Applications of Big Data Techniques to Astronomy

BY

B HAREESH GAUTHAM
2011B5A7536H

Under the supervision of

Prof. Ajit Kembhavi
Emeritus Professor,
Inter-University Centre for Astronomy and Astrophysics.

and

Mr. Sanjeev Kumar Singh
Lecturer,
Birla Institute of Technology and Science,
Pilani-Hyderabad Campus.

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI-HYDERABAD CAMPUS

April 29, 2016

ACKNOWLEDGEMENTS

I would like to thank Prof. Ajit Kembhavi and Mr. Sanjeev Kumar for their help and guidance. I express my gratitude to Dr. Sheelu Abraham who helped me a lot throughout the course of this thesis work. She is also one of the authors of the catalogue which is used primarily used in this thesis work. I would like to thank Mr. Kaustubh Vaghmare, who helped me in writing the software and offered useful suggestions on the how results should be presented. I am grateful to Dr. Ashish Mahabal and Dr. Yogesh Wadadekar who provided insights on astronomical aspects of the problem. I would also like to thank IUCAA and the Computer science department of BITS-PILANI, Hyderabad Campus for providing me this opportunity.

CERTIFICATE

This is to certify that the thesis entitled, "APPLICATIONS OF BIG DATA TECHNIQUES TO ASTRONOMY" and submitted by, B Hareesh Gautham, ID No. 2011B5A7536H in partial fulfillment of the requirement of BITS F421T Thesis embodies the work done by him under my supervision.

Signature of the supervisor:

Name:
Prof. Ajit Kembhavi
Emeritus Professor,
Inter-University Centre for Astronomy and Astrophysics.

Date: April 29, 2016

Contents

List of Figures

Chapter 1

Abstract

Spectral energy distribution(SED) is a plot of flux density against wavelength of the light emitted by an object. They can be used to classify celestial objects. Photometry data from different catalogues from different surveys (which usually operate at different wavelengths) needs to be collected and processed to build SEDs. To build SED for a given object, data from different catalogues needs to cross matched by their positions in the sky. Different catalogues store photometry data in different units. Hence, all the data needs to be converted to same units(usually Jy) before SEDs are plot.

A simple client program (which uses Table Access Protocol (TAP) and Astronomical Data Query Language(ADQL)) was written using linux sockets. This program sent queries to TAP servers to recieve photometry data from multiple catalogs. A crossmatching program which uses k-d tree data structure was also written. This allowed data to be processed locally. Extensions to the crossmatching program were written to assemble photometric data from multiple catalogues and then to convert them to common units.

Using the above programs SEDs were built for multiple objects. Some of these objects were confirmed to be stars and quasars. These were then compared to the SEDs of unknown objects. Since stars and quasars have distinct SEDs, unknown objects could be classified using their SEDs.

Chapter 2

Introduction

Quasars or quasi-stellar radio sources are sources of extremely luminous electromagnetic energy. They usually emit in radio and optical wavelengths and exhibit high redshifts. High redshifts indicates that most of them were formed approximately 12 billion year ago. They appear similar to stars but unlike stars their spectra has broad emmision lines. That is why they are called quasi stellar. Quasars were first discovered in late 1960s as radio sources(3C 48 and 3C 273) in the sky. It is now known that only ten percent of all known quasars are radio loud. The name quasi stellar objects(QSO) is used to refer to these objects. Quasars produce luminous energy at a rate comparable to a whole galaxy which typically contains about 10^{11} stars. Sloan Digital sky survey (SDSS) is an all sky which discovered most of known quasars. SDSS provides photometry data for about 357 million objects. Less than 1% of these have been spectroscopically observed. Hence the exact nature of most of the objects in the survey remain unknown. It is, therefore, necessary to develop techniques to identify objects reliably using photometric data.

"Photometric Catalogue of quasars and other point sources in the Sloan Digital Sky Survey"(Vizier ID: J/MNRAS/419/80) by Abraham et. al(2011) is one such attempt at classifying objects using photometry data. The clasification is done using a difference boosting neural network classifier which is a Bayesian supervised learning algorithm. Photometry data of 5 SDSS bands were used to get the corresponding 10 colors. Colors of spectroscopically confirmed stars and quasars were used to train the neural network. Unknown objects were then classified into stars, galaxies and quasars using the trained neural network. Catalogue has classified 2430625 objects as quasars, 3544036 as stars and 63586 as unresolved galaxies. As a part of this thesis work, the this catalog is used as the primary source of celestial objects.

Large catalogues of astronomical data are available from different sky surveys. Different sky surveys operate at different wavelengths, for example Sloan Digital Sky Survey (SDSS) is an optical survey, whereas Wide-field Infrared Survey Explorer (WISE) images in infrared wavelengths. Surveys like Catalina Real-Time Transient Survey (CRTS) scans a specific portion of the sky again

and again to observe transient phenomenon. To study objects in the sky across multiple wavelengths and times(epochs), data from different surveys needs to be correlated by their positions in the sky and hence analyzed. Spectral energy distributions(SEDs) (i.e. a plot of brightness of the object at different wavelengths) can be obtained which offers useful insights to classify an astronomical source. Using SEDs objects can be classified more reliably. Since a large amount of data is involved in such an analysis, application of big data techniques is essential.

Chapter 3

Basics of Astronomy

This chapter discusses some of the concepts from astronomy used in this thesis work.

3.1 Celestial Coordinates

Celestial coordinates specify the position of an object in the sky. They usually specify the direction in which the object can be found in the sky. There are many celestial coordinate systems which differ from each in their choice of fundamental plane(plane of zero latitude), center of coordinate system and primary direction(plane of zero longitude). Some of the common coordinate systems are horizontal, equitorial, ecliptic and galactic. In horizontal coordinate system, observer is at the center point, horizon is the fundamental plane while either north or south can be choosen as the fundamental direction. Since earth rotates around its own axis and revolves around the sun, coordinate of an object in horizontal system varies with time. In ecliptic system, fundamental plane is the plane of earth's orbit(also called ecliptic). It can be centered either at the center of sun(heliocentric) or at the center of earth(heliocentric). Vernal equinox is considered to be the fundamental direction. Galactic coordinate system uses galactic plane as the fundamental plane. It is centered at the center of sun. The direction of galactic center from the sun is taken as the fundamental direction. Equitorial coordinate system is centered at earth's center, it takes celestial equator(nothing but projection of earth's equator on celestial sphere) as fundamental plane and vernal equinox as the fundamental direction. Coordinates in this system are refered to as declination(latitude) and right ascension(longitude). Equitorial coordinates change beacuse of the precission of axis of earth. Hence, equitorial coordinates are associated with an epoch to indicate the specific reference frame in which the measurements are made.Some of the popular choices of epochs are B1950 and J2000. In this thesis work all coordinates are in equitorial coordinate system using J2000 epoch. Figure 3.1 shows the equilorial coordinate system.

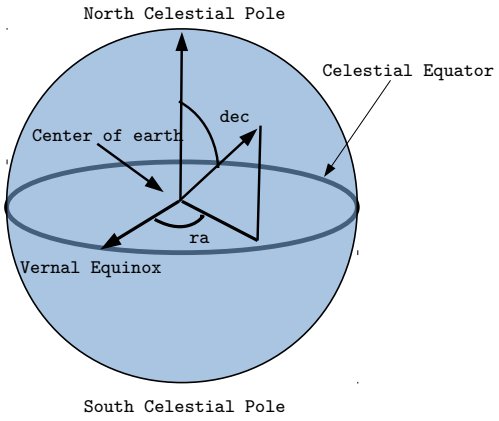


Figure 3.1: Equatorial coordinate system

3.2 Photometry

Almost all the information about celestial objects is recieved in the form of electromagnetic waves. Excepts parts of radio and optical wavelengths much of the electromagnetic spectrum is blocked by the atmosphere. To study objects at wavelengths blocked by the atmosphere one needs to observe them using telescopes and detectors out in space. Spectrophotometry aims to study an astronomical object by measuring the energy emitted by the object across multiple wavelengths. This is quantified by the spectral flux which is energy recieved per second, per unit area, per unit wavelength or frequency interval. To get true spectra, some sort of dispersion needs to be used which splits the light from celestial objects into different wavelengths. This requires large telescopes(to capture more photons) and can be done for relatively brighter objects. To study fainter objects, one can use filters that pass only certain wavelengths. Since no dispersion is used, more light is available (over broader waveband) for measurement. This is equivalent to low resolution spectrophotometry. Filter photometry (or photometry) is easier to do than spectrophotometry and requires less equipment. A set of well defined filters with a known sensitivity to incident radiation is known as a Photometric System.

There are many factors which effect the measurement of energy from a source. Earth's atmosphere absorbs the light from celestial objects. The amount of light absorbed depends on the wavelength of the light and changes with time.

The instrumental response to light depends on the reflecting surface of the mirror,the sensitivity of the filters and quantum efficiency of the detector. Hence it is very difficult to measure absolute spectral flux of an object. In practice only the ratio of flux with respect to some standard stars (called standard candles), like Vega is calculated. Doing so cancels the above mentioned factors as they affect the measurement of both the objects and hence not their ratio. Absolute fluxes of these stars have been carefully calculated which can be used to calculate absolute fluxes of other objects.

In Astronomy, magnitudes are used to quantify brightness of an object. Difference of magnitude of objects is defined by,

$$m_1 - m_2 = -2.5 \times log_{10}(f_1/f_2) \tag{3.1}$$

where f_1 and f_2 are fluxes. Brighter objects have smaller magnitude and vice versa. Star Vega is assigned a magnitude of zero (since it is a standard candle). Magnitude of any other star is given by,

$$m_1 = -2.5 \times log_{10}(f_1/f_{vega}) \tag{3.2}$$

Hence objects brighter than vega have negative magnitude and objects fainter than vega have positive magnitude. Astronomical colors are defined as the difference in magnitudes of different filters. They corresponds to ratio of fluxes in each of the filters. Astronomically, colors have more significance than the magnitude itself. Within a system of filters(also known as photometric system) colors can be measured more accurately than the magnitude.

Chapter 4

TAP SED Builder

IVOA(International Virtual Observatory Alliance) is a scientific organisation which facilitates access to data gathered by astronomical observatories. It develops standards to ease global and integrated access to astronomical data. This project uses two such standards namely TAP and ADQL. TAPVizieR (an implementation of TAP by CDS's VizieR service) is used in this work for acquiring the data to plot SEDs.

4.1 Table Access Protocol(TAP)

Table Access Protocol is a web-service protocol that gives access to large sets of astronomical catalogs. TAP servers accepts queries from various client programs. Queries can be posted in many languages(like ADQL, PQL and SQL) but the support for ADQL[1](Astronomical Data Query Language) is mandatory. After executing the queries, server (where the catalogues are stored) sends the corresponding data to the client in the format specifed by the client.

A TAP query can be made to execute in either synchronous or asynchronous mode. Support for both synchronous and asynchnorous execution is mandatory. Synchronous queries get an immediate response. Clients have to wait for the query to execute. Hence it is ideal for queries which execute quickly. Asynchronous queries require the clients and server(TAP service) share a state. Clients can use the state to get the status of execution, to get an estimate of how much time it will take for the execution of query and when the execution is complete, to retrieve the results. Queries which can take a lot of time should be executed in asynchronous mode.

TAP service is represented as a tree of web resources. The web resource at the root represent the service as a whole. For example, root for VizieR's TAP service(TAPVizieR) is represented by the URL http://TAPVizieR.u-strasbg.fr /TAPVizieR/tap/. Following web resources should be accesible as part of any TAP service:

[1]see section 3.2

- /sync This web resource is a direct child of the root URL. Requests sent to relative URL /sync get a synchronous reply.
- /async This web resource is a direct child of the root URL. Requests sent to relative URL /async get an asynchronous reply.

There are other resources like /availablity, /capabilities and /tables which i did not use in this project[2].

TAP serivce is implemented over HTTP using POST and GET methods. TAP clients specify key/value pairs of various parameters recognized by TAP, in the body of the HTTP form.

Asynchronous execution of queries require the server and client to store the state of the request. To post an asynchronous query, a HTTP POST request is sent to <root URL>/async/ with parameters REQUEST(=doQuery), LANG(the query language to be used, can be ADQL) and QUERY (query to be processed in the specified query language) specifed in the body of the HTTP form in proper encoding. TAP server replies by assigning a job identifier to the client (in a message with HTTP reponse code 303 "See Other"). In the figure shown below, the client is assigned a job identifier of 1455629057992A, which the client uses to control the execution of query and eventually when execution of is complete, retrieve the results. For example to track the progress of the execution of the query, a HTTP GET request can be sent to <root URL>/async/<job identifier>/phase/, which returns one of these values: QUEUED, COMPLETED, ERROR, EXECUTING or ABORTED. Once the execution of query is complete,HTTP GET request to the URL <root URL>/async/<job identifier>/phase/ returns COMPLETED. The results can then be retrived from the relative URL /results/result/. A job in executing or queued phase can be aborted with a HTTP POST request to <root URL>/async/<job identifier>/phase/. Following figure shows a typical exchange of message between TAP server and client.

4.2 Astronomical Data Query Language (ADQL)

The Astronomical Data Query Language (ADQL) is the language used by the IVOA to represent astronomy queries. ADQL is an extension to SQL to support queries that are specific to astronomy. It includes some geometries and geometric functions which simplify astronomical queries.

Following are some of the geometries specified in ADQL:

- POINT('coordinate system', right ascension, declination) which defines a point with the coordinates in the specified coordinate system.
- CIRCLE('coordinate system',right ascension, declination, radius) This function expresses a circular region on the sky (a cone in space) around the coordinates (right ascension, declination) in the specified coordinate system. The radius must be in degrees.

[2]see [4]for complete TAP specification

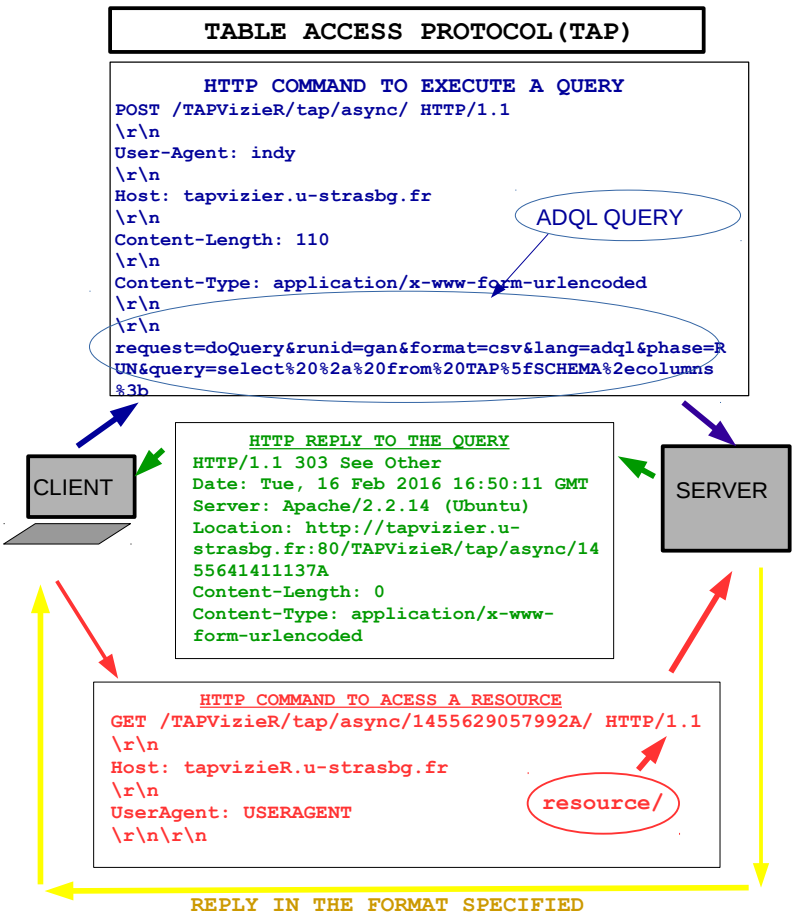


Figure 4.1: Exchange of messages between a TAP server and client

- BOX('coordinate system', right ascension center, declination center, width, height) This function defines a BOX centered in a position(right ascension center, declination center) of width "width" in degrees and height "height" in degrees.

- POLYGON('coordinate system', coordinate point,.....,coordinate point) This function expresses a region on the sky with sides denoted by great circles passing through specified coordinates. The coordinates of each point are defined by a right ascension and a declination expressed in degrees and in the specified coordinate system.

Following are some of the geometric functions, ADQL defines:

- DISTANCE(point1, point2) This function returns distance between two points. Arguments to the functions are should be the POINT geometry defined above.

- CONTAINS(region1, region2) This function returns true if region2 is within region1, false otherwise. Regions can any combination of the above defined geometries.

- INTERSECTS(region1, region2) This function returns true if region2 intersect region1, false otherwise. Regions can any combination of the above defined geometries.

The above geometries and functions are very useful for querying large catalogs. Following query crossmatches two different catalogues:

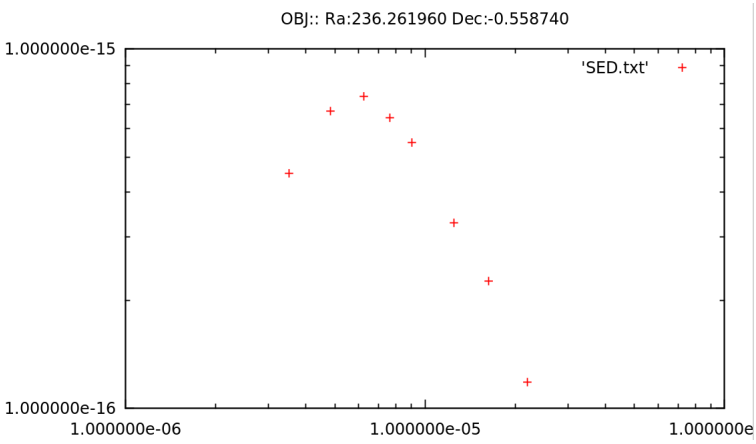
```
SELECT *
FROM "J/MNRAS/419/80/catalog","II/319/las9"
WHERE
1=CONTAINS(POINT('ICRS',"II/319/las9".RAJ2000,"II/319/las9".DEJ2000),
CIRCLE(
'ICRS',"J/MNRAS/419/80/catalog".RAJ2000,"J/MNRAS/419/80/catalog".DEJ2000, 2/3600.)
)
```

Here, "J/MNRAS/419/80/catalog" and "II/319/las9" are names of the catalogs used for crossmatch. For each point in the "J/MNRAS/419/80/catalog" catalog a cone search is performed on the "II/319/las9" catalog . The radius of the cone search is 2/3600 i.e. 2 arcseconds.

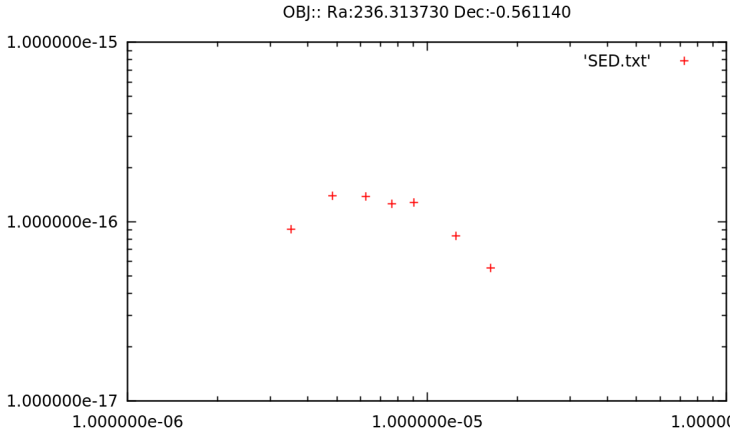
4.3 The Client Program

Using the above two IVOA standards a client program was written in C using linux sockets to build SEDs. Individual objects were crossmatched using using the ADQL queries posted through the TAP service. Only the nearest object in the query result was plot in the SEDs. Photometric measurements from different catalogues need to be converted to same units before being plot in a SED. This

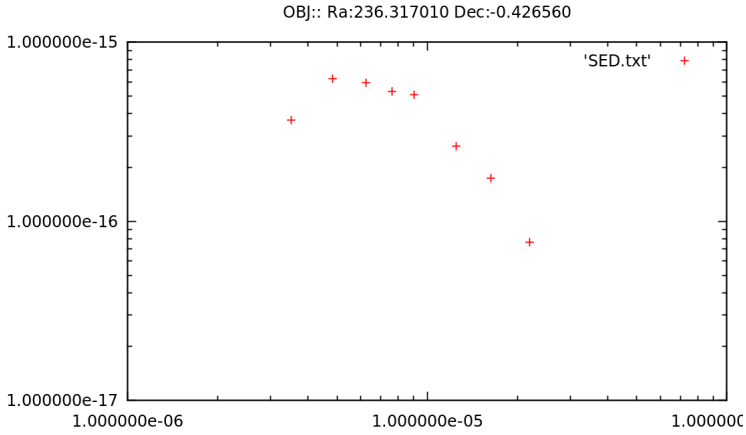
is done using various META tables in VizieR: METAtab,METAcols,METAsed and METAfltr. The results were compared against the web-based VizieR's photometry viewer[2]. Results from both were in agreement with each other. Gnuplot was used to plot the SEDs. Following are some of the SEDs plot using the client program.



(a)



(b)



(c)

Figure 4.2: SEDs plot using the client program. Flux density (in Jy, 1e-24 watts) is plotted against wavelength(in micro meter) for three random objects from the 6 million candidate quasar objects. Both x and y axes are plot in log scale.

Chapter 5

Crossmatching Program

The problem of crossmatching is essentially of searching for objects in a set which are within a specified distance to the given object. This is essentially a range searching problem which can be solved by using many methods. Brute force method would be to do it in linear time by browsing through the whole set of objects. For higher dimensions this method can outperform space partitioning approch. Space partitioning approach partitions the search space (into disjoint subsets) using a data structure to reduce search time. Two such data structures are k-d tree and R tree. R-tree uses a minimum bounding rectangle (hence the letter R) to group nearby objects. When done recursively, a hierarchy is formed using which a tree can be built. Each leaf represents an individual object. The other data structure, k-d Tree is used in this thesis work:

5.1 k-d Tree

k-d tree stores k dimensional data(hence the name) in the form of a binary tree. Every node in the k-d tree corresponds to some point in the data set and every point in the data set has a corresponding node in the tree. The tree is constructed by recursively partitioning the search space (at each node of the tree) in one of the k dimensions(called the splitting dimension) using the points in the data set. Every node of k-d tree has two attributes, its splitting dimension and the point it corresponds to. At any node of a k-d tree, all points whose splitting dimension coordinate is less than the splitting dimension coordinate of current node's point belong to left subtree. all other nodes belong to the right subtree. The splitting dimension is choosen either in a cyclic manner or in a way which is optimal for the given data. It should be noted that all nodes of a given level in the tree have the same splitting dimension. It represents the dimension along which the search space is partitioned at the current node. For all nodes of a given level the splitting dimension(if choosen in a cyclic manner) is given by:

$$splittingDim = (level - 1) \mod k \tag{5.1}$$

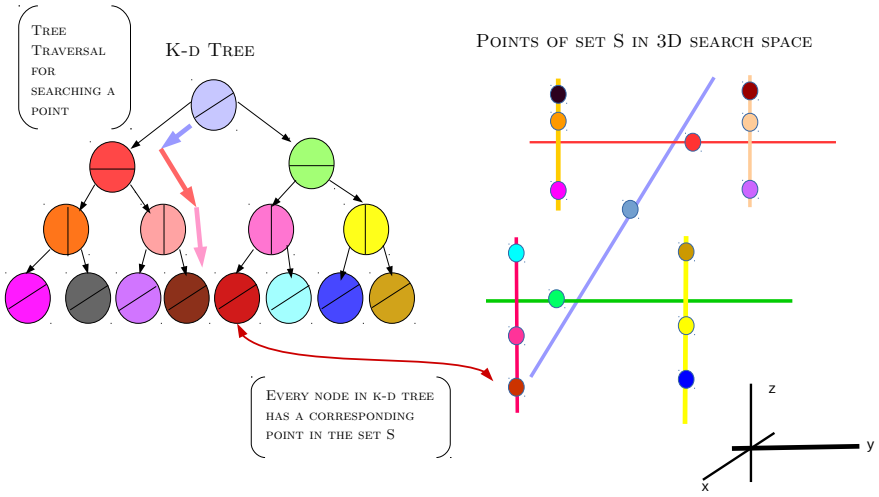


Figure 5.1: This figure shows the construction of k-d tree. Both the points in the three dimensional data set and the corresponding k-d tree is illustrated. Each node in the k-d tree has a corresponding point in the data set. In this figure, nodes have the same color as the point they represent. Lines on the nodes represent their splitting dimension. a slant line represents x axis, a vertical line represents z axis and a horizontal line represents y axis. It can be seen that all nodes of same level have same splitting dimension. Splitting dimension is given by the relation 4.1. It should be noted for that, for any node, all points in the left subtree have a smaller value of splitting dimension coordinate and opposite is true for right subtree. The three coordinate axes are also given for reference.

Searching for an object in k-d tree can be done by traversing the tree. Since it is a binary tree, a simple search would take O(log n) time. K-d tree is a k dimensional generalization of the of a binary search tree(which we can call 1-d tree).

Range searches can be done efficiently using k-d tree. To search for all the objects in the data set which are within a given distance from some query object, the corresponding k-d tree of the data set is traversed. It is traversed in the same way a binary tree is traversed, except in this case the current node's splitting dimension's coordinates are compared. k-d tree divides the space into disjoint regions. Each node of a k-d tree represents a plane of constant coordinate value.

For example, in 2-d, each node represents a line of either constant x or constant y value. While traversing the tree, it should be ensured that if the search region cuts the current node's plane of constant coordinate value, both subtrees of the given node must be traversed.

5.1.1 Implementation

k-d Tree can be constructed in many ways. A trivial way is to add each point of the data set by traversing throught the already constructed tree and creating a new node as a child of one of the leaves. This method doesnot always lead to balanced trees. For k-d Tree range searches to be efficient. trees need to be balanced. One way to do that is to use O(n) median finding algorithm to select points along which to split the space. Doing so will ensure that there will be equal number of points on either side of the split and hence equal number of nodes in both the subtrees. So the steps involved in constructing a k-d tree using such a algorithm would be:

```
buildK-DTree(p,sDim)
//p is the data set array
//sDim is the splitting dimension
{
splittingPoint=median(p); ///get the median point
newNode=NULL; //create new node
pl=getLTSubSet(splittingPoint,sDim); //get subset of points whose
splitting dimension's coordinate is less than
splitting point's splitting dimension's coordinate.

pg=getGTSubSet(splittingPoint,sDim); //get subset of points whose
splitting dimension's coordinate is greater than
splitting point's splitting dimension's coordinate.

node->sDim; //assign splitting dimension
node->left=buildK-DTree(pl,(sDim+1)%k); // build left subtree
node->right=buildK-DTree(pg,(sDim+1)%k); //build right subtree
return node;
}
```

In the above pseudo code, let n be the number of points in the k dimensional array p. Median finding algorithm takes O(n) time. Splitting the array to two sets will take another O(n) time. The arguments of two recursive calls (pl and pg) will have approximately half the number of points. Hence the time complexity of of the above method is given by the recurrence relation, $T(n) = 2T(n/2) + O(n)$,which has the solution of $T(n) = O(nlog(n))$.

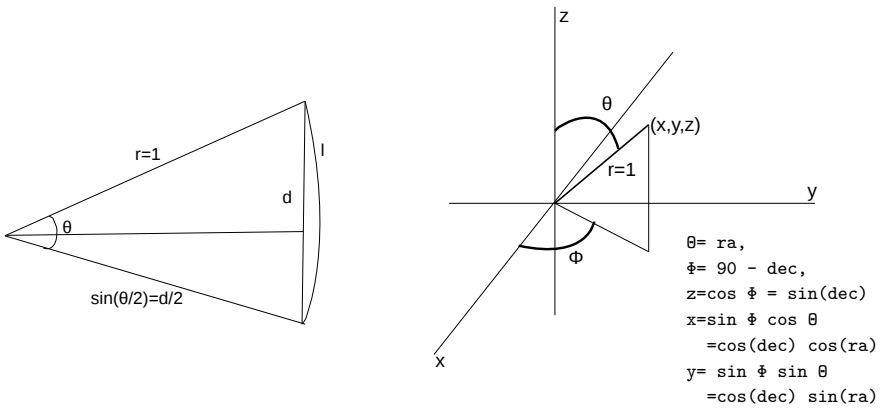
Instead of using a complex O(n) median finding algorithm one can use sorting to find the splitting points. In this method, the data is presorted in each of the

k dimensions. k sorted lists (one for each dimension) is formed by sorting the points in the data set by the coordinates of the dimension the list corresponds to. It should be noted that sorted lists contain only indices of the data and not the data itself. Only the indices to the data array are shuffeled during the course of the algorithm and not the data itself. Given a splitting dimension, median of the corresponding sorted list of points is chosen and the data is split across the choosen point(trivially). The arrays which correspond to other k-1 dimensions are shuffeled to reflect the partitioning (while preserving their sorted order) . A node is formed for the choosen point. Points above the splitting point in the splitting dimension sorted list form the left subtree and those below form the right subtree. The whole process is repeated till all points are added to the tree. Following algorithm illustrates the process:

```
S[N][k]; //input array

sortedArray[N][k];  // to store sorted points' indices.

sort(S,sortedArray) // to sort each column
of the newArray by the corresonding coordinate values

if(all elements of S are not in Tree):

find the median of the current splitting dimension's sorted array

create node using the median and current splitting dimension

shuffle the other columns of the newArray to match splitting
while preserving the sorting order.

recurse on the other upper and lower halves of the
current sorted array
```

Sorting for k dimensions takes O(knlog(n)) complexity. Recursive partitioning is done for $log(n)$ levels. Copying elements of sorted lists to do the shuffeleing requires another $O((k+1)nlog(n))$ time. Hence the overall complexity is $O(nklog(n))$. This algorithm requires a storage of n k-dimensional arrays(to store the data set), a n-element temporary array, and a k n-dimensional index arrays(for sorted lists).

An alternative to the above method is to use a smaller randomly choosen subset of points and sort them and use those sorted points to split the space. This method works well in practice. Parallel execution can be easily achieved for range searching problem. The data to be searched can be partitioned among as many processes as one wants and then the search can be performed independently. After the search is done all processes can be made to send their respective results to a master process which can then write the them to file.



(a) (b)

Figure 5.2: Convertion of coordinates from (ra,dec) to cartesian(x,y,z). This is done because k-d tree range search works for eucledian distances. Part(a) show how to convert angular radius search(θ) to distance between the points, d. Part(b) shows how to convert from (ra,dec) to (x,y,z)

Since k-d tree works only on euclidiean distances and all positions in astronomy are given in angles(Right acession and declination), one need to convert the points to corresponding three dimensional coordinates. Figure 4.2 illustrates how to do that.

5.1.2 Performance

The code for $O(knlog(n))$ k-d tree was implemented in c using the method detailed in Brown, 2015. Experiments were performed on a 2.2 GHz Intel i7 processor. Random catalogues were built and crossmatched to analyze the performance of the program. Figure 4.3 shows the time taken to build the k-d tree(x-axis) (including the sorting) plotted against the number of points in the data set(in the form of nlog(n))(y-axis). The data is plot for the range, $2^{24} \geq n \geq 2^{18}$. Plot is fit to a straight line which adequately establishes $O(nlog(n))$ dependence of time taken to build the tree. Time taken to read data and convert them to cartesian coordinates is not included in the results.

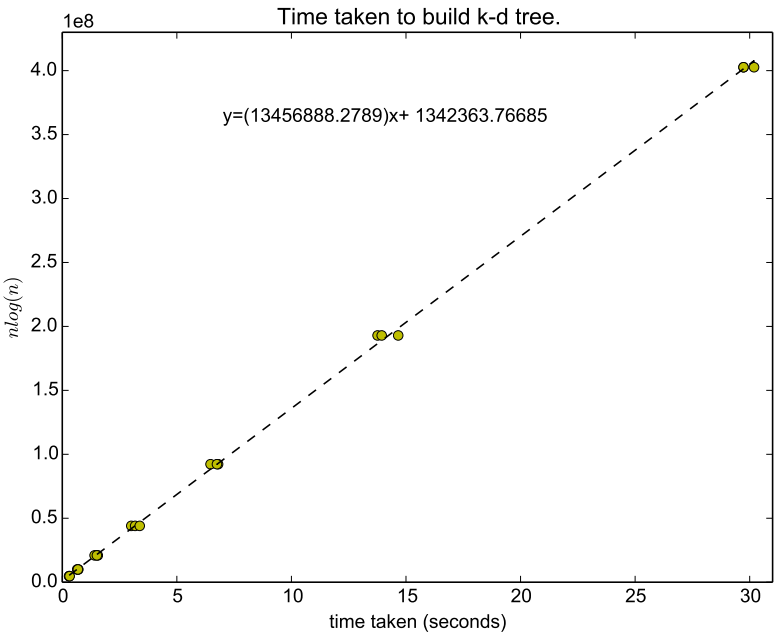


Figure 5.3: Number of points in the data set(in the form of $nlog(n)$) vs total time taken to build the k-d tree for randomly generated 3 dimensional data set.

Chapter 6

Results

Photometry based catalogue by Abraham et. al (vizier ID: J/MNRAS/419/80/catalog) was used as the primary catalogue to do crossmatches. It has about 6 million objects out of which 2,430,625 have been classified as quasars, 3,544,036 as stars and 63,586 as unresolved galaxies. This catalogue was crossmatched with a catalogue of confirmed quasars, The Half Million Quasars Catalogue(HMQ) (Vizier Id: VII/273). There are 510764 quasars in this catalogues . Following are the statistics of the crossmatch:

Cat. Name(Vizier ID)	No. of objects	Pos. error
The Half Million Quasars catalogue(VII/273/hmq)	510764	0.1”
Photometric catalogue based on SDSS DR7 (J/MNRAS/419/80/catalog)	6038247	2”

Results

No. of Objects Crossmatched	Search radius	Average distance
230973	5”	0.083”

The crossmatched result contains quasars which belong to both HMQ catalogue and our primary catalogue and has 230973 objects. These objects were crossmatched with other photometric catalogues to generate SEDs. The catalogs which were used to do this include, SDSS DR7(Vizier Id. II/294), 2MASS (II/246/out), ALLWISE(II/328/allwise) and FIRST(VIII/92/first14). SDSS has an rms error of 0.1” , 2mass has an error of 0.1-0.2”, allwise has an error of 0.01” and FIRST has an error of 1” in their respective astrometry. These catalogues have photometric data at different wavelengths. All the data is then converted to mJy and assembled to get SEDs. Following are the statistics of the above said crossmatches:

2MASS crossmatch results

No. of Objects	Search radius	Average distance
10063	2”	0.36”

ALLWISE crossmatch results

No. of Objects	Search radius	Average distance
194298	2”	0.37”

SDSS crossmatch results

No. of Objects	Search radius	Average distance
231176	2”	0.04”

FIRST crossmatch results

No. of Objects	Search radius	Average distance
13685	2”	0.40”

Figure 4.1 contains four SEDs built from the above crossmatches.

Similar exercise is repeated for a SDSS catalogue of spectroscopically confirmed stars. It has 78,597 objects. This catalogue was crossmatched to other photometric catalogues to build SEDs. Following are the statistics of the crossmatching exercise.

2MASS crossmatch results

No. of Objects	Search radius	Average distance
30204	2”	0.32”

ALLWISE crossmatch results

No. of Objects	Search radius	Average distance
47380	2”	0.44”

SDSS crossmatch results

No. of Objects	Search radius	Average distance
80619	2”	0.09”

FIRST crossmatch results

No. of Objects	Search radius	Average distance
588	2”	1.30”

Figure 5.2 comprises of SEDs built from the above crossmatches. One can see a clear distinction between SEDs of stars and quasars. Figure 5.3 consists of SEDs of four random objects from our primary catalogue. It is apparent that the first and last plots are of stars and other two are of quasars.

Since the primary catalogue used in this thesis work contains classification of objects into stars, quasars and galaxies, the crossmatch results were analysed for the correctedness of the classification. Following are the results of the analysis.

Half million quasar catalogue(SDSS) crossmatched with Photometric catalogue.

Cat. Name(Vizier ID)	No. of objects	Pos. error
The Half Million Quasars catalogue(VII/273/hmq)	510764	2”
Photometric catalogue based on SDSS DR7 (J/MNRAS/419/80/catalog)	6038247	0.1”

Results

No. of Objects Crossmatched	Search radius	Average distance
230973	5”	0.083”

Classification of Results

	Star	Quasar	Galaxies
No.of objects	17677 (7.65%)	212797(92.13%)	504(0.22%)
Average dist.	0.19”	0.07”	0.17”
Avg. Prob. %	97.5	99.02	94.1

Star catalogue(SDSS) crossmatched with Photometric catalogue.

Cat. Name(Vizier ID)	No. of objects	Pos. error
Star Catalogue	78597	0.1”
Photometric catalogue based on SDSS DR7 (J/MNRAS/419/80/catalog)	6038247	0.1”

Results

No. of Objects Crossmatched	Search radius	Average distance
16638	1”	0.04”

Classification of Results

	Star	Quasar	Galaxies
No.of objects	16495 (99.14%)	138 (0.83%)	5 (0.03%)
Average dist.	0.04”	0.05”	0.09”
Avg. Prob. %	99.85	95.59	100.0

Results of the crossmatch are hence statistically consistent with the classification of objects in the catalog.

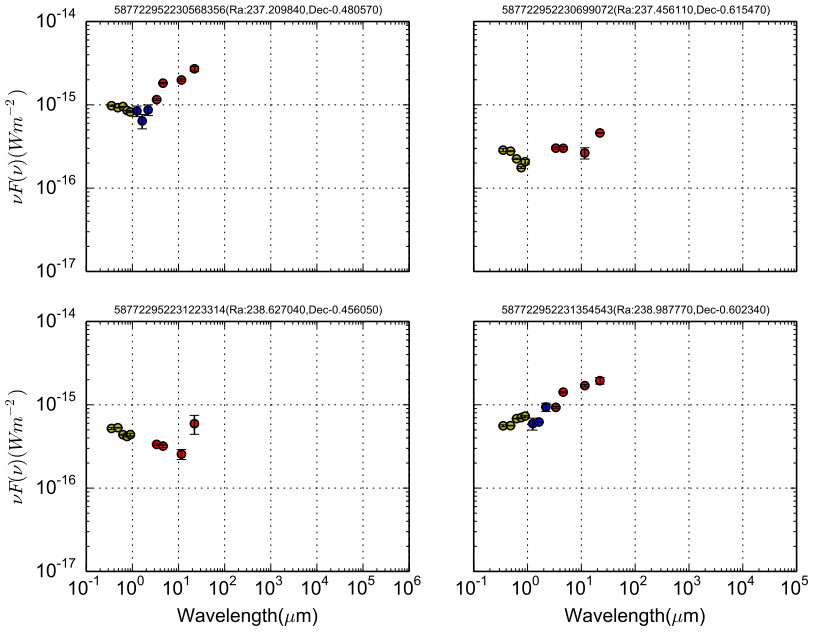


Figure 6.1: SEDs built for a set of quasars which belong to both the Photometric catalog and the Half million quasar catalog(HMQ). These objects were crossmatched with SDSS DR7, ALLWISE and 2MASS. search radius of 2" was used for all these crossmatches. Objects in the above figures are represented by their SDSS IDs and coordinates. Yellow points are from SDSS, blue points from 2MASS and red from ALLWISE.

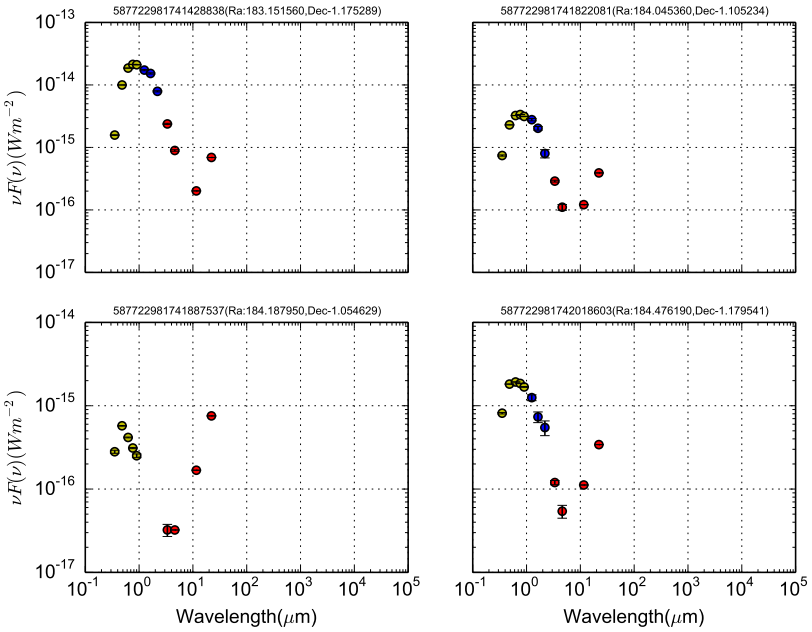


Figure 6.2: SEDs built for a set of stars. These objects were crossmatched with SDSS DR7, ALLWISE and 2MASS. search radius of 2" was used for all these crossmatches. Objects in the above figures are represented by their SDSS IDs and coordinates. Yellow points are from SDSS, blue points from 2MASS and red from ALLWISE.

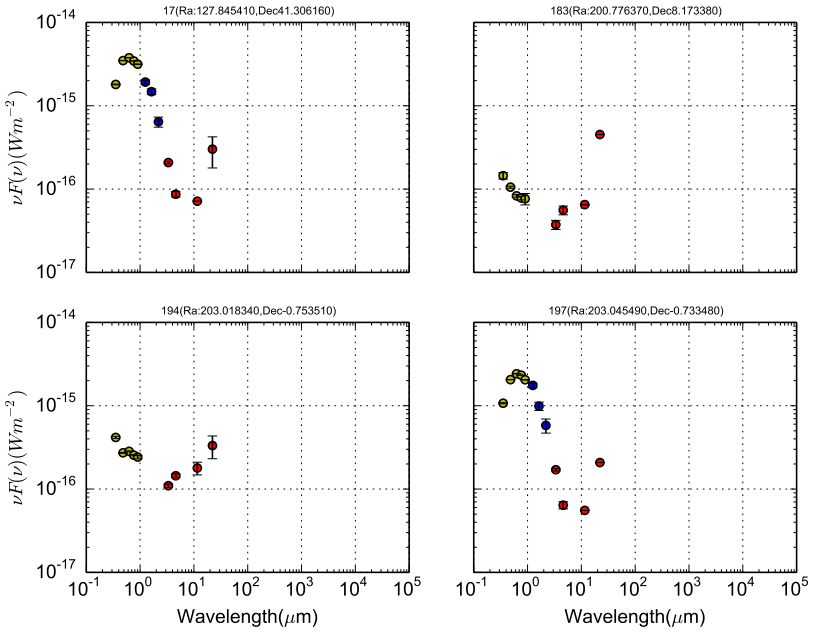


Figure 6.3: SEDs built for a set of random objects form the photometric catalog. These objects were crossmatched with SDSS DR7, ALLWISE and 2MASS. search radius of 2" was used for all these crossmatches. It is apparent visually that first and fourth objects are stars and other two are quasars. Yellow points are from SDSS, blue points from 2MASS and red from ALLWISE.

Chapter 7

Conclusions

SEDs offer a physical insight into the type of object under study. Various IVOA services including TAP and ADQL were used to build a client program which collects photometry data from multiple catalogues and builds SEDs (plots of flux density vs frequency/wavelength). These catalogues usually operate at different wavelengths and have different units of measurement. Using the data made available through META data tables, all the measurements have been converted to same units and hence plotted. A crossmatching program using k-d tree was also written to perform crossmatching locally. Using the crossmatching program, different catalogues were crossmatched and SEDs of multiple objects were built. These objects included spectroscopically confirmed quasars and stars. Stars and quasars have very distinct SEDs. Attempts to classify unknown objects into stars and quasars is made by analyzing their SEDs.

Bibliography

[1] Photometry viewer by Vizier: http://vizier.u-strasbg.fr/vizier/sed/

[2] Astronomy Catalog: Abraham S. et. al, A photometric catalogue of quasars and other point sources in the Sloan Digital Sky Survey (J/MNRAS/419/80/catalog).

[3] TAP client by Vizier: http://tapvizier.u-strasbg.fr/adql/

[4] IVOA TAP Recommendation: http://www.ivoa.net/documents/TAP /20100327/REC-TAP-1.0.pdf

[5] IVOA ADQL Recommendation: http://www.ivoa.net/documents/REC /ADQL/ADQL-20081030.pdf

[6] Book: Keir Davis, John W. Turner, and Nathan Yocom, The Definitive Guide to Linux Network Programming, Apress; 1 edition (August 3, 2004)

[7] CDS crossmatching service: http://cdsxmatch.u-strasbg.fr/xmatch.

[8] A Journal Paper: Russell A. Brown, Building a Balanced k -d Tree in O(kn log n) Time, Journal of Computer Graphics Techniques Vol. 4, No. 1, 2015.

[9] A Journal Paper: Sheelu Abraham, Ninan Sajeeth Philip, Ajit Kembhavi, Yogesh G. Wadadekar, and Rita Sinha, A photometric catalogue of quasars and other point sources in the Sloan Digital Sky Survey, Monthly Notices of the Royal Astronomical Society 419, 80-94 (2012).

[10] A Journal Paper: Multidimensional Binary Search Trees Used for Associative Searching, Multidimensional Binary Search Trees Used for Associative Searching, Communications of the ACM,Volume 18,Number 9, September 1975.

[11] A Journal Paper: Eric W. Flesch, The Half Million Quasars (HMQ) Catalogue, Pub. Astron. Soc. Australia 32, 10 (2015).

[12] Book: Ajit k. Kembhavi and Javanth V. Narlikar, Quasars and Active Galactic Nuclei : An Introduction, Cambridge University Press, 1999.

[13] Book: Computational Geometry An Introduction, Franco P. Preparata and Michael Ian Shamos, Springer-Verlag,1985.

[13] An Introduction to astrophysics, Second Edition. Baidyanath Basu, Tanuka Chattopadhyay and Sudhindra Nath Biswas, PHI Learning Private Ltd. , New Delhi , 2010.

[14] Book: An Introduction to Astronomical Photometry Using CCDs, W. Romanishin, University of Oklahoma.