Google
Summer of Code

cdli

CDLI

Multiple Layer Annotations Querying

9th April 2019

Name	Sagar
Organization	CDLI
Project	Multiple Layer Annotations Querying

ABSTRACT

Currently, there is no tool available to integrate into a website that has the capacity to query through multiple layers of linguistic annotations like morphology, syntax and semantics annotations. Also, the data has to be queried from RDF format which is flexible and will help integrate into a linked open data toolbox for computation. A tool named ANNIS already exists to query through multiple layers of annotations but it has some limitations which have to be overcome. Also, the tool must have a query language which must be easy to use for non-professionals. It must also highlight the results in the retrieved document which resulted in the retrieval of the document. The tool must also return data in a format that allows for easy integration into any website.

A SAMPLE TASK USING ANNIS

A sample task was done by me, showcasing some of the aspects of the project. The work was done on the P340098 file.

1. The P340098 file in ATF format can be seen here. (This is a part of big ATF file `cdliatf_unblocked.atf` taken from https://github.com/cdli-gh/data)
2. This ATF file was converted to CDLI-CoNLL format using ATF-CoNLL Converter tool.
3. The CDLI-CoNLL format file of P340098 can be seen here.
4. CDLI-CoNLL file was converted to CoNLL -U format the CoNLL -CoNLLU Converter tool.
5. The CoNLL-U format file of P340098 can be seen here.
6. The file from CoNLL-U format to RDF format was converted using CoNLLU-RDF tool.
7. The RDF format with .ttl extension can be found here.
8. For the task of querying a multi-layer corpus, a separate multi-layer corpus which was available preloaded on a free public ANNIS installation was used. The data was queried using AQL which is the query language of ANNIS.

9. The instance of the single-word multi-layer querying using AQL in ANNIS can be seen here (This query queries all the documents that have the lemma “art” and this lemma has POS tag as “NN”).
10. Further, an instance of multiple-layers multiple-word querying can be found here (This query queries all the documents that have the lemma “possible” immediately followed by a word that has a POS tag which starts with “NN”).
11. This folder contains all the files mentioned above.

OUTLINE AND METHODOLOGY

- **Studying about the complete system**
  - Study extensively about SPARQL from this book by Orelli authored by Bob DuCharme.
  - For multi-layer querying I would be studying about:-
    - ANNIS3 paper
    - PubAnnotation paper
  - Study the documentation of Apache Jena that would be used to set up the SPARQL endpoint.
- **Data Format Conversion**
  - All the data that is present with CDLI is in CDLI-ATF format which is not annotated. An example of the data is as follows:-

```
&P340098 = BPOA 1, 1443
#atf: lang sux
@object tag
@obverse
1. pisan-dub-ba
#tr.en: Basket-of-tablets:
2. kiszib3 dabx(U8)?-ba a2 erin2-na-ka
#tr.en: sealed documents of dab, labor of the (labor-)troops,
3. lugal-ku3-zu nu-banda3-gu4
#tr.en: Lugal-kuzu, manager of oxen,
4. iti 1(u) 4(disz)-kam
#tr.en: (a period of) 14 months,
5. iti sze-sag11-ku5
#tr.en: from month “Harvest,”
# some text moved to next line
```

```
6. mu na-ru2-a-mah ba-du3-ta
#tr.en: year: “Big-Stele was erected”
@reverse
1. iti sze-sag11-ku5
#tr.en: to month “Harvest,”
2. mu ma-da za-ab-sza-li{ki} ba-hul-sze3
#tr.en: year: “The lands of Zabšali were destroyed,”
3. mu na-ru2-a-mah ba-du3
#tr.en: year: “Big-Stele was erected.”
```

- These annotations are converted to CDLI-CoNLL format, a format developed by CDLI which is easy to annotate. For this task, ATF-CoNLL Converter tool has already been developed by CDLI. An example of CDLI-CoNLL format is as follows:-

```
#new_text=P340098
# ID   FORM   SEGM   XPOSTAG      HEAD   DEPREL MISC
o.1.1  pisan-dub-ba bisagdubak[filing_basket] N      _      _      _
o.2.1  kiszib3      kiszib[seal] N      _      _      _
o.2.2  dabx(U8)-ba  dab[seize]-a NF.V.PT      _      _      _
o.2.3  a2     a[wage]      N      _      _      _
o.2.4  erin2-na-ka  erin[people]-ak-ak  N.GEN.GEN     _      _      _
o.3.1  lugal-ku3-zu Lugalkuzu[1] PN     _      _      _
o.3.2  nu-banda3-gu4nubandagu[ox_overseer][-ak]      N.GEN  _      _      _
o.4.1  iti    iti[month]   N      _      _      _
o.4.2  1(u)   1(u)[ten]    NU     _      _      _
o.4.3  4(disz)-kam  4(disz)[one][-ak][-am]    NU.GEN.COP-3-SG     _      _      _
o.5.1  iti    iti[month]   N      _      _      _
o.5.2  sze-sag11-ku5Szesagkud[1] MN     _      _      _
o.6.1  mu     mu[year]     N      _      _      _
o.6.2  na-ru2-a-mah naruamah[magnificant_stele][-ø]  N.ABS  _      _      _
o.6.3  ba-du3-ta    ba-du[build][-ø]-ta MID.V.3-SG-S.ABL    _      _      _
r.1.1  iti    iti[month]   N      _      _      _
r.1.2  sze-sag11-ku5Szesagkud[1] MN     _      _      _
r.2.1  mu     mu[year]     N      _      _      _
r.2.2  ma-da  mada[land]   N      _      _      _
r.2.3  za-ab-sza-li{ki}    Zabszali[1][-ak]    SN.GEN _      _      _
r.2.4  ba-hul-sze3  ba-hulu[destroy][-ø]-sze  MID.V.3-SG-S.TERM   _      _      _
r.3.1  mu     mu[year]     N      _      _      _
r.3.2  na-ru2-a-mah naruamah[great_stele][-ø] N.ABS  _      _      _
r.3.3  ba-du3 ba-du[build][-ø]    MID.V.3-SG-S _      _      _
```

- Once these annotations are complete they have to be converted to CoNLL-U format using CoNLL-CoNLLU Converter developed by CDLI. An example of data in CoNLL-U format is as follows:-

```
#new_text=P340098
#ID    FORM   LEMMA  UPOSTAG      XPOSTAG      FEATS  HEAD   DEPREL DEPS   MISC
o.1.1  pisan-dub-ba bisagdubak[filing_basket] NOUN   N      Number=Sing  _      _
       _      _
o.2.1  kiszib3      kiszib[seal] NOUN   N      Number=Sing  _      _      _      _
o.2.2  dabx(U8)-ba  dab[seize]   VERB   V      _      _      _      _      _
o.2.3  a2     a[wage]      NOUN   N      Number=Sing  _      _      _      _
o.2.4  erin2-na-ka  erin[people] NOUN   N      Case=Gen,Gen|Number=Sing  _      _
       _      _
o.3.1  lugal-ku3-zu Lugalkuzu[1] PROPN  PN     Animacy=Hum|Number=Sing   _      _
       _      _
o.3.2  nu-banda3-gu4nubandagu[ox_overseer]    NOUN   N      Case=Gen|Number=Sing
       _      _      _      _
o.4.1  iti    iti[month]   NOUN   N      Number=Sing  _      _      _      _
o.4.2  1(u)   1(u)[ten]    NUM    NU     _      _      _      _      _
o.4.3  4(disz)-kam  4(disz)[one] NUM    NU     Case=Gen     _      _      _      _
o.5.1  iti    iti[month]   NOUN   N      Number=Sing  _      _      _      _
o.5.2  sze-sag11-ku5Szesagkud[1] PROPN  MN     Number=Sing  _      _      _      _
o.6.1  mu     mu[year]     NOUN   N      Number=Sing  _      _      _      _
o.6.2  na-ru2-a-mah naruamah[magnificant_stele]      NOUN   N
Case=Abs|Number=Sing       _      _      _      _
o.6.3  ba-du3-ta    du[build]    VERB   V      Voice=Mid|Person=3|Number=Sing   _
       _      _      _
r.1.1  iti    iti[month]   NOUN   N      Number=Sing  _      _      _      _
r.1.2  sze-sag11-ku5Szesagkud[1] PROPN  MN     Number=Sing  _      _      _      _
r.2.1  mu     mu[year]     NOUN   N      Number=Sing  _      _      _      _
r.2.2  ma-da  mada[land]   NOUN   N      Number=Sing  _      _      _      _
r.2.3  za-ab-sza-li{ki}    Zabszali[1]  PROPN  SN     Case=Gen|Number=Sing      _
       _      _      _
r.2.4  ba-hul-sze3  hulu[destroy]VERB   V      Voice=Mid|Person=3|Number=Sing   _
       _      _      _
r.3.1  mu     mu[year]     NOUN   N      Number=Sing  _      _      _      _
r.3.2  na-ru2-a-mah naruamah[great_stele]     NOUN   N      Case=Abs|Number=Sing
       _      _      _      _
r.3.3  ba-du3 du[build]    VERB   V      Voice=Mid|Person=3|Number=Sing   _      _
       _      _
```

- Now since the data has multiple annotations, we need to make a query from multiple layers of annotation system. For this, CoNLL-U format data is then converted to RDF format which can be realized as a graph data and can be queried using SPARQL query. An example of data in RDF format is as follows:-

```
#new_text=P340098   #ID FORM LEMMA UPOSTAG XPOSTAG FEATS HEAD DEPREL DEPS MISC
@prefix : <https://github.com/UniversalDependencies/UD_English#> .
@prefix terms: <http://purl.org/acoli/open-ie/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix conll: <http://ufal.mff.cuni.cz/conll2009-st/task-description.html#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix nif: <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#> .
:s1_0 a nif:Sentence .
:s1_o.1.1 a nif:Word; conll:FEATS "Number=Sing"; conll:FORM "pisan-dub-ba";
conll:ID "o.1.1"; conll:LEMMA "bisagdubak[filing_basket]"; conll:UPOSTAG "NOUN";
conll:XPOSTAG "N"; nif:nextWord :s1_o.2.1 .
:s1_o.2.1 a nif:Word; conll:FEATS "Number=Sing"; conll:FORM "kiszib3"; conll:ID
"o.2.1"; conll:LEMMA "kiszib[seal]"; conll:UPOSTAG "NOUN"; conll:XPOSTAG "N";
nif:nextWord :s1_o.2.2 .
:s1_o.2.2 a nif:Word; conll:FORM "dabx(U8)-ba"; conll:ID "o.2.2"; conll:LEMMA
"dab[seize]"; conll:UPOSTAG "VERB"; conll:XPOSTAG "V"; nif:nextWord :s1_o.2.3 .
...
...
...
```

*These are just the first few lines of the data in RDF format.

- **Setting up SPARQL endpoint**
  - The Docker Container for Apache Jena would be used and would be added to the Docker Container of CDLI website.
  - The data in RDF format would be loaded and queried.
- **Making a SPARQL query system to search for a single word with multiple layers of annotations in a document**
  - Single-Layer query search that targets searching for a lemma.
  - It refers to retrieving all the documents that have the given lemma in it.

- For eg:- A case where we want to find all the documents that have the lemma **blind** in it.
- For this, I would be working on writing a query to retrieve documents that have the given lemma in it.
- Further, it would be extended to search using the multiple layers of annotation of that token.
- For this, a SPARQL query would be used to retrieve documents by applying filters on the documents. This would be done by applying complex SPARQL queries that concatenate multiple conditions for different layers of annotations with it.
- For eg:- a case where we want to find all the documents that have the word **blind** as an **adverb**.

- **Making a SPARQL query to search for multiple words with multiple layers of annotations in a document**
  - Firstly, the system would be made to search for single-layer of annotations for multiple lemmas.
  - It refers to searching all the documents that have all the given lemmas in it.
  - For eg:- a query to retrieve all documents that have both the words **blind** and a word **person** in it.
  - This would be done by the intersection of all the documents retrieved from both the queries separately.
  - This would be extended to multiple layers of annotations by extending it to the intersection of multiple-layers of single-word queries.
  - It retrieves all the documents that have all the given word with the given annotations in it.
  - For eg:- a query to retrieve all documents that have the word **blind** as an **adverb** along with a **noun** which has a **named entity tag** as **‘Person’** class.

- **Querying the data in an easy format (for non-professionals)**
    - The system would be extended to provide an easy format for non-professionals to query.
    - Work on determining an easy format to query would be researched on from the beginning itself.
    - A sample for single word multi-layer query in user-friendly query format :- **“ lemma = ‘blind’ | pos = adverb; “** (can be improved and made even simpler)
    - Here I have specified the word and its required annotations.
    - A sample for multiple words multi-layer query:-**” lemma = ‘blind’ | pos = adverb & pos = noun | ner = person ;“**
    - Here I have a combination of multiple words. This query tries to retrieve documents with word **blind which is an adverb** and another word which is a **noun** and has an NER tag of **Person** class.
- **Returning the data in different formats (for humans and for machines)**
    - For humans, the retrieved data would be the list of all the documents that satisfy the query.
    - Also, the words in the retrieved document would be highlighted which resulted in its retrieval.
    - For machines, the retrieved documents would be all the documents that satisfy the query in RDF/XML format and JSON format.
    - Marking these words in the retrieved documents in the JSON and XML/RDF format so that these words can be identified easily on other platforms and would be easy to highlight them.
- **Making the system integrable to any website**
    - Since all the documents are returned in RDF/XML format as well as in JSON format, the system would be easily integrable to any of the web services.
    - Further, the system can be extended to many other formats which would make it easy to integrate it to any other web service.

SCHEDULE

1. Community Bonding (May 6 - May 27, 2019)

- Getting familiar with CDLI and data and annotations available with CDLI
- Setting up the system with the required dependencies to build the query system.
- Study about ANNIS-3 research paper and other research papers.
- Study about SPARQL from the book.
- Study about Apache Jena ARQ for SPARQL.
- Setting up the blog under the research section of https://cdli-gh.github.io/ (official documentation website) which I would be updating every week.

2. Week 1 (May 27 - June 3)

- Building a system to convert a document to RDF format.
- Converting all the data to RDF format.
- Automating the tool to trigger whenever data is uploaded. Would automatically be converting the data to RDF format and updating the database.
- Setting up the SPARQL endpoint using Apache Jena.
- Writing the blog about the work done.

3. Week 2 (June 3 - June 10)

- Loading the data to make a SPARQL query.
- Making a SPARQL query to search for a single word in a sentence.
- Writing tests for the system.

- Writing the blog about the work done.

4. Week 3 (June 10 - june 17)

- Making a SPARQL query to search for a single word by defining the annotations of that word.
- Working on querying the data in a normal format to be used by non-professionals.
- Writing tests for the system.
- Writing a blog to update all the work done in the week.

5. Week 4 (June 17 - June 24)

- Continuing the work from the previous week.
- Setting up the search system on the CDLI interface.
- Integrating the SPARQL single-word multi-layer query search system into the CDLI interface.
- Start researching and discussing the user-friendly querying system for single word multi-layer querying system.
- Testing the search system and the query system on the CDLI website.
- Writing the blog for the work done in both the weeks.

First Phase Evaluation (June 24 - June 28)

6. Week 5 (June 24 - July 1)

- Extending the system to search for a combination of words in a sentence.
- Writing tests for the work done.
- Integrate the SPARQL multi-word query search system into the CDLI interface.

- Testing the integrated system.
- Writing the blog for the work done.

7. Week 6 (July 1 - July 8)

- A system to search for a combination of words by defining the annotations of all the words in the query.
- Testing the system.
- Integrating the SPARQL single-word multi-layer query search system into the CDLI interface.
- Research and discuss the user-friendly query system for non-professionals for multiple words and multilayer querying system.
- Testing the search system on the website.
- Writing a blog for the learnings in the week.
- **If not able to do the project and face any issues, would shift to the Fallback Plan.**

8. Week 7 (July 8 - July 15)

- Working on querying the data in a form easy to query for non-professionals. (Which was researched and discussed in previous weeks and would be implemented here).
- Integrating the query system with the CDLI interface.
- Testing the query system developed.
- Complete integrating the system with the CDLI interface and solve any bugs that were missed.
- Writing Tests for the work done.
- Writing a blog for the work done in the week.

9. Week 8 (July 15 - July 22)

- Continuing the work from the previous week.
- Writing the blog work done in the past week.

Would be shifting to Fallback Plan, if things are not working out. And would be showing the results using the Fallback Plan in the Second Evaluation.

Second Phase Evaluation(July 22 - 26)

10. Week 9 (July 22 - July 29)

- Working on returning the data in readable formats for humans.
- Working on highlighting the words in the documents that match the query in each of the results.
- Working on returning the data for machines like in JSON, RDF/XML, etc and marking/highlighting the words in the documents in these formats also.
- Writing a short tutorial for non-professionals to query using the new user-friendly language developed.
- Write a blog for the work done in the week.

11. Week 10 (July 29 - August 5)

- Making the system integrable to any web service.
- Writing a tutorial explaining the process of how to integrate the service.
- Updating all the work done in the blog.

12. Week 11 (August 5 - August 12)

- Writing README on how to set up the system.
- Write the documentation and comments for the code.

- Updating all the work done in the blog.

13. Week 12(August 12 - August 19)

- Buffer Week to do any of the remaining tasks.

Final Evaluation (August 20 - August 26)

DELIVERABLES

I would be developing a Multiple Layer Annotation Querying tool that would be having the following properties:-

- Would be able to search through the complete data when a word or the annotations of the word or both are defined.
- Also, the system would be able to search for a combination of such word queries.
- A user-friendly language so that even non-professionals could easily query (the language would be limited to only some core functionalities).
- Also, the system would be able to return data in all types of formats, both readable by humans and the formats through which different services can communicate like JSON, XML, etc.
- Easily integrable to any web services.
- Tutorials on how to set up the system and use the system with other web services.
- Commented code explaining everything and complete documentation of the system.

TECH-STACK

- Apache Jena SPARQL endpoint
- Virtuoso (as a backup, if due to certain reasons, things not compatible with Apache Jena)
- SPARQL as a query language
- Java (to work with Apache Jena)
- PHP (The web framework of CDLI website is in PHP(CakePHP))
- Python (for many other tasks)

FUTURE WORK

The system can be extended a lot in future. If time permits, I would like to work on these extensions of the project. Some of the possible extensions are:-

- User-friendly query system can also be extended to provide more options for a single annotation of a single word itself. For eg:-
    - A query to find all the documents that have the word **blind** as an **adverb or blind** as a **NOUN.**
- The user-friendly query system can be extended further to search for **phrasal queries** (searching for a given phrase or sentence) and also to **proximity search** (searching for a query by providing the distance/no. of words between the words in the query). For eg:-
    - A query to find all the documents that have the phrase “**blind person”** in it.
    - A query to find all the documents that have the phrase “**blind** as an **adverb** followed by a word which is a **noun**”
- The ranking mechanism of the retrieved documents could also be done. They can be ranked on the basis of the frequency the query is satisfied in a document.
    - Given a query, to retrieve all documents where the word blind is an adverb. I this case sentences having the lemma “blind” with POS tag as adverb more number of times would be ranked higher as compared to documents that have the word “blind” lesser number of times.
- It can be extended to return data in various other types.
- For searching for a combination of words in all the sentences, the system can be optimized and can be used to reduce the query time.
- If time permits, the tool can be further improved to reduce the query time of the metadata of the document itself. This can be done by setting up a hierarchy for retrieving the documents for document specific details. For eg:- if we want to retrieve the documents for a specific query, and want the documents only from a specific period of years, then we can:-
    - Retrieve the documents from that particular range of years from MySQL database.
    - Apply the query on those retrieved documents.

FALLBACK PLAN

If due to uncontrolled conditions or errors, I am not able to go with the plan designed by me, then I would be working on the following fallback plan. I would be shifting to the Fallback Plan by the end of June, before the second evaluation and would make sure to show results in the Second Evaluation.

The minimal outcome of the project would be a system that would automate the task of converting the data to RDF format as soon as it is uploaded. Also, the CDLI corpus would be imported to ANNIS and then it would be queried. Then, the system would be integrated with ANNIS, which is an already existing tool for multiple layer querying of annotations.

The data would be uploaded on the ANNIS server and the queries fired from the CDLI website would be queried from ANNIS server using its API. After querying the retrieved documents would also be displayed on the CDLI website.

In the end, it would be made sure that the users are able to query the multi-layer annotated corpus and can see the results on the CDLI website.

OTHER OPEN SOURCE CONTRIBUTIONS

1. https://github.com/aimacode/aima-python/pull/1043 (not-merged)
2. https://github.com/aimacode/aima-python/pull/1042 (not-merged)
3. https://github.com/aimacode/aima-python/pull/1000 (approved) (not-merged)
4. https://github.com/aimacode/aima-python/pull/991 (approved) (not-merged)
5. https://github.com/aimacode/aima-python/pull/996 (approved) (not-merged)
6. https://github.com/aimacode/aima-python/pull/1010 (not-merged)
7. https://github.com/aimacode/aima-python/pull/1012 (merged)
8. https://github.com/aimacode/aima-python/pull/1013 (merged)
9. https://github.com/aimacode/aima-python/pull/1014 (merged)
10. https://github.com/aimacode/aima-python/pull/1016 (merged)
11. https://github.com/aimacode/aima-python/pull/993 (merged)
12. https://github.com/aimacode/aima-python/pull/1008 (merged)

13. https://github.com/aimacode/aima-python/pull/1017 (merged)
14. https://github.com/aimacode/aima-python/pull/1019 (merged)
15. https://github.com/aimacode/aima-python/pull/1034 (merged)
16. https://github.com/aimacode/aima-python/pull/1039 (merged)
17. https://github.com/cltk/cltk/pull/864 (merged)
18. https://github.com/jainaman224/Algo_Ds_Notes/pull/953 (not-merged)

WHY I AM BEST SUITED FOR THE PROJECT

The opportunity to work on this project would be very exciting and would help me showcase my coding abilities and my interest in the field of NLP. I plan to be a long term contributor and would be highly interested to support the organization and solve the issues in the tool developed by me even after GSoC 2019 has ended.

PAST PROJECTS

1. **Named Entity Recognition(NER) for Indian Languages**
    - Was a part of the shared task under FIRE 2018
    - Working Notes.
    - https://github.com/sagar-sehgal/Named-Entity-Recognition
    - This project is very near to me and took many days to study completely the literature and also write the Working Notes. An end to end Language Independent Deep Learning model built using Keras to predict the Named Entity for Indian Languages. The model makes use of the different contextual information of the words along with the Word-Level and Character-Level features that are helpful in predicting the various Named Entity Classes. No domain-specific knowledge and no handcrafted rules of any sort to attain good results. We used 2CNN+LSTM model consisting of CNN as the character-level encoder, CNN as the word-level encoder and BiLSTM as the Tag Decoder.
2. **Monument Search IR**
    - https://github.com/sagar-sehgal/MonumentSearch
    - This project was built using Lucene in Java. In this project, I built a retrieval system on the Indian Monuments. A small dataset for the images of

monuments and their corresponding text (small description of that monument). Monuments and their documents were indexed with their Soundex codes. Since it is difficult to write the correct spelling of monuments and its location, the monuments are indexed and more weight is given to the name of the document and less to its description. Correct monuments are retrieved using the Soundex code of the query given by the user.

3. **Chatbot-Song-Recommender**
    - https://github.com/sagar-sehgal/Chatbot_Song_Recommender
    - This project, even though not a big one is very close to me as it was done as a freelancing task and took a few sleepless nights to complete the project. A system which would recommend songs to a user based emotion of the person determined by chit-chatting with the user using a Chit-Chat Chatbot. IBM Watson's emotional API was used to determine the emotion of every response from the user. The response from the user, along with its emotion is sent to the pre-trained chit-chat chatbot model Cake-Chat. Also based on the emotion of the conversation the songs are recommended to the user based on the annotations from `last.fm` the dataset in Million Song Dataset.
4. **Freelancing Portal**
    - https://github.com/sagar-sehgal/FreelancingIIITS
    - This project is also very close to my heart and it took a complete semester to make a full-fledged freelancing platform. This project is a web portal built using Django to connect Freelancer to the employers (people who want to get the work done). Projects are added by the employer and then tasks are also added for each project. Then acc. to the skill of a freelancer the tasks are shown to a freelancer in his feed. The employer can select a freelancer for each of the tasks and all of the applicants would be notified about their acceptance and rejection. Further, the freelancer and the employer can rate each other, thus leading to a total rating of a user.
5. **Judicial Cases Dataset**
    - https://github.com/sagar-sehgal/Judicial-Cases-Dataset

- This is a Selenium based automation bot written in Python which was used to crawl all the documents from the website of the High Court. The PDF of all the cases in all the districts of the state were crawled and downloaded. The repository also contains the PDF of all the cases of some of the districts in the state. Due to space constraints, the data from all the districts were not downloaded and was limited to only few districts in the city..

6. **Cors**
   - https://github.com/sagar-sehgal/Cors
   - This is a web portal that consists of a database of all the tutorials of various technologies. Here a user can visit and view the various tutorials of various technologies. A user upvotes the tutorials if he likes any of them. Then these tutorials are shown according to upvotes. Further filters can be used to filter the tutorials on the basis of price, Video Tutorial/ Reading Tutorial and according to the level of difficulty.
7. **Water Management System**
   - https://github.com/sagar-sehgal/Water_Management_System
   - It is the system developed in Django where people can log in and monitor the condition of their plant. It can be used to remotely monitor the plant and control the water supply and check the health of the plant. The sensors are placed near the plant which sends the information to the hosted server and then this information is processed and shown using graphs.
8. **Abandoned-Object-Detection**
   - https://github.com/sagar-sehgal/Abandoned-Object-Detection)
   - This is a project developed in MATLAB, which is used to detect abandoned objects automatically from a video, particularly useful for identifying suspicious abandoned luggage at busy places like railway stations and bus stands. For this, the first frame of the video is assumed as the background image. The video is converted to frames of images and then each frame is subtracted from the background image and if an object remains static at one place for a fixed number of frames, then it is declared as an abandoned object and an alarm is raised.

ABOUT ME

1. Contact Details

a. **Name:-** Sagar
b. **Email:-** sagar.r16@iiits.in
c. **Telephone:-** +91-XXXXXXXXXX
d. **Timezone:-** Indian Standard Time (UTC+5:30)

2. Profiles

a. **Github Username:-** sagar-sehgal
b. **Gitter Username:-** @sagar-sehgal
c. **Email Address:-** sagar.r16@iiits.in
d. **Alternative Email Address:-** sehgal100sagar@gmail.com
e. **Homepage:-** https://sagar-sehgal.github.io/
f. **Codechef:-** https://www.codechef.com/users/sagar_sehgal
g. **Hackerrank:-** https://www.hackerrank.com/sagar_sehgal

3. Current Education Details

a. **College:** Indian Institute of Information Technology, Sri City, Chittoor
b. **Degree:** B.Tech. in Computer Science Engineering
c. **Current Year:** 3rd Year
d. **Expected to graduate:** May 2020

4. Achievements, Work Experience and Positions of Responsibilities

a. Won **Bronze Medal** in Hackerrank 101 Hack 53 (Certificate).
b. Scored a **rank of 94** in Codechef November Cook Off 2018 out of a total of **1730 participants**.
c. **Summer Research Intern:** I was the Summer Research Intern at IIIT Hyderabad under Dr. Ponnurangam Kumaraguru where I worked on the shared task from FIRE 2018 on Named Entity Recognition system for Indian Languages. (Certificate)
d. **Summer School Website Developer:** I was the Web-Developer for the Summer School on Privacy and Security in Social Media conducted by Dr. Ponnurangam Kumaraguru at IIIT Hyderabad in 2018.

e. **Chief Event Organizer for Med-Tech Hackathon:** I was the Chief event organizer for the Med-Tech Hackathon which was the flagship event of the college's first technical fest Techfesia 2018. I also brought a Start-up as one of the partners for the event which supported the event for providing the Problem Statements and Evaluating the solutions developed. (Certificate)
f. I have been contributing to **AIMA-Python** and have a contribution in **CLTK** as well.
g. **Java Mentor at Learn IT Girl:** I am currently a mentor for Java language at Learn IT Girl. It is an open Mentorship Program to help women learn a programming language while doing an awesome project. Each mentor is assigned to a single student and we as a mentor have to guide the student in doing their project. (Acceptance Mail)
h. **Contributor at Girlscript Summer of Code (GSSOC):** I am currently an open source contributor at GSSOC under the project **Algo DS Notes**. (My Contributions)

OTHER COMMITMENTS DURING SUMMER

I am not participating in any other internships during the GSOC programme. I am currently a part of a few events which would be ending before GSoC starts. Towards July-end/early-August, I might not be available for 2-3 days as the new semester begins, except that I will be able to devote 35-40 Hours per week and try for more if required.