

ReDefine - Research Definitions Summarized

Third Increment Report

Team #7 - COGNITOS

Team Members:

Nikhita Sharma (37), Dig Vijay Kumar Yarlagadda (47), Harsha Komalla (14), Sai Madhavi Sunkari (39)

1.INTRODUCTION

Motivation:

The motivation behind our project is to provide a concise summary of the research articles. We realized the necessity of such a system when we observed that there are dozens of research papers available and in order to just get the gist of single paper, we need to go through whole article. We do agree that there are other information sources like Wikipedia, but they don't give us complete insight of technical concepts, which we are looking for.

We intend to provide a knowledge graph that can come in handy for the students, as a search tool, to find articles, summary, authors and domains of respective articles. This system has the potential to be useful in many different ways: a beginner in a field can explore research papers in a particular domain, a search engine can crawl through the knowledge graph for more relevant search results, etc.,

Objectives:

The objectives of our project are:

- Build a scalable system for summarization of research articles which would give a graphical representation of the contents of the research paper including the important key terms and the relation between them. This would help understand the contents of the paper and the topic of discussion.
- Create an interactive knowledge graph of research articles where the user can view a summarized representation research papers in a field of research along with the sub categories in a specific research field. It would also include author, co-author, publication information, keywords in the paper and other related papers written by same author or co-author.

Features:

- User can provide a corpus of research papers related to a field as input.
- Users are presented with an interactive graph UI, which can be explored to view the interactions between research categories under the research field, authors and papers written, co-authors and interaction between related articles.
- The summary generated can be read by clicking on a particular paper title which would give a summarization of the contents of the paper.

Expected Outcomes:

1. Knowledge Graph:

We expect this project to generate a Knowledge graph representing all the publications in a specific field of study selected based on the requirement of the user. This graph would provide important results about the different papers written in the field, the paper titles, information about author of the paper, co-authors' information, other related papers an author or co-author has written and important results about each paper itself like focus of a paper, key terms in the paper and other important information.

2. Text Summary:

This project will also generate a text summary of each research paper/publication. A graphical representation of a research article would show the most important keywords in the article and relation between them to help understand the focus of research of the author. It would also show the major fields and topics of discussion. The summary will give an overview of contents of each paper to the end user, without the user having to read the complete research paper.

2.DOMAIN OF THE PROJECT:

Our project focuses on building a knowledge graph model using NLP and ML techniques to automatically generate summaries from a dataset of research articles. Due to the difficulty of finding sources for obtaining research articles, we are limiting our dataset to a few thousand articles. However, as our system is based on Apache Spark and we are confident that it can scale to large volumes.

Further we are concentrated on improving the quality of summaries generated, we are not particular about generating output in different formats like text summaries or graphical user interfaces; the output will be generated as a knowledge graph.

3.DATA COLLECTION:

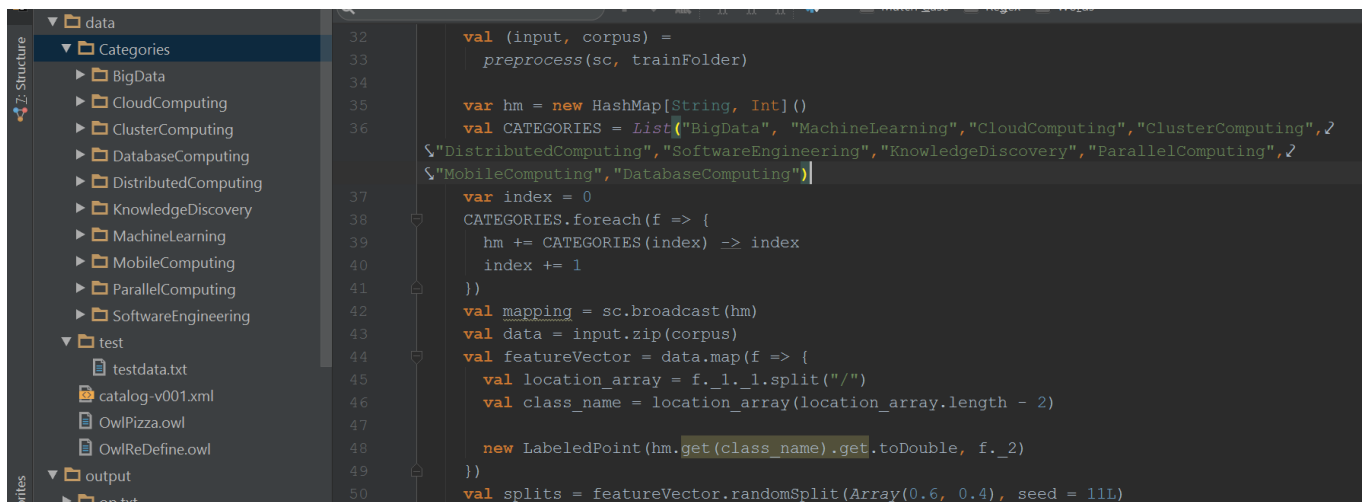
Finding an API which provides a dataset of entire research articles proved to be a difficult task, metadata of articles is available from many sources, but the actual full content of an article is not open sourced. We opted for using IEEE Xplore XML API.

We have currently used IBM Watson PDF2Word conversion API to convert the research papers to text format. Initially, we plan to perform summarization on 100 research articles and then extend to a larger dataset.

4.TASKS AND FEATURES IMPLEMENTED:

a. Pipeline: NLP- IR/IE-LDA-Feature Vector-ML-Ontology:

We have trained the Naive Bayes model using our input corpus divided into 10 categories: Big Data, Cloud Computing, Cluster Computing, database Computing, Distributed Computing, Knowledge Discovery, Machine Learning, Mobile Computing, Parallel Computing and Software Engineering. Then using LDA, when a new document is input to the system, the system can detect various terms in the documents as belonging to each of ten categories and an ontology is generated from it.



```
32  val (input, corpus) =
33      preprocess(sc, trainFolder)
34
35  var hm = new HashMap[String, Int]()
36  val CATEGORIES = List("BigData", "MachineLearning", "CloudComputing", "ClusterComputing",
37      "DistributedComputing", "SoftwareEngineering", "KnowledgeDiscovery", "ParallelComputing",
38      "MobileComputing", "DatabaseComputing")
39
40  var index = 0
41  CATEGORIES.foreach(f => {
42      hm += CATEGORIES(index) -> index
43      index += 1
44  })
45  val mapping = sc.broadcast(hm)
46  val data = input.zip(corpus)
47  val featureVector = data.map(f => {
48      val location_array = f._1._1.split("/")
49      val class_name = location_array(location_array.length - 2)
50      new LabeledPoint(hm.get(class_name).get.toDouble, f._2)
51  })
52  val splits = featureVector.randomSplit(Array(0.6, 0.4), seed = 11L)
```

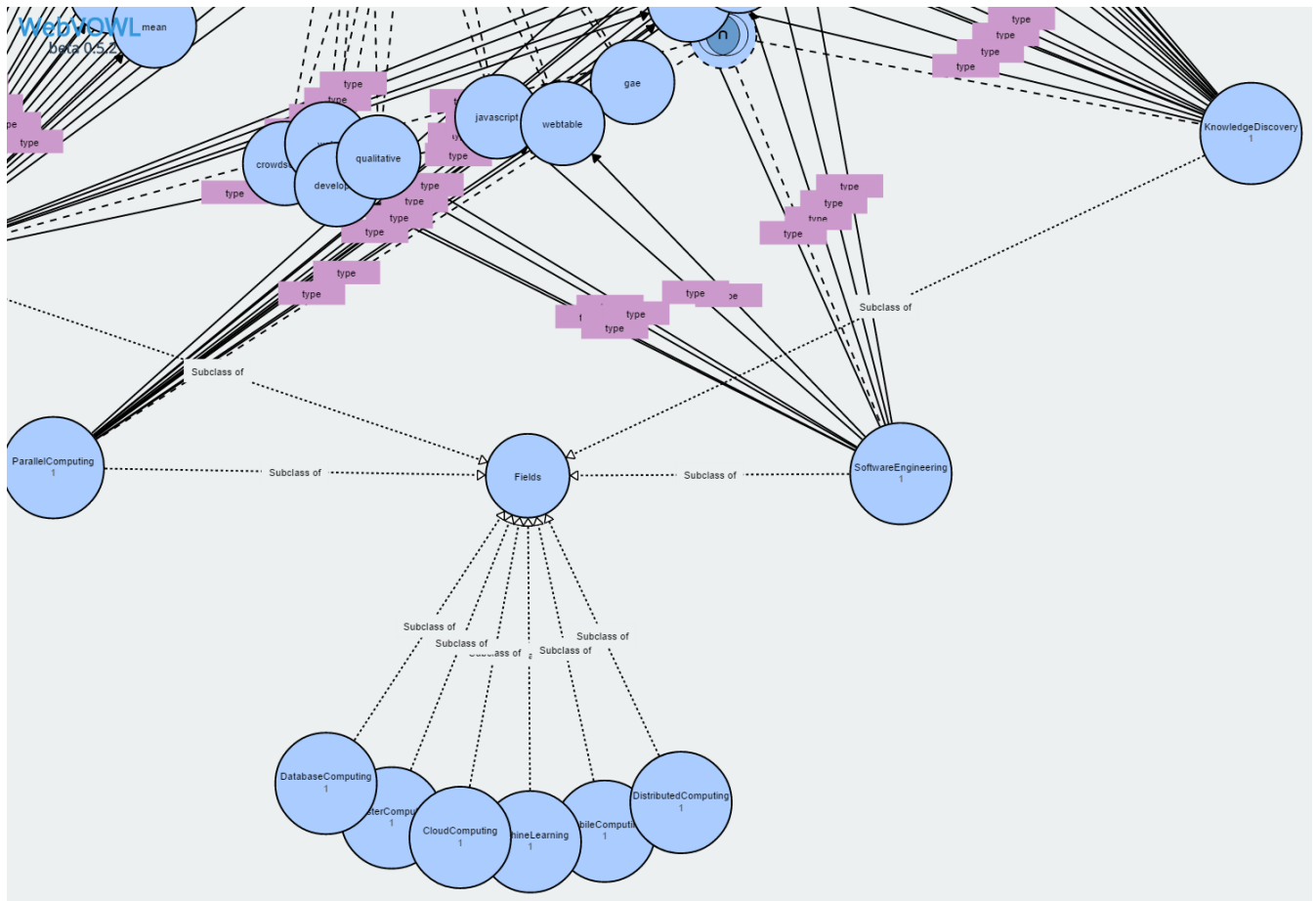
The screenshot shows a code editor with a file explorer on the left. The file explorer shows a directory structure with 'data' containing 'Categories' (a subdirectory with 10 files: BigData, CloudComputing, ClusterComputing, DatabaseComputing, DistributedComputing, KnowledgeDiscovery, MachineLearning, MobileComputing, ParallelComputing, SoftwareEngineering) and 'test' (containing testdata.txt, catalog-v001.xml, OwlPizza.owl, OwlReDefine.owl). The code editor shows Scala code for Naive Bayes model training, including preprocessing, feature vector creation, and splitting the data into training and testing sets.

```
Finished training LDA model.  Summary:
  Training time: 1.907966778 sec
  Training data average log likelihood: -22.410651889282054

10 topics:
peer persistent incremental processing structure d stream graphx gae percolator
peer uima entity recall ontology dbpedia precision qualitative crowdsourcing javascript
peer webtable graphs bigtable batch statistical clustering k vector supervise
peer webtable graphs bigtable batch crowdsourcing web javascript developer qualitative
peer gae pagerank pregel percolator graphx k vector clustering mean
peer version usability app android qos spark hive mapreduce hadoop
peer spanner oltp rdbms value key consistency relational hive mapreduce
peer webtable graphs batch bigtable app android usability qos version
peer spanner relational value key oltp consistency rdbms recall uima
peer k mean supervise clustering statistical vector relational rdbms value
0.0
7.0
0.0
0.0
0.0
7.0
6.0
0.0
0.0
0.0
Ontology Created

Process finished with exit code 0
```

b. Ontology and SPARQL and Rules:



SPARQL queries:

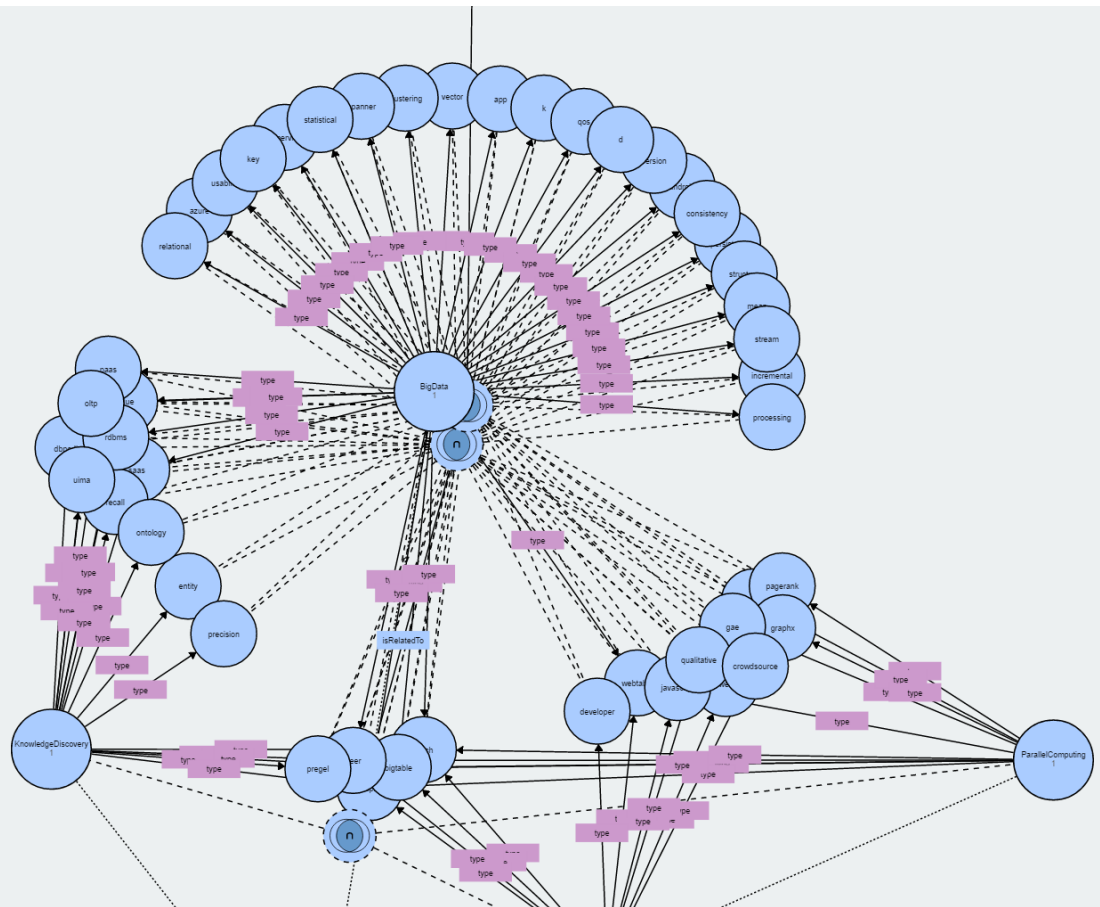
We did not use SPARQL in the system, but it can be used to extend the system using queries such as follows:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?subject ?predicate ?object
WHERE {?subject foaf:predicate ?object}
```

This query can be used to obtain the relations between two terms, which can be used to fill in spaces/gaps, if the data is ambiguous.

c. Application based on Ontology:

Using the base ontology, an input document is processed and terms in that document are classified as one of 10 categories. Based on this, we can classify the document as belonging to one of 10 categories.



OpenIE 4.1 is used to extract relations and the relations are saved in a JSON format as follows:

```
{
  "name": "A Distributed Machine learning System",
  "children": [
    {
      "name": "Machine learning (ML) and statistical techniques",
      "children": [
        {
          "name": "are",
          "children": [
            {
              "name": "key to transforming big data into actionable knowledge"
            }
          ]
        }
      ]
    }
  ],
  {
    "name": "MLbase",
    "children": [
      {
        "name": "provides way",
        "children": [
          {
            "name": "a simple declarative way to specify ML tasks"
          }
        ]
      },
      {
        "name": "can not optimally support",
        "children": [
          {
            "name": "every machine learning scenario"
          }
        ]
      },
      {
        "name": "uses",
        "children": [
          {
            "name": "a novel optimizer"
          }
        ]
      }
    ]
  }
]
```

5.IMPLEMENTATION SPECIFICATION:

a. Software Architecture & UML Model

System Architecture:

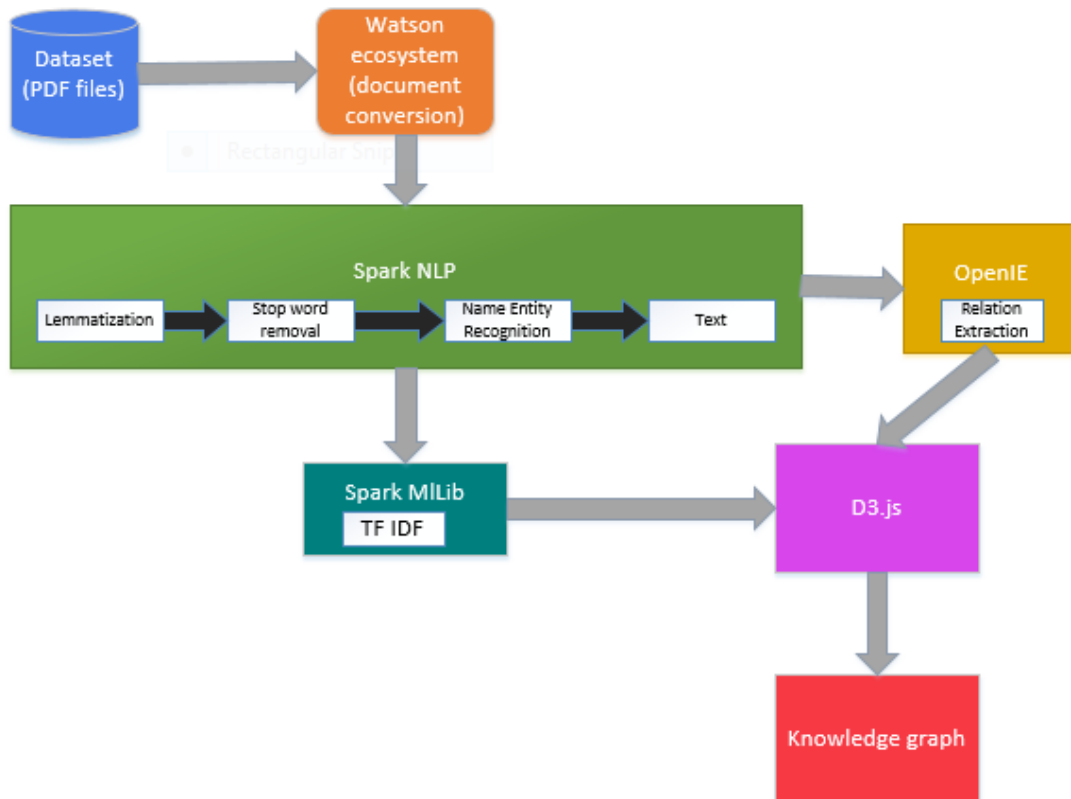


Fig 1: Architecture Diagram

Architecture Components:

1. [Watson Ecosystem SDK \(PDF2Word\):](#)

Watson Ecosystem SDK's PDF2Word Converter is one of the services provided by IBM to convert a PDF or a HTML document into plain text or JSON answer units. This document conversion tool detects the words using Optical Character Recognition (OCR) and also supports content in various languages.

2. [Spark NLP:](#)

Natural Language Processing is a part of Artificial Intelligence that has the ability to understand human speech. NLP tasks such as Sentence segmentation, tokenization, lemmatization, POS tagging, Named Entity extraction and Coreference resolution are provided through the Spark Core NLP package.

3. [OpenIE \(Relation Extraction\):](#)

OpenIE is used to extract relation triples from the document representing subject, relation and object triples. First a sentence is divided into clauses and each clause is minimized to small sentences and each sentence is divided into triples. We have used multiple OpenIE versions and extensions to extract relations like Stanford CoreNLP OpenIE, Ollie, Reverb and AllenAI OpenIE. We are planning to select the method which gives best triples results.

4. [SparkMLLib](#):

Spark's Machine Learning Library is a framework on top of Spark Core. It includes learning algorithms and utilities -regression, clustering and collaborative filtering and higher-level pipeline APIs, which makes machine learning easy and scalable. We will be using this package for Feature Vector generation and Machine learning

5. [D3.js](#):

D3.js is a JavaScript library which provides dynamic visualizations using SVG, HTML5 and CSS standards. SVG-objects are created by pre-built JavaScript functions and styled using CSS. Rich graphic charts are resulted on JSON/CSV as input file by using simple D3.js functions and SVG-objects. We will be using this tool to represent our summary and Knowledge Graph. We used Radial Tidy Tree and Collapsible Tree which are based on D3.js.

Class Diagram:

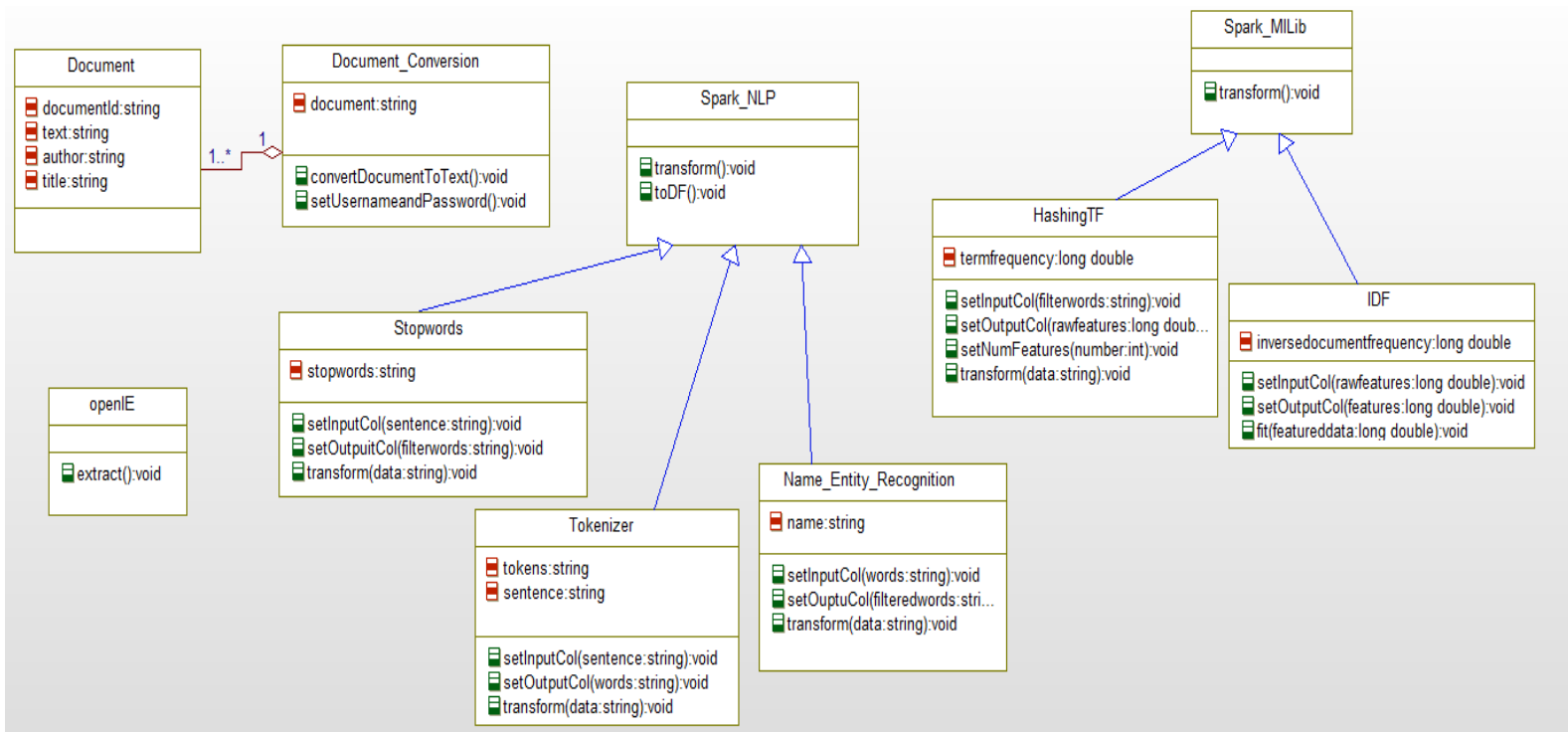


Fig 2: Class diagram

Sequence Diagram:

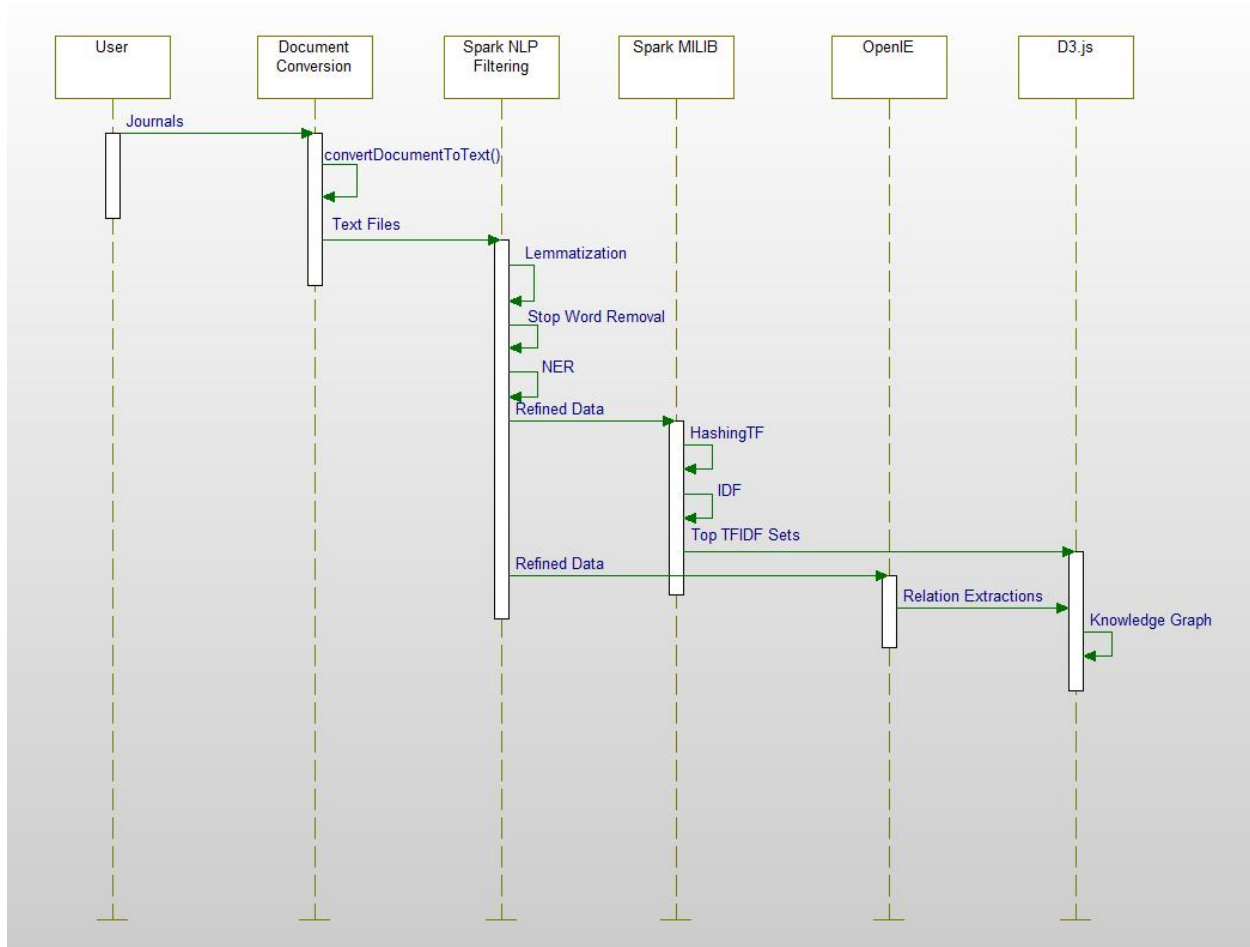


Fig 3: Sequence Diagram

Use Case Diagram:

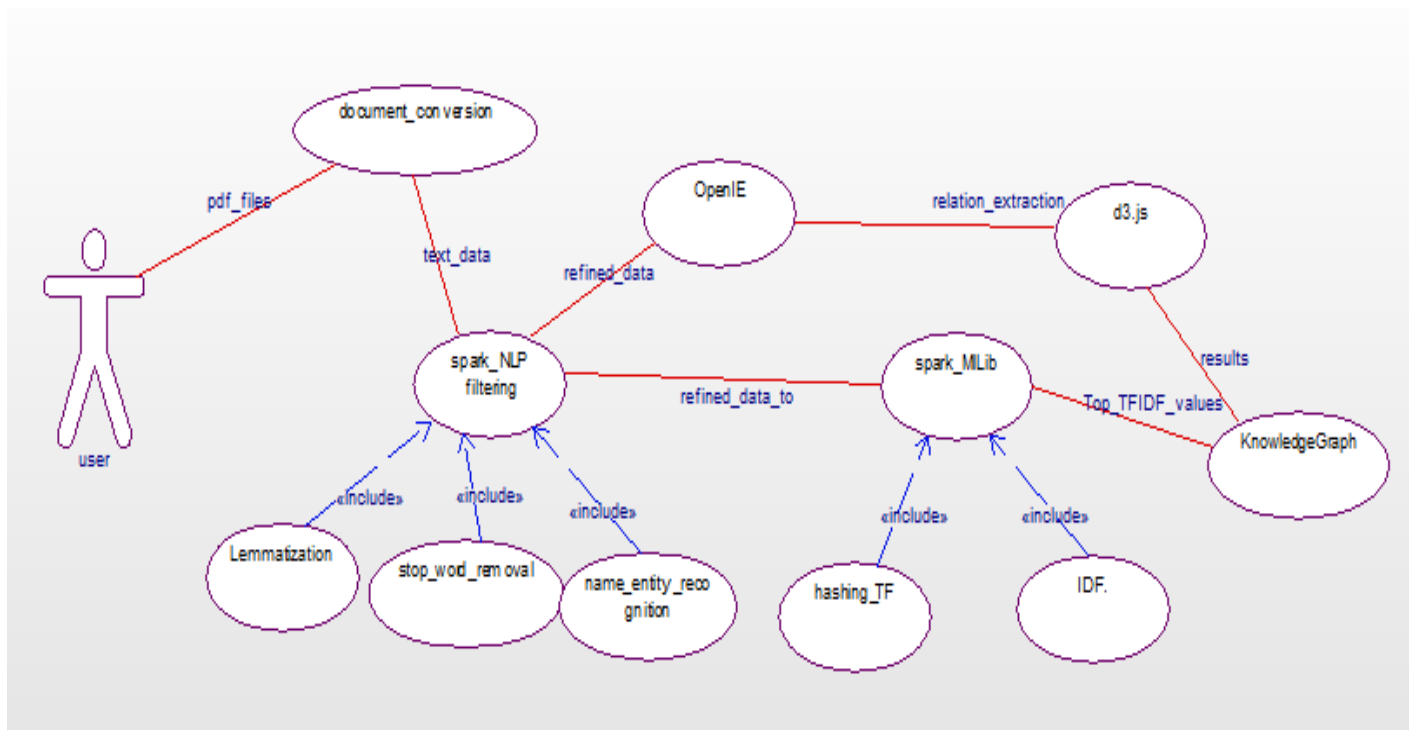


Fig 4: Use case diagram

b. Workflow

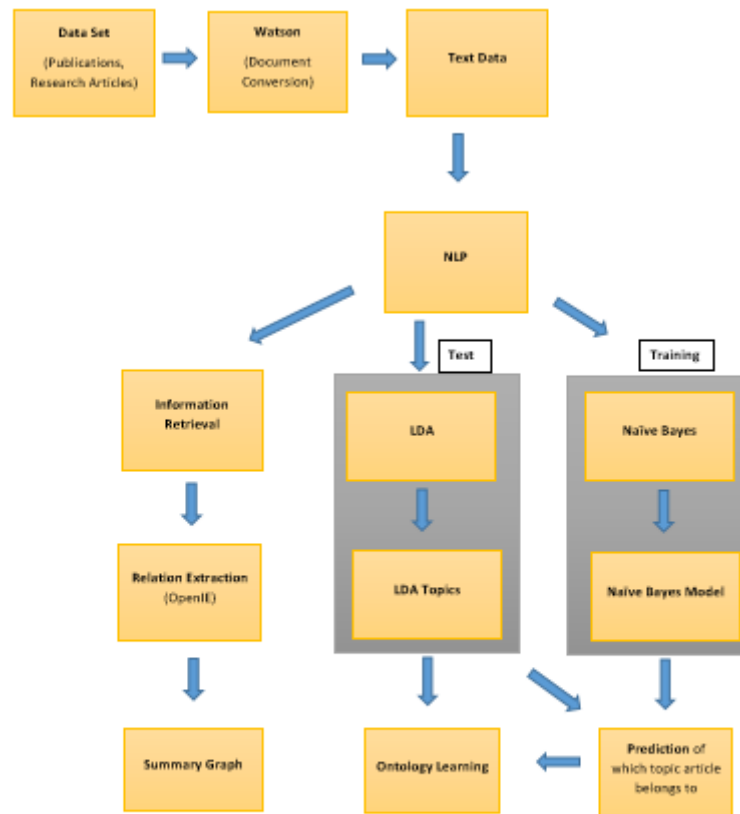


Fig 5: Workflow Diagram

c. Existing Services/APIs:

ReDefine uses different services and APIs for NLP and Information Extraction:

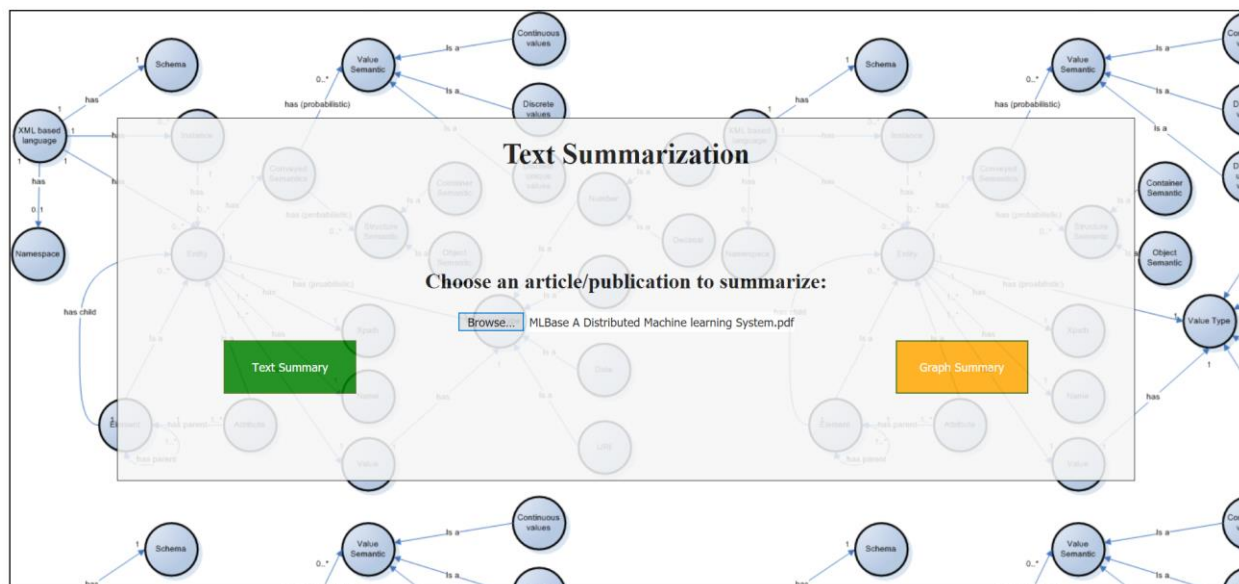
1. Document Conversion API in Watson Developer Cloud Java SDK using IBM Bluemix, for converting research documents in image or pdf formats into readable text files.
2. Open IE 4.1 for Relation Extraction
3. D3.js for visualizing knowledge graph

d. New services implemented:

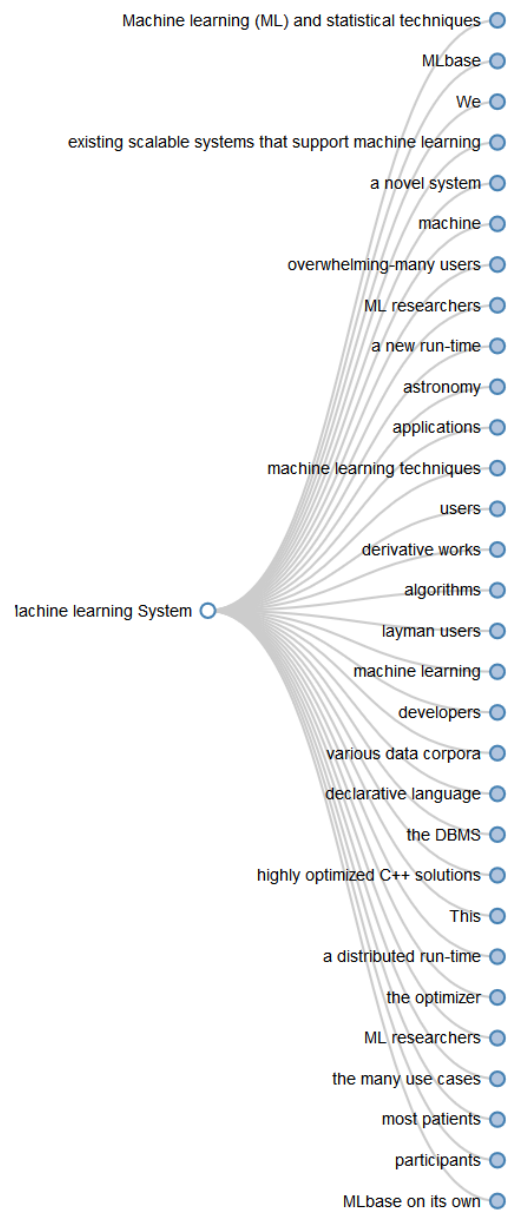
1. Dynamically created an ontology for our system, using Naive-Bayes and LDA for discovering topics among the text corpus. The text corpus has 10 categories, each representing a sub-field of computer science (Cloud Computing, Cluster Computing, Big Data etc.,). A model is generated based on the text corpus. When a new document is input to the model, it can detect various topics in the document and it can also classify terms in it as one of the ten categories.
2. Extracted relations using OpenIE 4.1 in JSON format and presented relations with high-confidence score in a graphical manner.

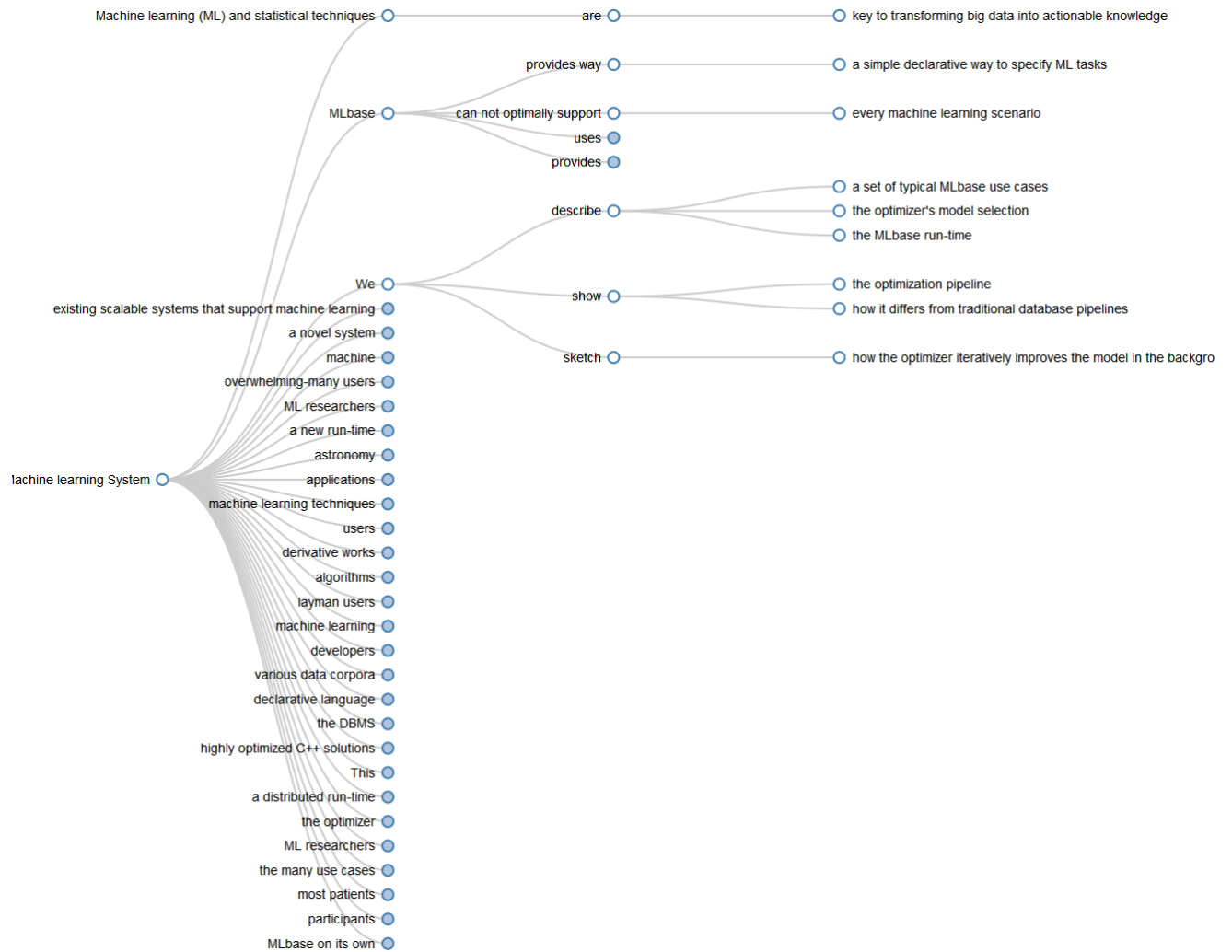
6.RESULTS & EVALUATION:

The system is trained on 128 publications/articles which are categorized into 10 different categories. (Training set with entire text data in the publication is not available, so we took open-source publication in PDF format, performed OCR and pruned the documents to generate a usable dataset). For a given input the system can display summary in two ways: Graph Summary or Text Summary.



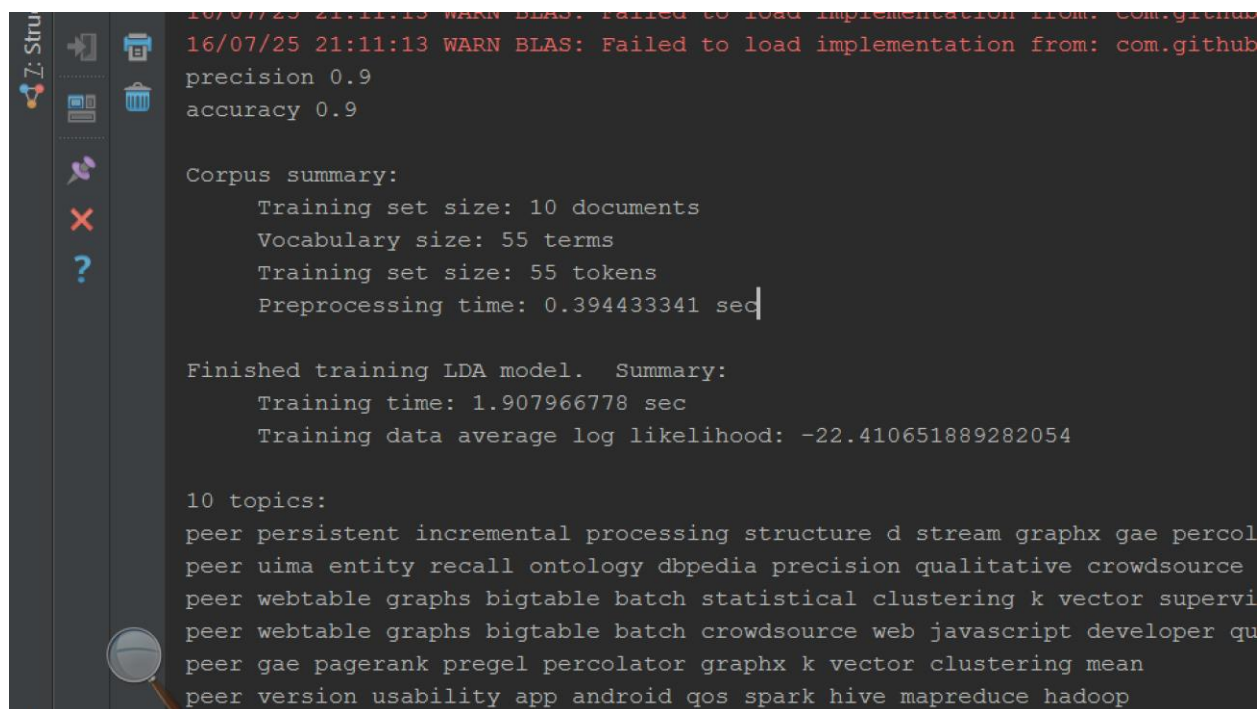
Graph Summary will give user an overview of important terms in the document. User can expand each term to further understand how that term is used in the document.





Precision and Accuracy:

The system exhibited a precision and accuracy ranging from 0.8 - 0.9. For example, a text publication had a precision and accuracy as follows:



```
16/07/25 21:11:13 WARN BLAS: Failed to load implementation from: com.github
16/07/25 21:11:13 WARN BLAS: Failed to load implementation from: com.github
precision 0.9
accuracy 0.9

Corpus summary:
  Training set size: 10 documents
  Vocabulary size: 55 terms
  Training set size: 55 tokens
  Preprocessing time: 0.394433341 sec

Finished training LDA model. Summary:
  Training time: 1.907966778 sec
  Training data average log likelihood: -22.410651889282054

10 topics:
peer persistent incremental processing structure d stream graphx gae percol
peer uima entity recall ontology dbpedia precision qualitative crowdsourcing
peer webtable graphs bigtable batch statistical clustering k vector supervi
peer webtable graphs bigtable batch crowdsourcing web javascript developer qu
peer gae pagerank pregel percolator graphx k vector clustering mean
peer version usability app android qos spark hive mapreduce hadoop
```

Performance:

Since our dataset is small, training of Naïve Bayes model using Quad Core CPU and 12 GB RAM assigned to application, only took 12 minutes. Actual classification of terms in document took an average of 1.8 seconds, making this system very much suitable for real-time applications.

Relation extraction using Open IE 4.1x required an approximate time of 6 minutes for each document and is a computationally intensive task.

7.PROJECT MANAGEMENT:

a. Team members contribution: (overall)

Nikhita Sharma - Ontology creation and relation extraction - 25%

Dig Vijay Kumar Yarlagadda - Relation Extraction, Ontology creation, Visualization- 25%

Harsha Komalla – Visualization ontology creation- 25%

Sai Madhavi Sunkari – SPARQL and Ontology generation 25%

b. Zenhub and Github URL/Statistics

Project GitHub repository:

https://github.com/nikhitasharma/KDM_Summer_2016_Cognitos

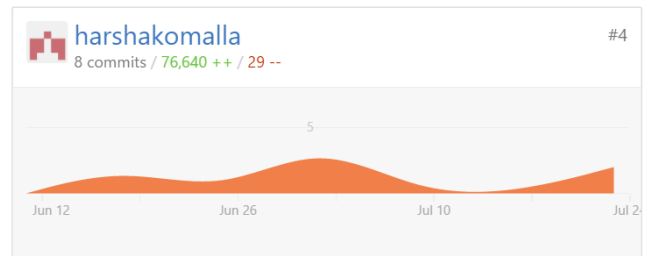
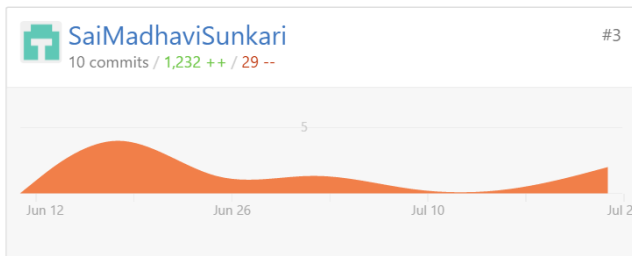
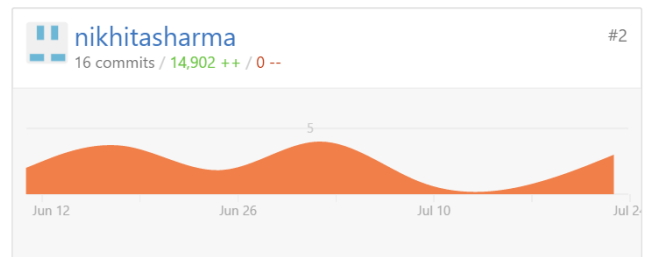
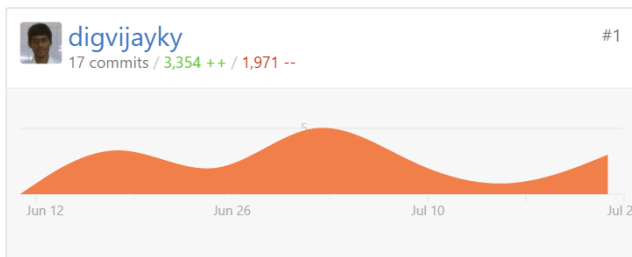
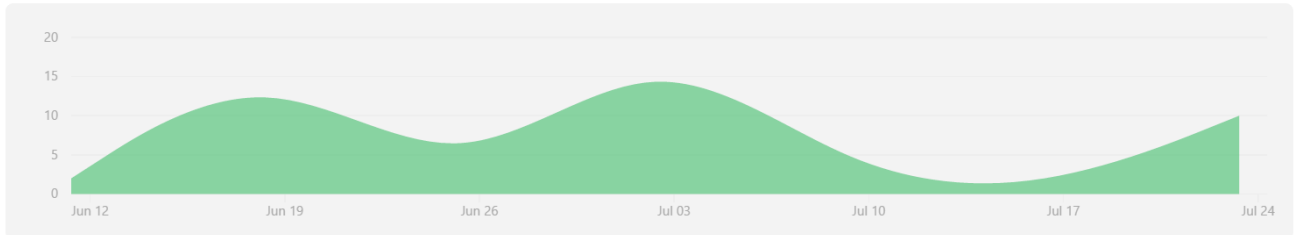
Zenhub statistics:

Team members contribution:

Jun 12, 2016 – Jul 25, 2016

Contributions: **Commits** ▼

Contributions to master, excluding merge commits



Zenhub Boards:

The screenshot displays a Zenhub Kanban board for the repository `nikhitasharma / KDM_Summer_2016_Cognitos`. The board is organized into six columns: **Icebox** (3 items), **Backlog** (1 item), **In Progress** (1 item), **Review/QA** (0 items), **Done** (2 items), and **Closed** (11 items). Each column contains task cards with titles, issue numbers, and labels like "enhancement" or "help wanted".

Icebox:

- KDM_Summer_2016_Cognitos #12: TextRank algorithm ambiguity! Abstraction or Extraction? (enhancement)
- KDM_Summer_2016_Cognitos #10: Compute Word2Vec model of dataset (enhancement)
- KDM_Summer_2016_Cognitos #8: Create knowledge graph of research articles (enhancement)

Backlog:

- KDM_Summer_2016_Cognitos #2: Collect open datasets of research articles (help wanted)

In Progress:

- KDM_Summer_2016_Cognitos #7: Extract relations in text (enhancement)

Done:

- KDM_Summer_2016_Cognitos #6: Extract Names, entities from dataset (enhancement)
- KDM_Summer_2016_Cognitos #5: Perform Lemmatization on data set (enhancement)

Closed:

- KDM_Summer_2016_Cognitos #15: Calculated TFIDF values but need to get top values (enhancement)
- KDM_Summer_2016_Cognitos #9: Compute TF-IDF on dataset (enhancement)
- KDM_Summer_2016_Cognitos #3: Convert research articles of pdf or image formats into text ... (enhancement)
- KDM_Summer_2016_Cognitos #11: API for collecting dataset of entire research articles not a ... (help wanted)
- KDM_Summer_2016_Cognitos #14: Perform tokenization on data (enhancement)
- KDM_Summer_2016_Cognitos #4: Remove stop words from data (enhancement)
- KDM_Summer_2016_Cognitos #1: Update README.md (enhancement)

Burndown Chart:



c. Concerns or Issues:

- Assigning weights to sentences in a documents is a complex task which is done using lot of procedures and only on clearly defined datasets, as explained in research papers. Moreover, most of the tools used in summarization research papers are proprietary.
- We have generated graph for a single document, but Generating a single knowledge graph from all input documents is
- In a knowledge graph, relationships between entities play a key role in understanding the gist of the concept discussed in a paper. In our case, after removing stop words from the data, we are performing lemmatization. We observed that this resulted in losing some important relationships between key terms. Since TextRunner values verb forms, it will not deliver sensible results on lemmatized dataset. We are mulling over this issue and trying with different articles to assess the impact of lemmatization on results generated by TextRunner.

d. Future Work:

- The system is designed to be scalable, but never been tested for large datasets due to lack of data and computing resources.

- Also, our knowledge graph can be further extended to represent different sub-fields in a field of research.

REFERENCES:

- [1]. <http://apache.org/>
- [2]. <http://spark.apache.org/docs/latest/mllib-guide.html>
- [3]. <https://spark.apache.org/docs/1.3.1/api/java/org/apache/spark/rdd/RDD.html>
- [4]. <https://spark.apache.org/docs/1.5.1/api/java/org/apache/spark/sql/DataFrame.html>
- [5]. <https://spark.apache.org/docs/1.1.0/mllib-feature-extraction.html>
- [6]. <https://github.com/watson-developer-cloud>
- [7]. <https://github.com/stanfordnlp/CoreNLP>
- [8]. <http://nlp.stanford.edu/software/openie.html>
- [9]. <http://spark.apache.org/docs/latest/mllib-feature-extraction.html>
- [10]. <http://openie.allenai.org/>
- [11]. <http://reverb.cs.washington.edu/>
- [12]. <http://knowitall.github.io/oilie/>
- [13]. <http://www.sciencedirect.com/>
- [14]. <https://journalofbigdata.springeropen.com/>
- [15]. <http://bigdata-madesimple.com/research-papers-that-changed-the-world-of-big-data/>
- [16]. <http://www.engpaper.net/big-data-research-papers-2013-2014.htm>
- [17]. <http://www.journals.elsevier.com/big-data-research/recent-articles>
- [18]. <https://www.researchgate.net/directory/publications>
- [19]. Open Information Extraction from the Web Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead and Oren Etzioni, Department of Computer Science and Engineering University of Washington.
- [20]. <https://bl.ocks.org/mbostock/2e12b0bd732e7fe4000e2d11ecab0268>
- [21]. <http://bl.ocks.org/mbostock/4339083>