**RESEARCH PAPERS SEARCH ENGINE**

**PHASE 1 – DATA CRAWLING AND INDEXING**

**ABSTRACT – 16PW24, 16PW33**

**Project Abstract :**

The aim is to develop a search engine which would search for papers related to a paper of interest to the user. It displays the user a list of relevant papers to the query. All the documents shown contain at least one of the query words and in the results shown, all query words are bold marked. The entire project is implemented in python.

**Scraping:**

The preliminary step to develop our search engine is to crawl from web to gather a huge corpus of research papers.

arXiv is a free distribution service and an open-access archive for 1,774,607 scholarly articles in the fields of physics, mathematics, computer science, quantitative biology, quantitative finance, statistics, electrical engineering and systems science, and economics.

arXiv supports and participates in the [Open Archives Initiative](http://www.openarchives.org/) (OAI). arXiv is a registered OAI-PMH [data-provider](http://www.openarchives.org/Register/BrowseSites.pl) and provides metadata. Each article in arXiv is modelled as an Item in the OAI-PMH interface. Metadata for each item (article) is available in several formats. We are going to use arxiv format [1].

The goal is to scrape the arxiv documents based on their categories (computer science, economics, physics etc.,) from arxiv website in XML format. Then the XML based content is parsed into json and the file is stored. The title, abstract, authors, URL, category, created date of the paper are stored.

PACKAGES TO BE USED:

* xml
* urlencode
* urlopen

**Indexing:**

The json file holds all the document information. For each document extract the title, abstract and authors and follow the pre-processing steps.

Pre-processing Steps:

* Tokenize
* Remove stop words
* Lemmatize
* Convert to lowercase

An inverted index is created. For each term, we can see which document contains the term, which enables to efficiently match documents containing specific terms. The terms are weighted using tf-idf value. Persistent indices allow for quick retrieval of previously indexed information. Persistence is to be achieved through use of the Python pickle module. This feature allows an index to be saved to disk and later restored, to avoid reindexing data.

PACKAGES TO BE USED:

* nltk
* pickle
* json

**References:**

1. http://export.arxiv.org/oai2?verb=GetRecord&identifier=oai:arXiv.org:0804.2273&metadataPrefix=arXiv