



北京大学软件工程研究所

Software Engineering Institute, Peking University

SOSE: Source cOde Search Engine

— Introduction to the design and
implementation

Software Institute, Peking University

Lijie Wang, Fei Liu, Yiyang Huang, Leye Wang, Nicholas Bertrand

Introduction of SOSE

- SOSE: Source cOde Search Engine for java language
- Developed by five students of Software Institute, Peking University.
- SOSE aims to provide support for programmers on **searching, understanding, and utilizing** existing source code on the internet to improve their work efficiency.
- Available at: <http://codesearch.seforge.org>



Agenda

- Motivation of the project
- Overview of SOSE
- Design and Implementation of SOSE
- Introduction of the project scale
- Conclusion



Motivation

- Many open source codes are available on the Internet.
- Finding codes from the Internet is a common practice for programmers.
- Some existing code search engines
 - Google code search, Koders, Krugle,...
- Different purposes for searching source codes of different users
 - Search for implementation of an API (class) to learn about the inner detail of the API
 - Search for usage examples of an API to learn about how to use the API.
 - ...
- For different searching purposes, code search engine could do something special to serve users better.



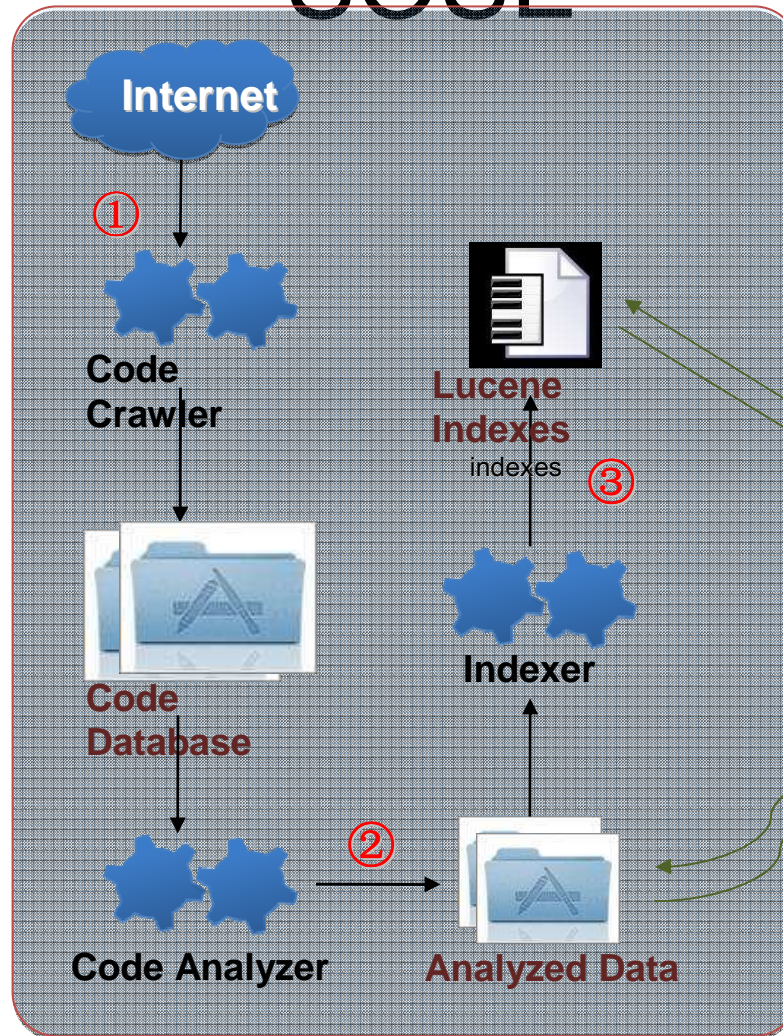
Overview of SOSE

- Distinguishing API **implementation** and API **usage examples** during searching process
- **Clustering usage examples** of an API. Each cluster stands for a typical usage of the API.
- Other assistance functions:
 - Query suggestion
 - Summary generation
 - Source code highlighting
 - ...

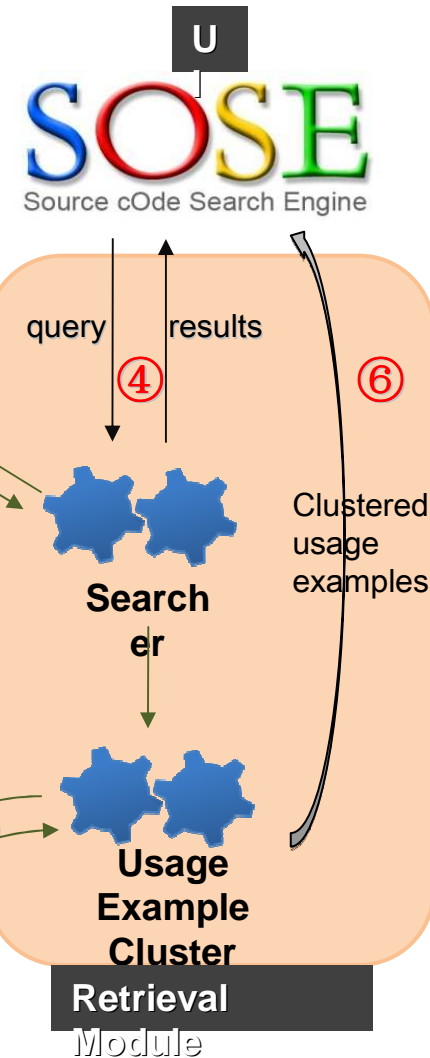


Design and Implementation of SOSE

System architecture



Data Preparation



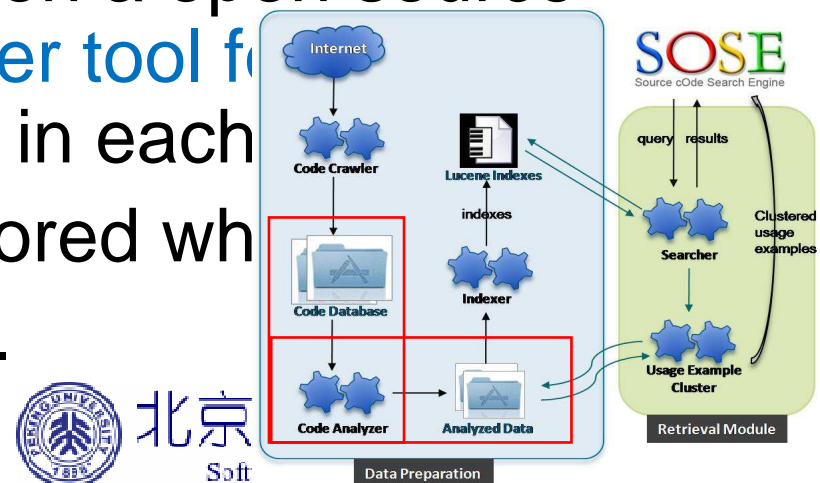
Design and Implementation of SOSE

- Data Preparation Module——Crawler
 - *Function*: Crawling source code (java file) from the Internet and storing them in local file system.
 - Two ways:
 - From source code repositories of open-source projects on sourceforge.



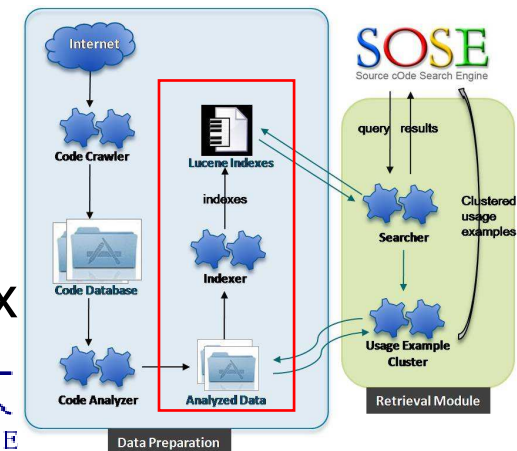
Design and Implementation of SOSE

- Data Preparation Module——Code Analyzer
 - *Function*: Parsing source code and extracts all useful information from source code including *class*, *interface*, *methods* and *comment* fields declared in source code.
 - It is implemented based on an open source Java parser *Antlr (Another tool for recognition)*. Information in each field is extracted and stored when parsing java source files.



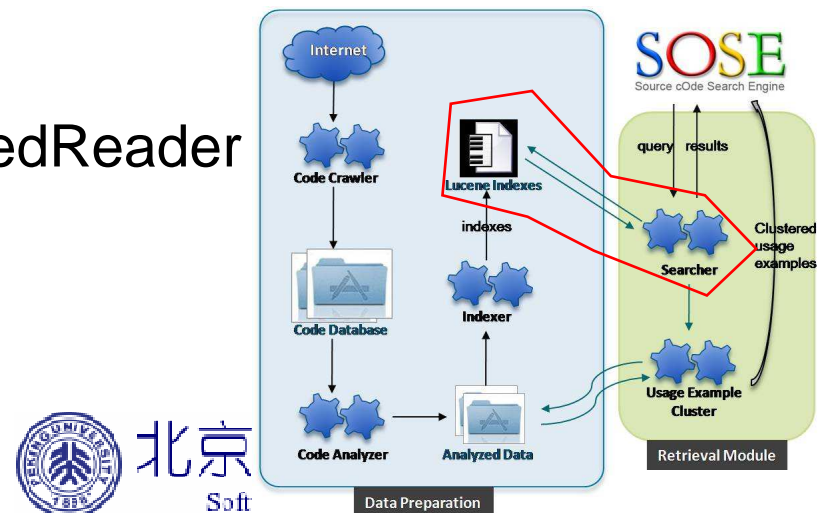
Design and Implementation of SOSE

- Data Preparation Module——Code Indexer
 - *Function*: Building indexes on fields of *class*, *methods*, *interface*, *comments* and *usage of other classes*.
 - Implemented based on Lucene.
 - Two steps:
 - Word splitting
`Java.io.InpustStream` → "Java io Input Stream"
`A_B` → "A B"
 - Indexing
 - Incremental Indexing:
merging new index files with original index



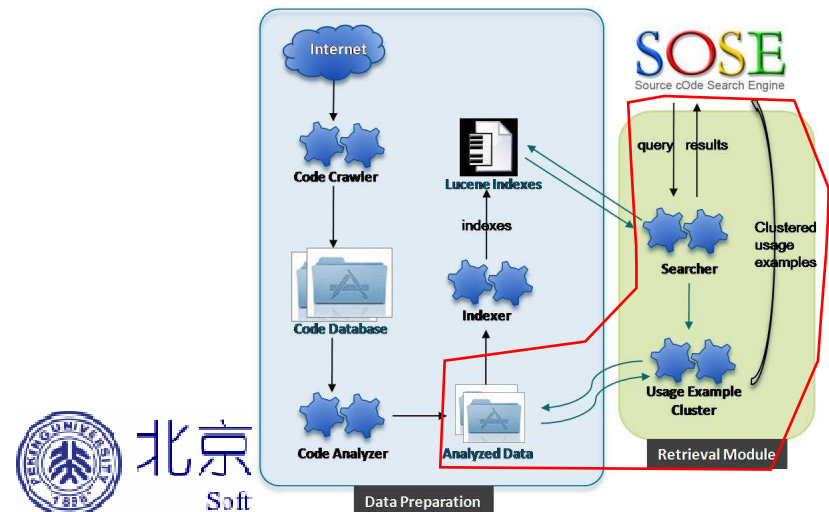
Design and Implementation of SOSE

- Retrieval Module——Searcher
 - *Function*: Conduct searching according to user's selection, in *class*, *method*, or *comment* field.
 - Based on Lucene
 - SOSE does not provide support for complex query grammar. User can select what he/she intend to search for.
 - Query examples:
 - Java.io.InputStream, BufferedReader
 - toString
 - Word splitting



Design and Implementation of SOSE

- Retrieval Module——Usage Example Clustering Tool
 - *Function*: Clustering usage example code of a specific API (class) according their usage.
 - Help programmers know different usages of the API by browsing less results.



Design and Implementation of SOSE

- Retrieval Module——Usage Example

Clustering Tool

INPUT: The vectors of all candidates

OUTPUT: The clustering result

Step 1: Treat each candidate as an individual cluster;

Step 2: REPEAT

 Choose the two most similar clusters x and y ;

IF for any element m in x and any element n in y , $sim(m,n) > threshold$

 Merge x and y into one cluster;

UNTIL the two most similar clusters can not be merged

Step 3: Return the remaining clusters;



Design and Implementation of SOSE

- Retrieval Module——Usage Example Clustering Tool
 - After clustering, SOSE selects an example code from each cluster as the representative.


$$x \in G \wedge \forall y \in G, \sum_{z \in G} sim(z, y) \leq \sum_{z \in G} sim(z, x)$$

– Ranking Strategy:

- examples from larger clusters will get higher ranks.
- the more comments the example has, the higher rank should be given.
- the examples in small size will be rank higher.



Design and Implementation of SOSE



The screenshot displays the SOSE (Source Code Search Engine) interface. At the top left is the SOSE logo and the text "Source Code Search Engine". To the right, there is a search bar with a "Search" button and a dropdown menu set to "Find In Any field". Below the search bar, a list of search results is shown, with the first result expanded to show Java code. The code includes a `LineCallback` interface, a `forEachLine` method, and a `main` method. The code uses `BufferedReader` for file reading. On the right side of the code, there is a vertical scrollbar with a dropdown menu showing the current line number (35) and a blue highlight at the bottom. At the bottom right of the page, there is a blue icon resembling a stylized 'S' or a similar symbol.

```
28     } catch (IOException e) {
29         if (done) throw e;
30     }
31 }
32 }
33 }
34 }
35 public interface LineCallback {
36     public void doForLine(String line) throws IOException;
37 }
38
39 public static void forEachLine(File file, final LineCallback lineCallback) throws IOException {
40     doWithFile(file, #(Reader reader) {
41         BufferedReader bufferedReader = new BufferedReader(reader);
42         String line;
43         while ( (line = bufferedReader.readLine()) != null) {
44             lineCallback.doForLine(line);
45         }
46     });
35 47 }
35 48
35 49
35 50 public static void main(String[] args) throws IOException {
35 51     forEachLine(new File(args[0]), #(String line) {
36 52         System.out.println("This is a line:" + line);
53     });
54 }
55
56 public static void theOldWay(String[] args) throws IOException {
57     BufferedReader reader = null;
58     boolean done = false;
59     try {
60         reader = new BufferedReader(new FileReader(new File(args[0])));
```

Project scale

- 4 developers; 2 months
- 12,000 LOC
- Crawled java source code: 4,428,182 LOC
- A prototype system is now available at:
<http://codesearch.seforge.org>



Summary

- SOSE distinguishes API implementation and API usage examples for code searching;
- SOSE clusters usage examples to reduce users' burden;
- SOSE supports searching in certain fields;
- Query suggestion;
- Summary generation;
- Java source code highlighting.



Thanks!

Q&A

- Any questions or advices are welcome. 😊

