



Knowledge Graph Generation of Diseases

CS - 563 : Neural Networks for Natural Language Processing

Indian Institute of Technology Guwahati

Submitted By :-

Atul Bhagat 224156004

Ashish Kumar 224156003

Atal Bhatia 190101017

Instructor - Dr. Amit Awekar

19 November 2022

Odd Semester

Academic Year 2022-23

Abstract

With exponentially increasing data in unstructured form, there arises a dire need to represent this data in machine understandable form like a knowledge graph. We are trying to extract entities and relationships between these entities from this data lump, which then can be built into a knowledge graph that can be used for many downstream tasks like answering queries from text, text summarisation etc. In this project, we tried three different approaches of extracting knowledge graphs of diseases

- 1) **SCIERO**, which is a fully neural pipeline recently published that extracts KGs based on an ensemble approach
- 2) **REBEL**, that is a readily available python library for extracting KGs
- 3) **Data Scrapping** from the summary tables of the Wikipedia Articles

1. Introduction

“Knowledge” is defined by using Wikipedia as "a familiarity, cognisance, or expertise of someone or something, such as statistics, abilities or gadgets." It exists in a myriad of information assets and formats - be it textual, graphical, auditory, or a combination of those. So as for computer systems to perceive and work with this concept of knowledge, knowledge desires to be modelled with as a minimum a few degrees of structure or form.

Even as knowledge graph offer a cleaner machine understandable representation of abstract records, they have to but, be constituted of the stated summary records. This entails converting large quantities of unstructured statistics like textual content, images, motion, pictures and many others to formularised semantic representations even though seemingly trivial for people, this venture proves to be quite tough to be carried out on a computer system. So, In this project we propose three methods to extract the data from any text and construct knowledge graph that can be utilised by any abstract downstream tasks.

There are generally three phases in a knowledge graph extraction pipeline i.e. Entity and Relation Extraction, Entity and Relation Refining and Merging and at last generating knowledge graphs of the final triplets.

2. Methodology

2.a) Using SCIERO

SCIERO is a deep learning and NLP approach for generating scientific knowledge graphs in the computer science domain. It is a large-scale utility that automatically generates knowledge graph composed by over **350M RDF triples** describing **41M statements** from **6.7M articles** about **10M entities** (e.g., tasks, methods, materials, metrics) linked by **179 semantic relations**. It is designed to support a large variety of intelligent services for analysing and making sense of research dynamics, supporting researchers in their daily job, and informing decision of founding bodies and research policy makers.

It was generated by applying an automatic pipeline that extracts entities and relationships using four tools: DyGIE++, Stanford CoreNLP, the CSO Classifier, and a new PoS Tagger. It then integrates and filters the resulting triples using a combination of deep learning and semantic technologies in order to produce a high quality knowledge graph. This pipeline was evaluated on a manually crafted gold standard yielding very competitive results.

The flow of the complete pipeline has been explained thoroughly in the next page by taking one example input that spans over all the processing that's being done to generate the final triples out of it.

Example - INPUT

```
{ "index": "aida_2021", "_type": "_doc", "id": "2466206094", "score": 2.0, "_ignored": ["abstract.keyword"], "_source": {
  "citationcount":
: 2, "confname": "icost 2016", "references": [80032331, 2058184275, 2106985027, 2121381553, 2043009189, 2026107757, 2001599770, 1991898702,
2158310867, 2081258511, 2016599119, 2065078650, 1585879837, 2018589614, 1979561374, 2052773675, 2140555382, 1987741348, 2146521180,
2980196493], "year": "2016-01-01", "topics": ["computer vision", "word error rate", "computer science", "artificial intelligence", "dead reckoning", "computer
security", "localization system", "pedestrian"],
},
  "papertitle": "smartswim an infrastructure free swimmer localization system based on smartphone sensors",
  "confseries": "ICOST",
  "language": [
    "en"
  ],
  "abstract": "Many works have focused their attention on the sports activity monitoring and recognition using inherit
sensors on the smartphone. However, distinct from many on-the-ground activities, swimming is not only hard to monitor
but also dangerous in the water. Knowing the position of a swimmer is crucial which can help a lot in rescuing people. In
this paper, we propose a system called SmartSwim employing smartphone as a sensor for swimming tracking and
localization. In detail, we first present a sensor based swimming status classification and moving length estimation. A
swimmer locating algorithm is then proposed drawing on the experience of pedestrian dead reckoning PDR concept. We
implemented the system on commercial smartphones and designed two prototype applications named WeSwim and
SafeSwim. Experimental results showed the accuracy of swimming status classification reaches more than 99i%4z% and
the Error Rate value for length estimation is lower than 7i%4z% overall.",
  "conferenceseriesid": 1180241966, "cso_old_enhanced": ["mobile phones", "smart phones", "sensors", "localization system", "dead reckoning", "cell phone",
"computer hardware", "telephone sets", "mobile telecommunication systems", "robot applications", "localization algorithm", "navigation systems", "cellular
telephone systems", "telephone", "telecommunication equipment", "computer science", "mobile devices", "communication channels (information theory)",
"telecommunication networks", "robots", "mobile robots", "wireless sensor networks", "mobile computing", "engineering", "wireless telecommunication systems",
"telecommunication systems", "robotics", "wireless communications", "energy utilization", "sensor nodes", "network protocols", "routing algorithms", "data
communication systems", "computer network", "communication systems", "computer systems", "electricity", "internet", "signal processing"], "confplace":
"Wuhan, China", "cso_syntactic_topics": ["smart phones", "dead reckoning", "localization system", "sensors"], "urls": ["https://link.springer.com/chapter/
10.1007/978-3-319-39601-9_20", "https://dl.acm.org/citation.cfm?id=2960876"], "cso_old_annotated": true, "cso_semantic_topics": ["localization system",
"smart phones", "sensors", "mobile phones", "classification models"], "confseriesname": "International Conference on Smart Homes and Health Telematics", "id":
2466206094, "cso_old_semantic": ["sensors", "localization system", "dead reckoning", "smart phones"], "cso_enhanced_topics": ["mobile phones", "sensors",
"classification models", "smart phones", "localization system", "dead reckoning", "computer hardware", "classification methods", "telephone sets", "mobile
telecommunication systems", "robot applications", "localization algorithm", "navigation systems", "computer science", "computer systems", "mobile devices",
"communication channels", "telecommunication networks", "robots", "mobile robots", "wireless sensor networks", "mobile computing", "wireless
telecommunication systems", "telecommunication systems", "robotics", "wireless communications", "energy utilization", "sensor nodes", "routing algorithms",
"data communication systems", "computer networks", "communication systems", "electricity", "engineering"], "doi": "10.1007/978-3-319-39601-9_20",
"authors": [{"country": "China", "affiliation": "Northwestern Polytechnical University", "name": "fei yi", "id": 2700521216, "gridid": "grid.440588.5", "affiliationid":
17145004, "order": 3
}, {"country": "United States", "affiliation": "Temple University", "name": "chiu c tan", "id": 2139448578, "gridid": "grid.264727.2", "affiliationid": 84392919, "order":
5
}, {"country": "China", "affiliation": "Northwestern Polytechnical University", "name": "zhiwen yu", "id": 2118377178, "gridid": "grid.440588.5", "affiliationid":
17145004, "order": 2
}, {"country": "China", "affiliation": "Xi'an University of Science and Technology", "name": "liang wang", "id": 2997442707, "gridid": "grid.440720.5", "affiliationid":
110440473, "order": 4
}, {"country": "China", "affiliation": "Northwestern Polytechnical University", "name": "bin guo", "id": 2105384808, "gridid": "grid.440588.5", "affiliationid":
17145004, "order": 6
}, {"country": "China", "affiliation": "Northwestern Polytechnical University", "name": "dong xiao", "id": 2526046322, "gridid": "grid.440588.5", "affiliationid":
17145004, "order": 1
}
], "cso_annotated": true, "grid_type": {
  "education":
: 100.00000000000001
}, "aida_annotated": false, "type": "academia", "countries": {
  "china":
: 83.33333333333334, "united states": 16.666666666666668
}, "citation_for_year": [{
  "year":
: "2020-01-01", "citationcount": 2
}
]
}}
```

Extraction Module

DYGIEPP Input

DYGI Epp -> <https://aclanthology.org/D19-1585.pdf>

Git Repository -> <https://github.com/dwadden/dygiepp>

In this, entities and relationships are extracted by using a transformer model able to capture the meaning of pieces of text that are associated with pre-defined entity types and find relationships among the entities.

Problem with this approach is that the **relationships** among entities are **not extracted from text**.

Model Used - science, a BERT-based model tuned for parsing computer science scientific text.

Entities - Method, Task, Material, Metric, *Other-Scientific-Term* and *Generic* (merged as Other-Text)

Relations - Used-for, hyponym-Of, Compare, Part-of, Conjunction, Feature-of, Evaluate-for

Sample Input -

```

{
  "clusters": ["", "", "", "", "", "", "", "", "", "sentences": ["smartswim", "an", "infrastructure", "free", "swimmer", "localization", "system", "based", "on", "smartphone", "sensors"], ["Many", "works", "have", "focused", "their", "attention", "on", "the", "sports", "activity", "monitoring", "and", "recognition", "using", "inherit", "sensors", "on", "the", "smartphone", "."], ["However", ",", ",", "distinct", "from", "many", "on-the-ground", "activities", ",", ",", "swimming", "is", "not", "only", "hard", "to", "monitor", "but", "also", "dangerous", "in", "the", "water", "."], ["Knowing", "the", "position", "of", "a", "swimmer", "is", "crucial", "which", "can", "help", "a", "lot", "in", "rescuing", "people", "."], ["In", "this", "paper", ",", ",", "we", "propose", "a", "system", "called", "SmartSwim", "employing", "smartphone", "as", "a", "sensor", "for", "swimming", "tracking", "and", "localization", "."], ["In", "detail", ",", ",", "we", "first", "present", "a", "sensor", "based", "swimming", "status", "classification", "and", "moving", "length", "estimation", "."], ["A", "swimmer", "locating", "algorithm", "is", "then", "proposed", "drawing", "on", "the", "experience", "of", "pedestrian", "dead", "reckoning", "PDR", "concept", "."], ["We", "implemented", "the", "system", "on", "commercial", "smartphones", "and", "designed", "two", "prototype", "applications", "named", "WeSwim", "and", "SafeSwim", "."], ["Experimental", "results", "showed", "the", "accuracy", "of", "swimming", "status", "classification", "reaches", "more", "than", "99", "%", "and", "the", "Error", "Rate", "value", "for", "length", "estimation", "is", "lower", "than", "7", "%", "overall", "."], "ner": ["", "", "", "", "", "", "", "", "relations": ["", "", "", "", "", "", "", "", "doc_key": "2466206094", "dataset": "scierc"
}

```

DYGIEPP Output

{**"doc_key"**: "2466206094", **"dataset"**: "scierc", **"sentences"**: ["smartswim", "an", "infrastructure", "free", "swimmer", "localization", "system", "based", "on", "smartphone", "sensors"], ["Many", "works", "have", "focused", "their", "attention", "on", "the", "sports", "activity", "monitoring", "and", "recognition", "using", "inherit", "sensors", "on", "the", "smartphone", "."], ["However", ",", "distinct", "from", "many", "on-the-ground", "activities", ",", "swimming", "is", "not", "only", "hard", "to", "monitor", "but", "also", "dangerous", "in", "the", "water", "."], ["Knowing", "the", "position", "of", "a", "swimmer", "is", "crucial", "which", "can", "help", "a", "lot", "in", "rescuing", "people", "."], ["In", "this", "paper", ",", "we", "propose", "a", "system", "called", "SmartSwim", "employing", "smartphone", "as", "a", "sensor", "for", "swimming", "tracking", "and", "localization", "."], ["In", "detail", ",", "we", "first", "present", "a", "sensor", "based", "swimming", "status", "classification", "and", "moving", "length", "estimation", "."], ["A", "swimmer", "locating", "algorithm", "is", "then", "proposed", "drawing", "on", "the", "experience", "of", "pedestrian", "dead", "reckoning", "PDR", "concept", "."], ["We", "implemented", "the", "system", "on", "commercial", "smartphones", "and", "designed", "two", "prototype", "applications", "named", "WeSwim", "and", "SafeSwim", "."], ["Experimental", "results", "showed", "the", "accuracy", "of", "swimming", "status", "classification", "reaches", "more", "than", "99%", "and", "the", "Error", "Rate", "value", "for", "length", "estimation", "is", "lower", "than", "7%", "overall", "."], **"ner"**: [{"O", "O", "O", "O", "O", "O", "O", "O"}, {"B-Me", "Method", "23.6889", "1.0"}, {"B-Me", "Method", "9.2639", "0.9999"}, {"I-10", "OtherScientificTerm", "16.3494", "1.0"}, {"I-19", "23,

"Task", 30.5581, 1.0], [25, 26, "OtherScientificTerm", 16.1658, 1.0], [29, 29, "OtherScientificTerm", 11.1627, 1.0]], [[36, 37, "OtherScientificTerm", 17.7275, 1.0], [39, 39, "OtherScientificTerm", 5.148, 0.7189]], [], [[77, 77, "Generic", 35.8234, 1.0], [79, 79, "Method", 29.7737, 1.0], [81, 81, "Material", 13.3827, 0.975], [84, 84, "OtherScientificTerm", 3.155, 0.9504], [86, 89, "Task", 40.1797, 1.0]], [[98, 102, "Task", 18.2747, 1.0], [104, 106, "Task", 28.3967, 1.0]], [[109, 111, "Method", 36.8435, 1.0], [120, 124, "OtherScientificTerm", 10.7493, 0.9999]], [[129, 129, "Generic", 36.9583, 1.0], [131, 132, "Material", 2.5155, 0.9251], [136, 137, "Generic", 19.5727, 1.0], [139, 139, "Method", 7.2664, 0.9749], [141, 141, "Method", 7.3634, 0.9975]], [[147, 147, "Metric", 34.6707, 1.0], [149, 151, "Task", 12.1183, 0.999], [159, 161, "Metric", 42.1028, 1.0], [163, 164, "Task", 21.8092, 1.0]], "relations": [[], [], [], [], [], [], [], [], [], []], "predicted_relations": [[], [], [], [], [], [], [], [], [], []], [77, 77, 86, 89, "USED-FOR", 25.4548, 1.0], [81, 81, 77, 77, "USED-FOR", 9.4836, 0.9999]], [[98, 102, 104, 106, "CONJUNCTION", 19.9681, 1.0]], [[120, 124, 109, 111, "USED-FOR", 23.9597, 1.0]], [[131, 132, 129, 129, "USED-FOR", 2.1679, 0.8973], [139, 139, 136, 137, "HYPONYM-OF", 31.6753, 1.0], [139, 139, 141, 141, "CONJUNCTION", 26.937, 1.0], [141, 141, 136, 137, "HYPONYM-OF", 28.8115, 1.0]], [[147, 147, 149, 151, "EVALUATE-FOR", 22.199, 1.0], [147, 147, 163, 164, "EVALUATE-FOR", 7.1789, 0.9992], [159, 161, 149, 151, "EVALUATE-FOR", 16.0454, 1.0], [159, 161, 163, 164, "EVALUATE-FOR", 24.961, 1.0]], "clusters": [[], [], [], [], [], [], [], [], [], []], "predicted_clusters": [[[0, 0], [39, 39], [77, 77], [79, 79], [129, 129]]]

CSO Classifier Module

It is an classifier built on top of the Computer Science Ontology

It uses unsupervised syntactic (n-gram comparisons) and semantic (Word2Vec, regular expressions on PoS Tag, and pre-decided similarity threshold) based classification components to detect research topics in a text.

Output of this sub-module is set of entities E-CSO.

CSO Topics

cross-language information retrieval document clustering image retrieval information dissemination information extraction information integration information retrieval system information storage and retrieval music information retrieval named entity recognition question answering recommender systems relevance model text processing vector space model machine learning classifier cluster analysis ensemble learning forecasting model inductive logic programming link prediction neural network parameter learning pattern recognition reinforcement learning supervised machine learning decision tree bayesian classifier dimensionality reduction graph neural network deep learning cluster analysis associative processing pattern classification hierarchical dirichlet process supervised machine learning cluster system text clustering abstracting indexing dialogue system information extraction machine translation part of speech question answering named entity recognition semantic similarity sentiment analysis syntactic parsing word segmentation dependency parsing natural language understanding natural language processing text processing concept similarity natural language text n gram language model relation extraction opinion mining sentiment classification translation model spoken language understanding wordnet dependency tree information retrieval technology natural language processing tool word vector nlp task natural language processing task natural language generation parsing algorithm document classification automatic translation spoken language processing question classification text classification

CoreNLP - OpenIE Module

It is an annotator that **extracts domain-independent triples** from a text.

Clauses (SVO) -> Shortened Clauses -> Triples

Resulting **triples are filtered** by checking the overlapping of the subjects or objects with the set E-CSO or E-DYGIapp

Relations must have **"Verb"** PoS tag to build the KG and Output is set of triples T-OpenIE

PoS Tag-Based Relationships Extractor Module

This associates a Part-Of-Speech tag with each term in an input text.

Used to detect verbs between pairs of entities (subset of E-CSO and E-DyGLEpp) in a sentence

Verbs are sought only between couples of entities whose distance (i.e. # of tokens between two entities) is equal or less than a predefined window size w.

The output of this module is the set of triples T-PoS

Path-based Relationships Extractor Module

Aim is extract meaningful triples by exploiting a set of pre-defined paths on the dependency trees of the sentences. Set of shortest paths P on the dependency trees containing at least one verb between the tokens of a pair of entities is generated. 50 most frequent shortest paths (P') are considered to be good paths.

Idea behind selecting the frequent paths is that a path must identify a certain amount of knowledge; a path that only detects a few triples is not of interest for the KG. 20 paths out of set P' were extracted and manually annotated by 4 computer science researchers and finally paths were evaluated considering the number of correct triples out of the 20 extracted (>60% correct). This resulted in selecting 12 final paths (P'')

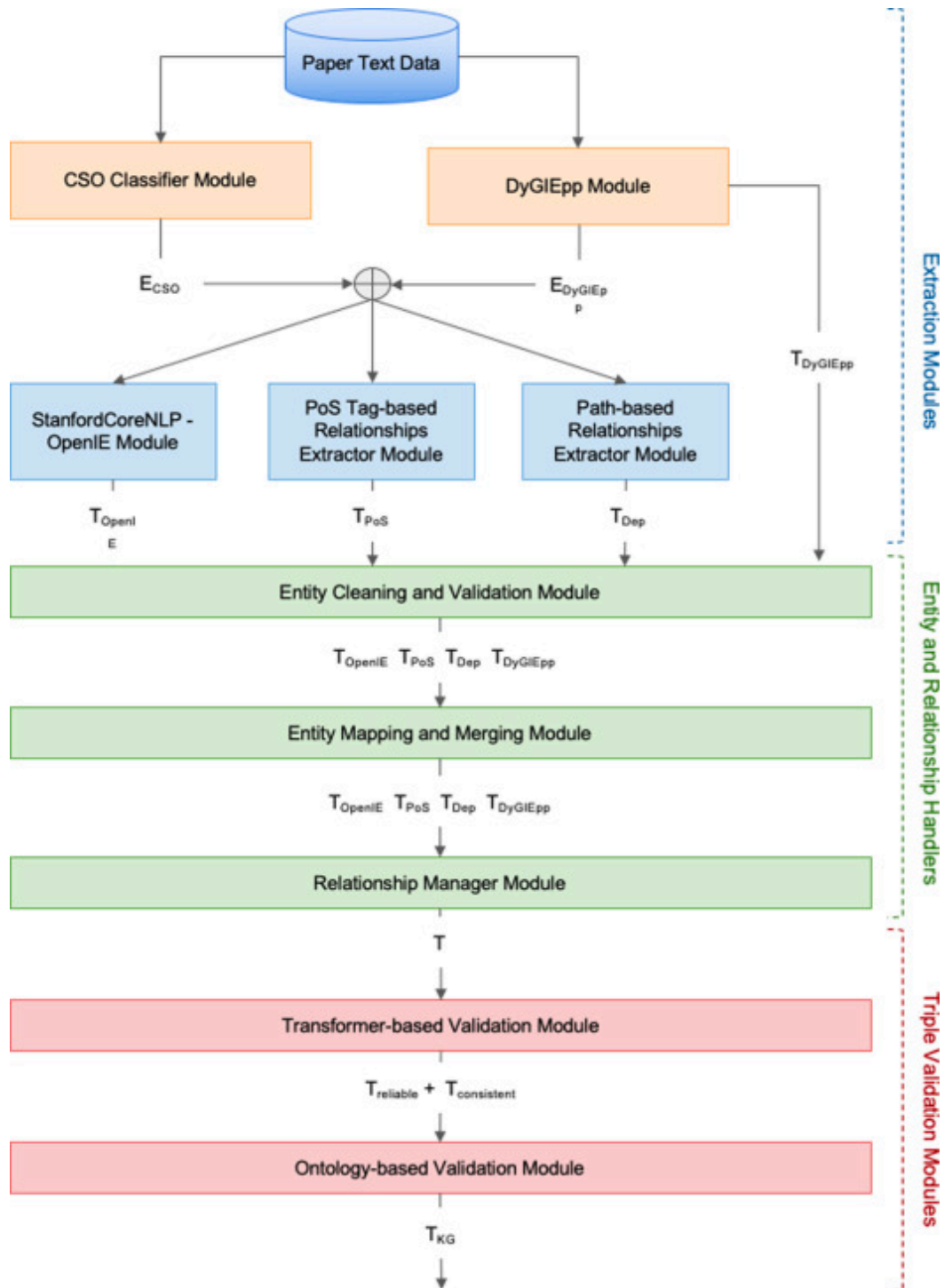
Now, given a new text and its entities, module builds its dependency tree, and uses P'' to extract relationships and the output of the module is set of triples T-DEP

Output : Extractor Module

```
{ "doc_key": "2466206094", "entities": [ ["sensor based swimming status classification", "Task"], ["inherit sensors", "OtherScientificTerm"], ["moving length estimation", "Task"], ["WeSwim", "Method"], ["commercial smartphones", "Material"], ["swimmer locating algorithm", "Method"], ["prototype applications", "Generic"], ["swimming status classification", "Task"], ["SmartSwim", "Method"], ["smartphone", "Material"], ["on-the-ground activities", "OtherScientificTerm"], ["accuracy", "Metric"], ["swimming tracking and localization", "Task"], ["sensors", "CSO Topic"], ["dead reckoning", "CSO Topic"], ["mobile phones", "CSO Topic"], ["localization system", "CSO Topic"], ["swimming", "OtherScientificTerm"], ["smartphone sensors", "OtherScientificTerm"], ["pedestrian dead reckoning PDR concept", "OtherScientificTerm"], ["smartswim", "Method"], ["Error Rate value", "Metric"], ["sports activity monitoring and recognition", "Task"], ["system", "Generic"], ["infrastructure free swimmer localization system", "Method"], ["length estimation", "Task"], ["classification models", "CSO Topic"], ["sensor", "OtherScientificTerm"], ["smart phones", "CSO Topic"], ["SafeSwim", "Method"]], "openie_triples": [ ["dead reckoning", "propose", "swimmer locating algorithm"], ["swimmer locating algorithm", "draw", "dead reckoning"], ["WeSwim", "implement", "localization system"], ["WeSwim", "design", "prototype applications"], ["WeSwim", "implement", "commercial smartphones"], ["pos_triples": [ ["swimmer locating algorithm", "draw", "pedestrian dead reckoning PDR concept"], ["system", "employ", "sensor"], ["swimmer locating algorithm", "draw", "dead reckoning"], ["smartswim", "base", "sensors"], ["commercial smartphones", "design", "WeSwim"], ["sports activity monitoring and recognition", "inherit", "smartphone"], ["smartswim", "base", "smartphone sensors"], ["system", "design", "WeSwim"], ["system", "design", "prototype applications"], ["commercial smartphones", "name", "SafeSwim"], ["system", "employ", "swimming tracking and localization"], ["commercial smartphones", "design", "SafeSwim"], ["commercial smartphones", "design", "prototype applications"], ["dependency_triples": [ ["prototype applications", "name", "WeSwim"]], "dygiepp_triples": [ ["accuracy", "EVALUATE-FOR", "swimming status classification"], ["smartphone", "USED-FOR", "system"], ["Error Rate value", "EVALUATE-FOR", "swimming status classification"], ["WeSwim", "CONJUNCTION", "SafeSwim"], ["WeSwim", "HYPONYM-OF", "prototype applications"], ["SafeSwim", "HYPONYM-OF", "prototype applications"], ["Error Rate value", "EVALUATE-FOR", "length estimation"]]
```


["accuracy", "EVALUATE-FOR", "length estimation"], ["sensor based swimming status classification", "CONJUNCTION", "moving length estimation"], ["system", "USED-FOR", "swimming tracking and localization"], ["commercial smartphones", "USED-FOR", "system"], ["pedestrian dead reckoning PDR concept", "USED-FOR", "swimmer locating algorithm"]]
}

SCIERO - Architecture



2.b) Using REBEL

Rebel (Relation Extraction By End-to-end Language Generation) is a seq2seq model based on BART that performs end-to-end relation extraction for more than 200 different relation types. It provides state-of-the-art results on any kind of text due to its wide range of pre-trained relations. For REBEL, we have as input the text from the dataset and, as output, the linearised triplets. If x is our input sentence and y the result of linearising the relations in x , the task for REBEL is to auto regressively generate y given x :

$$p_{BART}(y \mid x) = \prod_{i=1}^{len(y)} p_{BART}(y_i \mid y_{<i}, x)$$

Let's feed the below input text to our model.

Malaria is a mosquito-borne infectious disease that affects humans and other animals.^{[5][6][3]} Malaria causes symptoms that typically include fever, tiredness, vomiting, and headaches.^{[1][7]} In severe cases, it can cause jaundice, seizures, coma, or death.^[1] Symptoms usually begin ten to fifteen days after being bitten by an infected mosquito.^[3] If not properly treated, people may have recurrences of the disease months later.^[3] In those who have recently survived an infection, reinfection usually causes milder symptoms.^[1] This partial resistance disappears over months to years if the person has no continuing exposure to malaria.^[1]

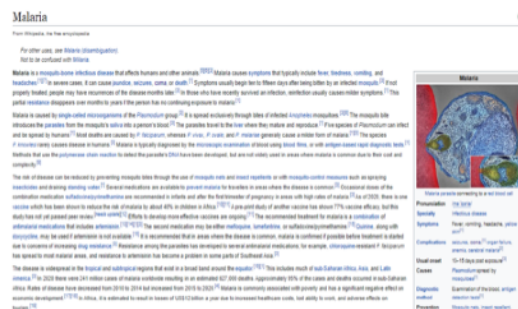
Malaria is caused by single-celled microorganisms of the *Plasmodium* group.^[3] It is spread exclusively through bites of infected *Anopheles* mosquitoes.^{[3][8]} The mosquito bite introduces the parasites from the mosquito's saliva into a person's blood.^[3] The parasites travel to the liver where they mature and reproduce.^[1] Five species of *Plasmodium* can infect and be spread by humans.^[1] Most deaths are caused by *P. falciparum*, whereas *P. vivax*, *P. ovale*, and *P. malariae* generally cause a milder form of malaria.^{[1][3]} The species *P. knowlesi* rarely causes disease in humans.^[3] Malaria is typically diagnosed by the microscopic examination of blood using blood films, or with antigen-based rapid diagnostic tests.^[1] Methods that use the polymerase chain reaction to detect the parasite's DNA have been developed, but are not widely used in areas where malaria is common due to their cost and complexity.^[9]

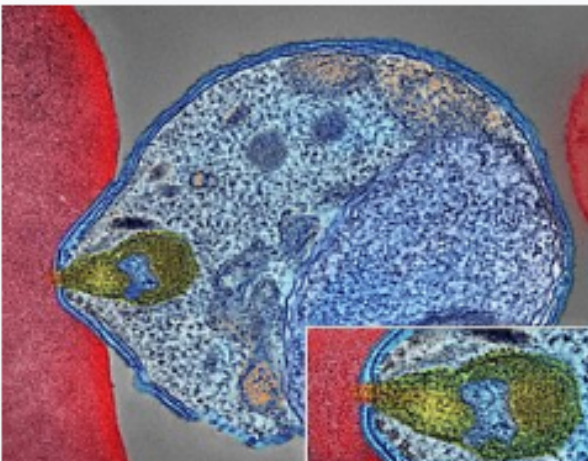
Below are the generated triplets for the above text. REBEL classifies triplets as '**head**'(subject), '**type**'(Relation) and '**tail**'(object) and classifies this as a single triplet.

```
{'head': 'malaria', 'type': 'subclass of', 'tail': 'infectious disease', 'meta': {'spans': [[0, 128]]}}
{'head': 'Malaria', 'type': 'subclass of', 'tail': 'mosquito-borne infectious disease', 'meta': {'spans': [[0, 128]]}}
{'head': 'Malaria', 'type': 'has cause', 'tail': 'Plasmodium', 'meta': {'spans': [[104, 232]]}}
{'head': 'Malaria', 'type': 'has cause', 'tail': 'Anopheles', 'meta': {'spans': [[104, 232]]}}
{'head': 'Anopheles', 'type': 'has effect', 'tail': 'Malaria', 'meta': {'spans': [[104, 232]]}}
{'head': 'Anopheles', 'type': 'has effect', 'tail': 'malaria', 'meta': {'spans': [[104, 232]]}}
{'head': 'Malaria', 'type': 'has cause', 'tail': 'Plasmodium group', 'meta': {'spans': [[104, 232]]}}
{'head': 'Plasmodium group', 'type': 'has effect', 'tail': 'Malaria', 'meta': {'spans': [[104, 232]]}}
{'head': 'malaria', 'type': 'has cause', 'tail': 'P. falciparum', 'meta': {'spans': [[208, 336]]}}
{'head': 'malaria', 'type': 'has cause', 'tail': 'malariae', 'meta': {'spans': [[208, 336]]}}
{'head': 'Malaria', 'type': 'has cause', 'tail': 'P. falciparum', 'meta': {'spans': [[208, 336]]}}
```

2.c) Scraping data from Wikipedia Summary Tables

- For scraping the data, we used Wikipedia table and generated triples.



Malaria	
	
Malaria parasite connecting to a red blood cell	
Pronunciation	/məˈleəriə/
Specialty	Infectious disease
Symptoms	Fever, vomiting, headache, yellow skin ^[1]
Complications	seizures, coma, ^[1] organ failure, anemia, cerebral malaria ^[2]
Usual onset	10–15 days post exposure ^[3]
Causes	<i>Plasmodium</i> spread by mosquitoes ^[1]
Diagnostic method	Examination of the blood, antigen detection tests ^[1]
Prevention	Mosquito nets, insect repellent, mosquito control, medications ^[1]
Medication	Antimalarial medication ^[3]
Frequency	241 million (2020) ^[4]
Deaths	627,000+ (2020) ^[4]

2. The code used is as follows:

```
In [ ]: import pandas as pd
my_table = pd.read_html('https://en.wikipedia.org/wiki/Malaria')
my_table[0]

In [ ]: mal=[]
for i in range(3,len(my_table[0]["Malaria"])-2):
    symptoms = my_table[0]["Malaria.1"][i].split(",")
    # print(symptoms)
    for s in symptoms:
        temp=["Malaria",my_table[0]["Malaria"][i],s]
        mal.append(temp)
print(mal)

In [ ]: |
df = pd.DataFrame(mal)
writer = pd.ExcelWriter('Malaria.xlsx', engine='xlsxwriter')
df.to_excel(writer, sheet_name='welcome', index=False)
writer.save()
```

3. The output is as follows:

```
[['Malaria', 'Specialty', 'Infectious disease'], ['Malaria', 'Symptoms', 'Fever'], ['Malaria', 'Symptoms', 'vomiting'], ['Malaria', 'Symptoms', 'headache'], ['Malaria', 'Symptoms', 'yellow skin[1]'], ['Malaria', 'Complications', 'seizures'], ['Malaria', 'Complications', 'coma'], ['Malaria', 'Complications', '[1] organ failure'], ['Malaria', 'Complications', 'anemia'], ['Malaria', 'Complications', 'cerebral malaria[2]'], ['Malaria', 'Usual onset', '10-15 days post exposure[3]'], ['Malaria', 'Cause s', 'Plasmodium spread by mosquitoes[1]'], ['Malaria', 'Diagnostic method', 'Examination of the blood'], ['Malaria', 'Diagnostic method', 'antigen detection tests[1]'], ['Malaria', 'Prevention', 'Mosquito nets'], ['Malaria', 'Prevention', 'insect repellent'], ['Malaria', 'Prevention', 'mosquito control'], ['Malaria', 'Prevention', 'medications[1]'], ['Malaria', 'Medication', 'Antimalarial medication[3]']]
```

4. The code for generating graph:

```
import networkx as nx
import matplotlib.pyplot as plt

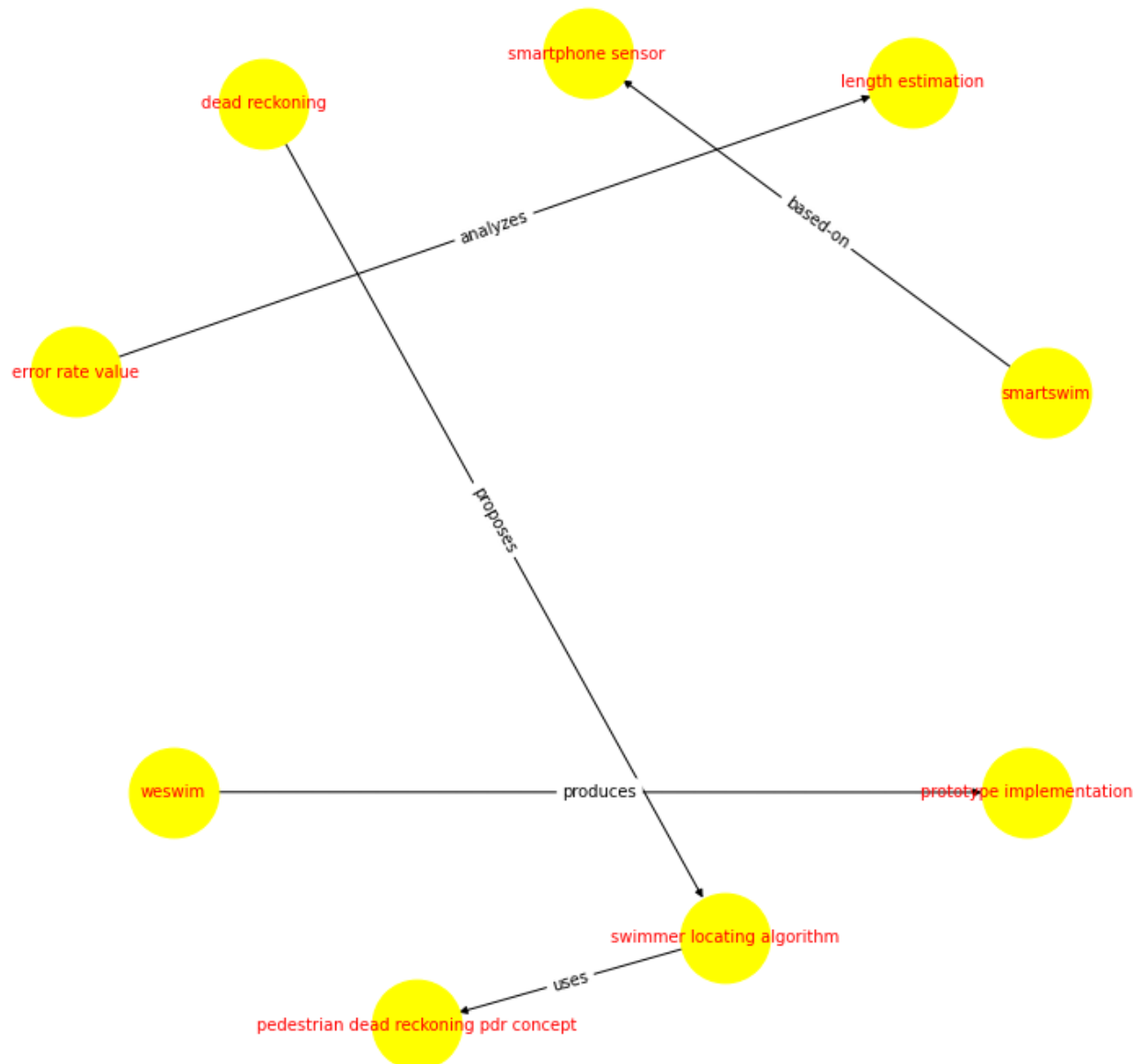
g = nx.Graph()
plt.figure(figsize=(15,15))
pos = nx.spring_layout(g, k=5)
for (die,cat,data) in mal:
    g.add_edge(die,cat)
    g.add_edge(cat,data)
nx.draw(g, with_labels = True, node_size = 3000, node_color="yellow")
plt.savefig("Malaria.JPEG")
```

3. Results

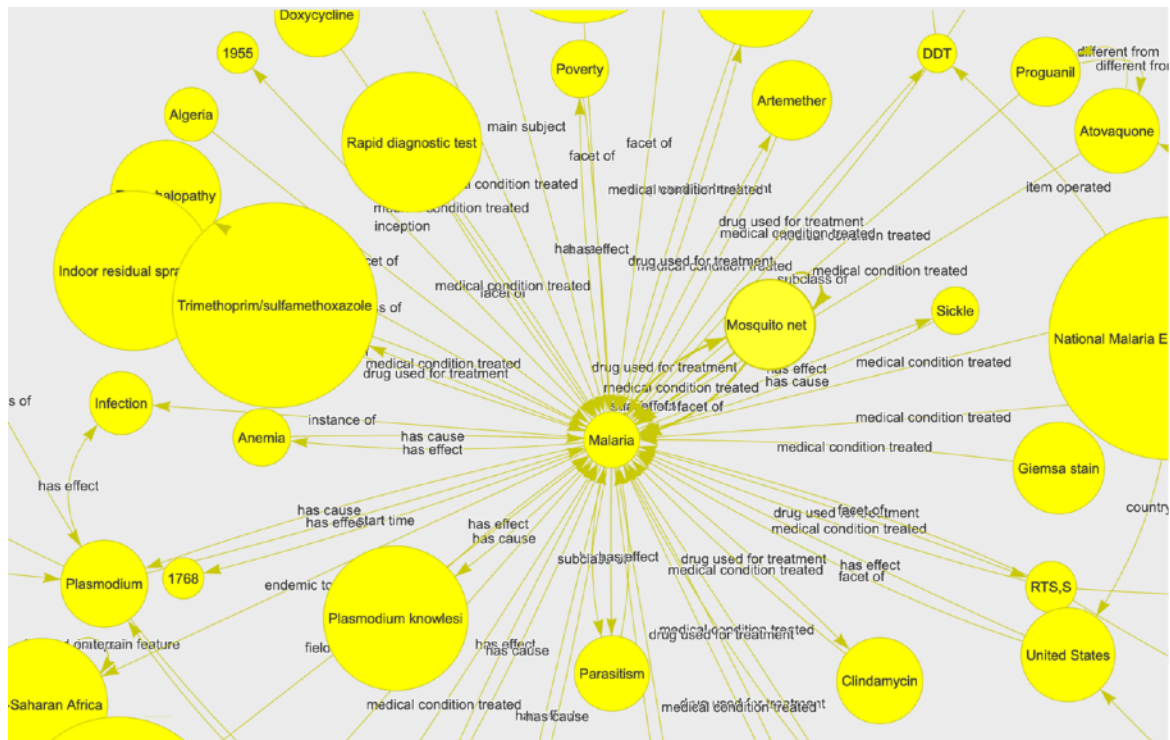
3.1 Methodology-1 Results

my_new_kg_final_data

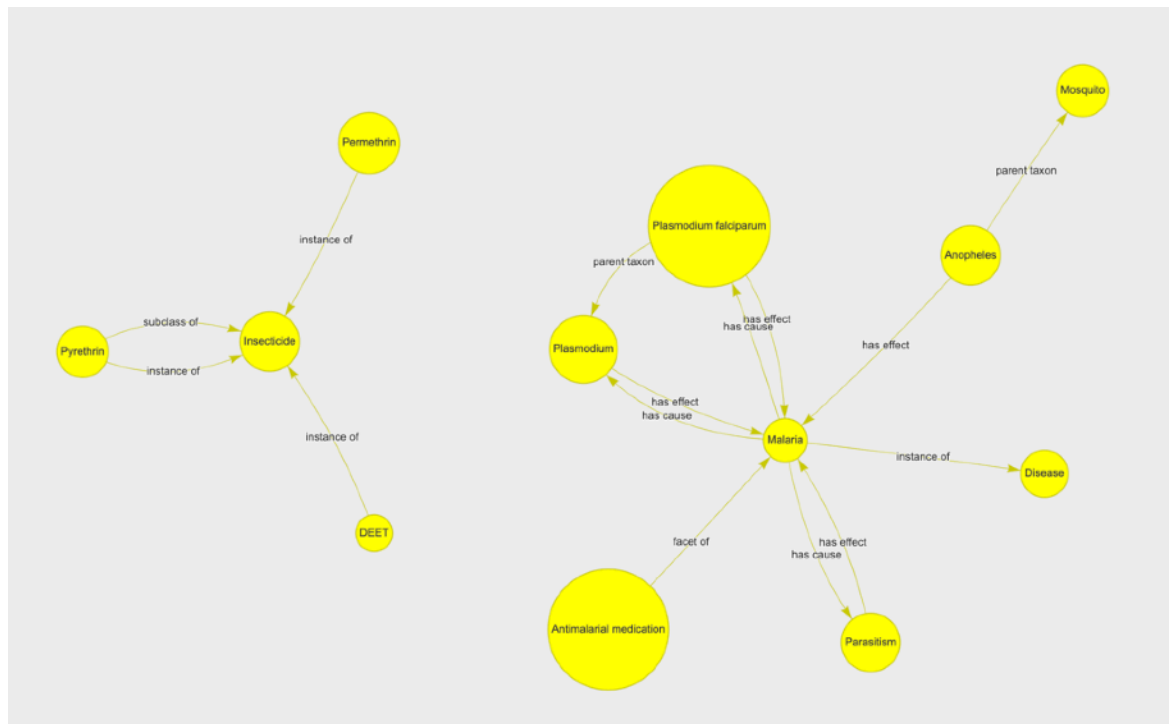
subj	rel	obj	support	sources	files	subj_type	obj_type
swimmer locating algorithm	uses	pedestrian dead reckoning pdr concept	1	{'dygiepp'}	{'2466206094'}	Method	OtherEntity
dead reckoning	proposes	swimmer locating algorithm	1	{'openle'}	{'2466206094'}	OtherEntity	Method
error rate value	analyzes	length estimation	1	{'dygiepp'}	{'2466206094'}	Metric	Task
weswim	produces	prototype implementation	1	{'openle'}	{'2466206094'}	Method	OtherEntity
smartswim	based-on	smartphone sensor	1	{'pos tagger'}	{'2466206094'}	Method	OtherEntity



3.2 Methodology-2 Results



Malaria - Wikipedia KG



Malaria - URM C KG3.3 Methodology-3 Results

3.3 Methodology-3 Results



Dengu Fever KG



Hepatitis B KG

3. Conclusions

With the three methodologies discussed above, it was found that the easiest of them is methodology 3 i.e. scrapping data from the Wikipedia tables, that is already in the structured format, however, this method doesn't work for any generic text also, the resulting triplets are very limited. Whereas the methodology 1, SCIERO is a very sophisticated system, that would generate very good results out of all the methods tried, but it very resource and time consuming. As of now the whole pipeline is set up just to extract the entities and relationships in the computer science domain and there needs a significant amount work and time to accommodate this pipeline to extract diseases knowledge graphs.

So, methodology 2, using REBEL, is the sweet spot that gives you the entities and relations in least amount of effort considering the generalised text it can go through. Although, the control over what entities to detect and the relationship span is fixed with the retrained model, it gave pretty good results out of the box.

Reference

D. Dessí, F. Osborne, D.R. Recupero et al., SCICERO: A deep learning and NLP approach for generating scientific knowledge graphs in the computer science domain, Knowledge-Based Systems (2022), doi: <https://doi.org/10.1016/j.knosys.2022.109945>

Git Repository

<https://github.com/danilo-dessi/SKG-pipeline>

Project Results, Python Notebooks

https://iitgoffice-my.sharepoint.com/:f:/g/personal/b_atul_iitg_ac_in/EnfLel6itJINvBLfVqZGnvkB6zEcBNLM-aE3liY68S_Ndw?e=P0OsQ6