

CSI4107 - Information Retrieval
Winter 2019

Search Engine Programming Project
Modules description and requirements

VERSION 1.0
January 15th 2019

This document contains a short description for each module to be included in your Search Engine. This document will undergo modifications as the semester progresses, as descriptions for modules will be added as we learn about them in class.

You might need additional modules as the "glue" to your system. That is fine. Just make sure to keep your design as modular as possible.

Some modules are "underspecified" in the descriptions below, which gives you freedom to implement them the way you want. Think flexible, modular and efficient.

Module 1 - Corpus Pre-processing

Purpose	Convert a collection of documents into a formatted corpus.
Input	A collection of documents as separate files, or a set of "documents" within a single file.
Output	<p>A formatted corpus that can be used by the dictionary building module and the retrieval modules. The output should be a physical output (not just in memory). The conversion should be performed once.</p> <p>The output file created by this corpus pre-processing module will become the input file for other modules.</p> <p>Each document should be uniquely identified with a docID.</p>
Requirements and challenges	<p>Any collection (UofO-courses, Reuters, web collection) will have a specific format. The corpus pre-processing module will transform each collection into a corpus defined with a <u>uniform format</u> so that the rest of your search engine is not dependent on the formatting of the original documents.</p> <p>Your challenge here is to think of a good "uniform format" and implement it.</p>
Modules depending on this module.	Dictionary building. Indexing. Retrieval modules.
Modules required by this module.	None. This could even be done in a separate project from the Search Engine.

Module 2 - User Interface

Purpose	Allow a user to access the search engine capabilities.
Input	None.
Output	Set of choices + query string.
Requirements and challenges	<p>The UI should include:</p> <ul style="list-style-type: none"> (a) The name for your Search Engine (b) Input box for user to write their query (c) Output as list of DocID (or titles), each one with a link (or icon) to open it, as well as an excerpt line (for now just take the first line of document), and a score. (d) The choice for model: Boolean, VSM, ... (e) The choice for collection: UofO catalog, .. <p>The UI should also allow to click a retrieved document link and see its full content (perhaps in another window).</p>
Modules depending on this module.	Retrieval modules.
Modules required by this module.	Any module giving a set of possibilities to show (list of models available, list of collections available).

Module 3 - Dictionary building

Purpose	Build a dictionary of terms to be indexed.
Input	A formatted corpus of documents (see output of Module 1).
Output	A set of terms to be indexed.
Requirements and challenges	<p>The dictionary building should have a few different modes/options (which you would eventually test when we do later evaluation):</p> <ul style="list-style-type: none"> - With/without stopword removal - With/without stemming (Porter Stemmer or other) - With/without normalization (hyphens, and period removals)
Modules depending on this module.	Indexing modules.
Modules required by this module.	Corpus Pre-processing module.

Module 4 - Inverted Index Construction

Purpose	Associate dictionary terms to documents.
Input	Document collection and dictionary.
Output	Term-Document index.
Requirements and challenges	<p>You can implement your indexing the way you desire, as long as it allows retrieval in <u>linear time</u>, in a way similar to the "postings sort-merge" algorithm we saw in class. This linear time constraint might not be important for small collections of documents, but it would be for larger collections.</p> <p>Your indexing should allow both the Boolean Model and the Vector Space Model (VSM) to work. As the VSM requires term weights associated with each document, the index should allow for this information to be included. The weight would not be used within the boolean model.</p> <p>The index should contain for each term, the set of docIDs for the documents it is found in (and its weight). This main index could eventually be complemented with additional indexes.</p>
Modules depending on this module.	All the different search modules.
Modules required by this module.	Dictionary building and corpus pre-processing.

Module 5 - Corpus access

Purpose	Access documents from the corpus.
Input	Set of document IDs.
Output	Set of corresponding documents (including title, excerpt line, link to full content).
Requirements and challenges	You can implement this any way you want, as long as it's possible to retrieve the information to be displayed in the UI from the set of docIDs.
Modules depending on this module.	UI module.
Modules required by this module.	Any retrieval model could provide a set of docIDs to this module.

Module 6 - Boolean Retrieval Model

Purpose	Implement the boolean retrieval model.
Input	1. A query from the user expressed with boolean operators. 2. Selected collection.
Output	A set document ids (corresponding to unique documents in your collection).
Requirements and challenges	<p>A query can contain boolean operators (AND, OR, AND_NOT), as well as parentheses () to delimit operators reach. To make things simpler, you could assume that the user will delimit everything using parentheses.</p> <p>Ex. printer AND laser OR ink → not required to deal with this form Ex. printer AND (laser OR ink) → yes, user should ask in this form</p> <p>Query Processing is challenging. An approach is to transform the infix format provided by the user (as above) to a postfix format.</p> <p>Ex. printer laser ink OR AND</p> <p>(see here for Python: http://interactivepython.org/runestone/static/pythonds/BasicDS/InfixPrefixandPostfixExpressions.html). Then, it is easier to process the query with postfix format.</p> <p>Also, a query could contain words ending with wildcards (ex. print*) which would then be transformed into a set of OR based on the dictionary.</p> <p>It is best to split this module into a Query Preprocessing Module (to handle the wildcard) and a Retrieval module to deal with the query. This would be good especially if you include the optional wildcard processing module (see optional modules).</p>
Modules depending on this module.	Corpus access.
Modules required by this module.	UI as it will provide the query and the collection on which to perform the search.

Module 7a - Vector Space Model (Weight calculation)

(Will be presented during Jan. 25th lecture)

Purpose	Include term weights in the index.
Input	Index. Collection of documents. (or only the index if the frequencies have been already included).
Output	A weighted index.
Requirements and challenges	According to the tf-idf weighting, you will need to calculate the weights assigned with each term in each document and include this in the index.
Modules depending on this module.	VSM retrieval
Modules required by this module.	Index construction.

Module 7b - Vector Space Model (Retrieval)

(Will be presented during Jan. 25th lecture)

Purpose	Implement the Vector Space Model for retrieval
Input	1. A query from the user expressed with a list of terms. 2. Selected collection.
Output	A set of ranked document ids (corresponding to unique documents in your collection).
Requirements and challenges	You will use the weighted indexed terms to retrieve (and rank) the documents.
Modules depending on this module.	Corpus Access.
Modules required by this module.	UI as it will provide the query and the collection on which to perform the search.

Optional Module - Phrase Indexing

(Will be presented during Jan. 25th lecture)

Purpose	Include phrases in the index.
Input	Document collection.
Output	Updated dictionary and updated index. (or could keep them separate)
Requirements and challenges	Identify frequent "phrases" (collocations) in the documents and add those to the dictionary. Modify the index accordingly.
Modules depending on this module.	<i>To be completed according to your design</i>
Modules required by this module.	<i>To be completed according to your design</i>

Optional Module - Spelling correction with Edit Distance

Purpose	Provide suggestions of corrected words to the user.
Input	Query
Output	Suggestion for a different query (if applicable). "Did you mean X? "
Requirements and challenges	<p>Make sure the edit distance calculation is performed rapidly, especially if the dictionary is quite large and the input query must be compared to all words.</p> <p>You can use some heuristics, such as the fact that people very rarely get the first letter wrong, so you can limit the search to the dictionary words starting with the same word.</p> <p>You could do weighted edit distance to have a lesser cost on particular characters (such as vowels or letters that are close by on keyboards).</p> <p>There might be more than one candidate word for the correction. You should show the top N most <i>likely</i> candidates. For likeliness, you could use their frequency in the corpus.</p>
Modules depending on this module.	UI for sure.... perhaps others.
Modules required by this module.	Dictionary building.

Optional Module - Spelling correction with Soundex

Purpose	Provide suggestions of corrected words to the user.
Input	Query
Output	Suggestion for a different query (if applicable). "Did you mean X? "
Requirements and challenges	<p>You could pre-process all dictionary terms to include their corresponding soundex codes. And then perform the comparison at query time.</p> <p>Soundex will quickly retrieve many words that are possible. You should show the top N most <i>likely</i> words. For likeliness, you could use the word frequency in the corpus.</p>
Modules depending on this module.	UI for sure.... perhaps others.
Modules required by this module.	Dictionary building.

Optional Module - Wildcard management

Purpose	Manage the wildcards include within words (not just at the end). This is assumed to work ONLY in the boolean model.
Input	Query
Output	It depends where you include this module, as query pre-processing, or within the boolean retrieval module.
Requirements and challenges	<p>End of word wildcards (e.g. print*) have been included in the boolean indexing module (or in a separate query preprocessing module).</p> <p>Here, we wish to deal with wildcards present anywhere in the word (e.g. pr*nt, or *graphy)</p> <p>To efficiently deal with middle-of-word wildcards, it is best to create a secondary index of bigram letters which lead to the primary index.</p>
Modules depending on this module.	<i>To be completed according to your design</i>
Modules required by this module.	<i>To be completed according to your design</i>

Optional Module - Relevance feedback

(description will be added after we see this in class)

Purpose	Either implicitly or explicitly capture the documents looked-at by the user.
Input	
Output	
Requirements and challenges	
Modules depending on this module.	
Modules required by this module.	

Optional Module - Probabilistic Retrieval Model

(description will be added after we see this in class)

Purpose	Implement the Probabilistic model for retrieval
Input	
Output	
Requirements and challenges	
Modules depending on this module.	
Modules required by this module.	