International Journal of Scientific Research in Computer Science and Engineering
Research Paper
Vol.10, Issue.6, pp.69-80, December (2022)
E-ISSN: 2320-7639

F-GSC, A Flexible Graph-based Subject Classifier for Scientific Articles using Topic Hierarchy

Soumya George

Dept. of Computer Applications, St. George's College, Aruvithura, India

Author's Mail Id: mariyam.george@gmail.com

Available online at: www.isroset.org

Received: 19/Sept/2022, Accepted: 21/Nov/2022, Online: 31/Dec/2022

***Abstract*—** Subject classification of articles using a topic hierarchy or classifier systems enables scholarly search engines to retrieve relevant articles in a fast and efficient manner. This work proposes a flexible knowledge graph-based topic hierarchy that is so flexible as to add new keywords to it. This is an extension of our previous work, which uses a knowledge graph-based topic hierarchy for the subject classification of scholarly articles based on a sequence-based full-text indexing approach into six levels, including the relevant subjects, categories, areas, disciplines, fields, and keywords. The proposed work is tested for efficiency and reliability using the arXiv dataset.

***Keywords*—** Scholarly search engine; Knowledge graph; Topic hierarchy; Flexible classifier; Subject classification; Sequence graph

I. INTRODUCTION

Subject classification of indexed documents facilitates any information retrieval system to search and retrieve relevant results in a fast and efficient manner based on user queries. In order to enhance this feature, scholarly search engines require a well-defined topic hierarchy of keywords arranged under different subject categories. Usually, only metadata of the articles like title, abstract, author-supplied keywords, etc., are used to classify each scientific article.

Subject classification of scientific articles usually follows two methods: (i) insists the corresponding authors enter keywords either by themselves or to use a classification system imposed by the journal itself, e.g., ACMCCS Taxonomy ii) using subject experts to manually classify each article. [1-4]. Generally, authors hesitate to classify their articles based on a given taxonomy, and also it's difficult for authors to find a correct classification when their articles are of interdisciplinary nature. Classification using subject experts incurs high costs and is also time-consuming, but it's the best approach.

This work proposes a flexible topic hierarchy that easily allows the updating of new keywords. This is an extension of our previous work that uses a six-level pre-indexed topic hierarchy to automatically label the subject and subcategories of each article at the indexing time itself based on a sequence graph-based full-text indexing approach [5].

II. RELATED WORK

Subject labeling of articles needs a well-defined topic hierarchy to label each article into its relevant subjects and subfields. There are many classification systems available, including ACM CCS - ACM Computing Classification System, Microsoft academic graph dataset's (MAG) FieldsOfStudy.txt, and FieldOfStudy Hierarchy.txt files, IEEE Taxonomy, Springer Nature's NPG Subjects ontology, Mathematics Subject Classification system – MSC, etc. [6-10]. ACM Computing Classification System consists of only keywords that belong to the computing domain [6]. MAG dataset is a four-level taxonomy with 47,989 fields of study at level 3, 1,966 at level 2, 293 at level 1, and 18 at level 0 [7]. IEEE Taxonomy, created by IEEE, is a three-level hierarchy that can be used to classify articles into subject and field and consists of top-most terms of IEEE thesaurus [8]. NPG Subjects ontology consists of around 2750 terms that can be used to classify only the main subject area of the article [9]. Mathematics Subject Classification, MSC consists of only keywords in the area of mathematics [10]. The journal-based classifier systems include NSF classification of fields of study, WoS - Web of Science classification, ERA - Evaluation of Research Excellence classification, and ARC- Australian Research Council.

There are many existing manually developed classification systems. A three-level Classifier hierarchy is constructed in [11] of around 23,104 keywords that initially cluster documents based on citation relations and then subject label research areas by selecting highly relevant terms based on relevance score utilizing the metadata of each article. An open journal ontology, a three-level classification tree was constructed in [12] with multilingual in nature by using existing ERA, CHA, and ISI categories as seeds. The authors constructed a semantic topic hierarchy by extracting hypernym-hyponym pairs from the contents using the CiteSeerX dataset in [13].

Most of the available topic hierarchies are either subject-related like ACM CCS, MCS, etc., or don't have enough keywords to subject label the articles properly. Again most of the journals that have subject experts to classify articles or insist users enter the subject area of their submitted article classify each article into its main subject and category like 'computer science' as the subject with 'machine learning as a category. Scalability with flexibility is the main issue in most of the works, and it is not easy to update existing hierarchies automatically with new keywords.

III. KNOWLEDGE GRAPH-BASED SUBJECT LABELING OF SCHOLARLY ARTICLES – REVIEW OF THE PREVIOUS WORK

The Graph-based Subject Classifier, GSC used for the previous work is a six-level classifier hierarchy, which consists of a total of 81189 keys that belongs to 7 different subjects grouped into categories, areas, disciplines, fields, and keywords, as shown in Figure 1. A node will be created for each unique key with the name of the key and node type as 'key_phrase'. All multi-word keys will be expanded to their individual words in order as a sequence of nodes connected by meaningful relationships. A unique node will be created for each non-stop word term, and all stop words in between were concatenated into one string and stored as 'stop_word' property in the connecting edge in between the 'expand' and 'key_next' relationship. Each expanded key node word will be labeled with 'key' in order to distinguish it from other types of nodes in the graph.

The word Sequence Graph, WSG representation of a multi-word subject key 'mathematics and statistics, is shown in Figure 2.

Each key will be connected to its alias or abbreviations node using "abbrev" or "alias" type as shown in Figure 3, represented by blue and red relationships, respectively. Each document, 'di' in the collection, is represented as a sequence graph for each sentence by merging the contents into the pre-indexed classifier model to generate a Key Sequence Graph, KSG model for each document. A document node will be created for each document, and a key sequence graph will be created for the title and for each sentence in the abstract and contents with meaningful relationships to differentiate between the sentence type each belongs to.

All terms other than keys are considered as stop word terms in the Key Sequence Graph, KSG model. The first non-stop word and other non-stop words in the sentence of abstract will be connected with 'abstract' and 'abstract_next' relationship and with 'contents' and 'next_seq' for contents along with concatenated stopword string in between, sequence id, relationship id, etc. as edge properties. The algorithm used to create the KSG model representation of each article and the steps for primary labeling of keys are detailed in Algorithm 1.

Cermine, the Content Extractor, and Miner are used to extract different sections of a research article, and each article node will be connected to each of the reference files, authors, and other details like the already given subject or categories in the dataset. The sequence graph-based representation of an arXiv article is shown in Figure 4. Each article will then be subject-labeled after the primary labeling of keys that classify each article into its respective subjects, categories, areas, disciplines, fields, and keywords, as shown in Figure 5. The subject labeling algorithm used is detailed in Algorithm 2.

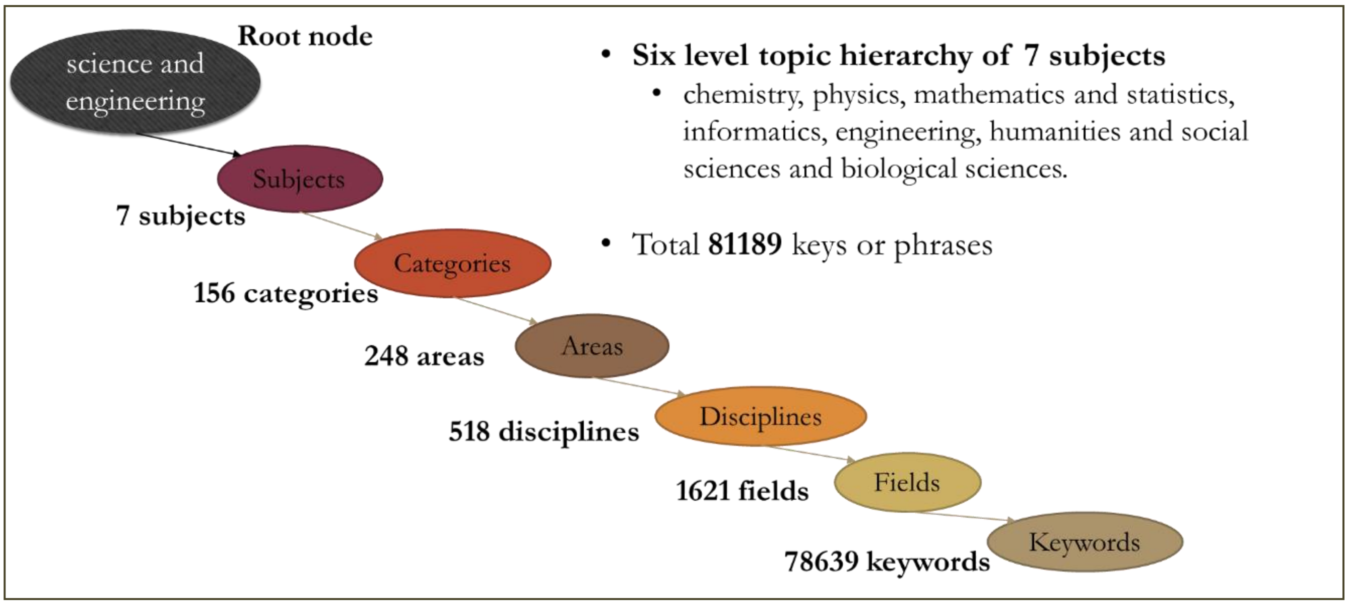

Root node
science and engineering
Subjects
7 subjects
Categories
156 categories
Areas
248 areas
Disciplines
518 disciplines
Fields
1621 fields
Keywords
78639 keywords
• Six level topic hierarchy of 7 subjects
• chemistry, physics, mathematics and statistics, informatics, engineering, humanities and social sciences and biological sciences.
• Total 81189 keys or phrases

Figure 1. GSC six-level topic hierarchy

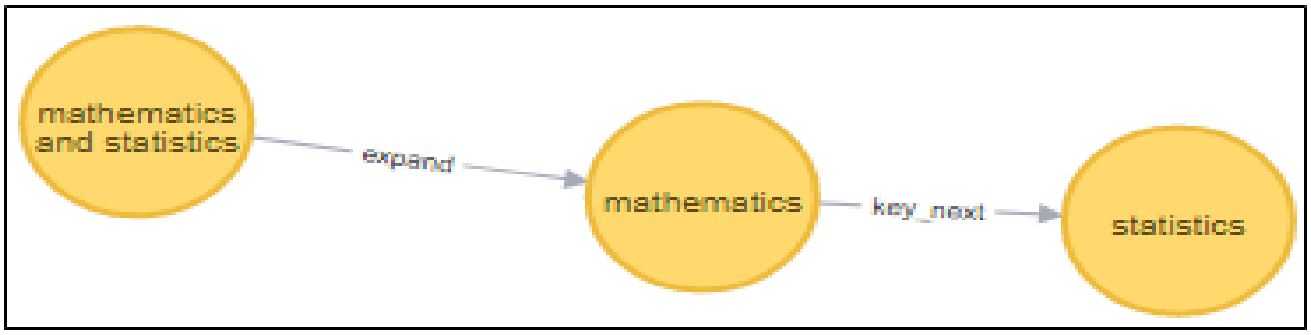


Figure 2. Sequence graph representation of keys

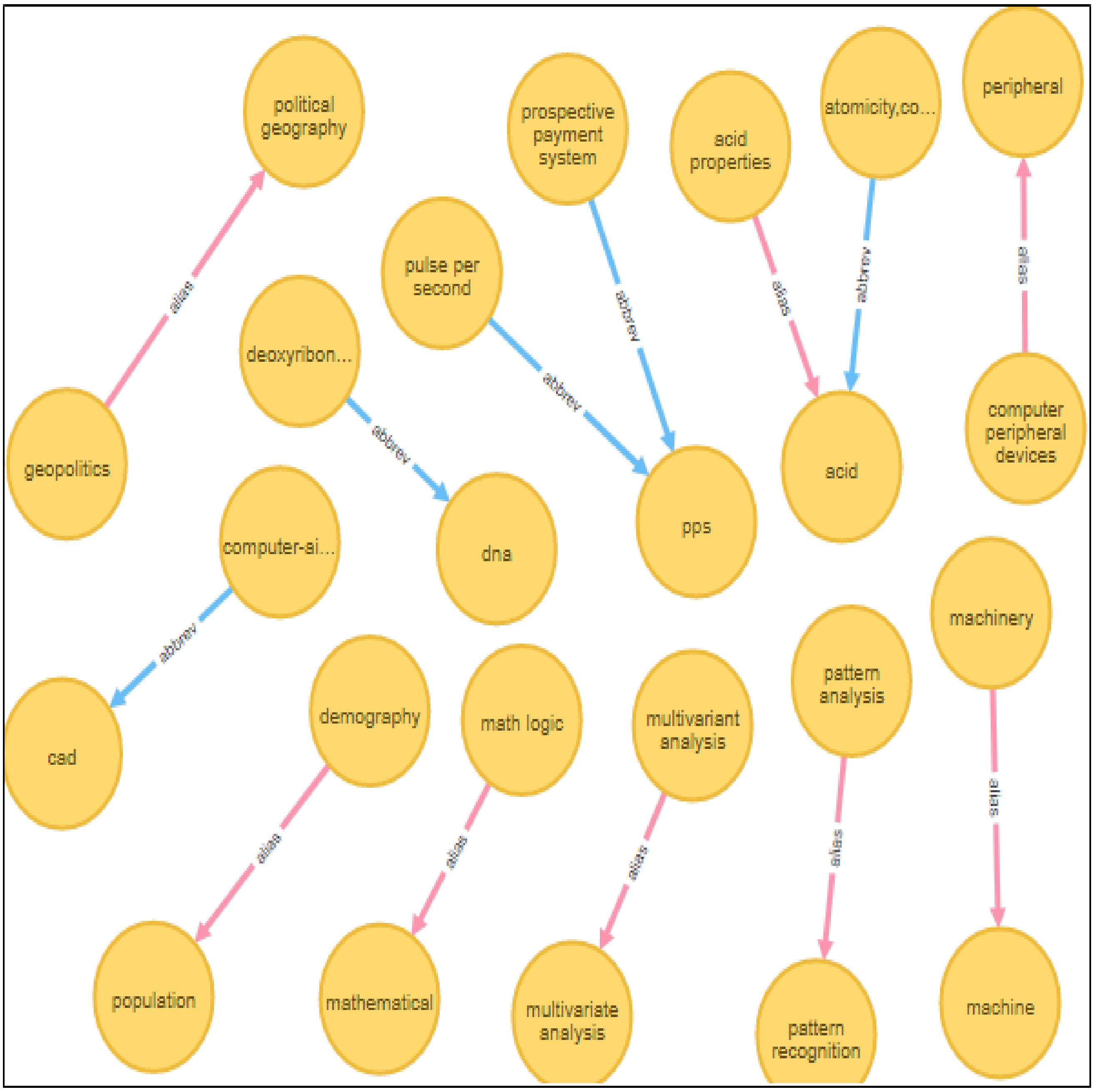
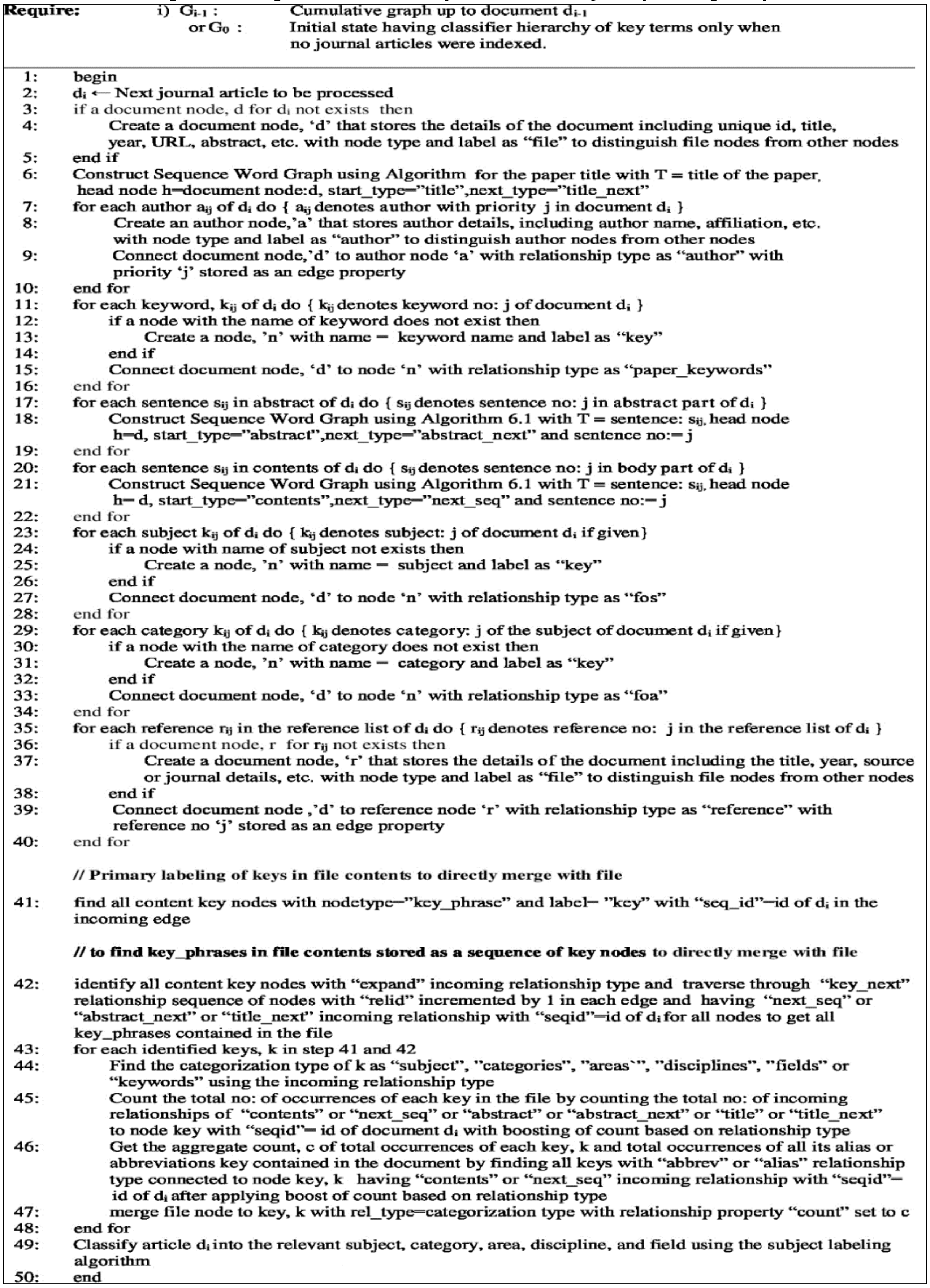


Figure 3. ‘alias’ and ‘abbrev’ relation of keys

Algorithm 1. Algorithm to index each journal article and primary labeling of keys



```
Require:     i) $G_{i-1}$ :   Cumulative graph up to document $d_{i-1}$
               or $G_0$ :     Initial state having classifier hierarchy of key terms only when
                              no journal articles were indexed.

 1:  begin
 2:  $d_i \leftarrow$ Next journal article to be processed
 3:  if a document node, d for $d_i$ not exists then
 4:      Create a document node, 'd' that stores the details of the document including unique id, title,
         year, URL, abstract, etc. with node type and label as "file" to distinguish file nodes from other nodes
 5:  end if
 6:  Construct Sequence Word Graph using Algorithm for the paper title with T = title of the paper,
      head node h=document node:d, start_type="title",next_type="title_next"
 7:  for each author $a_{ij}$ of $d_i$ do { $a_{ij}$ denotes author with priority j in document $d_i$ }
 8:       Create an author node,'a' that stores author details, including author name, affiliation, etc.
          with node type and label as "author" to distinguish author nodes from other nodes
 9:       Connect document node,'d' to author node 'a' with relationship type as "author" with
          priority 'j' stored as an edge property
10:  end for
11:  for each keyword, $k_{ij}$ of $d_i$ do { $k_{ij}$ denotes keyword no: j of document $d_i$ }
12:      if a node with the name of keyword does not exist then
13:          Create a node, 'n' with name = keyword name and label as "key"
14:      end if
15:      Connect document node, 'd' to node 'n' with relationship type as "paper_keywords"
16:  end for
17:  for each sentence $s_{ij}$ in abstract of $d_i$ do { $s_{ij}$ denotes sentence no: j in abstract part of $d_i$ }
18:      Construct Sequence Word Graph using Algorithm 6.1 with T = sentence: $s_{ij}$, head node
          h=d, start_type="abstract",next_type="abstract_next" and sentence no:= j
19:  end for
20:  for each sentence $s_{ij}$ in contents of $d_i$ do { $s_{ij}$ denotes sentence no: j in body part of $d_i$ }
21:      Construct Sequence Word Graph using Algorithm 6.1 with T = sentence: $s_{ij}$, head node
          h= d, start_type="contents",next_type="next_seq" and sentence no:= j
22:  end for
23:  for each subject $k_{ij}$ of $d_i$ do { $k_{ij}$ denotes subject: j of document $d_i$ if given}
24:      if a node with name of subject not exists then
25:          Create a node, 'n' with name = subject and label as "key"
26:      end if
27:      Connect document node, 'd' to node 'n' with relationship type as "fos"
28:  end for
29:  for each category $k_{ij}$ of $d_i$ do { $k_{ij}$ denotes category: j of the subject of document $d_i$ if given}
30:      if a node with the name of category does not exist then
31:          Create a node, 'n' with name = category and label as "key"
32:      end if
33:      Connect document node, 'd' to node 'n' with relationship type as "foa"
34:  end for
35:  for each reference $r_{ij}$ in the reference list of $d_i$ do { $r_{ij}$ denotes reference no: j in the reference list of $d_i$ }
36:      if a document node, r for $r_{ij}$ not exists then
37:          Create a document node, 'r' that stores the details of the document including the title, year, source
             or journal details, etc. with node type and label as "file" to distinguish file nodes from other nodes
38:      end if
39:       Connect document node ,'d' to reference node 'r' with relationship type as "reference" with
          reference no 'j' stored as an edge property
40:  end for

     // Primary labeling of keys in file contents to directly merge with file

41:  find all content key nodes with nodetype="key_phrase" and label= "key" with "seq_id"=id of $d_i$ in the
     incoming edge

     // to find key_phrases in file contents stored as a sequence of key nodes to directly merge with file

42:  identify all content key nodes with "expand" incoming relationship type and traverse through "key_next"
     relationship sequence of nodes with "relid" incremented by 1 in each edge and having "next_seq" or
     "abstract_next" or "title_next" incoming relationship with "seqid"=id of $d_i$ for all nodes to get all
     key_phrases contained in the file
43:  for each identified keys, k in step 41 and 42
44:      Find the categorization type of k as "subject", "categories", "areas`", "disciplines", "fields" or
         "keywords" using the incoming relationship type
45:      Count the total no: of occurrences of each key in the file by counting the total no: of incoming
         relationships of "contents" or "next_seq" or "abstract" or "abstract_next" or "title" or "title_next"
         to node key with "seqid"= id of document $d_i$ with boosting of count based on relationship type
46:      Get the aggregate count, c of total occurrences of each key, k and total occurrences of all its alias or
         abbreviations key contained in the document by finding all keys with "abbrev" or "alias" relationship
         type connected to node key, k having "contents" or "next_seq" incoming relationship with "seqid"=
          id of $d_i$ after applying boost of count based on relationship type
47:      merge file node to key, k with rel_type=categorization type with relationship property "count" set to c
48:  end for
49:  Classify article $d_i$ into the relevant subject, category, area, discipline, and field using the subject labeling
     algorithm
50:  end
```

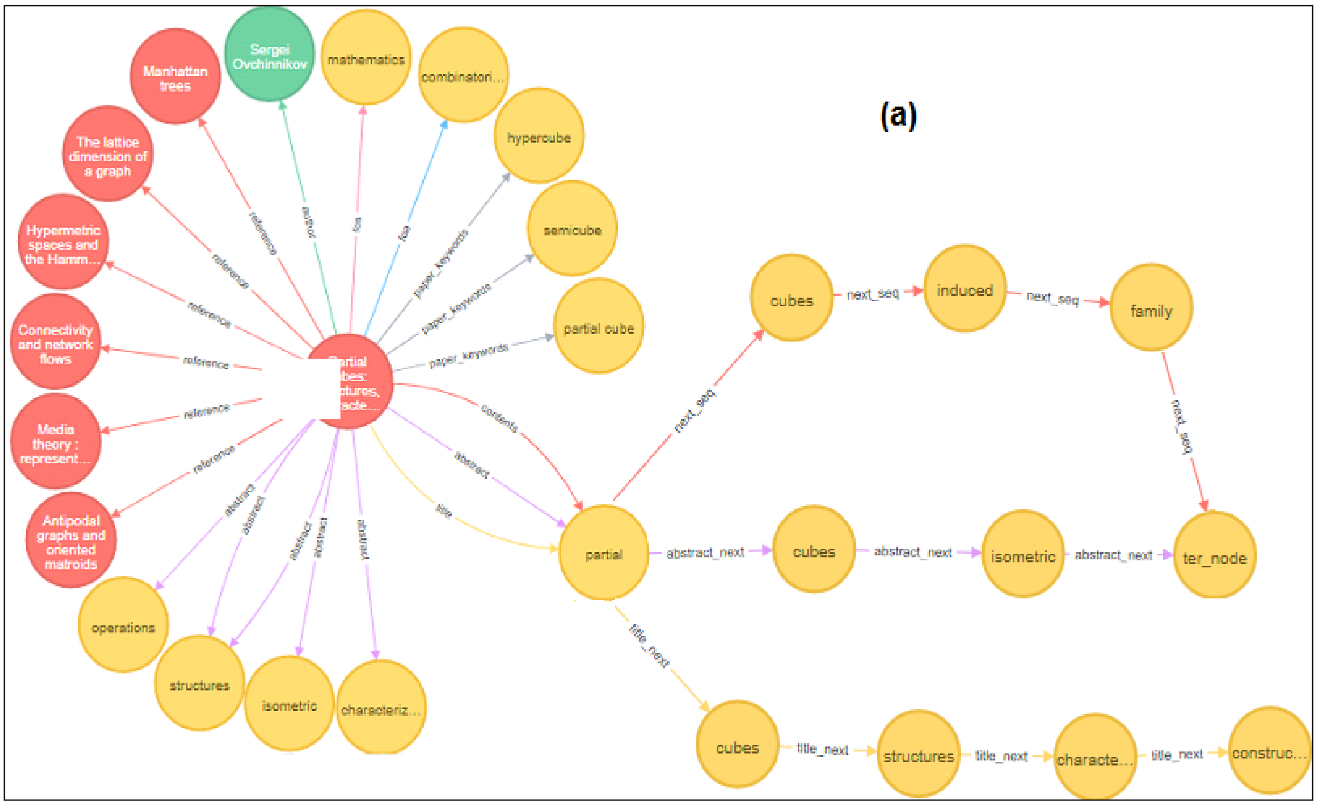


Figure 4. Graph-based representation of the article

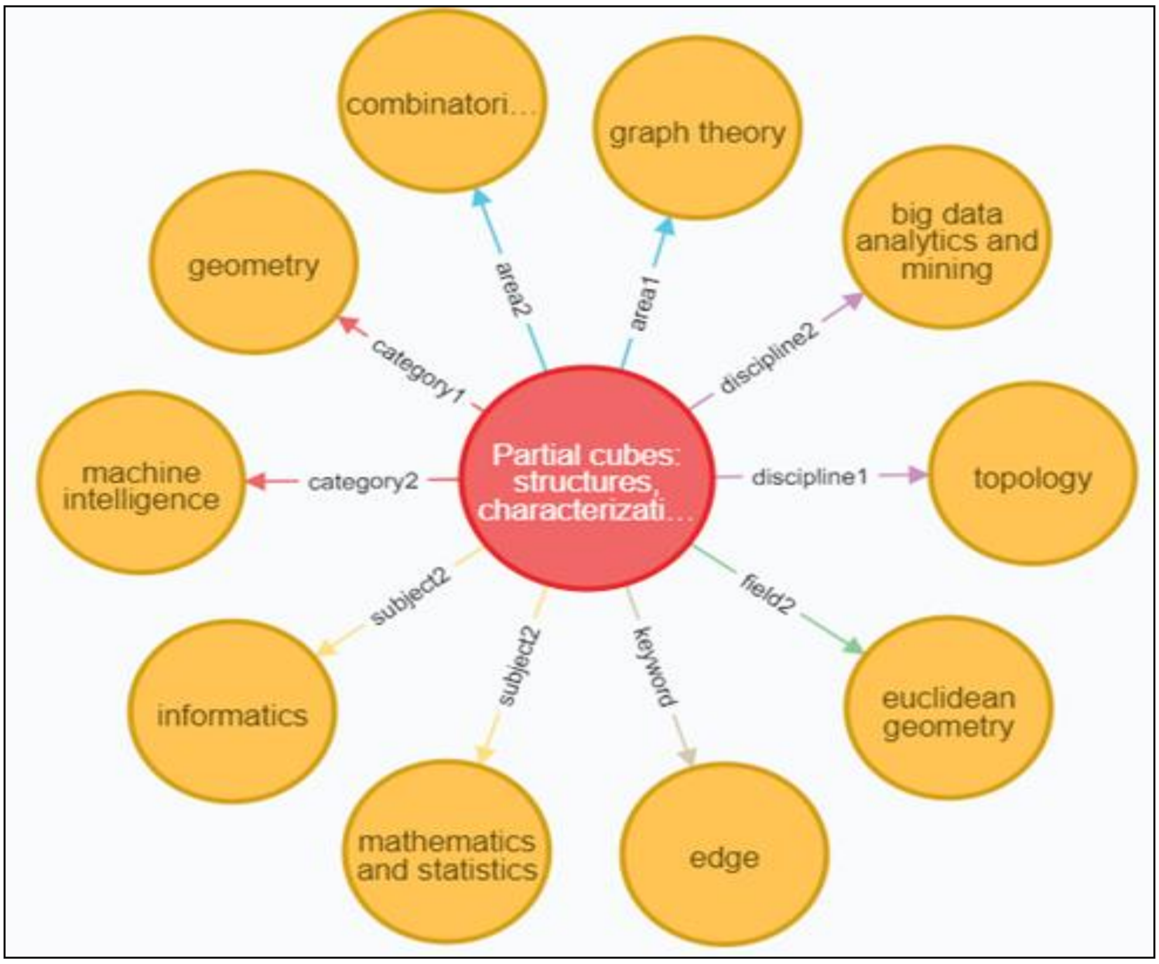


Figure 5. Subject labeled article

Algorithm 2. Subject labeling algorithm

```
Require:    i) $G_{i-1}$: Cumulative graph up to document $d_i$ with primary keys with count labeled for $d_i$

 1:   begin
 2:   $d_i$ ← Next journal article to be classified with primary labeling of keys done

      // based on keys of each categorization type present in article contents
 3:   for each categorization type ,t { subject, categories, areas, disciplines, fields}
 4:       max ← 0 {initially set to 0}, pkey← null {key with highest aggregate count for each categorization
          type,  t}, count← 0 {initially set to 0 }
 5:       for each key, k merged  to file by categorization type, t and count, c with classifier level $l_i$
 6:           set count = c
 7:           while `level $l_{i-1}$<=1{traverse from the current node to child nodes recursively till level 1,
              "keywords" }
 8:               for each key, p merged to both key k and file by categorization type, y and count, cc with
                  level $l_{i-1}$
 9:                   set count = count + cc
10:               end for
11:           end while
12:           if count > max then
13:               set pkey = k  and  max=count
14:           end if
15:       end for
16:       connect document node, '$d_i$' to node pkey with relationship type as categorization type, s of each
          categorization type , t and count = max  { s = "subject1" for  subject, "category1" for categories,
         "area1" for    areas,  "discipline1" for disciplines and  "field1" for fields }
17:   end for

      // based on all keys of each categorization type
18:   for each categorization type ,t { subject, categories, areas, disciplines, fields}
19:       max← 0 {initially set to 0}, smax ← 0 {initially set to 0, ,  needed only if "subject1" not present },
          pkey← null {key with highest aggregate count for each categorization type,  t}, skey← null   {key
          with next highest aggregate count , only for "subject"  categorization type,  if "subject1" not present},
          count← 0 {initially set to 0 }
20:       for each key, k of categorization type, t in level $l_i$
21:           repeat steps 7-14
22:            if subject1 does not present for $d_i$ then
23:                if count > smax and count < max
24:                    set skey=k and smax=count
25:               end if
26:           end if
27:       end for
28:       connect document node, '$d_i$' to node pkey with relationship type as categorization type, r appended
          with "2" of each categorization type , t and count = max  { r = "subject2" for  subject,  "category2" for
          categories,  "area2" for areas,  "discipline2" for disciplines and  "field2"  for fields }
29:       if subject1 not present for $d_i$ and only one subject classified as  "subject2" for $d_i$ then
30:           connect document node, '$d_i$' to node skey with relationship type as categorization type, r of each
              categorization type , t and count = smax  { r = "subject2" for  subject,  "category2" for
             categories,  "area2" for areas,  "discipline2" for disciplines and  "field2"  for fields }
31:       end if
32:   end for
33:  end
```

IV. F-GSC, A FLEXIBLE GRAPH-BASED SUBJECT CLASSIFIER

- New technologies are introducing day-by-day in almost all areas of disciplines which will introduce new keywords into existing subjects. So a topic hierarchy should be flexible and flexible enough to incorporate these new rising keywords into the existing taxonomy. The main feature of the pre-indexed graph-based classifier, GSC, is its flexibility to update the existing hierarchy with new keywords. The main source of new keywords is the author-supplied paper keywords, which are not in the classifier hierarchy. Or new keywords can be manually added to the existing taxonomy into its correct subject and level. Whenever a new key is introduced into the existing taxonomy, a reclassification of already indexed documents that contain this newly added key should be done for the accurate subject labeling of the article. Steps of key update in the topic hierarchy and relabeling of articles include:

- Upgrade each newly introduced key, K, to the key node by removing from 'stop_word' property and properly inserted in the correct order in the sequence of key nodes of file contents
- Update key node in the topic hierarchy in the correct level
- Relabeling of all indexed articles containing this term

The detailed pseudocode for key up-gradation and reclassification of documents is given in Algorithm 1.

A. Up-gradation of key

The initial step is the key up-gradation. In the KSG model, all terms other than classifier keywords are considered as stop words. So the initial phase is to find out all the target edges that contain newly added keys in the 'stop_word' property and then upgrade it to a key node in the correct order in the sequence. Key up-gradation is done by splitting the concatenated stop word by new tern and form a new key node with this term and insert in between for each edge sequence of a sentence. The start node and end node of the edge remain the same for the newly inserted key node by adding a new edge in between to and from the key node by maintaining the common properties before copying from the respective target edge and then delete the target edge.

All properties, including "seqid", the sentence no: etc., will be added to each of the newly added relationships. The case of the newly added key will also be determined and stored in its incoming edge as before. The preceded stop word, if any after splitting the concatenated stop word string that contains the new key, will be added as 'stop_word' property in the incoming edge of the new key node, and the succeeding stop word will be added as 'stop_word' property in the respective outgoing edge of the new key node. The next step is to update the "relid", that maintains the sequence order of each sentence by copying the same 'relid' value of the target edge to the incoming relationship of the key node and then will be incremented by 1 for the rest of the edges in the sequence of the same sentence. After all properties are copied, delete the target edge. If it's a multi-word key, the only difference is the creation of key nodes in sequence for all the non-stop word terms in key as in KSG. A 'file_list' will be created to store the list of all files containing the new key for key updating and reclassification easier.

B.Update key node in the topic hierarchy

The next phase is to update the key node in the topic hierarchy by merging it into the relevant subject as a 'keyword'. In order to update the key node to one of the correct six levels in the topic hierarchy, manual intervention is needed, and it's easy to deal with. In this paper, only an automatic updating is done by adding the new key as a 'keyword' under the correct 'subject'. For that, first find the appropriate subject, S, having the highest aggregate count by considering relationship type, "subject1", or "subject2" of the classified subjects of all files in the 'file_list' and then merge the new key as 'keyword' to the classified subject. If the key is a multi-word term, then expand it using WSG, the Word Sequence Graph model [5].

C. Relabeling of Documents

The next step is the reclassification of all files in the 'file_list', which includes the initial primary labeling of keys and then subject labeling using the labeling algorithm. Primary labeling calculates the Term frequency, TF, that counts the number of occurrences of the newly added key in each file of the 'file_list', and connects each of the file nodes to the key node with edge type as 'keywords' with property 'count' that stores the TF. Then subject labeling of each article file into the relevant subject is done using the subject labeling algorithm.

A simple example of Key upgradation is explained next in the section. Initially, let the term 'supernovae' be not there in the classifier hierarchy as a key and hence considered as a stop word in KSG representation of each document that contains this term. Consider the article shown in Figure 6. It consists of the word 'supernovae' in the title and also in the first line of the abstract and the Key Sequence Graph representation of the title and abstract of this article is shown in Figure 7 and Figure 8 with graphical and tabular representation in (a) and (b) respectively.

The term 'supernovae' is considered as a stop word in both representations, as highlighted in the red outline in each of the figures. Let 'R' be the edge that consists of the term 'supernovae' in the 'stop_word' property in each case. Since the title terminates with this word alone with no other substrings, the only thing to do is to connect its previous key node 'collapse' with a new edge 'Rn' of type 'title_next' to the 'supernovae' key node after it key creation and upgradation in the classifier hierarchy and then copy all the common properties like 'seqid', 'id', 'stop_word' etc. from 'R' to 'Rn' followed by deletion of the relation 'R'.

Since the term appears somewhat in the middle in the abstract part, there is a need to keep track of the entire sequence from its start node 'collapse' to the end node, 'ter_node' as indicated with a blue outline. The word appears at beginning of 'stop_word' property and hence it will be first split by this term to get the suffix, 's' (i.e., s = 'lies in need for an accurate'). Then directly connect the key node 'collapse' to 'supernovae' key node as done earlier in the itle by creating a new edge 'Rn' of type 'abstract_next' along with updated properties in the edges.

Algorithm 1 Classifier key up-gradation and reclassification of documents

// Key up-gradation

```
for each key K,
    1. Identify each relationship, R containing K in its stop_word property
    2. Identify the start node, start and end node, end of R
    3. Identify the incoming relationship type, rel_in and outgoing relationship, rel_out for K using relationship type, rel_type of R
        a. rel_in = rel_type itself
        b. if rel_type = "contents" then rel_out = "next_seq"
        c. else if rel_type = "abstract" then rel_out = "abstract_next"
        d. else if rel_type = "title" then rel_out = "title_next"
        e. else rel_out= rel_type
    4. Identify the case of K as 'U', 'S' or 'N'
        a. convert K to lowercase if case= 'U' or 'S'
        b. else store as such
    5. Create a node for K with node_type as "key_phrase" and label as "key"
    6. Create a relationship, rn1 from start node, start to key node, K and another relationship, rn2 from K to end node, end of R
    7. Add all common properties of R like "seqid", "sentence" (exists only for title and abstract) to both rn1 and rn2
    8. Add case property if exists in R, to rn2
    9. if case of K = 'U' or 'S' , then store case in rn1
    10. Split the stop_word by the key K into a stop array, stop
        a. Add stop[0] to rn1 and stop[1] to rn2 if both exists
        b. else if stop_word starts with K then add stop[0] to rn2
        c. else add stop[0] to rn1
    11. Update "relid" property of R to rn1 and relid+1 as "relid" to rn2
    12. Increment the value of "relid" property by 1 of each key node in the chain of key of sentence corresponds to R starting with end node, end of R
    13. Identify the file that needs to be reclassified as a result of presence of K in the contents using "seqid" of R and add it to file_list
    14. Delete relationship R
```

//Update the key node in the classifier

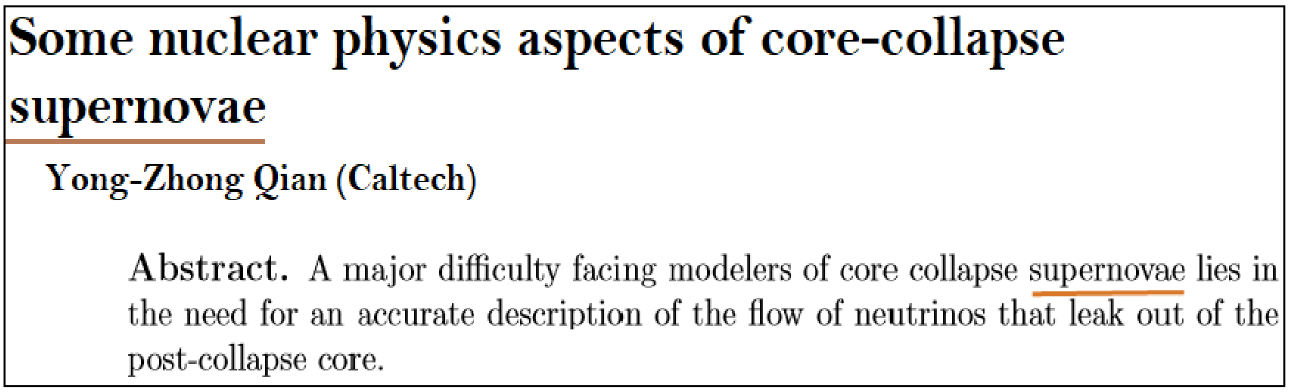
```
for each key K,

1. Find the appropriate subject, S with highest aggregate count of keyword K by considering classified
   subjects of all files with id in the file_list of K connected by relationship type , "subject1" or "subject2"
2. Merge key K to the subject node , 'S' with relationship type "keywords"
3. if key K is a phrase with length > 1 then
    a. Construct WSG representation of its expansion terms with "expand" and "key_next" type
```

//Primary labeling of keys

```
1. for each file, F of file_list of Key, K,
2. Connect file node, F to key node of K with relationship type as "keywords" and count of occurrences
```

//Classify each article file, F into the relevant subject using the GASE subject labeling algorithm

```
1. Merge all files in file_list of all updated keys, $K_i$ to form the final list of all files that contains any of the
   updated keys for relabeling.
2. for each file, F of file_list do
    a. Delete all current subject and sub category labeling relation of F by finding outgoing relationship of
       type "subject1" or "subject2" etc.
    b. Classify article F into relevant subject using the GASE subject labeling algorithm
```



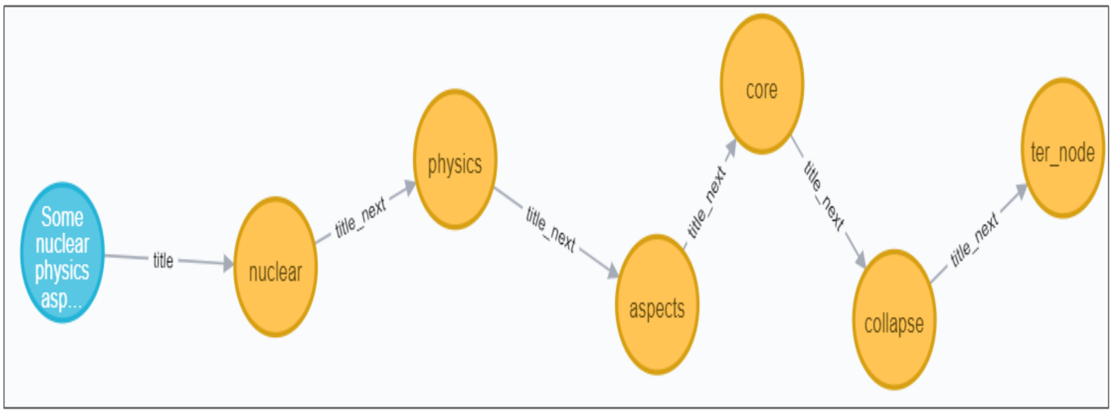
Some nuclear physics aspects of core-collapse supernovae

Yong-Zhong Qian (Caltech)

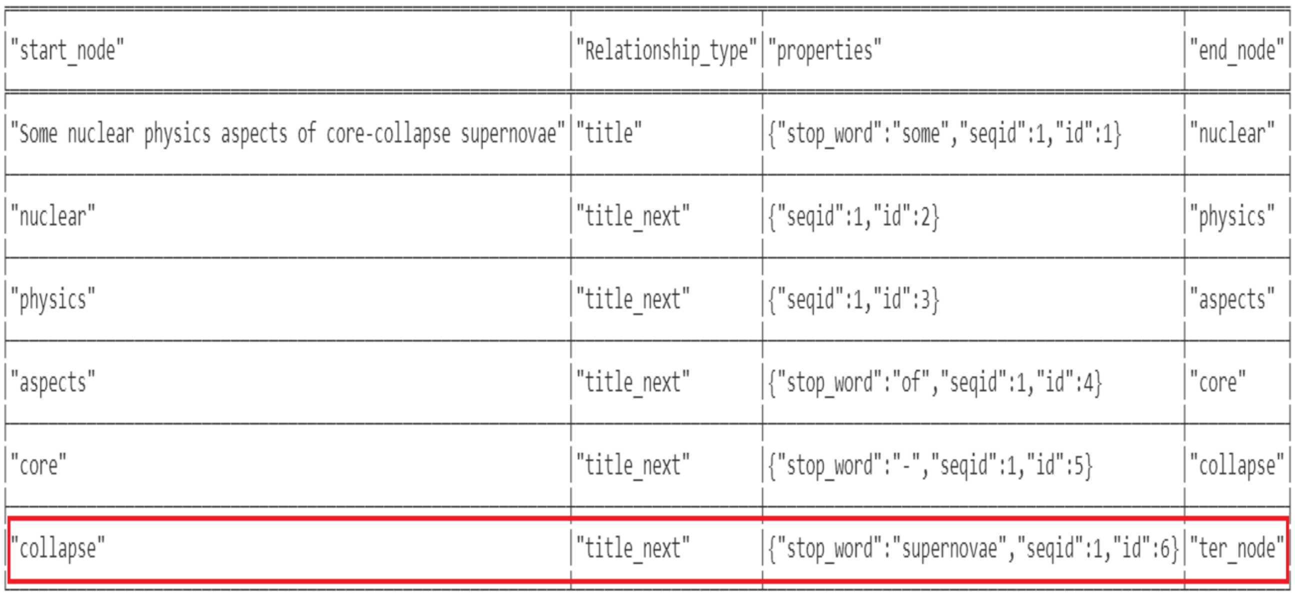
Abstract. A major difficulty facing modelers of core collapse supernovae lies in the need for an accurate description of the flow of neutrinos that leak out of the post-collapse core.

Figure 6. A sample article containing the term 'supernovae' in the 'title' and 'abstract

(a)



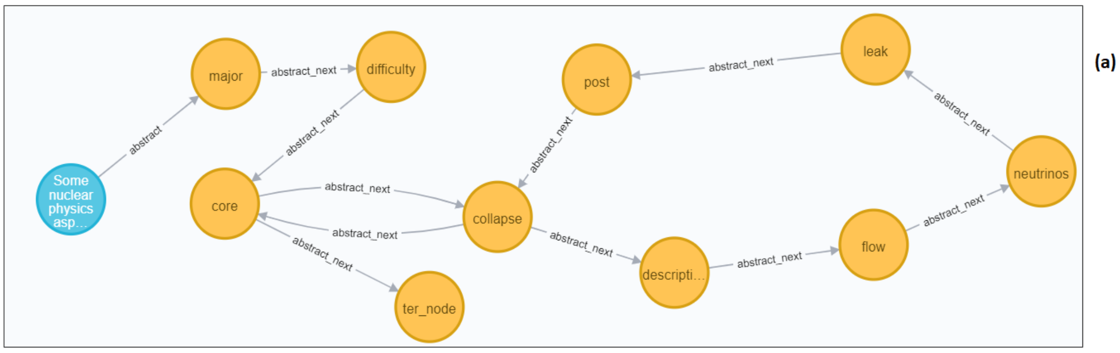
(b)



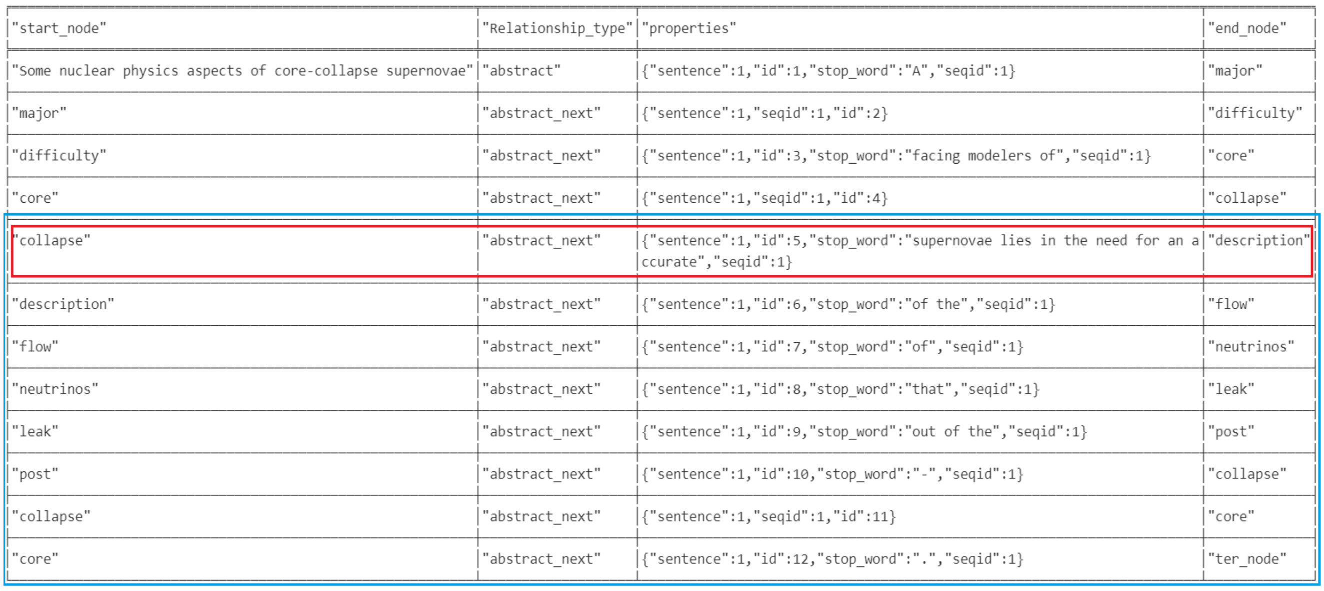
"start_node"	"Relationship_type"	"properties"	"end_node"
"Some nuclear physics aspects of core-collapse supernovae"	"title"	{"stop_word":"some","seqid":1,"id":1}	"nuclear"
"nuclear"	"title_next"	{"seqid":1,"id":2}	"physics"
"physics"	"title_next"	{"seqid":1,"id":3}	"aspects"
"aspects"	"title_next"	{"stop_word":"of","seqid":1,"id":4}	"core"
"core"	"title_next"	{"stop_word":"-","seqid":1,"id":5}	"collapse"
"collapse"	"title_next"	{"stop_word":"supernovae","seqid":1,"id":6}	"ter_node"

Figure 7. The word 'supernovae'as stop_word in the sequence graph representation of article's title shown in Figure 6

A new edge 'Rnn' will be created next labeled with type 'abstract_next' to connect 'supernovae' node to end node 'description' of 'R'. The next step is to duplicate all the edge properties of 'R' to 'Rnn' and add the extracted suffix as 'stop_word' property in 'Rnn' and add the edge sequence id 'id', which is one greater than that of 'R' (i.e., id=6 in the relation connecting 'supernovae' node to 'description' node). The final step is to maintain the sequence order of the sentence by updating the 'id' property till the end node, 'ter_node', which is one greater than the previous connecting edge. Figure 9 and Figure 10 represent the modified Key Sequence Graph and tabular representation.

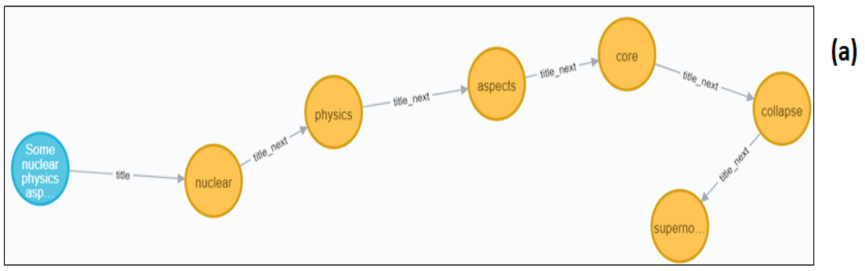


(b)

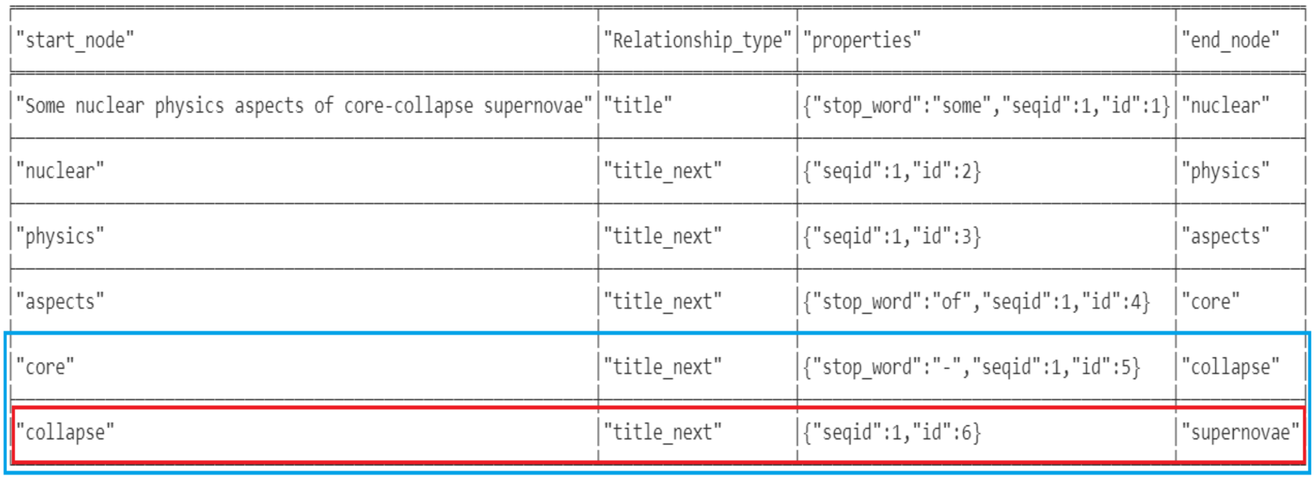


"start_node"	"Relationship_type"	"properties"	"end_node"
"Some nuclear physics aspects of core-collapse supernovae"	"abstract"	{"sentence":1,"id":1,"stop_word":"A","seqid":1}	"major"
"major"	"abstract_next"	{"sentence":1,"seqid":1,"id":2}	"difficulty"
"difficulty"	"abstract_next"	{"sentence":1,"id":3,"stop_word":"facing modelers of","seqid":1}	"core"
"core"	"abstract_next"	{"sentence":1,"seqid":1,"id":4}	"collapse"
"collapse"	"abstract_next"	{"sentence":1,"id":5,"stop_word":"supernovae lies in the need for an accurate","seqid":1}	"description"
"description"	"abstract_next"	{"sentence":1,"id":6,"stop_word":"of the","seqid":1}	"flow"
"flow"	"abstract_next"	{"sentence":1,"id":7,"stop_word":"of","seqid":1}	"neutrinos"
"neutrinos"	"abstract_next"	{"sentence":1,"id":8,"stop_word":"that","seqid":1}	"leak"
"leak"	"abstract_next"	{"sentence":1,"id":9,"stop_word":"out of the","seqid":1}	"post"
"post"	"abstract_next"	{"sentence":1,"id":10,"stop_word":"-","seqid":1}	"collapse"
"collapse"	"abstract_next"	{"sentence":1,"seqid":1,"id":11}	"core"
"core"	"abstract_next"	{"sentence":1,"id":12,"stop_word":".","seqid":1}	"ter_node"

Figure 8. The word 'supernovae'as stop_word in the sequence graph representation of abstract of the article shown in Figure 6



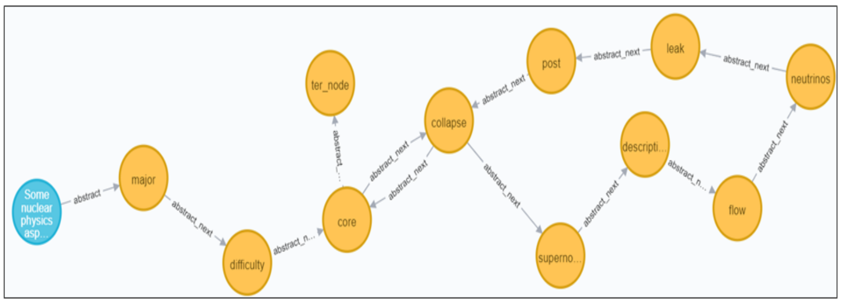
(b)



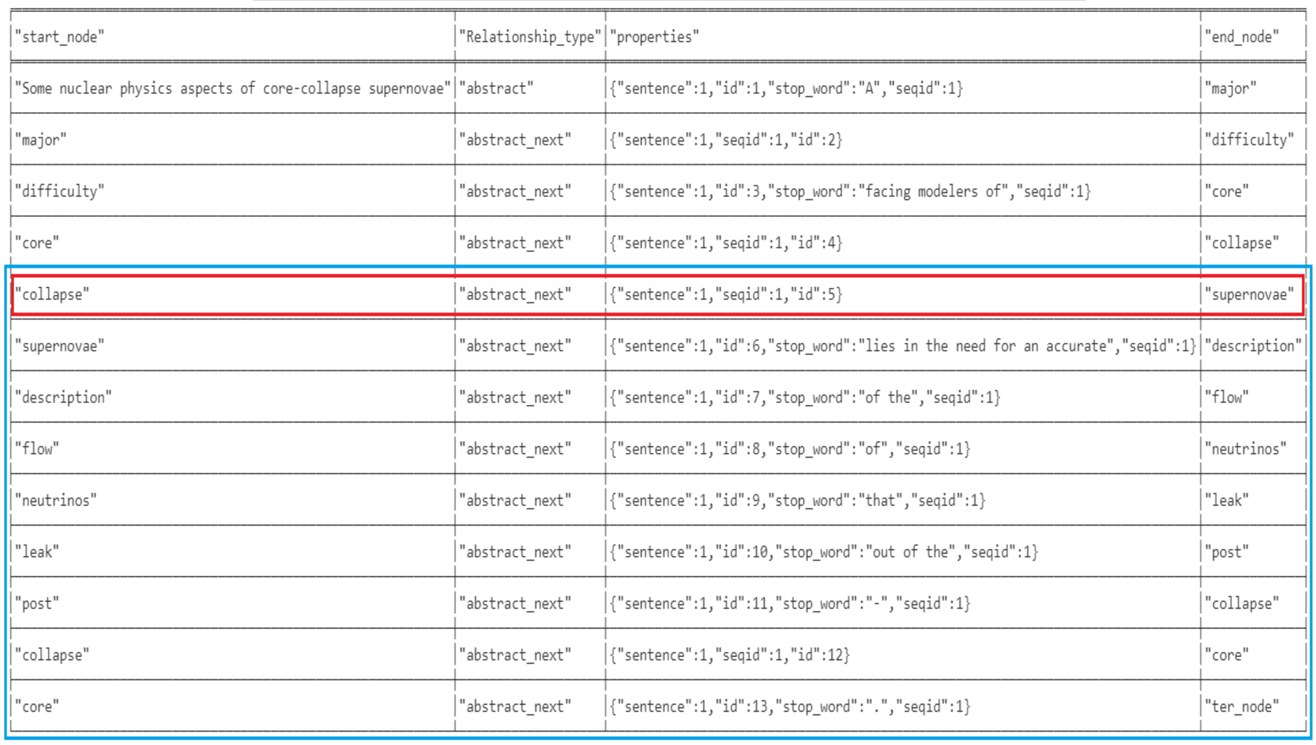
"start_node"	"Relationship_type"	"properties"	"end_node"
"Some nuclear physics aspects of core-collapse supernovae"	"title"	{"stop_word":"some","seqid":1,"id":1}	"nuclear"
"nuclear"	"title_next"	{"seqid":1,"id":2}	"physics"
"physics"	"title_next"	{"seqid":1,"id":3}	"aspects"
"aspects"	"title_next"	{"stop_word":"of","seqid":1,"id":4}	"core"
"core"	"title_next"	{"stop_word":"-","seqid":1,"id":5}	"collapse"
"collapse"	"title_next"	{"seqid":1,"id":6}	"supernovae"

Figure 9. The word 'supernovae' changed to key node after key upgradation in title

(a)



(b)



"start_node"	"Relationship_type"	"properties"	"end_node"
"Some nuclear physics aspects of core-collapse supernovae"	"abstract"	{"sentence":1,"id":1,"stop_word":"A","seqid":1}	"major"
"major"	"abstract_next"	{"sentence":1,"seqid":1,"id":2}	"difficulty"
"difficulty"	"abstract_next"	{"sentence":1,"id":3,"stop_word":"facing modelers of","seqid":1}	"core"
"core"	"abstract_next"	{"sentence":1,"seqid":1,"id":4}	"collapse"
"collapse"	"abstract_next"	{"sentence":1,"seqid":1,"id":5}	"supernovae"
"supernovae"	"abstract_next"	{"sentence":1,"id":6,"stop_word":"lies in the need for an accurate","seqid":1}	"description"
"description"	"abstract_next"	{"sentence":1,"id":7,"stop_word":"of the","seqid":1}	"flow"
"flow"	"abstract_next"	{"sentence":1,"id":8,"stop_word":"of","seqid":1}	"neutrinos"
"neutrinos"	"abstract_next"	{"sentence":1,"id":9,"stop_word":"that","seqid":1}	"leak"
"leak"	"abstract_next"	{"sentence":1,"id":10,"stop_word":"out of the","seqid":1}	"post"
"post"	"abstract_next"	{"sentence":1,"id":11,"stop_word":"-","seqid":1}	"collapse"
"collapse"	"abstract_next"	{"sentence":1,"seqid":1,"id":12}	"core"
"core"	"abstract_next"	{"sentence":1,"id":13,"stop_word":".","seqid":1}	"ter_node"

Figure 10. The term ‘supernovae’ changed to key node after key upgradation in abstract part

V. EXPERIMENTAL EVALUATIONS & RESULTS

In order to evaluate the GSC flexibility for key update and relabelling, arXiv articles already indexed for our previous work to create a sequence graph-based academic search engine is used. A total of 1307 papers were used, of which 426 papers belong to 'mathematics and statistics', 390 papers belong to 'computer science', and 428 papers belonged to 'physics'. To evaluate the accuracy of key update and relabelling, we searched and find out paper keywords given by the author, which is not in the classifier taxonomy. 11 author-supplied keywords not in the topic hierarchy were used. The accuracy of the labelled subject of newly added 11 keywords was manually tested, and only one among them is misclassified. The result is shown in Table 1.

Table 1. Evaluation results of key update

Key	Subject	Accuracy based on source file?
hypre	informatics, engineering	√
isoparametric submanifolds	mathematics and statistics, informatics	√
pulsar	mathematics and statistics	× (physics is more accurate)
starburst	physics	√
polytetrafluoroethylene	biological sciences, physics	√
subellipticity	mathematics and statistics, engineering	√
semicube	mathematics and statistics	√
supergiants	physics	√
cusp	mathematics and statistics	√
lerpa	informatics, engineering	√
supernovae	physics	√

Relabeling of the already indexed articles containing the newly added keywords changed 22314 relationships, and one file, which is wrongly classified into the area of 'mathematics and statistics' was correctly classified to the relevant subject ‘physics'.

VI. CONCLUSIONS AND FUTURE SCOPE

Subject classification of indexed scientific articles provides a better search experience for researchers or academicians to search articles by subject or category. This paper presents a flexible classifier, F-GSC, which facilitates automatic subject labelling of articles at the indexing time itself to six levels of topic hierarchy that helps users to search scientific articles by subject, category, area, discipline, field, or keyword. F-GSC is flexible in the sense that it is flexible enough to automatically add new keywords to the topic hierarchy. An experimental evaluation of automatically updating new keywords and relabelling the already indexed articles containing the newly added keywords were also done to evaluate the work. The results proved that the classifier is flexible enough to update with new keywords.

ACKNOWLEDGMENT

I sincerely thank my guide, Dr. Sudheep Elayidom, Professor, Division of Engineering, SOE, CUSAT, and my co-guide Dr. T Santhankrishnan, Scientist-F, DRDO, NPOL, Kakkanad for their valuable suggestions and corrections.

REFERENCES

[1] Rous, B, Major update to ACM's computing classification system. *Communications of the ACM, 55*(**11), 12-12, 2012**.

[2] Harris, A. L. Impact factor.

[3] Kang, M., Shin, J. D., & Kim, B., Automatic subject classification of korean journals based on kscd. *Indian Journal of Science and Technology*, **8(S1), 452-456, 2015**.

[4] Soumya George, M. Sudheep Elayidom, T. Santhanakrishnan, Knowledge Graph Based Subject Classification of Scholarly Articles", *Journal of Advanced Research in Dynamical & Control Systems, (JARDCS),* Volume. **11, 02**-Special Issue, **2019**

[5] Sammet, J. E., & Ralston, A,The new (1982) computing reviews classification system—final version. *Communications of the ACM*, ***25*(1), 13-25, 1982**.

[6] Herrmannova, D., & Knoth, P, An analysis of the microsoft academic graph. *D-lib Magazine*, ***22*(9/10), 37, 2016**.

[7] Kengeri, R., Seals, C. D., Harley, H. D., Reddy, H. P., & Fox, E. A., Usability study of digital libraries: Acm, ieee-cs, ncstrl, ndltd. *International Journal on Digital Libraries*, ***2*(2), 157-169, 1999**.

[8] Hammond, T., & Pasin, M, The nature. com Ontologies Portal. In *LISC@ ISWC* (pp. **2-14**), **2015**.

[9] Dunne, E., & Hulek, K, Mathematics subject classification 2020. *European Mathematical Society Magazine*, **(115), 5-6,2020**.

[10] Waltman, L., & Eck, N. J, A new methodology for constructing a publication level classification system of science. *Journal of the Association for Information Science and Technology,* **63(12), 2378-2392, 2012**.

[11] Archambault, É., Beauchesne, O. H., & Caruso, J,Towards a multilingual, comprehensive and open scientific journal ontology. *In Proceedings of the 13th international conference of the international society for scientometrics and informetrics* (pp. **66-77**). South Africa: Durban, **2011, July**

[12] Al-Zaidy, R. A., & Giles, C. L., Extracting Semantic Relations for Scholarly Knowledge Base Construction. *In Semantic Computing (ICSC), 2018 IEEE 12th International Conference* on (pp. **56-63**). IEEE, **2018, January**.

AUTHORS PROFILE

Dr. Soumya George is an Asst. Professor at St. George's College, Aruvithura. She completed her Ph. D in Computer Science from Cochin University of Science and Technology. Her research areas of interests include data mining, information retrieval etc. She had many journal publications to her credit.. She has around 8 years of teaching experience and around 4 years of research experience.