Search Medium Write

From Text to Knowledge: The Information Extraction Pipeline

Implementation of information extraction pipeline that includes coreference resolution, entity linking, and relationship extraction techniques.

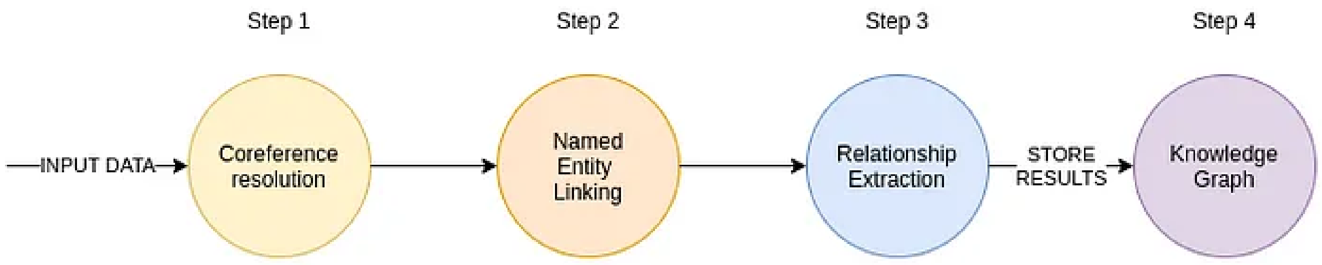
Tomaz Bratanic · Following
Published in Towards Data Science · 11 min read · Feb 12, 2021

1.8K 17

I am thrilled to present my latest project I have been working on. If you have been following my posts, you know that I am passionate about combining natural language processing and knowledge graphs. In this blog post, I will present my implementation of an information extraction data pipeline. Later on, I will also explain why I see the combination of NLP and graphs as one of the paths to explainable AI.

Information extraction pipeline

What exactly is an information extraction pipeline? To put it in simple terms, information extraction is the task of extracting structured information from unstructured data such as text.



Steps in my implementation of the IE pipeline. Image by author

My implementation of the information extraction pipeline consists of four parts. In the first step, we run the input text through a coreference resolution model. The coreference resolution is the task of finding all expressions that refer to a specific entity. To put it simply, it links all the pronouns to the referred entity. Once that step is finished, it splits the text into sentences and removes the punctuations. I have noticed that the specific ML model used for named entity linking works better when we first remove the punctuations. In the named entity linking part of the pipeline, we try to extract all the mentioned entities and connect them to a target knowledge base. The target knowledge base, in this case, is Wikipedia. Named entity linking is beneficial because it also deals with entity disambiguation, which can be a big problem.

Once we have extracted the mentioned entities, the IE pipeline tries to infer relationships between entities that make sense based on the text's context. The IE pipeline results are entities and their relationships, so it makes sense to use a graph database to store the output. I will show how to save the IE information to Neo4j.

I'll use the following excerpt from Wikipedia to walk you through the IE pipeline.

```
Elon Musk is a business magnate, industrial designer, and engineer.
He is the founder, CEO, CTO, and chief designer of SpaceX. He is also
early investor, CEO, and product architect of Tesla, Inc. He is also
the founder of The Boring Company and the co-founder of Neuralink. A
centibillionaire, Musk became the richest person in the world in
January 2021, with an estimated net worth of $185 billion at the
time, surpassing Jeff Bezos. Musk was born to a Canadian mother and
South African father and raised in Pretoria, South Africa. He briefly
attended the University of Pretoria before moving to Canada aged 17
to attend Queen's University. He transferred to the University of
Pennsylvania two years later, where he received dual bachelor's
degrees in economics and physics. He moved to California in 1995 to
attend Stanford University, but decided instead to pursue a business
career. He went on co-founding a web software company Zip2 with his
brother Kimbal Musk.
```

Text is copied from https://en.wikipedia.org/wiki/Elon_Musk and is available under CC BY-SA 3.0 license.

Step 1: Coreference resolution

As mentioned, the coreference resolution tries to find all expressions in the text that refer to a specific entity. In my implementation, I have used the Neuralcoref model from Huggingface that runs on top of the SpaCy framework. I have used the default parameters of the Neuralcoref model. One thing I did notice along the way is that the Neuralcoref model doesn't work well with location pronouns. I have also borrowed a small improvement code from one of the GitHub issues. The code for the coreference resolution part is the following:

If we run our example text through the coref_resolution function, we'll get the following output:

```
Elon Musk is a business magnate, industrial designer, and engineer.
Elon Musk is the founder, CEO, CTO, and chief designer of SpaceX.
Elon Musk is also early investor, CEO, and product architect of
Tesla, Inc. Elon Musk is also the founder of The Boring Company and
the co-founder of Neuralink. A centibillionaire, Musk became the
richest person in the world in January 2021, with an estimated net
```

```
worth of $185 billion at the time, surpassing Jeff Bezos. Musk was
born to a Canadian mother and South African father and raised in
Pretoria, South Africa. Elon Musk briefly attended the University of
Pretoria before moving to Canada aged 17 to attend Queen's
University. Elon Musk transferred to the University of Pennsylvania
two years later, where Elon Musk received dual bachelor's degrees in
economics and physics. Elon Musk moved to California in 1995 to
attend Stanford University, but decided instead to pursue a business
career. Elon Musk went on co-founding a web software company Zip2
with Elon Musk brother Kimbal Musk.
```

In this example, there are no advanced coreference resolution techniques required. The Neuralcoref model changed a couple of pronouns "He" to "Elon Musk". While it might seem very simple, this is an important step that will increase the overall efficiency of our IE pipeline.

Step 2: Named Entity Linking

Just recently, I have published a blog post using Named Entity Linking to construct a knowledge graph. Here, I wanted to use a different named entity linking model. I first tried to use the Facebook BLINK model, but I quickly realized it wouldn't work on my laptop. It needs at least 50GB of free space, which is not a big problem per se, but it also requires 32GB of RAM. My laptop has only 16GB of RAM, and we still need other parts of the pipeline to work. So I reverted to use the good old Wikifier API, which has already shown to be useful. And it's totally free. If you want to find more information about the API, look at my previous blog post or the official documentation.

Before we run our input text through the Wikifier API, we will split the text into sentences and remove the punctuations. Overall, the code for this step is as follows:

I forgot to mention that the Wikifier API returns all the classes that an entity belongs to. It looks at the **INSTANCE_OF** and **SUBCLASS_OF** classes and traverses all the way through the class hierarchy. I decided to filter out entities with categories that would belong to a person, organization, or location. If we run our example text through the Named Entity Linking part of the pipeline, we will get the following output.

A nice thing about the wikification process is that we also get the corresponding WikiData ids for entities along with their titles. Having the WikiData ids takes care of the entity disambiguation problem. You might wonder then what happens if an entity does not exist on Wikipedia. In that case, unfortunately, the Wikifier will not recognize it. I wouldn't worry too much about it, though, as Wikipedia has more than 100 million entities if I recall correctly.

If you look closely at the results, you'll notice that Pretoria is wrongly classified as an Organization. I tried to solve this issue, but the Wikipedia class hierarchy is complicated and usually spans five or six hops. If there are some Wiki class experts out there, I will happily listen to your advice.

Step 3: Relationship extraction

I have already presented all of the concepts until this point. I have never delved into relationship extraction before. So far, we have only played around with co-occurrence networks. So, I am excited to present a working relationship extraction process. I spend a lot of time searching for any open-source models that might do a decent job. I was delighted to stumble upon the OpenNRE project. It features five open-source relationship extraction models that were trained on either the Wiki80 or Tacred dataset. Because I am such a big fan of everything Wiki, I decided to use the Wiki80 dataset. Models trained on the Wiki80 dataset can infer 80 relationship types. I haven't tried the models trained on the Tacred dataset. You might try that on your own. In the IE pipeline implementation, I have used the `wiki80_bert_softmax` model. As the name implies, it uses the BERT encoder under the hood. One thing is sure. If you don't have a GPU, you are not going to have a good time.

If we look at an example relationship extraction call in the OpenNRE library, we'll notice that it only infers relationships and doesn't try to extract named entities. We have to provide a pair of entities with the `h` and `t` parameters and then the model tries to infer a relationship.

```
model.infer({'text': 'He was the son of Máel Dúin mac Máele Fithrich,
and grandson of the high king Áed Uaridnach (died 612).', 'h':
{'pos': (18, 46)}, 't': {'pos': (78, 91)}})
('father', 0.5108704566955566)
```

The results output a relationship type as well as the confidence level of the prediction. My not so spotless code for relationship extraction looks like this:

We have to use the results of the named entity linking as an input to the relationship extraction process. We iterate over every permutation of a pair of entities and try to infer a relationship. As you can see by the code, we also have a relation_threshold parameter to omit relationships with a small confidence level. You will later see why we use permutations and not combinations of entities.

So, if we run our example text through the relationship extraction pipeline, the results are the following:

Relationship extraction is a challenging problem to tackle, so don't expect perfect results. I must say that this IE pipeline works as well, if not better than some of the commercial solutions out there. And obviously, other commercial solutions are way better.

Step 4: Knowledge graph

As we are dealing with entities and their relationships, it only makes sense to store the results in a graph database. I used Neo4j in my example.

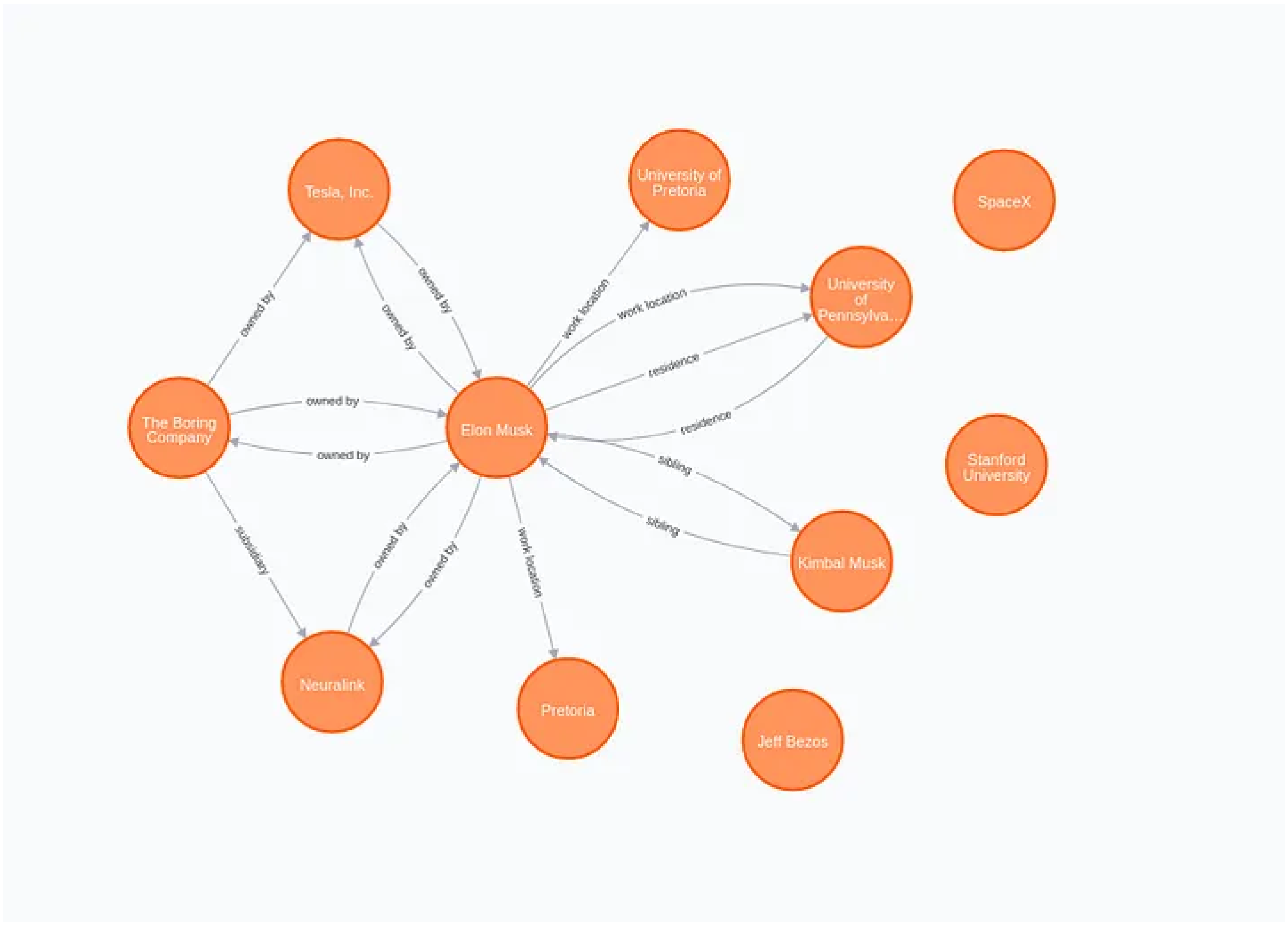


Image by author

Remember, I said that we would try to infer a relationship between all permutations of pairs of entities instead of combinations. Looking at table results, it would be harder to spot why. In a graph visualization, it is easy to observe that while most of the relationships are inferred in both directions, that is not true in all cases. For example, the work location relationship between Elon Musk and the University of Pennsylvania is assumed in a single direction only. That brings us to another shortcoming of the OpenNRE model. The direction of the relationship isn't as precise as we would like it to be.

A practical example of IE pipeline

To not leave you empty-handed, I will show you how you can use my IE implementation in your projects. We will run the IE pipeline through the BBC News Dataset found on Kaggle. The hardest part about the IE pipeline implementation was to set up all the dependencies. I want you to retain your mental sanity, so I built a docker image that you can use. Run the following command to get it up and running:

```
docker run -p 5000:5000 tomasonjo/trinityie
```

On the first run, the OpenNRE models have to be downloaded, so definitely don't use `-rm` option. If you want to do some changes to the project and built your own version, I have also prepared a GitHub repository.

As we will be storing the results into Neo4j, you will also have to download and set it up. In the above example, I have used a simple graph schema, where nodes represent entities and relationships represent, well, relationships. Now we will refactor our graph schema a bit. We want to store entities and relationships in the graph but also save the original text. Having an audit trail is very useful in real-world scenarios as we already know that the IE pipeline is not perfect.

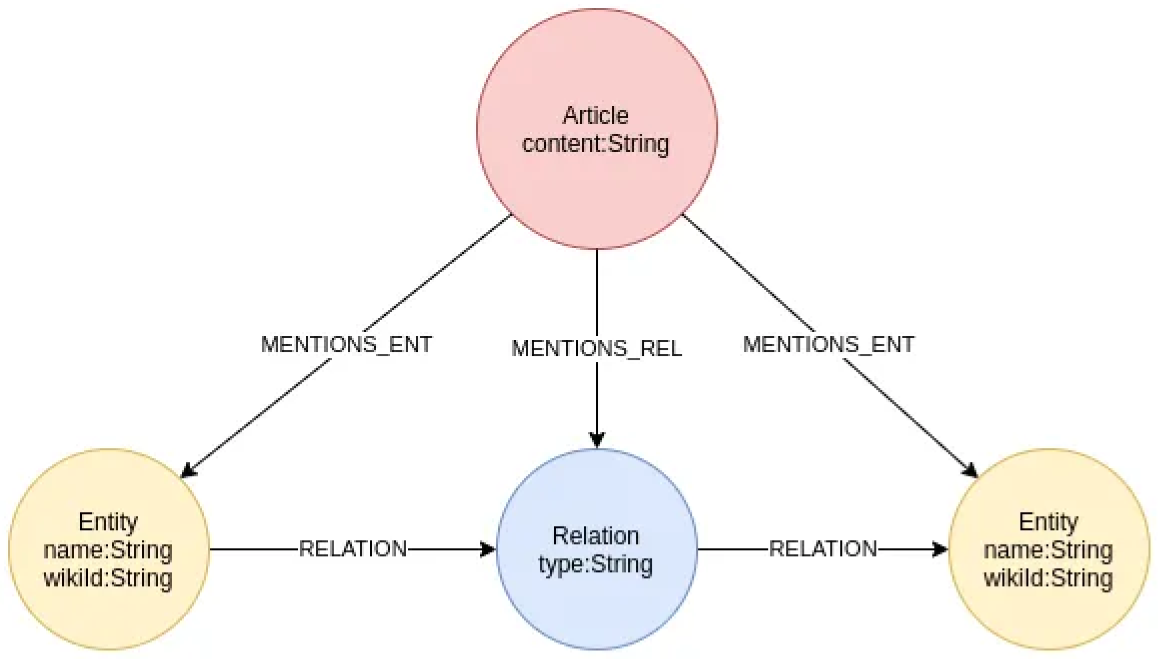


Image by author

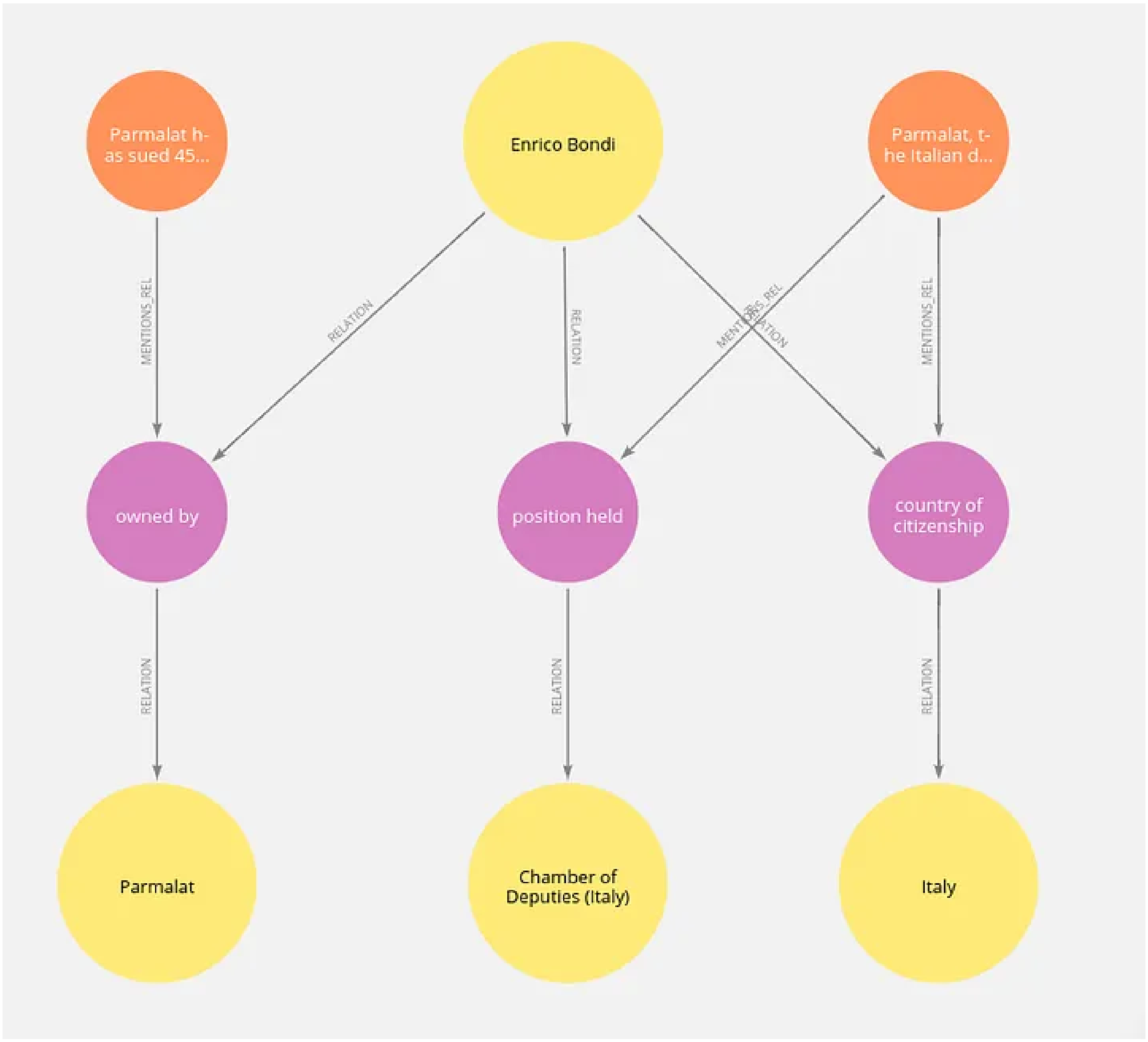
It might be a bit counter-intuitive to refactor a relationship into an intermediate node. The problem we are facing is that we can't have a relationship pointing to another relationship. Given this issue, I have decided to refactor a relationship into an intermediate node. I could have used my imagination to produce better relationship types and node labels, but it is what it is. I only wanted for the relationship direction to retain its function.

The code to import 500 articles in the BBC news dataset to Neo4j is the following. You'll have to have the trinityIE docker running for the IE pipeline to work.

The code is also available in the form of a Jupyter Notebook on GitHub. Depending on your GPU capabilities, the IE pipeline might take some time. Let's now inspect the output. Obviously, I chose results that make sense. Run the following query:

```
MATCH p=(e:Entity{name:'Enrico Bondi'})-[:RELATION]->(r)-[:RELATION]-
>(),
          (r)<-[:MENTIONS_REL]-(s)
RETURN *
```

Results



Results of IE extraction on BBC news dataset. Image by author

We can observe that Enrico Bondi is an Italian citizen. He held a position at Italy's Chamber of Deputies. Another relationship was inferred that he also owns Parmalat. After a short Google search, it seems that this data is more or less at least in the realms of possible.

Path to explainable AI

You might wonder, what has this got to do with explainable AI. I'll give you a real-world example. This research paper is titled Drug Repurposing for COVID-19 via Knowledge Graph Completion. I'm not a doctor, so don't expect a detailed presentation, but I can give a high-level overview. There are a lot of medical research papers available online. There are also online medical entities databases such as MeSH or Ensembl. Suppose you run a Named Entity Linking model on biomedical research papers and use one of the online medical databases as a target knowledge base. In that case, you can extract mentioned entities in articles. The more challenging part is the relationship extraction. Because this is such an important field, great minds have come together and extracted those relationships.

Probably there are more projects, but I am aware of the SemMedDB project, which was also used in the mentioned article. Now that you have your knowledge graph, you can try to predict new purposes for existing drugs. In network science, this is referred to as link prediction. When you are trying to predict links as well as their relationship types, then the scientific community calls it knowledge graph completion. Imagine we have predicted some new use cases for existing drugs and show our results to a doctor or a pharmacologist. His response would probably be, that's nice, but what makes you think this new use case will work? The machine learning models are a black box, so that's not really helpful. But what you can give to the doctor is all the connections between the existing drug and the new disease it could treat. And not only direct relationships, but also those that are two or three hops away. I'll make up an example, so it might not make sense to a biomedical researcher. Suppose the existing drug inhibits a gene that is correlated to the disease. There might be many direct or indirect connections between the drug and the disease that might make sense. Hence, we have embarked on a step towards an explainable AI.

Conclusion

I am really delighted with how this project worked out. I've been tinkering with combining NLP and Knowledge graphs for the last year or so, and now I have poured all of my knowledge into a single post. I hope you enjoyed it!

p.s. If you want to make some changes to the IE pipeline, the code is available as a Github repository. The code for reproducing this blog post is also available as a Jupyter Notebook.

Neo4j NLP Graph Spacy Information Extraction

View list

Written by Tomaz Bratanic

Following

6.2K Followers · Writer for Towards Data Science

Data explorer. Turn everything into a graph. Author of Graph algorithms for Data Science at Manning publication. http://mng.bz/GGVN

More from Tomaz Bratanic and Towards Data Science

Tomaz Bratanic in Neo4j Developer Blog

LangChain Library Adds Full Support for Neo4j Vector Index

Streamlining data ingestion and querying in Retrieval-Augmented Generation…

6 min read · Aug 31

318 6

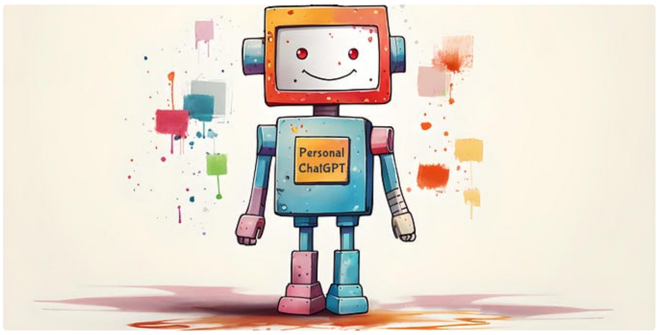
Antonis Makropoulos in Towards Data Science

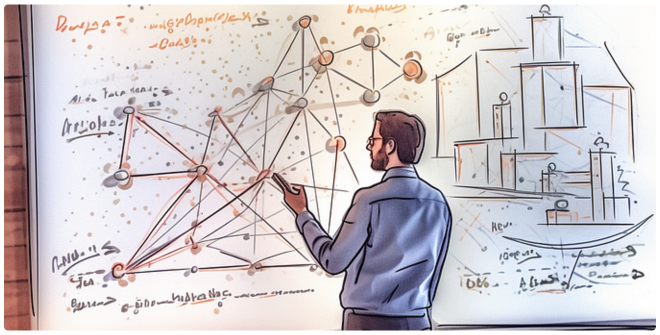
How to Build a Multi-GPU System for Deep Learning in 2023

This story provides a guide on how to build a multi-GPU system for deep learning and…

10 min read · Sep 17

549 11





Robert A. Gonsalves in Towards Data Science

Your Own Personal ChatGPT

How you can fine-tune OpenAI's GPT-3.5 Turbo model to perform new tasks using you...

✦ · 15 min read · Sep 8

595 7

Tomaz Bratanic in Neo4j Developer Blog

Knowledge Graphs & LLMs: Real-Time Graph Analytics

Understanding data points through the context of their relationships

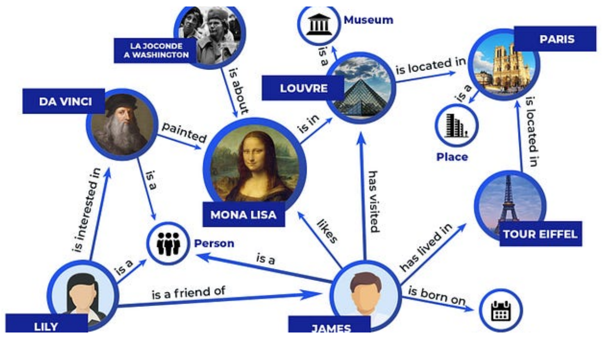
9 min read · Jul 13

232

See all from Tomaz Bratanic

See all from Towards Data Science

Recommended from Medium



Alla Chepurova in DeepPavlov

Improving Knowledge Graph Completion with Generative LM...

Combining both internal LM knowledge and external data from KG

13 min read · Sep 5

36

0xdevshah in AI Skunks

Knowledge Graphs: A Comprehensive Analysis

Knowledge graphs are becoming increasingly important in NLP due to their ability to mode...

8 min read · Apr 9

63

Lists

Natural Language Processing

669 stories · 283 saves

The New Chatbots: ChatGPT, Bard, and Beyond

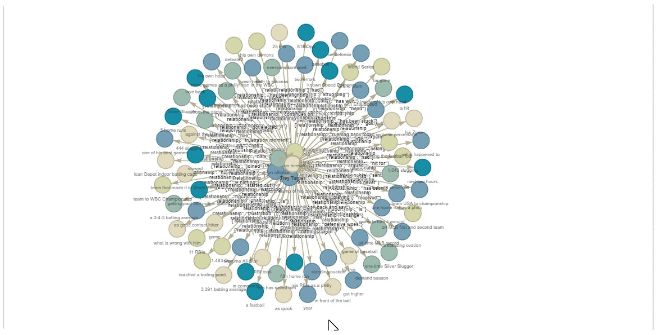
13 stories · 133 saves

New_Reading_List

174 stories · 133 saves

Staff Picks

465 stories · 317 saves



Wenqi Glantz in Better Programming

7 Query Strategies for Navigating Knowledge Graphs With...

Exploring NebulaGraph RAG Pipeline with the Philadelphia Phillies

· 17 min read · 4 days ago

Alex Reed

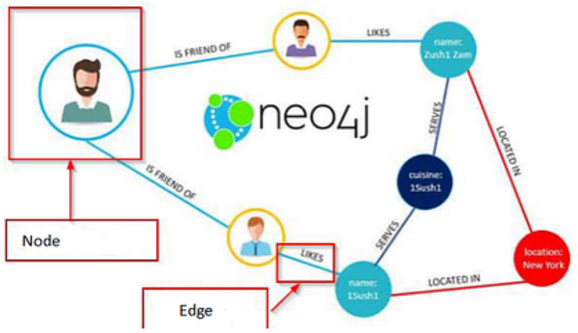
Writing Your First NLP Python Script: From Text Preprocessing t...

Welcome to the world of Natural Language Processing (NLP)! NLP is a fascinating field a...

5 min read · Sep 22

501 4

56



Ajay Verma

Knowledge Graph from Text Data

How to generate knowledge graph for any text data

2 min read · Sep 18

63

Serbu Claudiaa

How to use Neo4j

A graph-based database management system represents an alternative to the...

9 min read · Sep 15

50

See more recommendations