DBpedia Hindi Chapter

Mentors: Sanju Tiwari, Ronak Panchal, Ananya Hooda

Personal information

- Name: Debarghya Datta
- GitHub User ID: @deba-iitbh
- Email: debarghyad@iitbhilai.ac.in
- Phone: +91 xxxxxxxxxx
- Location: Kolkata, India
- LinkedIn: https://www.linkedin.com/in/deba-iitbh/
- College: Indian Institute of Technology, Bhilai, Chhattisgarh, India
- Discipline: Computer Science
- Year: 2nd
- CV: https://d2atta.github.io/cv.pdf

Project

- ## Problem description:

  DBpedia, a dynamic open knowledge graph, undergoes continuous evolution. In contrast to Wikidata, where RDF content is directly edited like a wiki, DBpedia solely relies on Wikipedia. This means that every triple in DBpedia can be traced back to some infobox, sentence, or table cell in Wikipedia. Currently, the existing knowledge graph is derived from the English version of Wikidata (e.g., https://dbpedia.org/page/India).

  Our objective now is to construct a knowledge graph (KG) exclusively from Hindi Wikipedia. This would enable us to utilize the generated KG directly using the "Devanagari" script (e.g., https://hi.dbpedia.org/page/भारत).

  Current methods for generating triples typically depend on the Extraction Framework for infobox extraction or utilize novel NLP-based approaches such as the Neural Extraction Framework. However, these frameworks primarily support the English language. Hence, there's a need to extend the Neural Extraction Framework to encompass multilingual text, particularly Hindi.

  So, as part of the project, we want to achieve the following:

  1. Create the knowledge graph with data from Hindi Wikipedia.
  2. Expose the knowledge graph to make it browsable via web(hi.dbpedia.org).
  3. Create a SPARQL endpoint to make it queryable

  ### Impact

  - **Cultural and Educational Enrichment**: Languages carry the traditions, history, and identity of a community. Multilingual KGs can be used to store and share cultural information in local languages. This will help empower Hindi-speaking users with culturally relevant and easily accessible knowledge, fostering educational enrichment and linguistic inclusivity.
  - **Fostering Intercultural Understanding**: Multilingual KGs can facilitate communication and understanding between different cultures. By enabling people to access information about other cultures in their own languages, It can break down language barriers and promote empathy and respect.
  - **Semantic Search and NLP Applications**: Hindi knowledge graphs can be used to create more accurate and relevant search results and recommendations for people who use the internet in Hindi. Enabling advanced semantic search and natural language processing (NLP) applications in Hindi, opening avenues for innovation in information retrieval and analysis. This can be particularly beneficial for areas like education, healthcare, and e-commerce.

- **Community Engagement**: Encourage community contributions, feedback, and collaboration in maintaining and expanding the Hindi ontology, ensuring continuous improvement and relevance

Basic Ideology:

Based on the warm-up task, I have covered the following concepts:

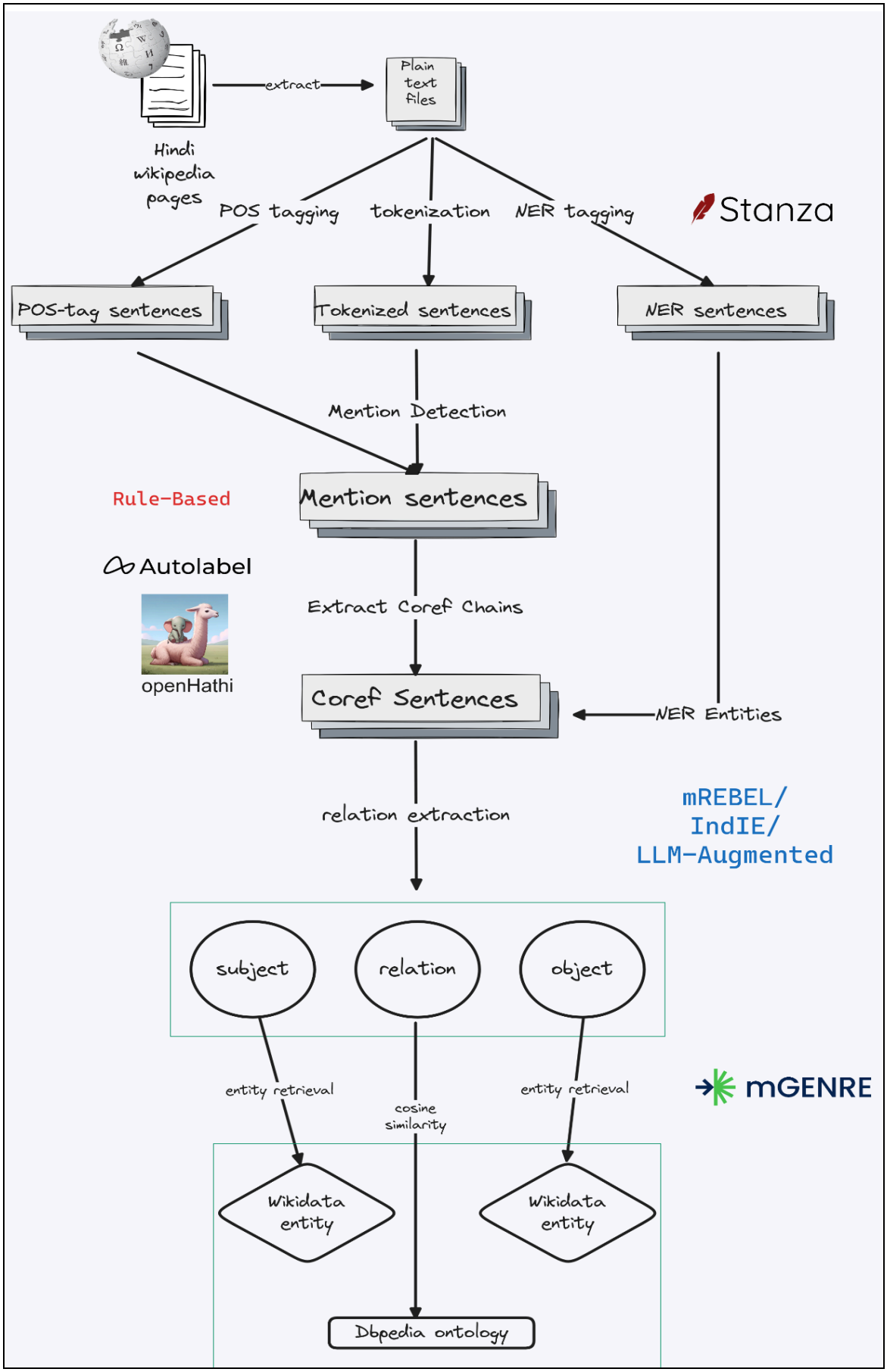
- overview on creating new DBpedia Chapters.
- DBpedia – A large-scale, multilingual knowledge base paper (2015).
- Internationalization of Linked Data: The case of the Greek DBpedia edition by Kontokostas et al.
- Building a Japanese DBpedia Chapter paper (2014).
- A Step towards the Arabic DBpedia paper (2013).
- Explored tutorial to use SPARQL query. [Tutorial]
- IndIE: A Multilingual Open Information Extraction Tool For Indic Languages by Ritwik et al (Relation Extraction for Hindi).
- REDFM: a Filtered and Multilingual Relation Extraction Dataset by Pere-Lluís et al (Relation Extraction).
- Multilingual Autoregressive Entity Linking by De et al (Entity Linking).
- HindiBenchIE - For evaluating the triple extraction quality
- The mapping in Hindi of the DBpedia ontology.
- SPARQL query format for the DBpedia endpoint.
- local DBpedia Virtuoso endpoint (For running SPARQL queries locally).

Also, explored the approach and codes for the following repositories:

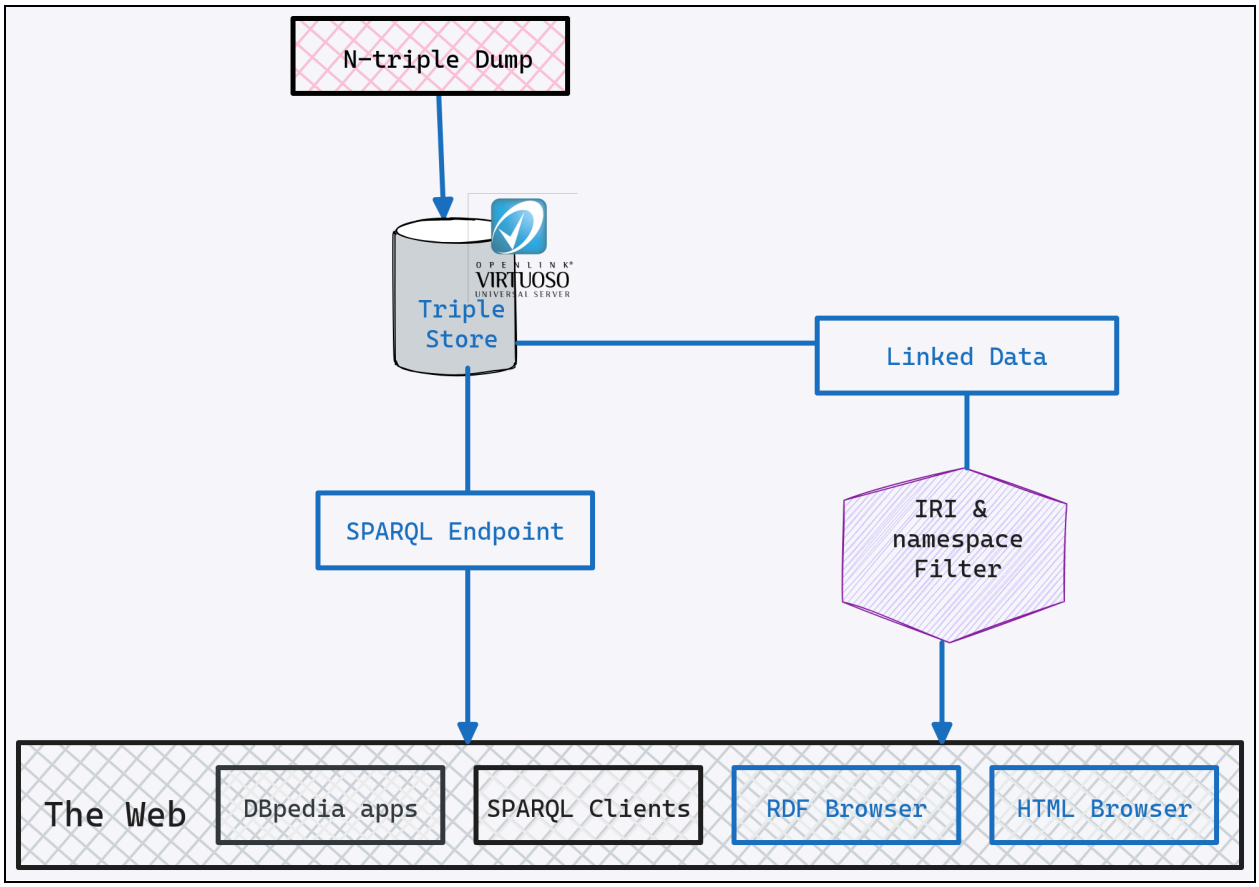
- DBpedia Extraction Framework (software used to transform Wikipedia infobox data into RDF triples)
- Neural Extraction Framework (software used to extract information from the english wikipedia articles)

Objective

1. Creation of Hindi Knowledge Graph from the whole Wikipedia Article(Including Infoboxes) using the given pipeline:
    i. Navigating to Wikipedia Page(language: hindi)
    ii. Extract the plain text using Wikipedia
    iii. Tokenization, POS tagging and NER using stanza (Stanford NLP)
    iv. Coreference resolution using an LLM (Prompt-based)
    v. Relation extraction from the coref-sentences (get entities and relations)
    vi. Get the DBpedia relation-URI form the ontology
    vii. Get the entity-URIs of the predicted entities using mGENRE (Wikipedia Article name)
    viii. Aggregate the triples (Converting each entity id to DBpedia URI)
    ix. Validate the triples (Check for any discrepancy)
2. Creation of Hindi RDF Collection
    i. Process the triples to include the type information of the entities
    ii. Convert the processed triples into RDF
    iii. Store the RDF in turtle(.ttl) file
3. Deploy using SPARQL endpoint
    i. Load the RDF into the virtuosa client
    ii. Test the SPARQL endpoint at `localhost:8890/sparql`



Pipeline for the Triple Extraction from Wikipedia Articles



Deployment Architecture for Hindi DBpedia

- Description of Approach:

Extracting pages from Wikipedia

In the Kaggle datasets, we have a cleaned Hindi Wikipedia dump, Hindi Wikipedia Articles - 172k | Kaggle, but as they are old, we may need to again extract the updated articles from wikipedia. For that, we can use the "wikipedia" package to extract any page from the wikipedia in any language. For our use-case, we will select Hindi (Language code: hi) and extract each page from the wikipedia.

Sample text from hi_wiki_dump:

```
भारत (आधिकारिक नाम: भारत गणराज्य, अंग्रेज़ी: Republic of India, लिप्यन्तरण: रिपब्लिक ऑफ़ इंडिया) दक्षिण एशिया में स्थित भारतीय उपमहाद्वीप का सबसे बड़ा देश है।
```

Tokenization, POS Tagging and NER of Hindi sentences

For tokenizing and POS tagging the sentences, we can use the SOTA multilingual tokenization "stanza" library from stanfordNLP, which has support for Hindi language. It uses the Hindi Universal Dependency TreeBank(HDTB).The HDTB is a part of an

ongoing effort of creating multi-layered treebanks for Hindi and Urdu developed at IIIT-H India. It has a 80+ MLAS (morphology-aware labeled attachment score), making it a fairly reliable model for our experimentation.

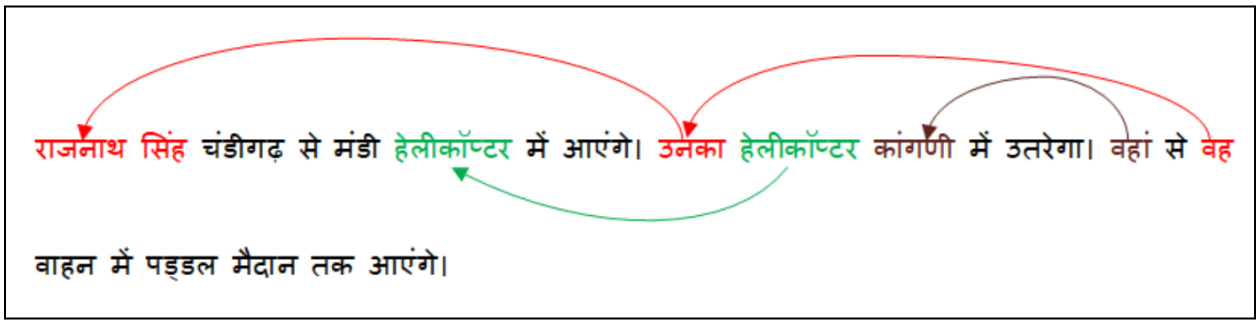
Though stanza doesn't support Hindi-NER officially, but we have a off-the shelf Hindi-NER system by Computation for Indian Language Technology (CFILT) called HiNER, which can be integrated with the stanza library to provide NED tags for the sentences. Mainly these named entities are extracted:

1. Names
2. Location
3. Organization
4. Language
5. Game
6. Literature
7. Religion
8. Festival
9. Miscellaneous

The authors of the paper have also released the pretrained model in Huggingface.
A sample of the tokenization by the stanza library is given below.

```
====== Sentence tokens =======
id: (1,)        text: भारत
id: (2,)        text: (
id: (3,)        text: आधिकारिक
id: (4,)        text: नाम
id: (5,)        text: :
id: (6,)        text: भारत
id: (7,)        text: गणराज्य
id: (8,)        text: ,
id: (9,)        text: अंग्रेज़ी
id: (10,)       text: :
id: (11,)       text: Republic
id: (12,)       text: of
id: (13,)       text: India
id: (14,)       text: ,
id: (15,)       text: लिप्यन्तरण
id: (16,)       text: :
id: (17,)       text: रिपब्लिक
id: (18,)       text: ऑफ़
id: (19,)       text: इंडिया
id: (20,)       text: )
id: (21,)       text: दक्षिण
id: (22,)       text: एशिया
id: (23,)       text: में
id: (24,)       text: स्थित
id: (25,)       text: भारतीय
id: (26,)       text: उपमहाद्वीप
id: (27,)       text: का
id: (28,)       text: सबसे
id: (29,)       text: बड़ा
id: (30,)       text: देश
id: (31,)       text: है
id: (32,)       text: |
```

Coreference resolution for the Tokenized sentences



Example of Coreference Resolution

Coreference resolution is a natural language processing task that involves identifying all the expressions in a text that refer to the same entity. This task is crucial for understanding the relationships between entities mentioned in a text. It helps in constructing a coherent representation of the text by replacing pronouns and other referring expressions with their appropriate antecedents. Coreference resolution algorithms typically involve machine learning techniques that analyze linguistic features such as syntactic structure, semantic similarity, and contextual cues to determine coreference relationships.
Significant progress has been seen in the field of Coreference Resolution over the past years. With the help of the Rule-based approach, Statistical-based approach, and Neural Networks-based, several innovative ideas have been proposed by researchers that have shown great success, mainly in the English language. However, in Hindi, less amount of research work is reported in the literature. Hindi is a morphologically rich language, with free word order and ambiguity exhibited in Hindi pronouns as explained by Coreference Annotation Scheme and Relation Types for Hindi. Due to these characteristics of the Hindi language, the CR task is more complicated. There is a need for syntactic, semantic, discourse, morphological, and lexical understanding to solve this challenge.

In our scenario, we can employ Airavata, a Hindi Large Language Model (LLM), which is a 7B OpenHathi model fine-tuned on the IndicInstruct dataset. This dataset comprises various instructional datasets like Anudesh, wikiHow, Flan v2, Dolly, Anthropic-HHH, OpenAssistant v1, and LymSys-Chat. Airavata is readily accessible via Huggingface. To accomplish our task, we can utilize both Zero-shot and Few-shot prompting techniques.

Since coreference resolution necessitates explicit marking of mentions, we propose employing a rule-based mention detector. This detector will label all instances of a mention using markdown-tag syntax "[mention](#)", enabling us to use them as prompts for Airavata. Mentions conveying similar meanings are linked within the same cluster.

After obtaining responses from Airavata, we can process them to identify unique clusters and assign a single Named Entity per cluster, resulting in coreferenced sentences.

Prompt:

```
Annotate all entity mentions in the following sentences separated by newline with
coreference clusters. Use Markdown tags to indicate clusters in the output, with the
following format [mention](#cluster_name)

[ब्रिटिश](#) [कोलम्बिया](#) , ( [अंग्रेज़ी](#) : [British](#) [Columbia](#) , [फ्राँसीसी:](#) a
[Colombi](#) [e-Britannique](#) ) [कनाडा](#) का एक प्रान्त है [जो](#) [कनाडा](#) के
[प्रशान्त](#) [महासागर](#) से लगते पश्चिमी तट पर स्थित है।

[यह](#) [कनाडा](#) का तीसरा सबसे बड़ा प्रान्त है [जिसका](#) क्षेत्रफल ९,९४,७३५ वर्ग किमी है।

[२००६](#) की जनगणना के अनुसार [इस](#) प्रान्त की कुल जनसंख्या ४१,१३,४८७ थी।

[इस](#) प्रान्त की राजधानी [विक्टोरिया](#) है और राज्य का सबसे बड़ा नगर [वैंकूवर](#) है।

[इसी](#) नगर में ब्रिटिश [कोलम्बिया](#) की लगभग आधी जनसंख्या निवास करती है ( २० लाख ) ।

अन्य बड़े नगर हैं : [केलोव्ना](#) , [अबोट्स्फोर्ड](#) , [कैम्लूप्स](#) , [नानाइमो](#) और [प्रिन्स](#)
[जॉर्ज](#)।
```

Output:

```
[ब्रिटिश](#1) [कोलम्बिया](#1) , ( [अंग्रेज़ी](#1) : [British](#1) [Columbia](#1) ,
[फ्राँसीसी:](#1) a [Colombi](#1) [e-Britannique](#1) ) [कनाडा](#2) का एक प्रान्त है
[जो](#2) [कनाडा](#2) के [प्रशान्त](#3) [महासागर](#3) से लगते पश्चिमी तट पर स्थित है।

[यह](#2) [कनाडा](#2) का तीसरा सबसे बड़ा प्रान्त है [जिसका](#2) क्षेत्रफल ९,९४,७३५ वर्ग किमी है।

[२००६](#4) की जनगणना के अनुसार [इस](#2) प्रान्त की कुल जनसंख्या ४१,१३,४८७ थी।

[इस](#2) प्रान्त की राजधानी [विक्टोरिया](#5) है और राज्य का सबसे बड़ा नगर [वैंकूवर](#6) है।

[इसी](#6) नगर में [ब्रिटिश](#1) [कोलम्बिया](#1) की लगभग आधी जनसंख्या निवास करती है ( २०
लाख )।

अन्य बड़े नगर हैं : [केलोव्ना](#7) , [अबोट्स्फोर्ड](#8) , [कैम्लूप्स](#9) , [नानाइमो](#10) और
[प्रिन्स](#11) [जॉर्ज](#11)।
```

Triple Extraction/Relation Extraction From the sentences

Relation extraction is the process of extracting information from a sentence, where X = {x1, x2, · · · , xn} consists of tokens xi. We will establish a function that maps X to a set Y

= {y1, y2, · · · , yj}, wherein each element is a tuple yi = {reli, arg1i, arg2i} encapsulating the information conveyed in sentence X.

As the triple extraction step is the most essential step of generating a Knowledge base, thus, I have listed three different approaches, with increasing complexity.

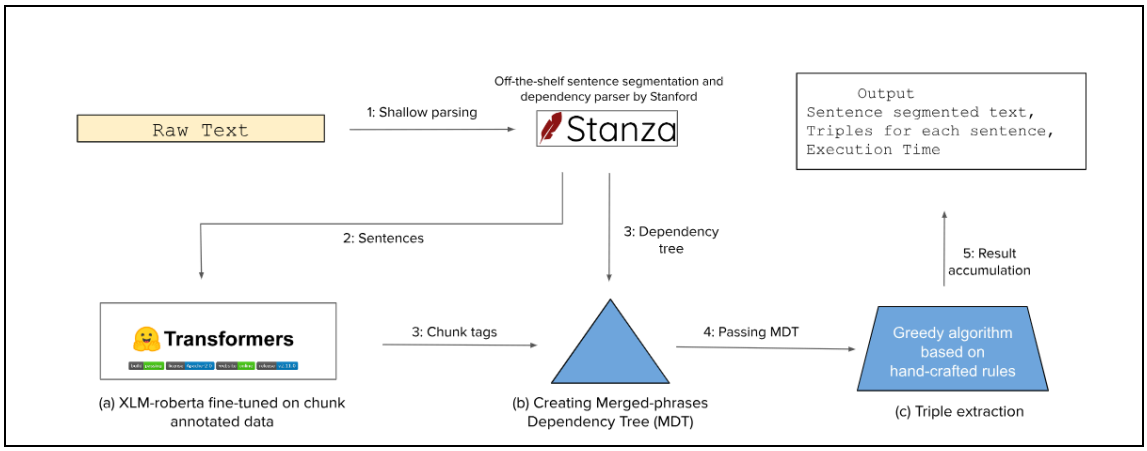
1. Rule-Based (IndIE)
2. Transformer-Based (mREBEL)
3. LLM-based (Prompting)

mREBEL (Multilingual Relation Extraction)

It has been extensively studied in the previous iteration of GSoC'23 under the project of Neural Extraction Framework. So, we will be using the same framework by just changing the base model to its multilingual counterpart, mREBEL. Although it has shown remarkable performance in various languages, still the precision for Hindi is only 8.3 Macro-F1 in its test set. Thus, during our experimentation, if we don't get satisfying results, we can also check out the work "A Data Bootstrapping Recipe for LowResource Multilingual Relation Classification", which showed >80 Macro-F1 metric.

Rule-based Relation Extraction

IndIE is a joint triple extraction tool for Indian languages, it performs chunking, creates a Merged-phrase Dependency Tree (MDT), and generates triples using hand-crafted rules. It also uses a fine-tuned transformer-based chunker, as it outperforms other traditional methods of chunking. The tool showed that the IndIE generates more informative and fine-grained triples than other traditional baselines.



Pipeline of IndIE

Although, some of the limitations discussed are given below:

1. Inability to identify overlapping predicates. For example, consider the following sentence "Nehru became the prime minister of India in 1947". Depending on the level of detail (granularity) in triples, two predicates that can be extracted among numerous possible predicates are "became" and "became the prime minister".

2. The utilization of hand-crafted rules in the triple extraction process imposes constraints on the scalability and versatility of the pipeline.

```
Sentence: मोहाली का मोहन एक अच्छा लड़का है ।
['मोहन', 'है', 'एक अच्छा लड़का']
['मोहन', 'है', 'मोहाली का']
```

Example of Triple Generation

LLM-augmented Triple Extraction

The potential of Large Language Models, in the context of Information Extraction has been utilized in several papers[openie_portg, doc_ie] and showed a significant improvement in improving the task at hand using the inherent knowledge in these pretrained Language Models.

In this technique, prompts are carefully crafted textual cues provided to the language model to guide it in generating desired outputs. These prompts serve as structured queries, instructing the model on what information to extract and how to structure it into triples. Some of the openly available LLMs which can be used are listed below.

Name	Model Size
LLaMA (Non-commercial)	65B
Falcon (Apache 2)	40B
LLaMa-2 (Non-commercial)	7/70B
Airavata (Non-Commercial)	7B

An example prompt, adapted from recent paper[1] is also given below.

```
Some text is provided below. Extract up to {max_triplets} knowledge
triplets in the form (subject, predicate, object) from the text.
-----------------------------------------------------------------
Examples:
Text: एबिलीन, टेक्सास संयुक्त राज्य अमेरिका में है।
Triplets:
(एबिलीन टेक्सास, देश, संयुक्त राज्य अमेरिका)

Text: संयुक्त राज्य अमेरिका में अफ्रीकी जातीय समूह शामिल है
अमेरिकी और अब्राहम ए रिबिकॉफ़ का जन्मस्थान है
जिन्होंने केसी रिबिकॉफ़ से शादी की है।
```

[1] https://arxiv.org/pdf/2312.01954.pdf

```
Triplets:
(अब्राहम ए. रिबिकॉफ, जीवनसाथी, केसी रिबिकॉफ)
(अब्राहम ए. रिबिकॉफ, जन्म स्थान, संयुक्त राज्य अमेरिका)
(संयुक्त राज्य अमेरिका, जातीय समूह, अफ़्रीकी अमेरिकी)
--------------------------------------------------------------------------
Triplet Extraction Prompt
Text: {text}
Triplets:
```

[openie_portg] https://aclanthology.org/2024.propor-1.13.pdf
[doc_ie] https://arxiv.org/pdf/2401.13598.pdf

Automating LLM Labeling

We have the capability to automate document labeling through a straightforward tool called "Autolabel." This tool labels sentences within documents using a Large Language Model (LLM) by following these steps:

1. Define labeling guidelines and specify the LLM model to be used in a JSON configuration.
2. Conduct a dry run to ensure that the final prompt appears satisfactory.
3. Execute the labeling script.

This tool is integral to our experiment, as we rely on the LLM for annotating substantial amounts of data, encompassing both coreference resolution and Relation Extraction tasks.

Entity Linking for the extracted triples

Similar to the Neural Extraction Framework, I will use the multilingual version of the base Entity Linking model, mGENRE. Which gave significant improvements over the alias table, showing that it is able to understand syntactic as well as semantic meanings of the mentions.

```
{
    "फिल्म महोत्सव" → Mohabbatein (Q1135349)
    "प्रकाश झा" → Prakash Jha (Q3401298)
}
```

Example of Entity linking to Wikidata

Given the limitations identified in the GENRE model, I plan to delve deeper into alternative methods for entity linking throughout the duration of the GSoC project.

Finding the closest relationship type from the DBpedia Ontology

Since there isn't a model specifically trained to recognize the Ontology within the generated relations, we'll utilize a multilingual sentence transformer, such as Hindi-sentence-similarity-sbert as mentioned in the paper. This transformer maps sentences into a dense vector representation of 768 dimensions. Consequently, we can compute the cosine similarity between the embeddings to rank the likely relations.

Sentence similarity using Sentence transformer

Creation of Hindi RDF Collection

1. Process Triples
    a. Read the list of (subject, predicate, object) triples.
    b. Identify the types of entities involved (if not provided).
    c. Include type information for each entity in the triples.

2. Convert Triples to RDF
    a. Define appropriate namespaces for the RDF serialization.
    b. Create an RDF graph.
    c. Iterate over the processed triples.
    d. Convert subjects, predicates, and objects to RDF resources or literals.

e. Add triples to the RDF graph.

3. Serialize RDF to Turtle Format
    a. Serialize the RDF graph to Turtle format (.ttl file).
    b. Ensure the file adheres to the proper Turtle syntax.
    c. Save the Turtle file containing RDF data.

Deploy using SPARQL endpoint

1. Load RDF into Virtuoso Client
    a. Connect to the Virtuoso database.
    b. Prepare the Virtuoso client for data upload.
    c. Upload the Turtle file containing RDF data into the Virtuoso database.

2. Test SPARQL Endpoint
    a. Access the SPARQL endpoint provided by the Virtuoso server.
    b. Ensure the SPARQL endpoint is configured and running correctly.
    c. Perform test queries against the SPARQL endpoint to verify functionality.
    d. Use a tool such as cURL or a web browser to query the endpoint.
    e. Verify that the SPARQL endpoint returns the expected results.

Example SPARQL Query:

```
# SPARQL query to extract the names of administrative Regions of
India
PREFIX dbpedia-owl: <http://hi.dbpedia.org/ontology/>

SELECT DISTINCT ?stateName
WHERE {
  ?state rdf:type dbpedia-owl:AdministrativeRegion .
  ?state dbpedia-owl:country dbr:भारत .
  ?state rdfs:label ?stateName .
}
```

Output:

```
-----------------------------------------
| stateName                             |
=========================================
|आंध्र_प्रदेश  |
|अरुणाचल_प्रदेश |
|असम           |
|बिहार         |
|छत्तीसगढ़      |
|गोवा          |
```

```
|गुजरात          |
|हरियाणा         |
|हिमाचल_प्रदेश  |
|जम्मू_और_कश्मीर |
...
-----------------------------------------
```

- Evaluation:

  Since we lack a standardized benchmark to gauge the effectiveness of our methodologies, we can resort to human evaluation. This entails randomly selecting 100 triples and scrutinizing whether they are pertinent and correctly associated with the respective entities. While this evaluation method isn't flawless, it provides insights into the level of disparity present in our approach.

- Timeline:
  - **May 1 - 26: (***Community bonding period*)
    - Keep in touch with mentors to clearly understand the implementation details.
    - Explore utilities to use the LLM for annotating the corpus.
    - Will discuss with my peers, to explore more options and approaches that can be tried to solve the problem.
  - **May 27 - 31:** (*Week 1*)
    - Creating a new Wikipedia dump, based on the previous Hindi dump by scraping the hi.wikipedia pages.
    - Add the Hindi-NER model in the `stanza` library
    - Update the Documentation with the progress
  - **June 3 - 7:** (Week 2)
    - Write rules for the “Mention Detection” module using the POS tags, to get the mentioned entities in the sentence.
    - Find the Coreference clusters for the mentioned entities in the sentence.
    - Explore different LLMs for better performance
  - **June 10 - 21:** (Week 3)
    - Transform the sentence with the “coref” and “ner” information
    - Automate the process for all the sentences using the `Autolabel` library

- **June 24 - 28:** (Week 4)
  - Explore the `IndIE` codebase to adapt it to our use-case.
  - Examine the extracted relations and the relation types.
- **July 1 - 5:** (Week 5)
  - Explore the `mREBEL` model to extract the triples.
  - Analyze the performance of the model compared to the previous Rule-based model
- **July 8 - 12 :** (Week 6)
  - Document the approaches, and fix any issues.
  - Compare the Rule-based and Transformer-based approaches of relation extraction.
- **July 12: Midterm evaluation due**
- **July 15 - August 2:** (Week 7)
  - Experimentation with the LLM-augmented Relation Extractor.
  - Automate the labeling process using `Autolabel` library
- **August 5 - 9 :** (Week 8)
  - Develop the ontology mapper, which maps the generated relation to one of the DBpedia ontologies.
  - Use the `mGENRE` model to get the wikidata id of the generated subject and object entities.
  - Convert the Wikidata ID to DBpedia ID using the existing mapping
- **August 12 - 20:** (Week 9)
  - Verify the generated triples.
  - Convert the triples into RDF format
- **August 21 - 23:** (Week 10**)**
  - Load the RDF into a virtuoso client and check the SPARQL endpoint.
  - Discuss with the mentors for system performance improvements.
- **August 26: Final evaluation due**

Note: I will also be blogging weekly through my personal website and updating my mentors after every 2-3 days about my progress during the program. The timeline might seem crude as at this moment I am not able to judge how much time each work is going to take. I will be proceeding accordingly.

Future Aspects:

Based on the outcome of this approach, I want to publish my work in a reputed conference along with the mentors of this project. As there are many Indic languages

which need to be attained, I am planning to help bring other chapters on board, with the help of the mentors of DBpedia.

- ## Why I choose this idea and DBpedia organization:

  As, during my Master's, I have some experience on working on Relation Extraction for Knowledge Graph, thus this project intrigued me. As with this project, I can learn the intricacies of handling multilingual data as well as use the knowledge gained from my thesis.

  Additionally, the concept of unraveling intricate inquiries through a straightforward graph featuring diverse real-world entities as nodes is not only intriguing but also seems almost magical. It presents a myriad of new possibilities. DBpedia has played a crucial role in the open-source Knowledge Graph landscape by furnishing a vast, multilingual knowledge repository that adheres to open standards, enabling smooth integration with various data sources. Therefore, I believe that making a meaningful contribution to the open-source Knowledge Graph community can be effectively realized through collaboration with DBpedia.

- ## Open Source:

  I have not participated in Open-Source projects before, but am a daily user of Open-source softwares, like Linux, DWM(Window Manager), NeoVim, Emacs, etc. I share my work with the community by regularly sharing different projects on my Github.

- ## Background and education:

  I have a Bachelor's degree in Computer Science, where I learned the basics of CS, like Operating Systems, Databases, Algorithms, etc., and used them to create a viable product during a 2 months internship at HighRadius. After that, I joined TCS as an End2End Quality Assurance Engineer, testing different execution flows and automating them for increased productivity. During my 1 year tenure at TCS, I learned the challenges of developing a real-world facing product, and how to make services reliable. To widen my knowledge, I joined the premiere institute of India, IIT(Bhilai) for my post-graduation, and explored the topics like Machine Learning, Natural Language Processing, Information Retrieval, etc.

- ## Research:

  My research started when I joined my Masters under Dr. Soumajit Pramanik, working on Medical Question Answering with the help of a Medical Knowledge Graph(MedKG). I explored the different steps in a Question Answering system, like tokenization, entity extraction, intent classification, etc. I have also worked on Entity Disambiguation during

that time, as in medical context, the frequency of colloquial terms are much more than other domains, along with insufficient training data. Thus forcing us to look into zero-shot or unsupervised ways of dealing with that problem.

Some of the projects, that I have worked:

1. Query Summarized Attention for TREC-CT 2022 [code]
    a. Used BM25 to retrieve the top 1000 Documents
    b. Created a Key-Word Summarizer module to create Query summaries
    c. Used BERT-Reranker to rank the retrieved documents
    d. Able to outperform the published models in 2/3 metrics
2. Information Extraction from Scientific Papers [code]
    a. Extracting contributions from Scientific Papers
    b. Replaced Relation extraction model Dygie++ with PL-Maker
    c. Improved task extraction precision by 33%
    d. Improved method extraction precision by 23%
3. Exploring the Realm of Drug-Drug Interaction [Submitted Paper]
    a. Predicting the Drug-Drug Interaction using Deep Learning
    b. Explored different Deep-graph models
        i. Graph Convolutional Network(GCN)
        ii. Relational Graph Convolutional Network(R-GCN)
4. Unsupervised Entity Linking for Low Resource Domains [Submitted Paper]
    a. We propose an unsupervised Group Steiner Tree based Entity Linking (GroSTEL) method which is capable of performing EL for low resource domains in the absence of any annotated data.
    b. We use multiple domain specific datasets and compare with multiple state-of-the-art baselines to demonstrate the superior performance of our model.
    c. Our proposed method (GroSTEL) is also shown to improve the performance of supervised EL models when used in conjunction.

- Technical Skills:
    - **Python** : (Advance), have used python extensively for most of my project that included the following:
        - **Machine learning** : Pytorch, Huggingface, Scikit-Learn
        - Numpy, Matplotlib, Pandas, NLTK: For manipulation, evaluation and plotting purposes.
        - **Web Server**: Fastapi, Flask
    - Git: for version control of almost all my projects
    - C and C++ : Intermediate
    - SPARQL : Basic (Learnt for this project)
    - Web Development: HTML, CSS, Js, React, Tailwind.
    - Linux: Bash, zsh, Core Utils (grep, sed, etc), Vim

- Summer Plans:
    1. **What city/country will you be spending this summer in?**
        a. Kolkata(and some amount of time in Bangalore), India
    2. **Do you have a full- or part-time job or internship planned for this summer?**
        a. No, i have no such plans currently
    3. **How many hours per week do you have available for a summer project?**
        a. 25-30 hrs per week

- About Me:

I am Debarghya Datta, a 2nd Year Master's student, currently studying in Indian Institute of Technology, Bhilai in the Computer Science Department specializing in Natural Language Processing. Completed my Bachelor's at Kalinga Institute of Industrial Technology, Bhubaneswar in Computer Science engineering discipline with 9+ CGPA. During my bachelor's, I interned at HighRadius where I built a dashboard for accounts receivable for partial payment collection. After graduation, I joined Tata Consultancy Services as E2E Quality Assurance engineering, where I tested different functional requirements extensively with custom test data.