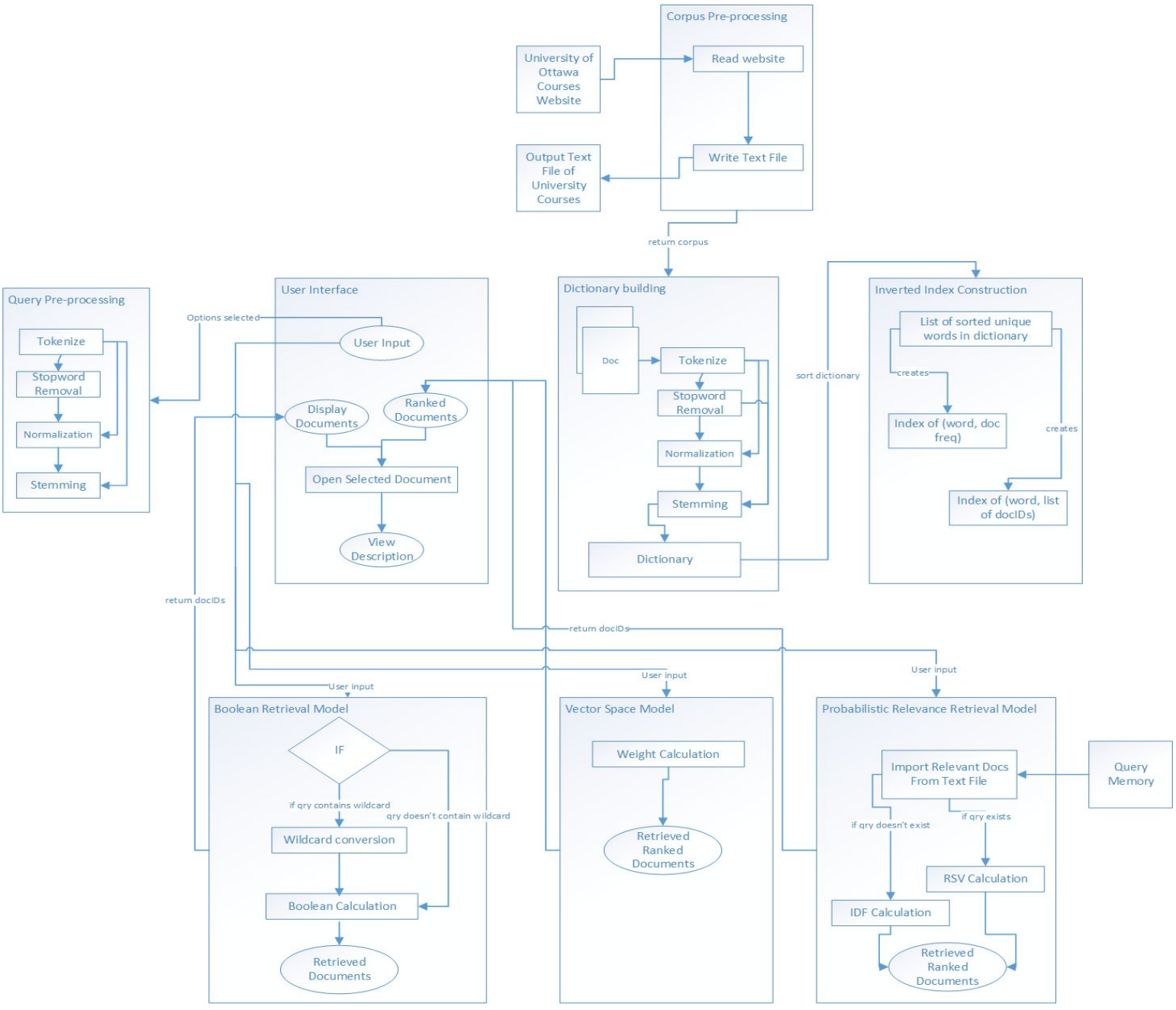
# Phase 1: Vanilla Search Engine

Course code: CSI 4107

Student 1: Catherine DesOrmeaux, 7746139

Student 2: NamChi Nguyen, 7236760

# System Architecture



# Module 1 - Corpus Pre-Processing

## Functionality

The class Preprocessor.java has the functionality of module 1 (corpus pre-processing). The Preprocessor constructor takes the course information directly from the website and stores them into a TreeMap, where the key is the docID and the value is the description. The constructor then calls the write function, where the output text file is created on the user's computer. The output text file will be saved in the same directory as the project. If the output text file already exists, the file will be overwritten.

## Limitations (cases not handled)

We have removed the french courses. However, the courses that included a description in french and english will be stored in the HashMap. For example: the course CSI 4900 has the description in both languages so it will be stored in the corpus.

Also, this module only handles the parsing of the CSI courses and won't work for a different website. However, in the final system, the preprocessing for the Reuters and web collections will be added.

## Problems encountered (if any, as you developed the module)

There were no problems encountered as we were able to generate the physical output (not just in memory), for this module.

# Module 2 - User Interface

## Functionality

The class MainPage.java has the functionality of module 2 (user interface). The user is able to write the query, choose the type of model, document collection. There is also the options of stopword removal, stemming, and/or normalization, which will modify both the dictionary (in DictionaryBuilder.java) and user input (QueryPreprocessor.java). Once the user is finished filling out the form, they can click the Search button, where the docID, Title, Excerpt, and Score will be displayed in a table. The user can select one document and click the View Details button to read the course description.

## Limitations (cases not handled)

The user input must contain a whitespace in between each word and parentheses.

For example: “(operating AND system)“ does not work with our search engine. However, the query “( operating AND system )” does work with our model. The difference is that there is one space between the parenthese and the word.

Also, the user input must be lowercase excluding the boolean operations AND, OR, AND_NOT since if the entire query was converted to lowercase, in the case of the Boolean model, the boolean operators would be treated as ‘stopwords’ instead, so we decided to keep the query format consistent for all retrieval models.

## Problems encountered (if any, as you developed the module)

We had problems displaying the components on the user interface. Once we discovered the functionality of WindowBuilder on Eclipse, we were able to add, modify, delete the components more efficiently. For example: we were able to use the design mode, which allowed us to view the location of each components. This avoided us to determine the exact location with the coordinates.

# Module 3 - Dictionary building

## Functionality

The class DictionaryBuilder.java has the functionality of module 3 (dictionary building). The dictionary building class tokenizes the description of each course with the built-in functionality of the white space tokenizer from Apache OpenNLP. In addition to this, if the word ended with a symbol (i.e. punctuation), we removed the symbol because we did not want to have a word in the dictionary ending with a symbol. Each token is lower-case. For example: “System” and “system” is the same word.

There are the options for stopword removal, which removes the stopword listed in the text file stored on the user’s computer, stemming, which uses the built-in PorterStemmer, and normalization, which removes all the periods and hyphens. For hyphens, we decided to concatenate the word as one instead of keeping them separate. For example: “object-oriented” becomes “objectoriented” when normalized.

## Limitations (cases not handled)

N/A - the search engines contains all the required functions

## Problems encountered (if any, as you developed the module)

As we were implementing the tokenize function, we noticed that Python had more built-in tokenizer options than Java. Apache OpenNLP only had 3 options which were: SimpleTokenizer, TokenizerMe, and WhitespaceTokenizer. After using the built-in WhitespaceTokenizer, we had to make sure that the word did not end with a symbol. Therefore, we added a part of code to remove them manually.

# Module 4 - Inverted Index Construction

## Functionality

The class DictionaryBuilder.java has the functionality of module 4 (inverted index construction). We have decided to include the functionality in the same class in order to avoid reading our collection of documents again in a different class. The function *buildInvertedIndex* builds the inverted index. In other words, a dictionary of terms are indexed. The function stores the information into two HashMap. The *keytodocID* HashMap stores each unique word with a list of its corresponding documents. The *keytofreq* HashMap stores each unique word with its corresponding document frequency.

## Limitations (cases not handled)

N/A

## Problems encountered (if any, as you developed the module)

As we were developing this module, the main challenge was to determine whether or not the dictionary was built properly. When we printed the words and its corresponding documents, Eclipse did not allow us to view all the words in the dictionary due to the vocabulary size. We were able to test with the UI. This allowed us to choose the word and see the corresponding documents.

# Module 5 - Corpus access

## Functionality

The class MainPage.java contains the functionality of module 5 (corpus access). The user can retrieve the documents corresponding to the query. All documents are showed in a table (title, excerpt line, score). The button view detail allows the user to see the full content.

For each retrieval model, the main page calls the getter methods from the following classes:

- DictionaryBuilder - retrieve the docIDs, course titles and descriptions
- Boolean_Model - retrieve the documents for the boolean model
- VSM - retrieve the documents from highest to lowest score for the VSM model
- Probabilistic - retrieve the documents from highest to lowest score based on the idf or RSV for the probabilistic model

## Limitations (cases not handled)

N/A - the search engine evaluates the query and displays the documents for the chosen model

## Problems encountered (if any, as you developed the module)

N/A

# Module 6 - Boolean Retrieval Model

## Functionality

The class Boolean_Model.java has the functionality of module 6 (boolean retrieval model).

First, the user input is converted from infix to postfix. Then, the query is passed into the *postfixEval* function, which returns an arrayList of docIDs.

The *postfixEval* function calls the following functions within the class:

- getList, which returns the list of documents associated to the operand
- performBooleanOperation, which determines the if the *union, intersection*, or *andNot* function should be called based on the operator (“AND”, “OR”, “AND_NOT”)
- union, which returns an arrayList of all the docIDs in the first arrayList or the second arrayList
- intersection, which returns an arrayList of the docIDs in both arrayList
- and_not, which returns an arrayList of all the docID in the first arrayList that are not in the second arrayList

## Limitations (cases not handled)

NA - the search engine returns the corresponding documents based on the query.

## Problems encountered (if any, as you developed the module)

As we were developing the *postEval* function, we had to modify the function where the stack could contain a string, which is the word, or an arrayList, which is the result of the *union, intersection, andNot* function. We tested the function by printing the steps within the *postEval* function.

# Module 7a - Vector Space Model (Weight calculation)

## Functionality

The class DictionaryBuilder.java has the functionality of module 7a (vector space model - weight calculation) since the inverted index is also built in DictionaryBuilder. It calculates the tf-idf weights required by the VSM with the following functions:

- getTermFreq - gets the tf of a term
- calculateWeight - calculates tf-idf
- vsmWeightList - stores the weights in a list

## Limitations (cases not handled)

N/A

## Problems encountered (if any, as you developed the module)

N/A

# Module 7b - Vector Space Model (Retrieval)

## Functionality

The class VSM.java has the functionality of module 7b (vector space model - retrieval). The search engine retrieves a list of ranked documents based on the weights calculated. There are other functions within the class that are being called.

Overall, the search engine returns the documents from highest to lowest score

## Limitations (cases not handled)

N/A

## Problems encountered (if any, as you developed the module)

N/A

# Optional Module - Wildcard management

## Functionality

The class Boolean_Model.java has the functionality the wildcard management optional module. If the query contains a wildcard, the query is passed into the *wildcardToInfixFormat* function. This function returns the equivalent query without the wildcard.

For example: the query is “( comput* OR graph* )”. The function returns the following query “( ( ( ( ( ( ( computation OR computational ) OR computations ) OR computer ) OR computer-based ) OR computerized ) OR computers ) OR computing ) OR ( ( ( graph OR graph-theoretical ) OR graphics ) OR graphs ) )”.

In other words, the function finds all the possible words with that starts with “comput” and puts the operator OR between each word. Same for the word graph. The search engines finds all the words starting with “graph” and puts the operator OR between each word.

## Limitations (cases not handled)

N/A

## Problems encountered (if any, as you developed the module)

As we were testing the different queries, the query “comput* AND graph*” does not work with our engine. In comparison, the queries “( comput* AND graph* )” and “( comput* ) AND ( graph* )” returns the corresponding results. We solved the problem by adding print statement in order to analyze the values in the variables. The error was that the if-else statement were incorrect.

# Optional Module - Relevance feedback

## Functionality

The class Probabilistic.java has the functionality of the relevance feedback optional module. Each time the user opens a document, the information are stored into a HashMap, where the key is the user input (query), and the value is another HashMap, where the key is the docID and the value is the number of times the document has been open by the user.

When the user opens the main page, all the information from the text file will be restored into the memory. Each time the user view the course description, all the information from the HashMap will save into the text file in the same directory as the project. This allows the search engine to keep track of relevant documents per query.

## Limitations (cases not handled)

N/A - the search engine stores the necessary info into the HashMap (memory) and saves them into a text file on the user's computer.

## Problems encountered (if any, as you developed the module)

As we were implementing the functions readFile and writeToFile, we were not sure how we would be able to distinguish the difference between the user input (query), the number of time the document has been open by the user. We decided to use the symbol "ééé", this would allow the search engine to separate the three values. We know that in the english language, there are no accents on the letter "e". Therefore, we chose that symbol.

# Optional Module - Probabilistic Relevance Retrieval Model

## Functionality

The class Probabilistic.java has the functionality of the probabilistic relevance retrieval optional module. The search engine checks if the query has exists in the HashMap (HashMap data is read from the text file to retrieve the information). In other words, the user has already written the query.

If the query exists in the memory, the score uses RSV (Relevance Status Value) for each document. There are functions to calculate the pi, ri, cw.

If the query does not exist in the memory, then the scores are calculated using the idf.

Overall, the documents are ranked from highest to lowest scores.

## Limitations (cases not handled)

N/A - the search engine can calculate the scores using idf and RSV and rank the documents in order using the information from the relevance feedback optional module.

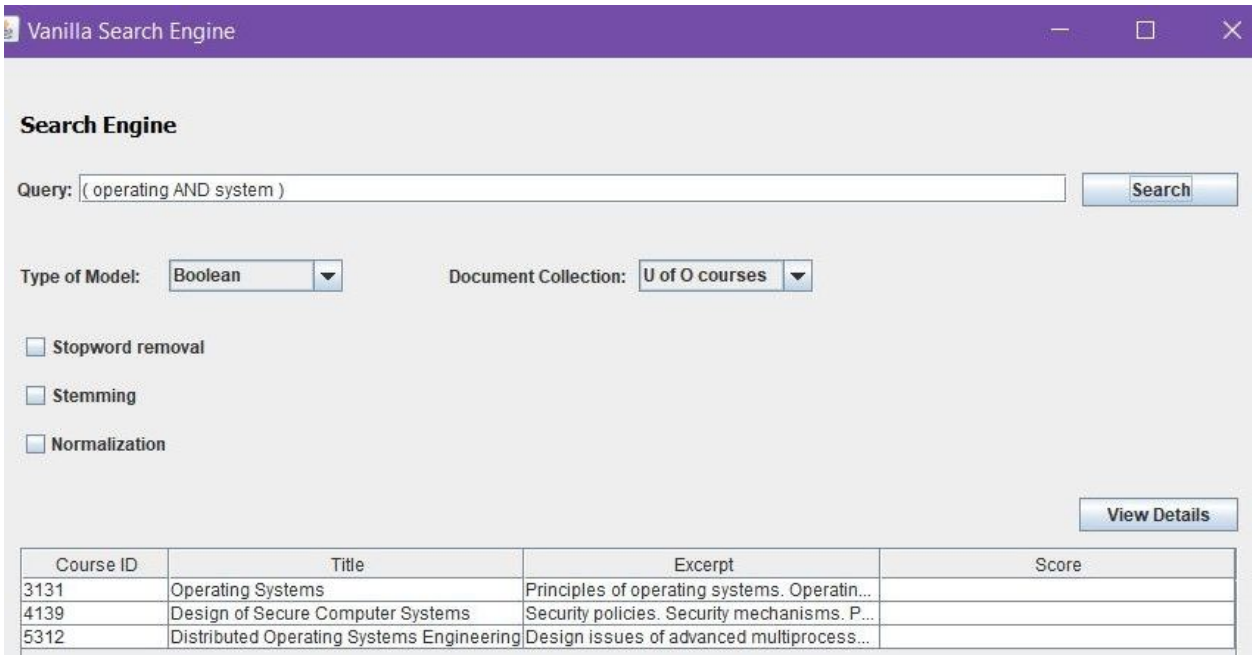
## Problems encountered (if any, as you developed the module)

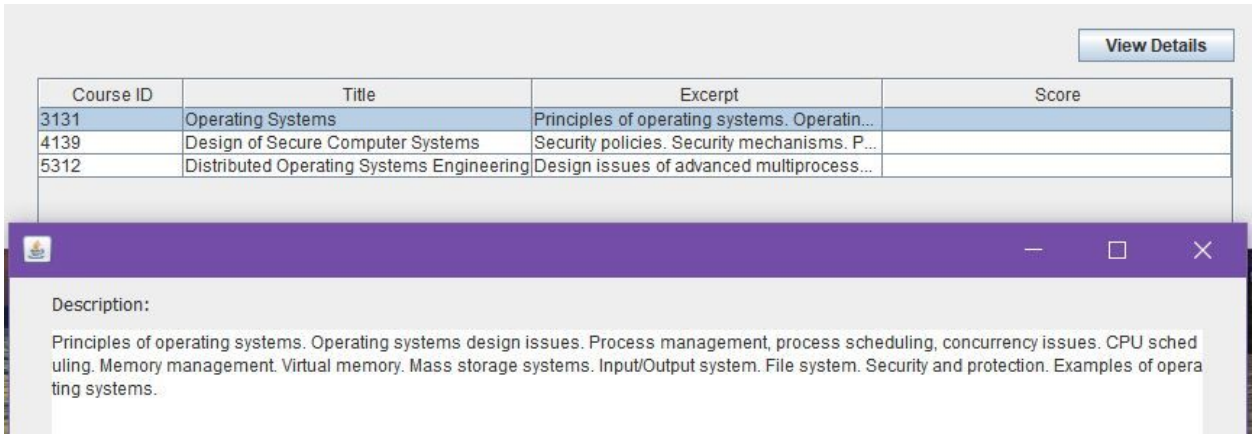
N/A

# Screenshots of the results

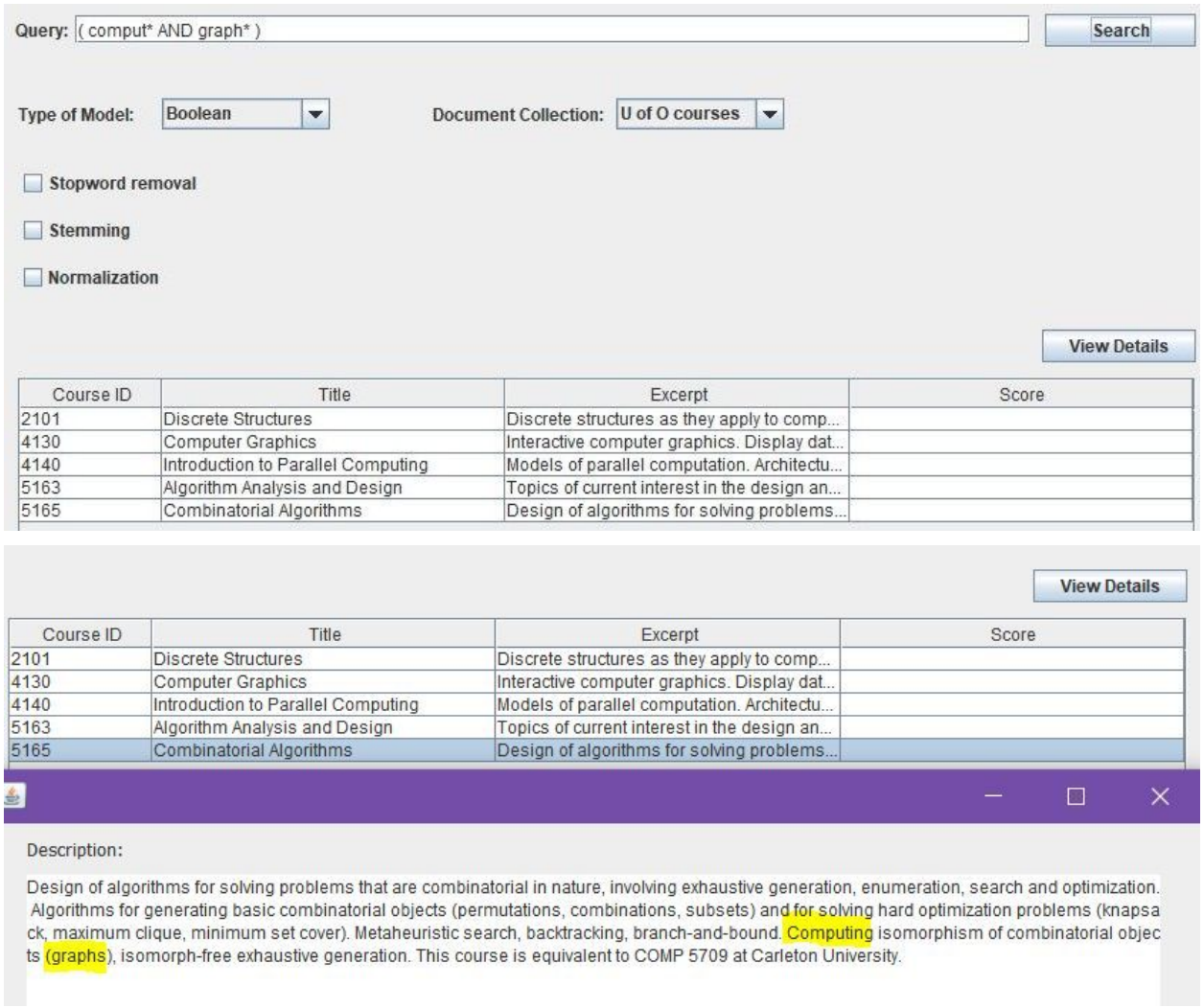
## Boolean Model

■ ( operating AND system )

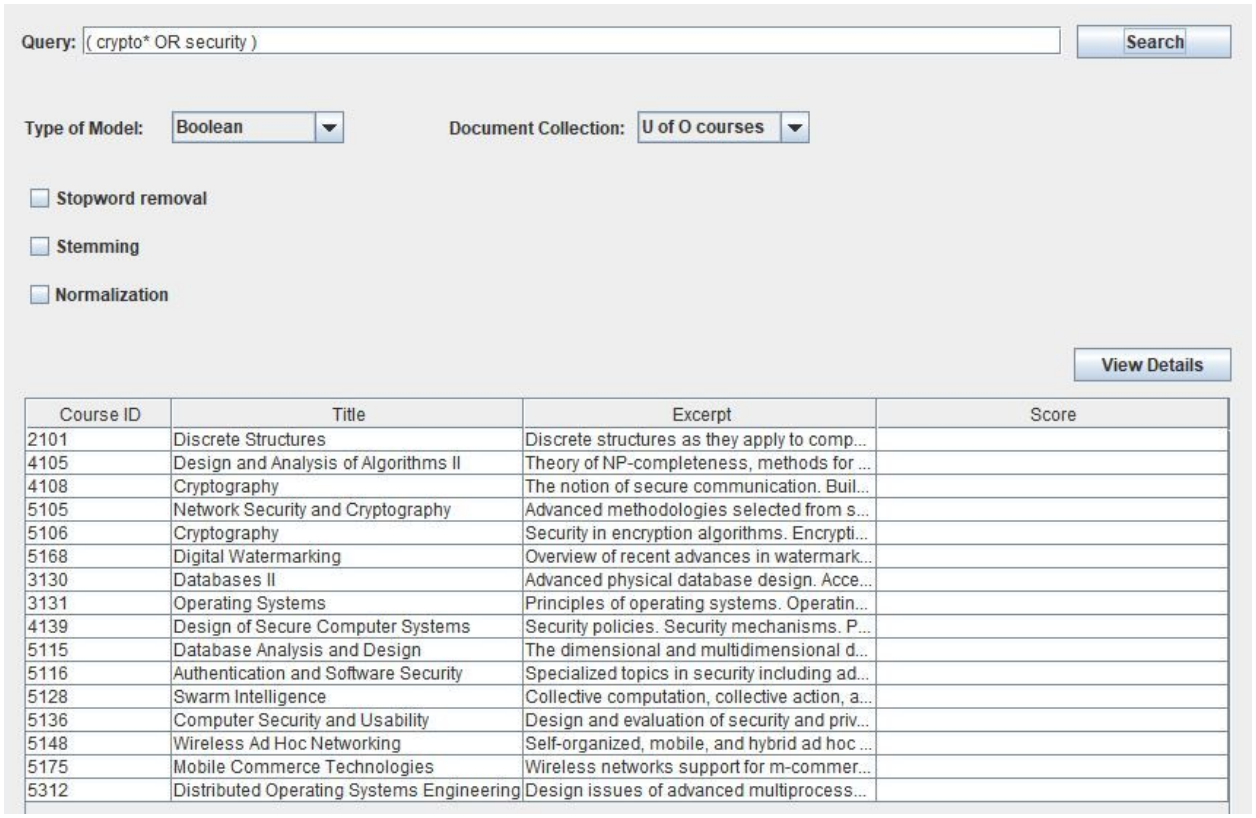


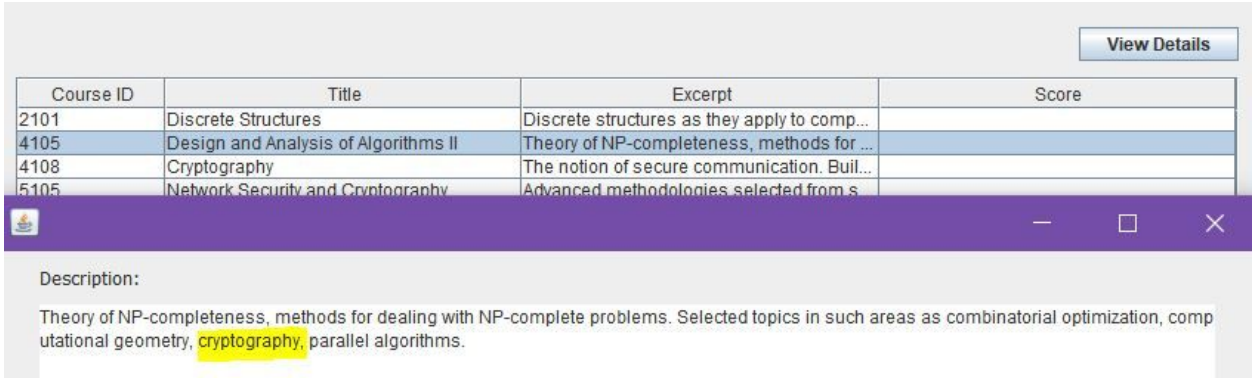
■ ( comput* AND graph* )



■ ( crypto* OR security )




| Course ID | Title | Excerpt | Score |
|---|---|---|---|
| 2101 | Discrete Structures | Discrete structures as they apply to comp... | |
| 4105 | Design and Analysis of Algorithms II | Theory of NP-completeness, methods for ... | |
| 4108 | Cryptography | The notion of secure communication. Buil... | |
| 5105 | Network Security and Cryptography | Advanced methodologies selected from s... | |
| 5106 | Cryptography | Security in encryption algorithms. Encrypti... | |
| 5168 | Digital Watermarking | Overview of recent advances in watermark... | |
| 3130 | Databases II | Advanced physical database design. Acce... | |
| 3131 | Operating Systems | Principles of operating systems. Operatin... | |
| 4139 | Design of Secure Computer Systems | Security policies. Security mechanisms. P... | |
| 5115 | Database Analysis and Design | The dimensional and multidimensional d... | |
| 5116 | Authentication and Software Security | Specialized topics in security including ad... | |
| 5128 | Swarm Intelligence | Collective computation, collective action, a... | |
| 5136 | Computer Security and Usability | Design and evaluation of security and priv... | |
| 5148 | Wireless Ad Hoc Networking | Self-organized, mobile, and hybrid ad hoc ... | |
| 5175 | Mobile Commerce Technologies | Wireless networks support for m-commer... | |
| 5312 | Distributed Operating Systems Engineering | Design issues of advanced multiprocess... | |




| Course ID | Title | Excerpt | Score |
|---|---|---|---|
| 2101 | Discrete Structures | Discrete structures as they apply to comp... | |
| 4105 | Design and Analysis of Algorithms II | Theory of NP-completeness, methods for ... | |
| 4108 | Cryptography | The notion of secure communication. Buil... | |
| 5105 | Network Security and Cryptography | Advanced methodologies selected from s... | |

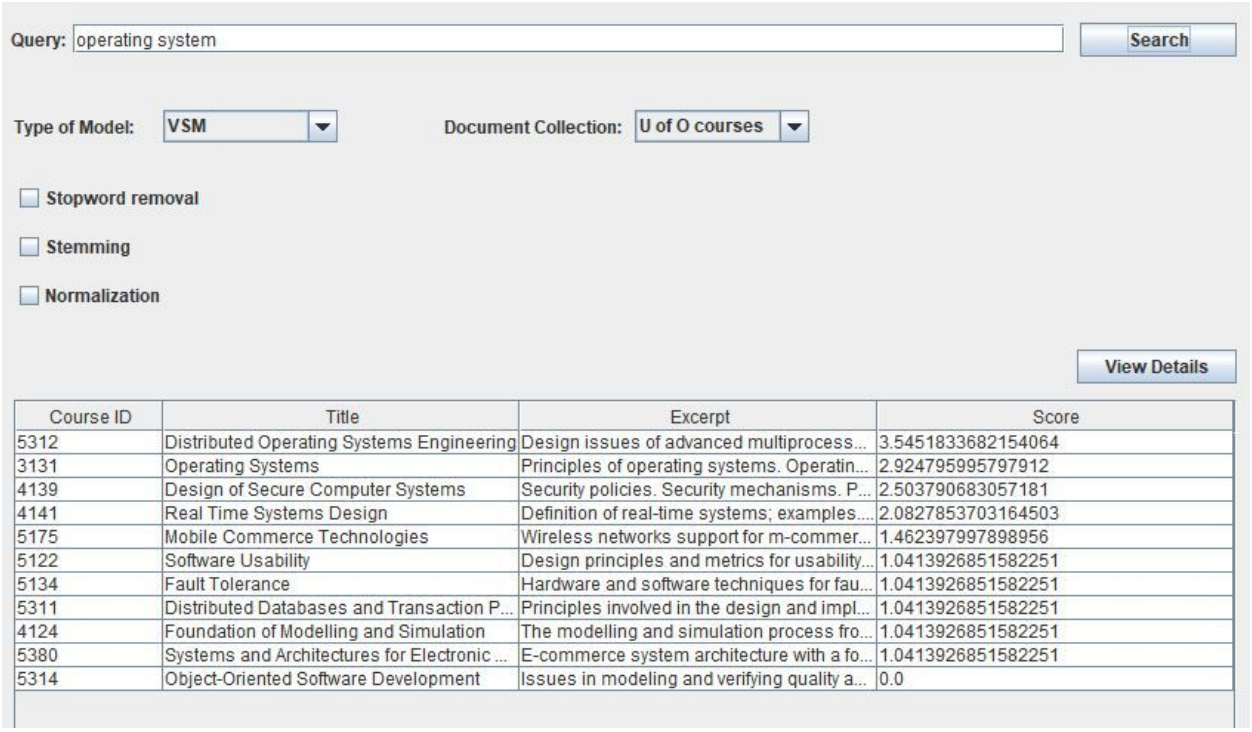


Description:

Theory of NP-completeness, methods for dealing with NP-complete problems. Selected topics in such areas as combinatorial optimization, computational geometry, cryptography, parallel algorithms.

# VSM

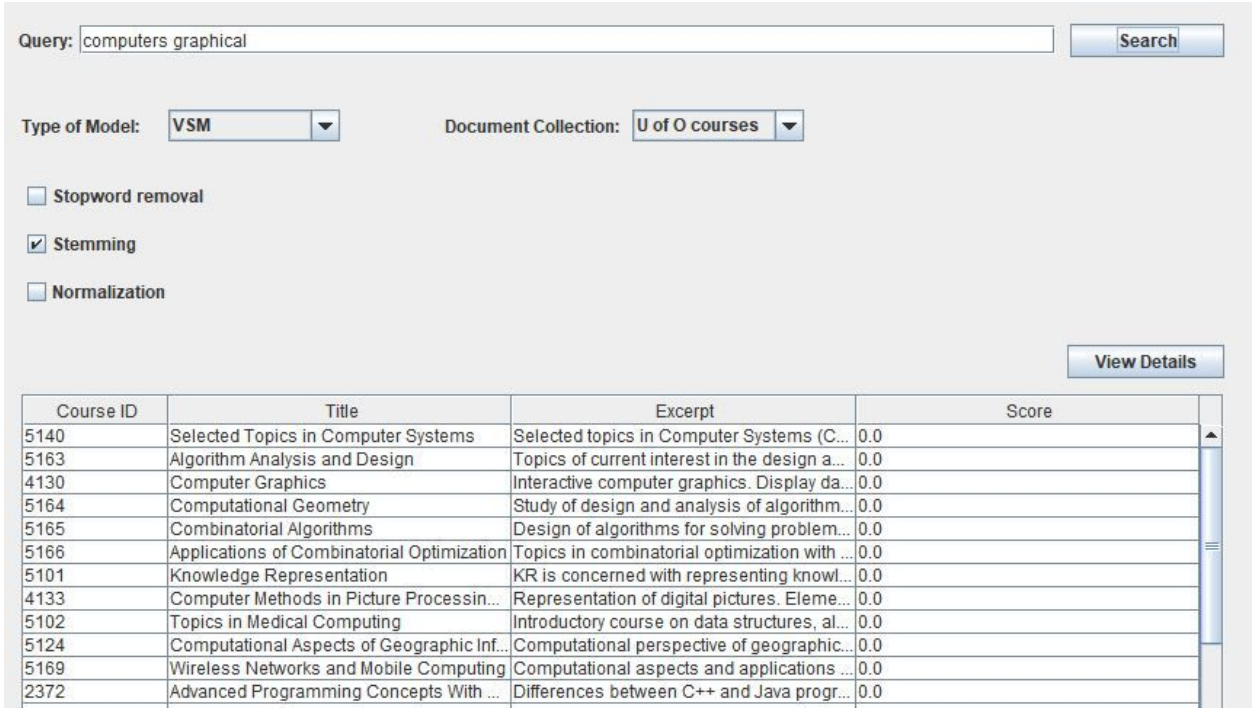
■ operating system




| Course ID | Title | Excerpt | Score |
|---|---|---|---|
| 5312 | Distributed Operating Systems Engineering | Design issues of advanced multiprocess... | 3.5451833682154064 |
| 3131 | Operating Systems | Principles of operating systems. Operatin... | 2.924795995797912 |
| 4139 | Design of Secure Computer Systems | Security policies. Security mechanisms. P... | 2.503790683057181 |
| 4141 | Real Time Systems Design | Definition of real-time systems; examples.... | 2.0827853703164503 |
| 5175 | Mobile Commerce Technologies | Wireless networks support for m-commer... | 1.462397997898956 |
| 5122 | Software Usability | Design principles and metrics for usability... | 1.0413926851582251 |
| 5134 | Fault Tolerance | Hardware and software techniques for fau... | 1.0413926851582251 |
| 5311 | Distributed Databases and Transaction P... | Principles involved in the design and impl... | 1.0413926851582251 |
| 4124 | Foundation of Modelling and Simulation | The modelling and simulation process fro... | 1.0413926851582251 |
| 5380 | Systems and Architectures for Electronic ... | E-commerce system architecture with a fo... | 1.0413926851582251 |
| 5314 | Object-Oriented Software Development | Issues in modeling and verifying quality a... | 0.0 |

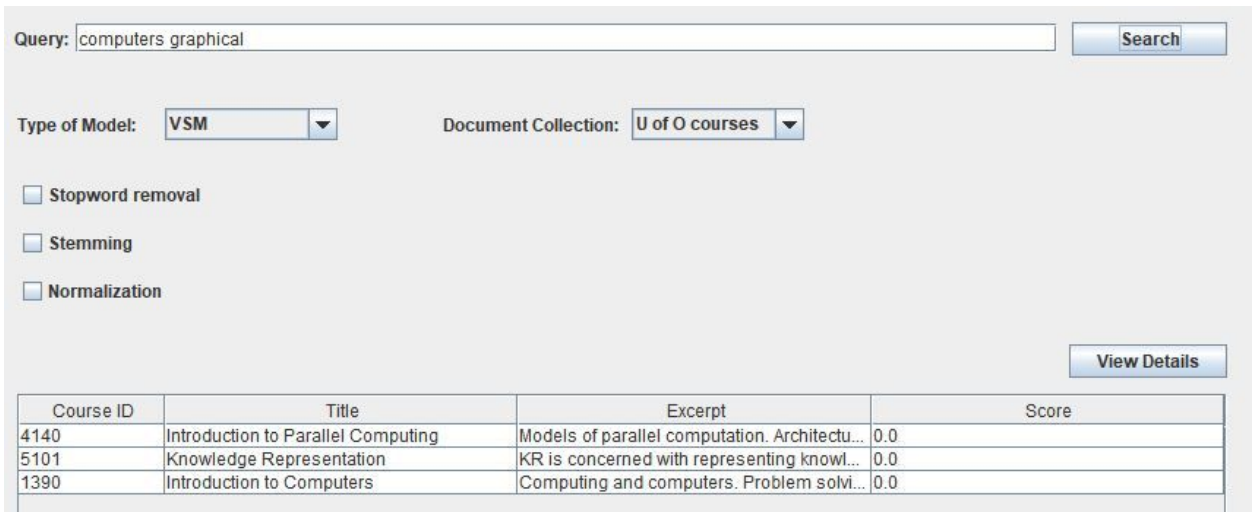
■ computers graphical

Note: Stemming is used here to retrieve relevant results such as ‘Computer Graphics’



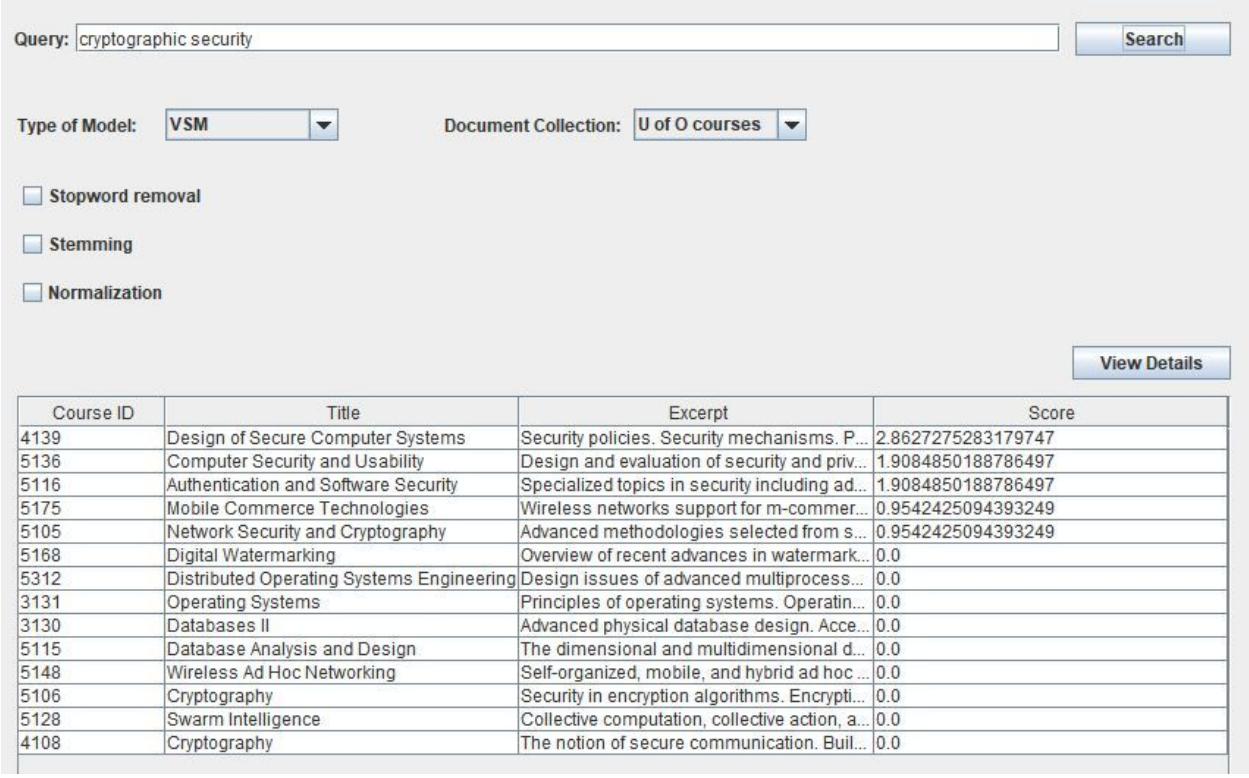

| Course ID | Title | Excerpt | Score |
|---|---|---|---|
| 5140 | Selected Topics in Computer Systems | Selected topics in Computer Systems (C... | 0.0 |
| 5163 | Algorithm Analysis and Design | Topics of current interest in the design a... | 0.0 |
| 4130 | Computer Graphics | Interactive computer graphics. Display da... | 0.0 |
| 5164 | Computational Geometry | Study of design and analysis of algorithm... | 0.0 |
| 5165 | Combinatorial Algorithms | Design of algorithms for solving problem... | 0.0 |
| 5166 | Applications of Combinatorial Optimization | Topics in combinatorial optimization with ... | 0.0 |
| 5101 | Knowledge Representation | KR is concerned with representing knowl... | 0.0 |
| 4133 | Computer Methods in Picture Processin... | Representation of digital pictures. Eleme... | 0.0 |
| 5102 | Topics in Medical Computing | Introductory course on data structures, al... | 0.0 |
| 5124 | Computational Aspects of Geographic Inf... | Computational perspective of geographic... | 0.0 |
| 5169 | Wireless Networks and Mobile Computing | Computational aspects and applications ... | 0.0 |
| 2372 | Advanced Programming Concepts With ... | Differences between C++ and Java progr... | 0.0 |

In comparison to no stemming which returns different results:




| Course ID | Title | Excerpt | Score |
|---|---|---|---|
| 4140 | Introduction to Parallel Computing | Models of parallel computation. Architectu... | 0.0 |
| 5101 | Knowledge Representation | KR is concerned with representing knowl... | 0.0 |
| 1390 | Introduction to Computers | Computing and computers. Problem solvi... | 0.0 |

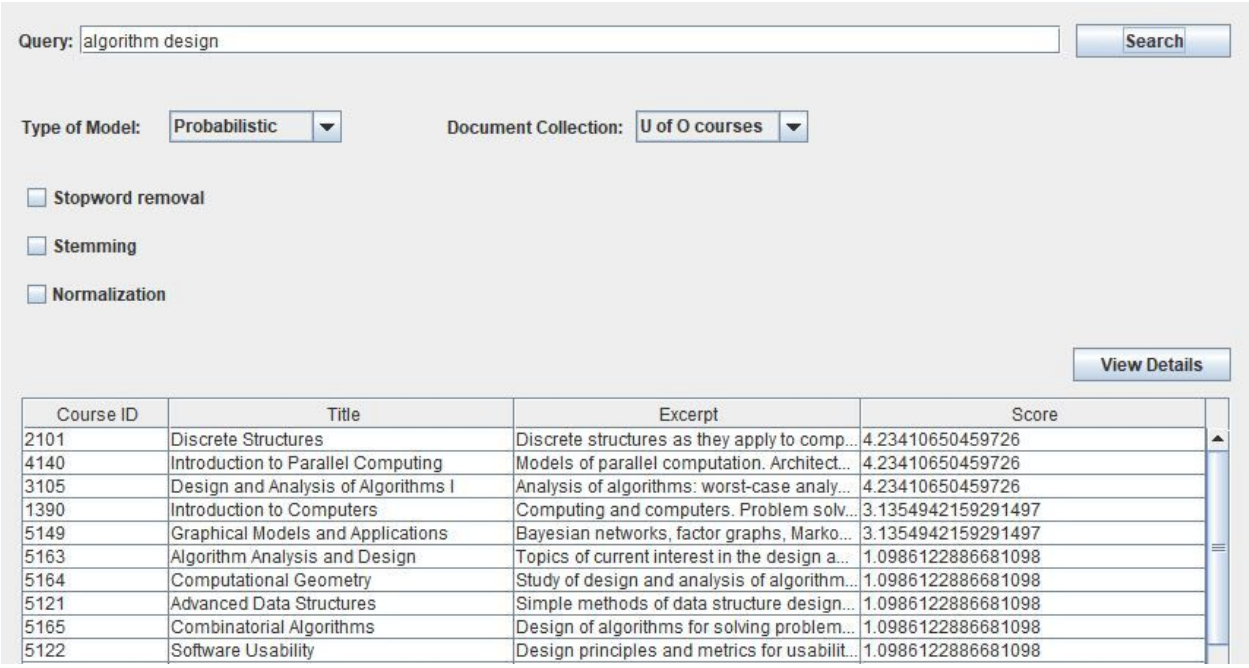
■ cryptographic security




| Course ID | Title | Excerpt | Score |
|---|---|---|---|
| 4139 | Design of Secure Computer Systems | Security policies. Security mechanisms. P... | 2.8627275283179747 |
| 5136 | Computer Security and Usability | Design and evaluation of security and priv... | 1.9084850188786497 |
| 5116 | Authentication and Software Security | Specialized topics in security including ad... | 1.9084850188786497 |
| 5175 | Mobile Commerce Technologies | Wireless networks support for m-commer... | 0.9542425094393249 |
| 5105 | Network Security and Cryptography | Advanced methodologies selected from s... | 0.9542425094393249 |
| 5168 | Digital Watermarking | Overview of recent advances in watermark... | 0.0 |
| 5312 | Distributed Operating Systems Engineering | Design issues of advanced multiprocess... | 0.0 |
| 3131 | Operating Systems | Principles of operating systems. Operatin... | 0.0 |
| 3130 | Databases II | Advanced physical database design. Acce... | 0.0 |
| 5115 | Database Analysis and Design | The dimensional and multidimensional d... | 0.0 |
| 5148 | Wireless Ad Hoc Networking | Self-organized, mobile, and hybrid ad hoc ... | 0.0 |
| 5106 | Cryptography | Security in encryption algorithms. Encrypti... | 0.0 |
| 5128 | Swarm Intelligence | Collective computation, collective action, a... | 0.0 |
| 4108 | Cryptography | The notion of secure communication. Buil... | 0.0 |

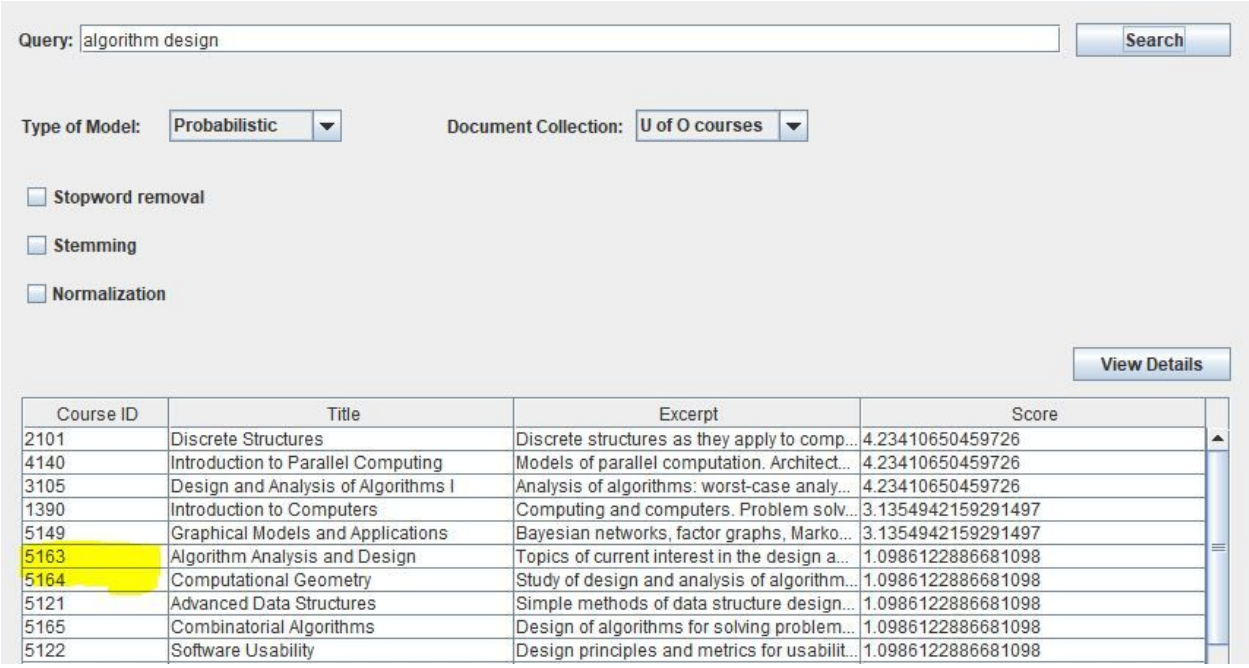
# Probabilistic

Initial results:



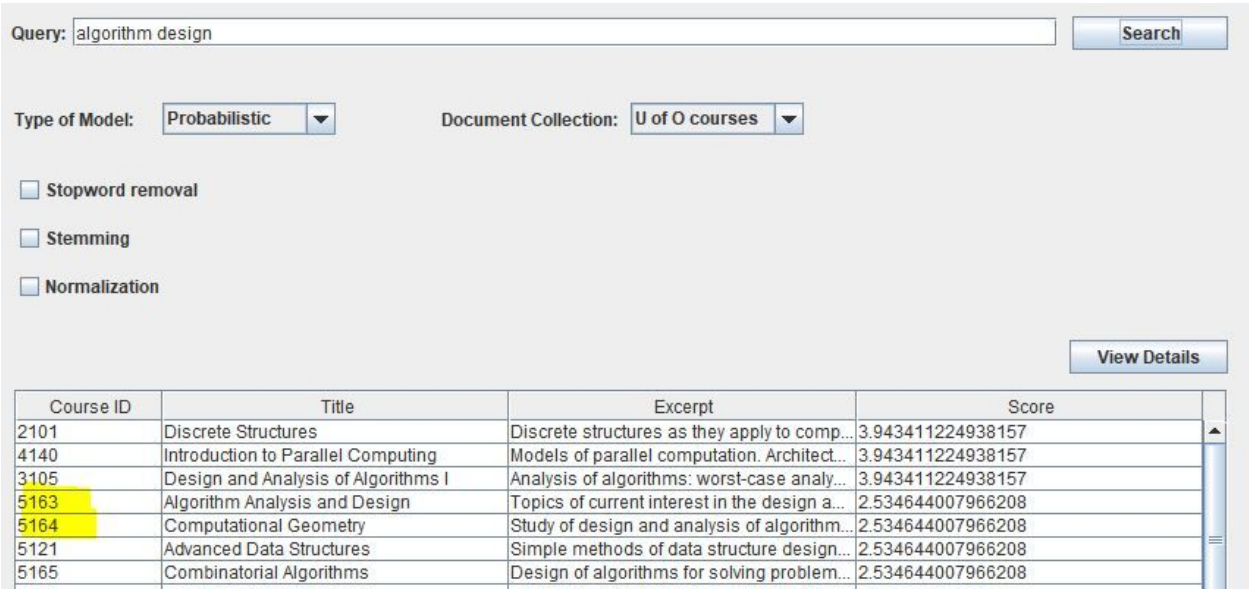

| Course ID | Title | Excerpt | Score |
|---|---|---|---|
| 2101 | Discrete Structures | Discrete structures as they apply to comp... | 4.23410650459726 |
| 4140 | Introduction to Parallel Computing | Models of parallel computation. Architect... | 4.23410650459726 |
| 3105 | Design and Analysis of Algorithms I | Analysis of algorithms: worst-case analy... | 4.23410650459726 |
| 1390 | Introduction to Computers | Computing and computers. Problem solv... | 3.1354942159291497 |
| 5149 | Graphical Models and Applications | Bayesian networks, factor graphs, Marko... | 3.1354942159291497 |
| 5163 | Algorithm Analysis and Design | Topics of current interest in the design a... | 1.0986122886681098 |
| 5164 | Computational Geometry | Study of design and analysis of algorithm... | 1.0986122886681098 |
| 5121 | Advanced Data Structures | Simple methods of data structure design... | 1.0986122886681098 |
| 5165 | Combinatorial Algorithms | Design of algorithms for solving problem... | 1.0986122886681098 |
| 5122 | Software Usability | Design principles and metrics for usabilit... | 1.0986122886681098 |

Let courseID = 5163 and courseID = 5164 be relevant documents (highlighted) after clicking View Details.
Note: These courses are below courseID = 5149.




| Course ID | Title | Excerpt | Score |
|---|---|---|---|
| 2101 | Discrete Structures | Discrete structures as they apply to comp... | 4.23410650459726 |
| 4140 | Introduction to Parallel Computing | Models of parallel computation. Architect... | 4.23410650459726 |
| 3105 | Design and Analysis of Algorithms I | Analysis of algorithms: worst-case analy... | 4.23410650459726 |
| 1390 | Introduction to Computers | Computing and computers. Problem solv... | 3.1354942159291497 |
| 5149 | Graphical Models and Applications | Bayesian networks, factor graphs, Marko... | 3.1354942159291497 |
| 5163 | Algorithm Analysis and Design | Topics of current interest in the design a... | 1.0986122886681098 |
| 5164 | Computational Geometry | Study of design and analysis of algorithm... | 1.0986122886681098 |
| 5121 | Advanced Data Structures | Simple methods of data structure design... | 1.0986122886681098 |
| 5165 | Combinatorial Algorithms | Design of algorithms for solving problem... | 1.0986122886681098 |
| 5122 | Software Usability | Design principles and metrics for usabilit... | 1.0986122886681098 |

When the query “algorithm design” is entered again, the RSV score of these courses have increased and moved up in rank.
Note: they are now underneath courseID = 3105 instead of 5149.

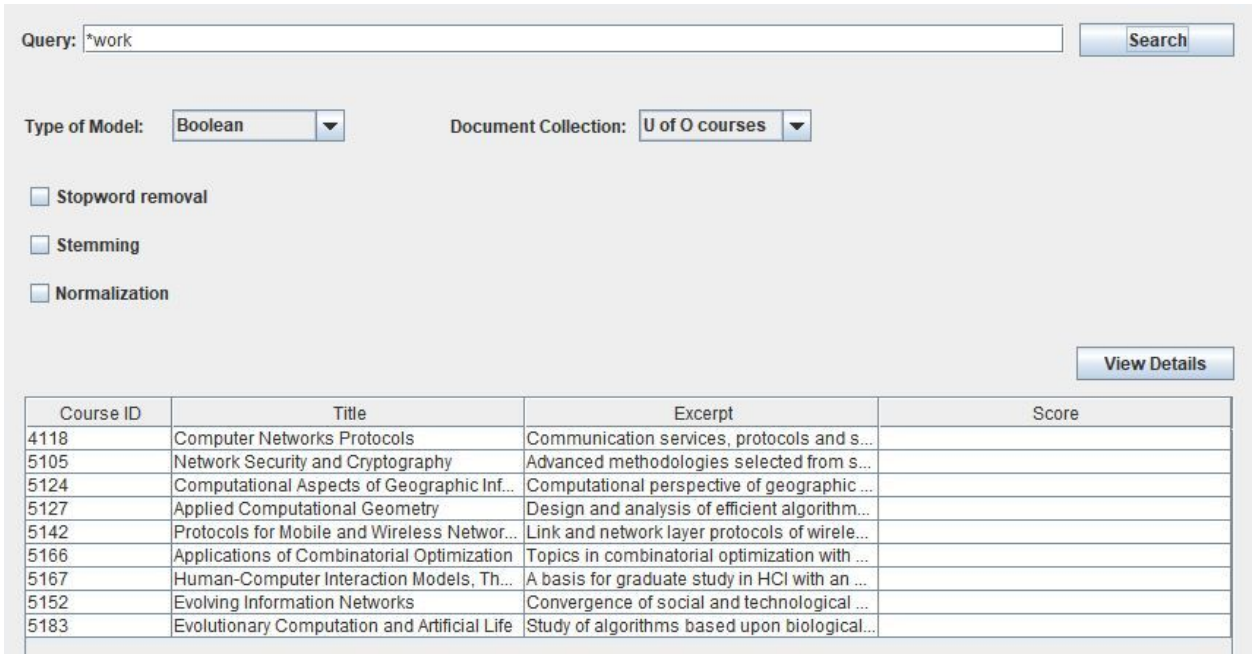



| Course ID | Title | Excerpt | Score |
|---|---|---|---|
| 2101 | Discrete Structures | Discrete structures as they apply to comp... | 3.943411224938157 |
| 4140 | Introduction to Parallel Computing | Models of parallel computation. Architect... | 3.943411224938157 |
| 3105 | Design and Analysis of Algorithms I | Analysis of algorithms: worst-case analy... | 3.943411224938157 |
| 5163 | Algorithm Analysis and Design | Topics of current interest in the design a... | 2.534644007966208 |
| 5164 | Computational Geometry | Study of design and analysis of algorithm... | 2.534644007966208 |
| 5121 | Advanced Data Structures | Simple methods of data structure design... | 2.534644007966208 |
| 5165 | Combinatorial Algorithms | Design of algorithms for solving problem... | 2.534644007966208 |

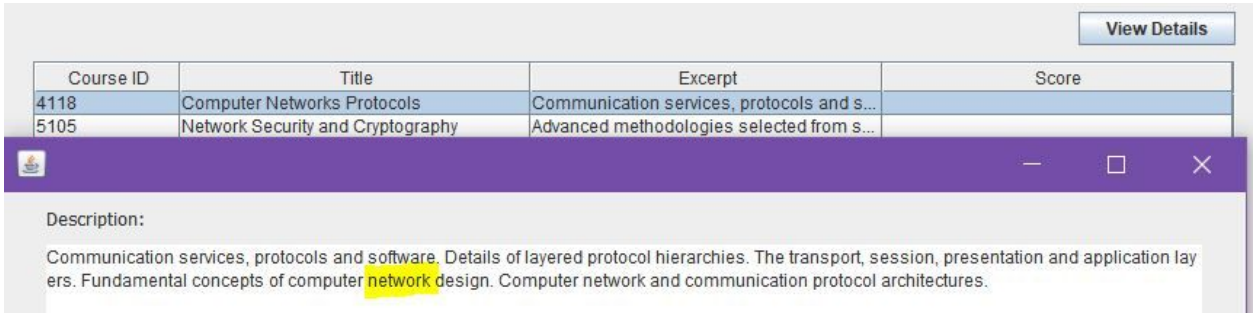
# Wildcard Management

* *at end of a word:* see screenshots for Boolean Model for the given queries

* *at beginning of a word:*

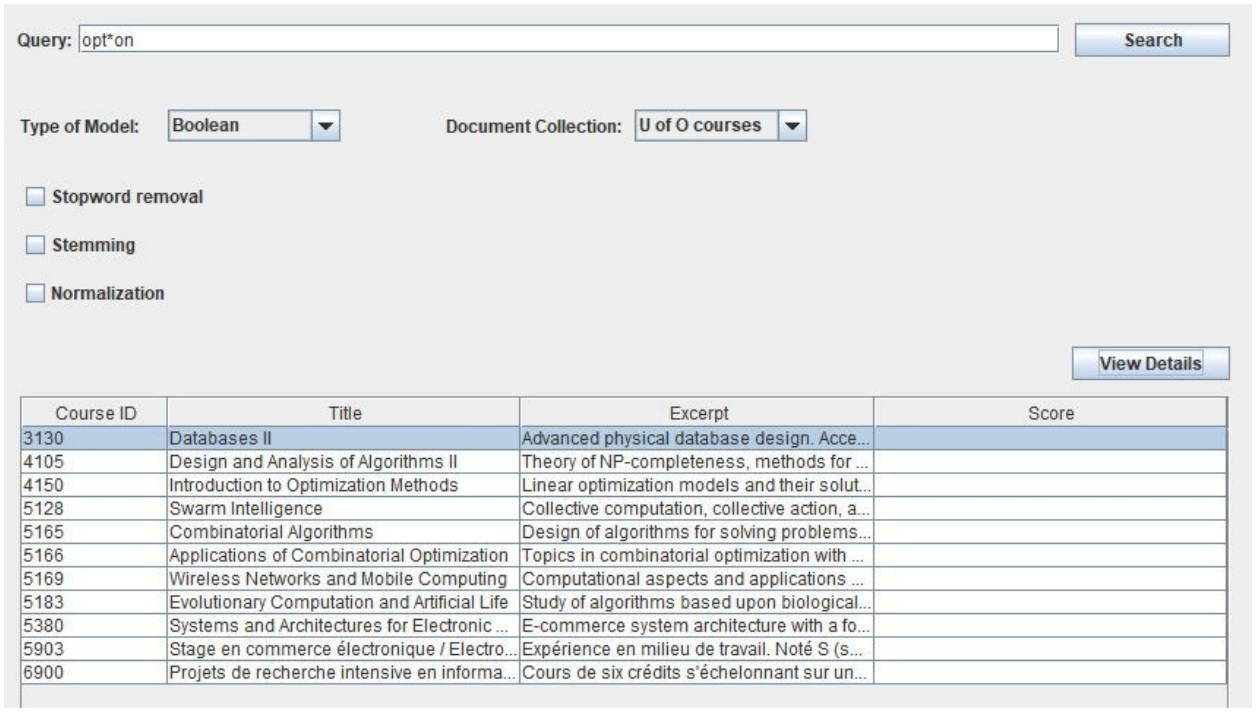



| Course ID | Title | Excerpt | Score |
|---|---|---|---|
| 4118 | Computer Networks Protocols | Communication services, protocols and s... | |
| 5105 | Network Security and Cryptography | Advanced methodologies selected from s... | |
| 5124 | Computational Aspects of Geographic Inf... | Computational perspective of geographic ... | |
| 5127 | Applied Computational Geometry | Design and analysis of efficient algorithm... | |
| 5142 | Protocols for Mobile and Wireless Networ... | Link and network layer protocols of wirele... | |
| 5166 | Applications of Combinatorial Optimization | Topics in combinatorial optimization with ... | |
| 5167 | Human-Computer Interaction Models, Th... | A basis for graduate study in HCI with an ... | |
| 5152 | Evolving Information Networks | Convergence of social and technological ... | |
| 5183 | Evolutionary Computation and Artificial Life | Study of algorithms based upon biological... | |

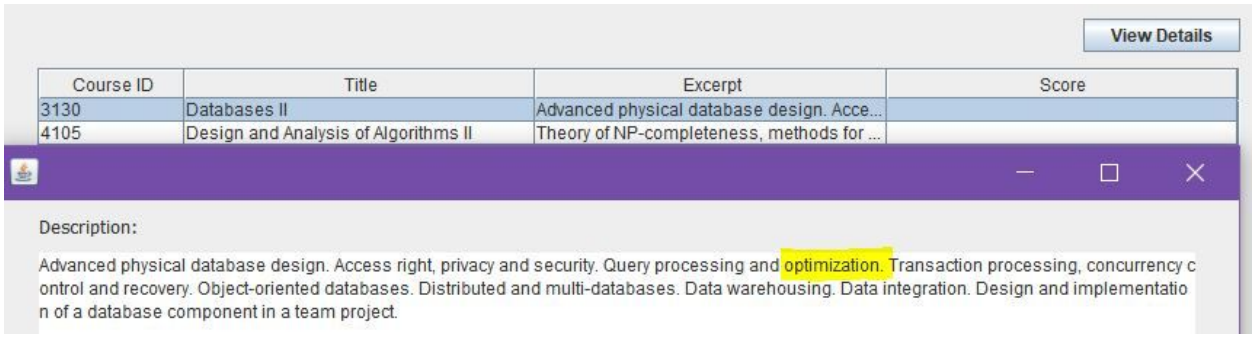



| Course ID | Title | Excerpt | Score |
|---|---|---|---|
| 4118 | Computer Networks Protocols | Communication services, protocols and s... | |
| 5105 | Network Security and Cryptography | Advanced methodologies selected from s... | |

** in-between a word:*



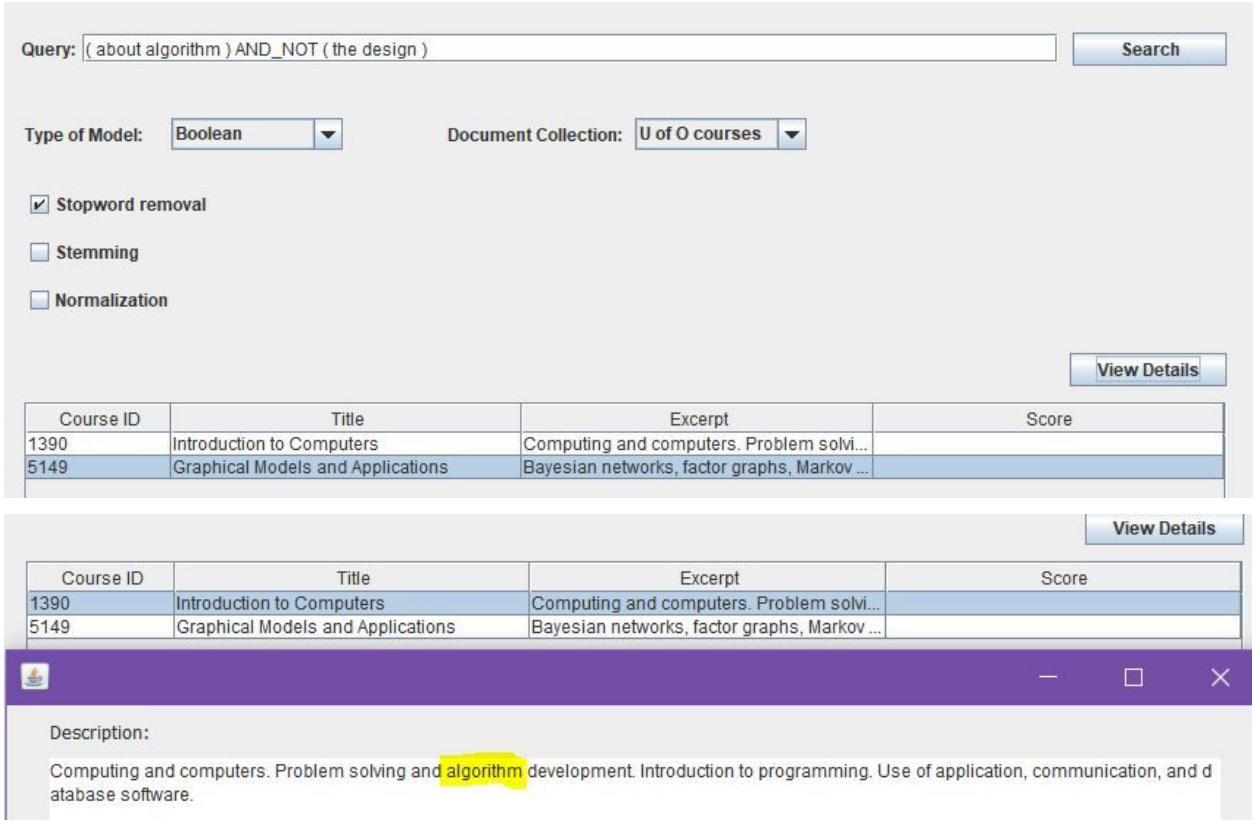

| Course ID | Title | Excerpt | Score |
|---|---|---|---|
| 3130 | Databases II | Advanced physical database design. Acce... | |
| 4105 | Design and Analysis of Algorithms II | Theory of NP-completeness, methods for ... | |
| 4150 | Introduction to Optimization Methods | Linear optimization models and their solut... | |
| 5128 | Swarm Intelligence | Collective computation, collective action, a... | |
| 5165 | Combinatorial Algorithms | Design of algorithms for solving problems... | |
| 5166 | Applications of Combinatorial Optimization | Topics in combinatorial optimization with ... | |
| 5169 | Wireless Networks and Mobile Computing | Computational aspects and applications ... | |
| 5183 | Evolutionary Computation and Artificial Life | Study of algorithms based upon biological... | |
| 5380 | Systems and Architectures for Electronic ... | E-commerce system architecture with a fo... | |
| 5903 | Stage en commerce électronique / Electro... | Expérience en milieu de travail. Noté S (s... | |
| 6900 | Projets de recherche intensive en informa... | Cours de six crédits s'échelonnant sur un... | |




| Course ID | Title | Excerpt | Score |
|---|---|---|---|
| 3130 | Databases II | Advanced physical database design. Acce... | |
| 4105 | Design and Analysis of Algorithms II | Theory of NP-completeness, methods for ... | |

# Stopword removal, stemming, normalization

Eg. Stopwords ‘about’ and ‘the’ are removed from the query which would then yield ( algorithm ) AND_NOT ( design )




Eg. Stemming: see a sample of stemming for VSM for the given query: computers graphical

Eg. Normalization of “a.i.” to “ai”: