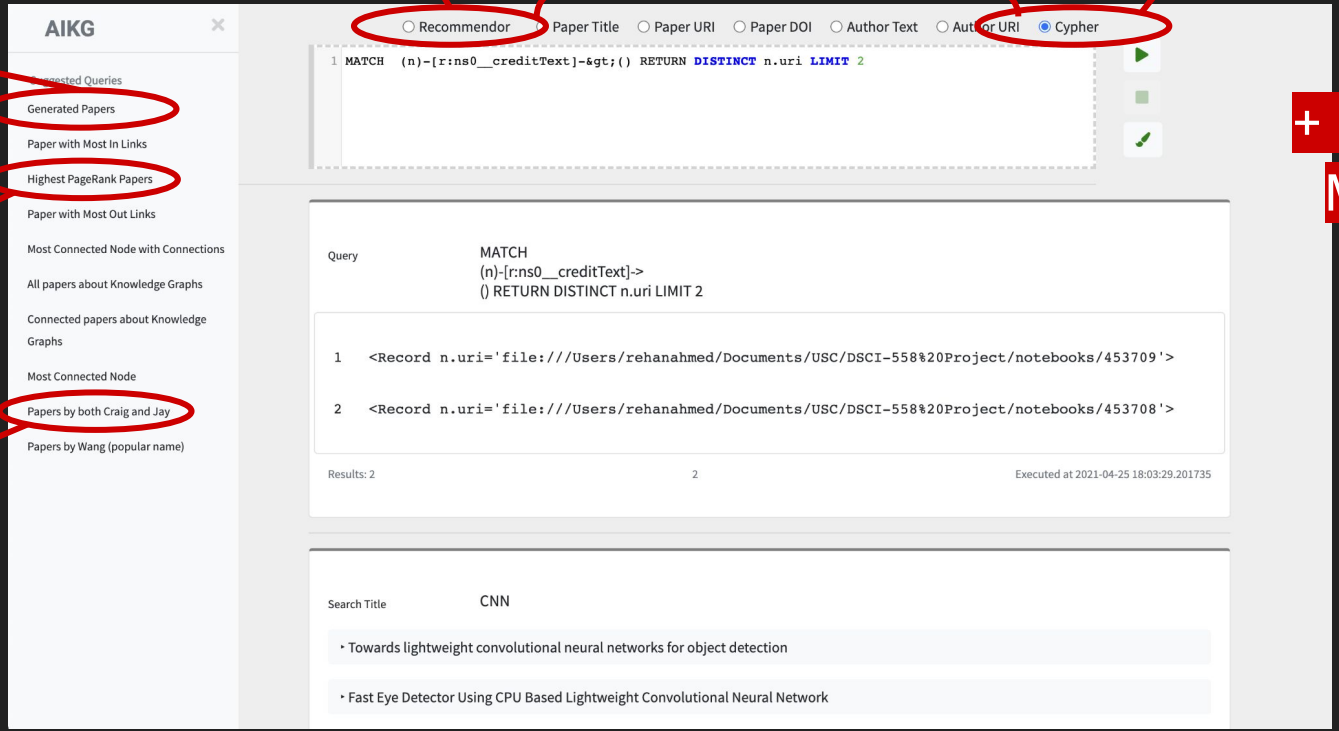
Neo4J Endpoint

Research Predictions

PageRank

DBpedia

+ LDA Topic Modeling



Neural-Net for Traffic Control on Foggy Days

'A sparse deep belief network with efficient fuzzy learning framework The algorithm is based on a deep neural network that is able to learn to distinguish between different types of blurred scenes . The algorithm can be applied to any scene, including static and dynamic ones, and it can also be used to improve the performance of existing algorithms . The authors demonstrate the use of deep learning in traffic prediction by using a network trained on real traffic data to train a model that could predict how fast a particular vehicle would be traveling at any given time of day, based on its speed in relation to other vehicles, such as cars, trucks, buses, motorcycles, bicycles, etc. In this way, the researchers demonstrate how deep networks can be used in a wide range of problems, including traffic management, vehicle safety, traffic congestion, urban planning, transportation planning and transportation planning . They hope that this work will inspire other researchers to explore the potential of this approach in other areas of this type of engineering . The work was supported by the U.S. National Science Foundation (NSF)'

Training a Facial Recognition Neural-Net from Brain Activity

"The reconstructions capture individual-specific information, such as the location of the eyes, nose, mouth, and other facial features, as well as their shape and size . This information can be used to improve the model's performance in a number of ways . For example, it is possible to use the reconstructed images to train a model to recognize faces . In this way, we are able to capture the individual features of a face and use them to predict the face's identity . We also use this information to enhance the performance of our model in other ways, for instance, by using it to detect faces that are not in the image . Our method is based on a combination of two approaches: (1) the use of a network-based approach and (2) a neural network based approach . Our model is able to extract features from the neural activity of a human subject, including the shape and shape of their dendritic trees . We show that our method can also be used as a tool for reconstruct"

typo

Thank You!

Questions?