

Learning Content Models for Semantic Search ADD

Professional Advisor:

Michael Elhadad

Academic Advisor:

Menahem Adler

Team Members:

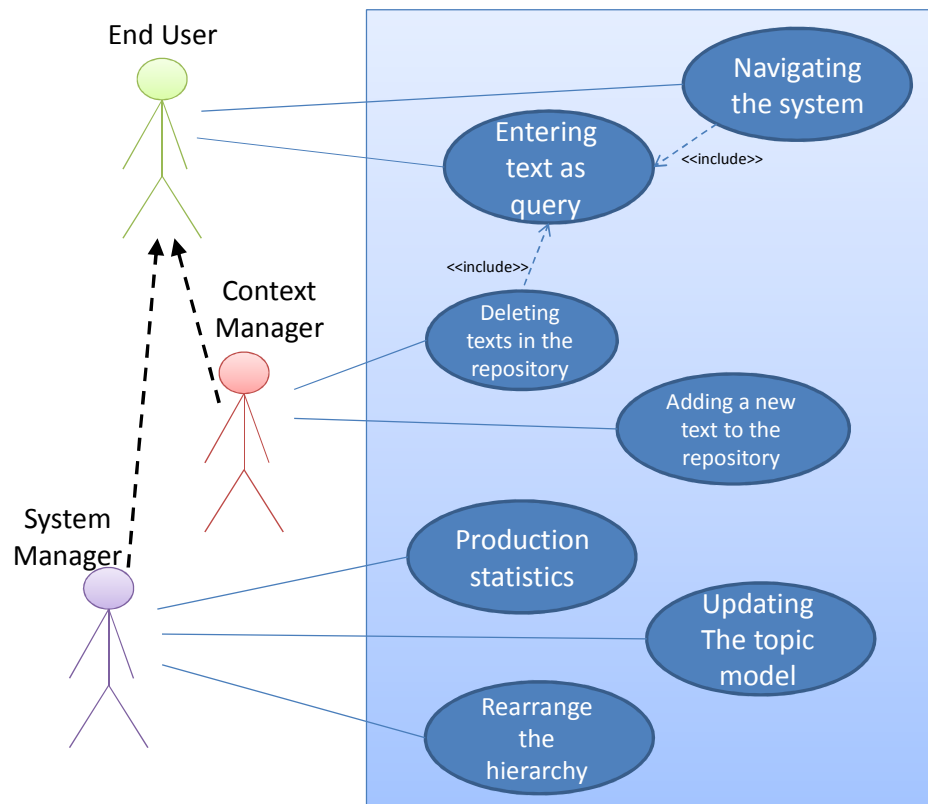
Eran Peer

Hila Shalom

Contents

Use Cases	3
Data Model	7
Description of Data Objects & Data Objects Relationships	7
Databases	8
Behavioral Analysis	11
Sequence Diagrams	11
Events	15
States	18
Object-Oriented Analysis	19
Class Diagrams	19
Class Description	21
Packages	27
Unit Testing	28
System Architecture	30
User Interface Draft	31
Testing	49
Task List	50
Appendix	52

1 Use Cases



Use Case 1

Name: Entering a text as query.

Primary Actor: The end-user

Pre-Condition: The application is running. The preprocessing has been done successfully. The repository is not empty.

Post-Condition: The system returns an answer to the user. The answer is like what described above.

Description: The user can enter a text as query for the purpose of searching some data he is looking for.

Flow of events:

1. The user entered the application.
2. The user enters a text as a query
3. The user submits the query.
4. The system processing the query.
5. The system searching for the query's relevant data.
6. The system presents to the user an answer, as described above.
7. The system saves the user's use (the texts and topics) at the statistics.
8. The user can choose one or more of the texts that are part of the answer.

Alternative flows: Instead of choosing texts, the user can navigate the system by choosing topics that are also part of the answer, or enter a new query (back to 3).

Use Case 2

Name: Navigating the system.

Primary Actor: The end-user

Pre-Condition: The application is running. The preprocessing has been done successfully. The repository is not empty. The user submitted a query and got an answer (by the use case "Entering a text as query").

Post-Condition: The system shows to the user the hierarchy topics tree. The hierarchy topics tree is like what described above.

Description: The user can navigatethe system by navigating the hierarchy topics tree for the purpose of searching some data he is looking for.

Flow of events:

1. The user chooses one of the topics.
2. If the topic is not a leaf in the tree, the system shows to the user the topic's subtopics (back to 1).
3. If the topic is a leaf in the tree, i.e. the texts of this topic, the system shows to the user the full references texts list.
 - 3.1. The user can choose the references to the texts that he wants.

Alternative flows: at any point the user can choose one or more from the given texts, or enter a new query.

Use Case 3

Name: Producing statistics

Primary Actor: System manager

Pre-Condition: The application is running. The user is registered to the system as the system's manager.

Post-Condition: The system returns the relevant statistics to the user.

Description: The user can produce statistics about the system.

These statistics:

- How many times a topic has been selected in searches.
- How many times a document has been selected in searches.
- How many times a document and a topic has been selected together in searches.

Flow of events:

1. The user chooses to produce statistics.
2. The system calculates and returns to the user the described above statistics.

Use Case 4

Name: Updating the topic model.

Primary Actor: System manager

Pre-Condition: The application is running. The user is registered to the system as the system's manager.

Post-Condition: The topic model has been updated.

Description: The system manager can update the topic model by changing the algorithm of topic modeling.

Flow of events:

1. The user chooses to update the topic model.
2. The system shows to the user the current topic model.
3. The user changes the topic model.
4. The user submits the changes.

Use Case 5

Name: Rearrange the hierarchy.

Primary Actor: System manager

Pre-Condition: The application is running. The user is registered to the system as the system's manager.

Post-Condition: The hierarchy topics tree has been rearranged.

Description: The system manager can update the hierarchy topics tree. He can add/remove/change topics names/change the hierarchy/... of the hierarchy topics tree.

Flow of events:

1. The user chooses to rearrange the hierarchy topics tree.
2. The system shows to the user the current the hierarchy topics tree. A
3. The user changes the hierarchy topics tree.
4. The user submits the changes

Use Case 6

Name: Adding a new text to the repository.

Primary Actor: Content manager

Pre-Condition: The application is running. The user is registered to the system as the content's manager.

Post-Condition: The new text has been added to the repository, annotated and with the suitable topics. The hierarchy topics model tree is updated respectively.

Description: The user can add new text files to the repository.

Flow of events:

1. The user chooses to add a new text file.
2. The user submits a text file.
3. The system adds this text to the corpus, and does the database preprocessing again (the morphology analyzing, the topic modeling and the indexing).

Use Case 7

Name: Deleting a text from the repository.

Primary Actor: Content manager

Pre-Condition: The application is running. The user is registered to the system as the content's manager. The user found the text he wants to delete (by the use case "Entering a text as query").

Post-Condition: The text has been deleted. The hierarchy topics model tree is updated respectively.

Description: The user can delete existing text from the repository.

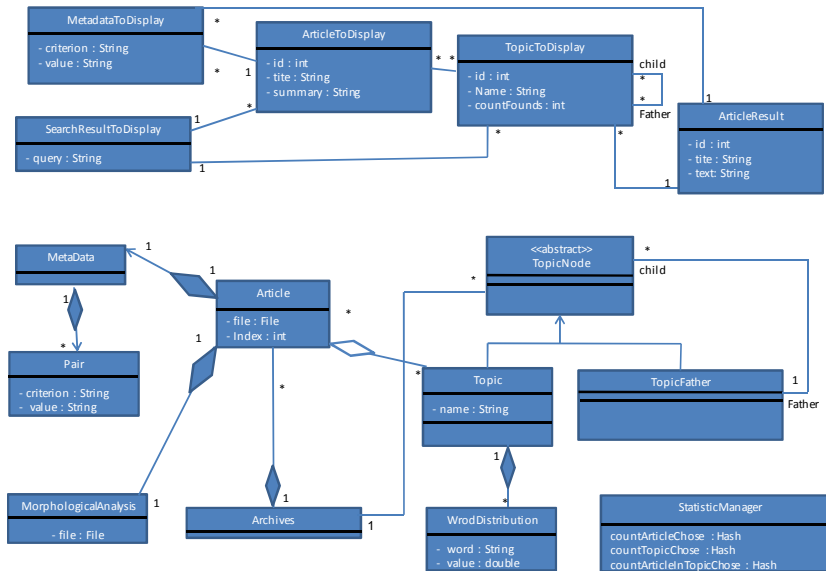
Flow of events:

1. The user chooses to delete the selected text.
2. The system removes this text from the corpus, and does the database preprocessing again (the morphology analyzing, the topic modeling and the indexing).

2 Data Model

2.1 Description of Data Objects

2.2 Data Objects Relationships



2.3 Databases

The major tables is the Articles table and the Topic table. Those tables have a relation between them – a single article can have many topics, and a single topic can be attached to many articles.

The Article table's fields are:

- id (Number) – primary key. Sequence number that uniquely describes each article. Article receives it when it added to the Archives.
- text(File) – Each article has its own unique text file, that represents the article's text.
- metadata(<criteria (Text), value (Text)>) – Each article has the metadata that helps to understand what there is in the article's text. Each metadata consists of the tuples <criteria, value>.
- morphological analysis (File) – Each article has a morphological analysis of its text.

The Topic table's fields are:

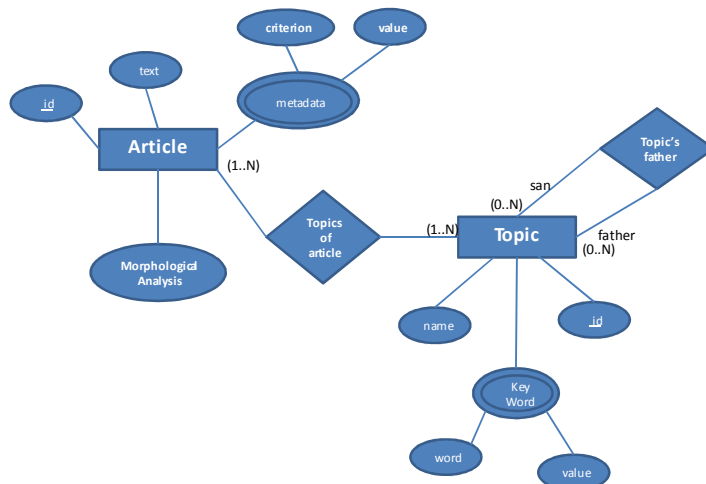
name (Text) – the name of the topic. Each topic has a unique name, but for convenience we chose to use an id field as the primary key.

key (<word (Text), value (Number)>) – Each topic has its own words that represents it. Each word in the topic has its distribution in the topic.

id (Number) – primary key. Sequence number that uniquely describes each topic. Topic receives it when it created.

Each topic (except of the root) has some topic fathers in the tree.

Each topic (except of the leafs) has some topic children in the tree.



Name: Article addition.

Description: Adding a new article to the system.

Affected Table: Article.

Added Fields: id, text, metadata.

Deleted Fields: None.

Updated Fields: None.

Added Relations:None.

Deleted Relations:None.

Updated Relations:None.

Name: Article removal.

Description: Removing an existing article from the system.

Affected Table: Article.

Added Fields: None.

Deleted Fields: id, text, metadata, morphological analysis.

Updated Fields: None.

Affected Table: Topic.

Added Fields: None.

Deleted Fields: None.

Updated Fields: None.

Added Relations:None.

Deleted Relations:Topics of articles.

Updated Relations:None.

Name: Morphological analysis addition.

Description: Adding morphological analysis to article.

Affected Table: Article.

Added Fields: morphological analysis.

Deleted Fields: None.

Updated Fields: None.

Added Relations:None.

Deleted Relations:None.

Updated Relations:None.

Name: Article-Topic relation addition.

Description: Adding relation an existing article an existing topic.

Affected Table: Article.

Added Fields: None.

Deleted Fields: None.

Updated Fields: None.

Affected Table: Topic.

Added Fields: None.

Deleted Fields: None.

Updated Fields: None.

Added Relations: Topics of articles.

Deleted Relations:None.

Updated Relations:None.

Name: Rebuilding the hierarchal topics tree.

Description: Executing the LDA algorithm, so it rebuilds the TopicModel and the hierarchical topics tree.

Affected Table: Topic.

Added Fields: None.

Deleted Fields: None.

Updated Fields: name, key, id.

Added Relations: Topic's father.

Deleted Relations: Topic's father.

Updated Relations: Topic's father.

Affected Table: Article.

Added Fields: None.

Deleted Fields: None.

Updated Fields: None.

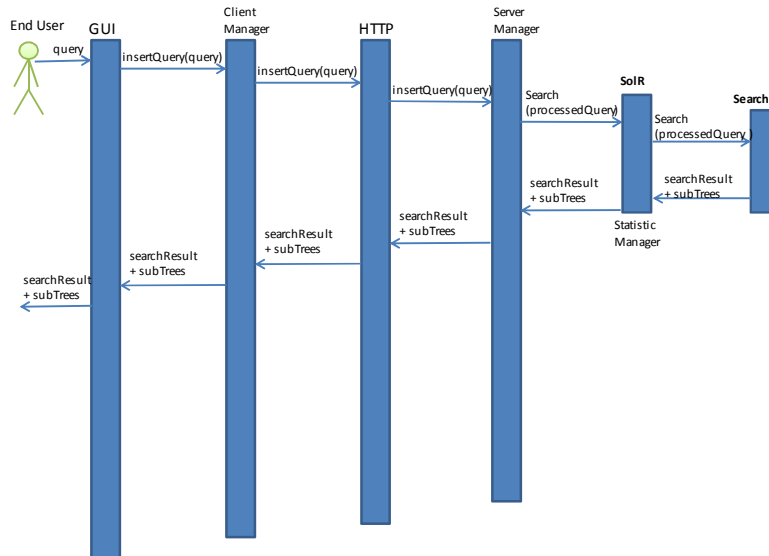
Added Relations: Topics of articles.

Deleted Relations: Topics of articles.

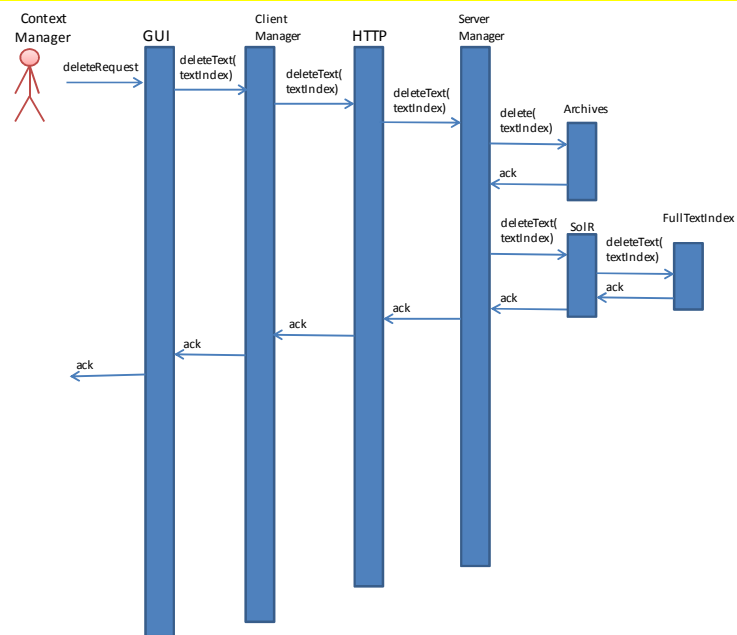
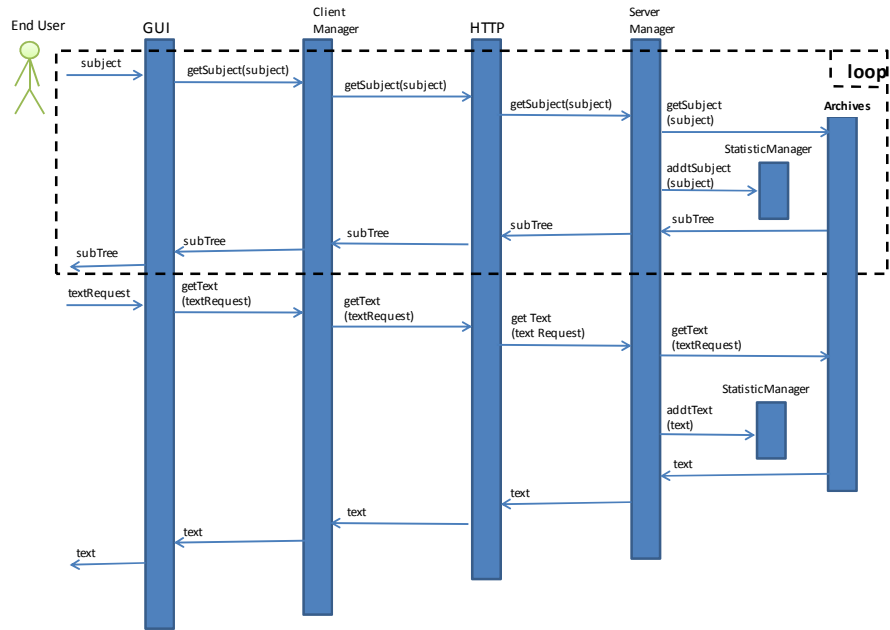
Updated Relations: Topics of articles.

3 Behavioral Analysis

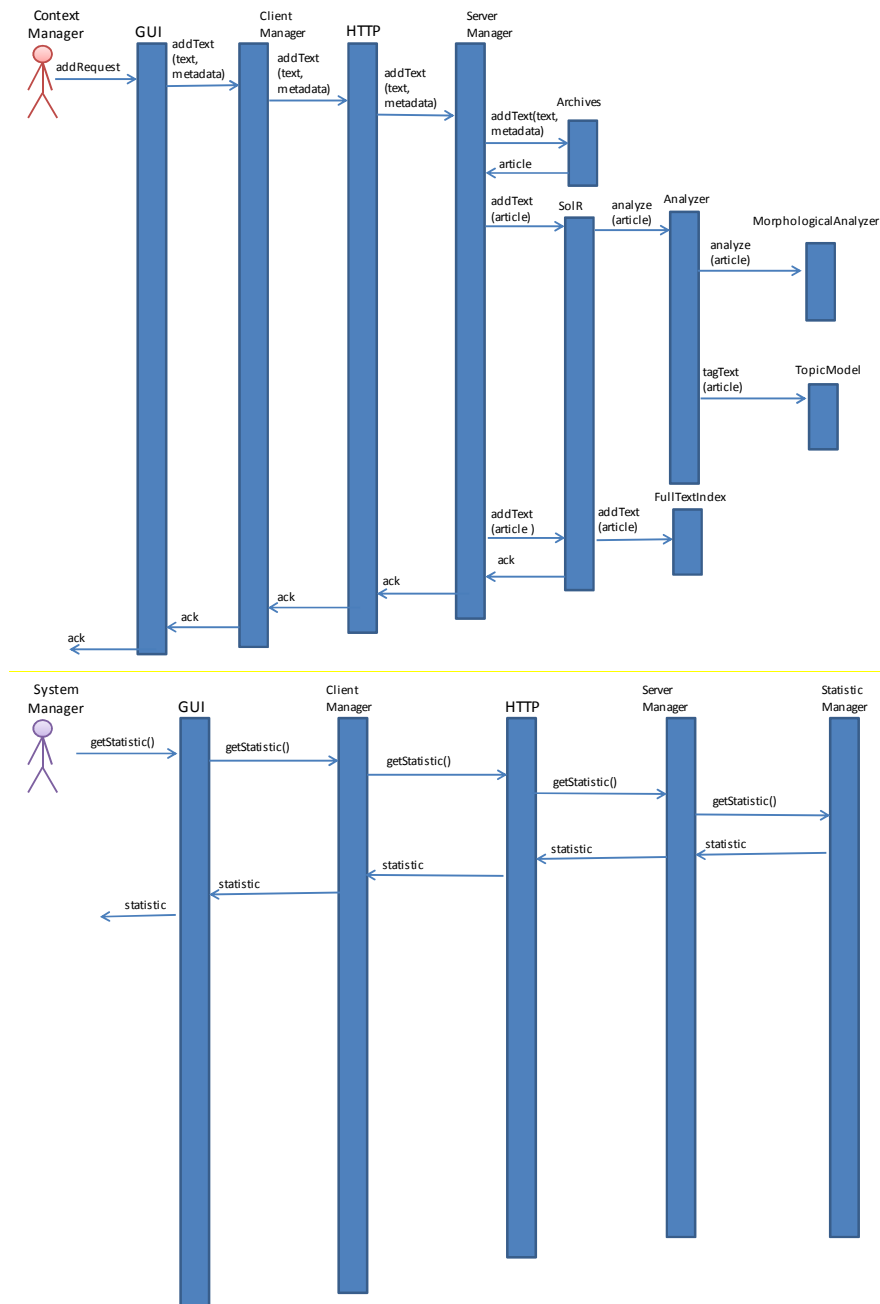
3.1 Sequence Diagrams



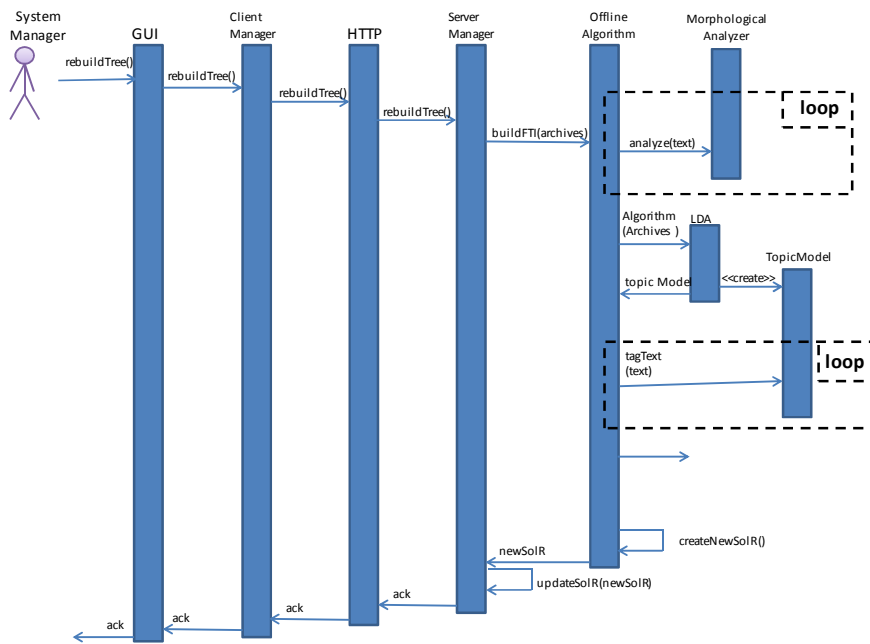
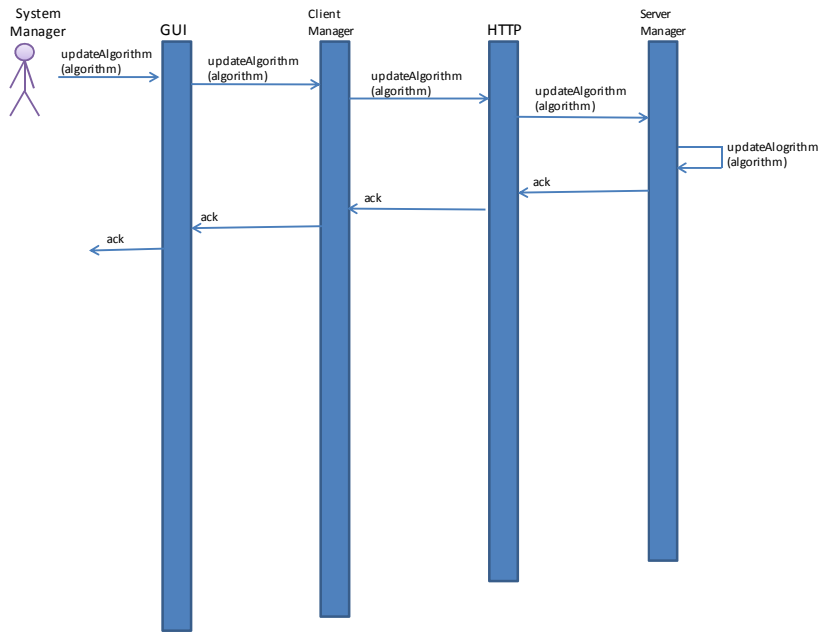
Learning Content Models for Semantic Search



Learning Content Models for Semantic Search



Learning Content Models for Semantic Search



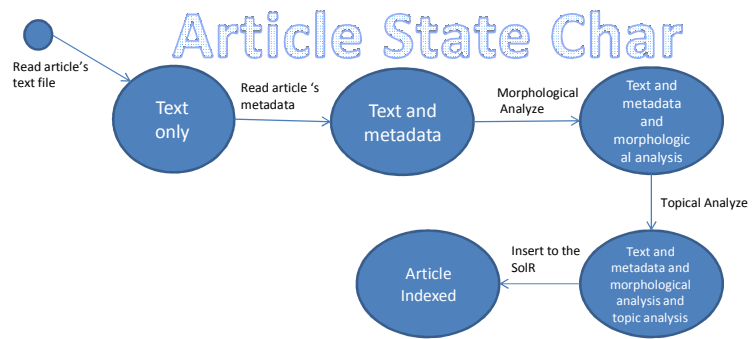
3.2 Events

Name	Sender	Receiver	Description	Action
Entering text as query	End User		The user can enter a text as query for the purpose of searching some data he is looking for.	SolR processes the given query with the QueryProcessor, searches for the processed query's result with the Search, and returns the search result and the relevant topics sub trees.
Navigating the system	End User		The user can navigate the system by navigating the hierarchy topics tree for the purpose of searching some data he is looking for.	While the user chooses a subject - The ServerManager asks and gets the relevant topics sub tree from the Archives and returns it. The ServerManager also "tells" the StatisticManager that the subject has been chosen so it can remember this user selection of the subject. When the user chooses a text - The ServerManager asks and gets the relevant text from the Archives and returns it. The ServerManager also "tells" the StatisticManager that the text has been chosen so it can remember this user selection of the text.
Deleting texts from the repository	Context Manager		The user can delete existing text from the repository.	The ServerManager asks the Archives and the SolR to delete a given text. SolR transfers this request to the FullTextIndex.
Adding a new text to the repository	Context Manager		The user can add new text files to the repository.	The ServerManager gets a new text and its metadata, adds them to the Archives, and gets a

				<p>new article.</p> <p>The ServerManager analyzes the new article by calling to SolR that calls the Analyzer that calls to the MorphologicalAnalyzer, which analyzes it. SolR also calls the TopicModel, which tags the new article. Now the text is analyzed and tagged, the ServerManager asks SolR to add the new article, which asks the FullTextIndex to add it.</p>
Productionstatistics	System Manager		<p>The user can produce statistics about the system. These statistics:</p> <ul style="list-style-type: none"> •How many times a topic has been selected in searches. • How many times a document has been selected in searches. •How many times a document and a topic have been selected together in searches. 	<p>The StatisticManager gets a request for statistics, as a result produces and returns the statistics.</p>
Updating the topic model	System Manager		<p>The system manager can update the topic model by changing the algorithm of topic modeling.</p>	<p>The ServerManager gets a request for updating the LDA algorithm, and updates it.</p>

Rearrange the hierarchy	System Manager		<p>The application is running. The user is registered to the system as the system's manager.</p>	<p>The ServerManager gets a request to rebuild the topics tree. It sends a request to the OfflineAlgorithm which:</p> <ol style="list-style-type: none"> 1. While there are unanalyzed texts in the archives - analyzes the text by calling to the MorphologicalAnalyzer's method "analyze" with the text. 2. Activates the LDA algorithm on the archives. The LDA creates and returns the new TopicModel. 3. While there are untagged texts in the archives - tags the text by calling to the TopicModel's method "tagText" with the text. 4. Creates a new SolR and returns it to the ServerManager, that updates its SolR to the new SolR.
-------------------------	----------------	--	--	---

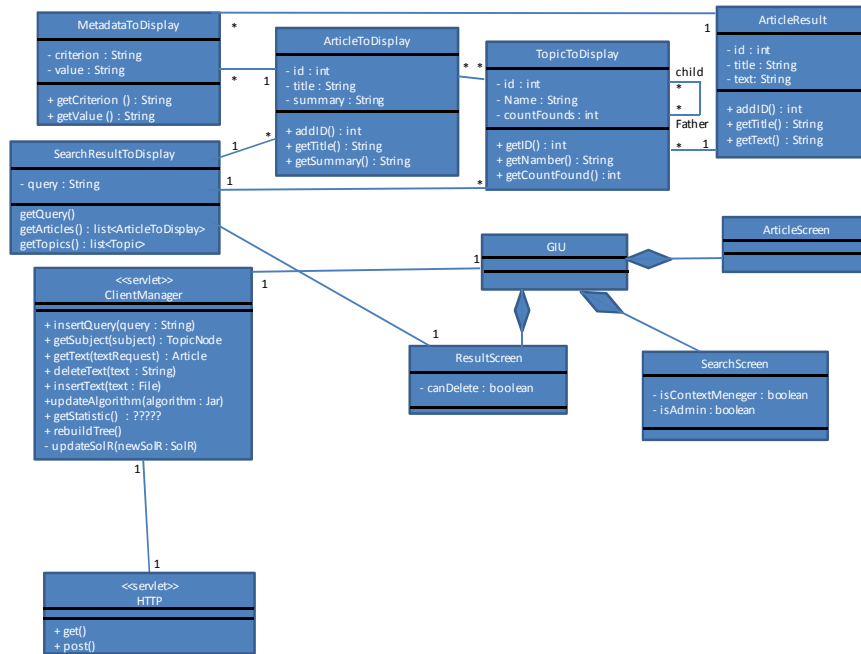
3.3 States

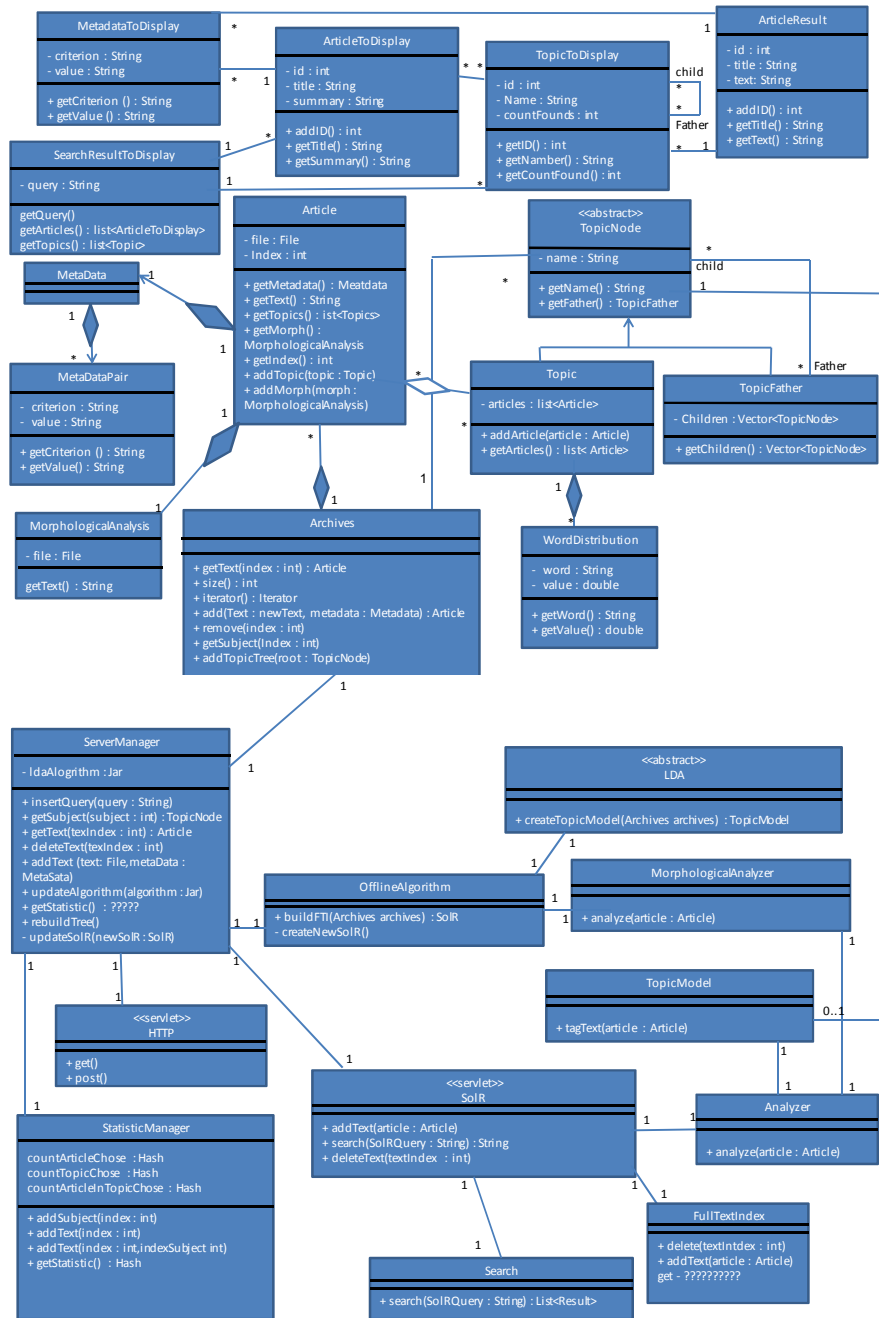


4 Object-Oriented Analysis

4.1 Class Diagrams

Client :



Server :

4.2 Class Description

Article

Article is an object that holds the data about a text.

The data is:

- the serial ID in the Archives (=repository)
- the text's File
- the text's metadata
- the text's morphological analyze as a File
- the topics' list, that the text deals with them

This object created and updated during the offline run. At the beginning it only has serial ID, the text's File and metadata. During the running the morphological analyze and the topics.

Main methods:

getText() – returns the text itself (as String) of the Article.

getMetadata() – returns the metadata of the Article.

getTopics() – returns the list of all the Article's topics.

@pre: the Article already has the complete topics list.

getMorph() – returns the Article's morphological analyze.

@pre: the Article already has the complete morphological analyze.

getIndex() – returns the Article's ID.

addTopic(Topic) –

- gets a topic (=subject).
- if the topic doesn't exists in the topics list, adds it to the topics list.

@post: If the given topic is not already in the list - list@post = list@pre+Topic.

addMorph(morph) –

- gets morphological analyze of the Article.
- adds the given morphological analyze as the Article's morphological analyze.

@pre: the Article doesn't have a morphological analyze. The given morphological analyze is a good morphological analyze of the Article.

@post: The morphological analyze of the Article is the given morphological analyze.

Archives

Archives is an object that holds the collection of all the Articles. It also has the hierarchy topics tree and the collection of all the topics.

Main methods:

addTopicTree(root) –

- gets the root of the hierarchy topics tree.
- adds all the topics to the topics collection.

@pre: the Archives doesn't have the hierarchy topics tree.

getText(index) –

- gets an index of the wanted text.
- returns the text's Article.

@invariant: the given index must be ≥ 0 and \leq the number of the Articles in the Archives.

size() – returns the number of the Articles in the Archives.

iterator() – returns the iterator that can iterate over all the Articles in the Archives.

add(text,metadata) –

- gets text and its metadata.
- makes a new Article with the given text and metadata.
- adds the new Article to the Archive.
- returns the new Article.

@post: a new Article added to the Archive.

remove(index) –

- gets the index of the wanted Article.
- removes the Article from the Archive.

@invariant: the given index must be ≥ 0 and \leq the number of the Articles in the Archives.

@pre: the requested Article is in the Archive.

@post: the requested Article is no longer in the Archive.

getSubject(index) –

- gets the index of the wanted topic.
- returns the topic from the topics collection.

@invariant: the given index must be ≥ 0 and \leq the number of the topics in the Archives.

@pre: the requested Topic is in the Archive's topics collection.

TopicNode, Topic, TopicFather

Those three classes are design pattern named Composite. In this case:

- TopicNode is the Component – can be Topic or TopicFather
- Topic is the Leaf – has collection of all the Articles that deals with it.
- TopicFather is the Composite – it has many TopicNodes

The role of this threesome is to represent the hierarchy topics tree.

Each topic would be an instance of the object Topic. Topic has the key words which can tell if a given text deals with this topic or not.

Each key word represented by the object WordDistribution. Each WordDistribution, by the contact of a Topic, has the word it represent and the word's distribution in the specific Topic.

Main methods:

For TopicNode:

getFather() – returns instance of TopicFather which is the first ancestor of the current TopicNode. If the current TopicNode is the earliest ancestor (Adam, Eve or the root of the three), the method returns null.

getName() – returns the topic's name.

For TopicFather:

getChildren() – returns vector of all the current topic's first children.

For Topic:

addArticle(article) – adds the given Article to the topic's Articles.

@invariant: the given article's text deals with the topic.

getArticles() – returns the collection of all the topic's Articles.

Metadata

Metadata is an object that its main role is to keep general data that helps to identify texts. This object consists of the object `MetaDataPair` which represents pairs of `<criteria, value>`. The criteria describes Archives' field and the value is the criteria's value.

MorphologicalAnalysis

This is an external library we use. You can read and use it at the following links:

<http://www.cs.bgu.ac.il/~sindany/nlp/db/>

<http://www.cs.bgu.ac.il/~sindany/nlp/>

<http://www.cs.bgu.ac.il/~adlerm/dat/thesis.pdf>

TopicModel

`TopicModel` implements the Command design pattern, this object has one major method that executes an algorithm.

`TopicModel`'s role is to tag a given article's text with the topics the text deals with them. This object has the hierarchy topics tree.

Main method:

`tagText(article)` – adds to a given article a list of topics that the article's text deals with. In addition, adds to each topic from the list, the article's text.

LDA

`LDA` implements the Command design pattern, this object has one major method that executes an algorithm.

`LDA`'s role is to make the `TopicModel` and the hierarchy topics tree.

Main method:

`createTopicModel(archives)` – builds, by the given archives, the hierarchy topics tree and updates the tree in the archives.

@pre: all the texts in the archives have `MorphologicalAnalysis`

MorphologicalAnalyzer

`MorphologicalAnalyzer` implements the Command design pattern, this object has one major method that executes an algorithm.

`MorphologicalAnalyzer`'s role is to do morphological analyze to a given article's text.

Main method:

`analyze(article)` – adds to a given article's text its morphological analysis.

OfflineAlgorithm

`OfflineAlgorithm` implements the Command design pattern, this object has one major method that executes an algorithm.

`OfflineAlgorithm`'s role is to execute the `MorphologicalAnalyzer` first, and the `LDA` second.

`LDA` creates the `TopicModel`, which tags all the articles' text. Finally the `OfflineAlgorithm` builds the SolR with all have been created.

Main method:

buildFTI(archives) –

- analyzes all the articles in the given archives with the MorphologicalAnalyzer.
- builds the hierarchy topics tree with the LDA.
- tags all the articles in the given archives with the TopicModel.
- builds the Full Text Index according to all the analyzed and tagged articles.
- builds the object SolR, that will use us to search articles.

SolR

This class uses the external library SolR, which keeps and searches in a data structure named Full Text Index.

SolR can add and remove articles from the Full Text Index.

SolR has the following plug-ins:

Search – the one that do the search. This class exists in SolR, we only need to fit it to our system, its algorithm works very fast.

Analyzer – the one that do morphological analysis and tags the topics to a given new article's text.

Main methods:

addText(article) –

- analyzes and tags the given article by using the Analyzer.
 - adds the analyzed and tagged article to the Full Text Index.
- @post: the analyzed and tagged article is added to the Full Text Index.

deleteText(index) – deletes an article by its given index.

@invariant: the given index must be ≥ 0 and \leq the number of the Articles in the Archives.

@pre: the requested Article is in the Full Text Index.

@post: the requested Article is no longer in the Full Text Index.

search(query) –

- gets a free text language text as query
- analyzes the given query with the QueryProcessor
- searches the analyzed query in the Full Text Index
- returns the search results

Analyzer

Analyzer is a SolR's plugin.

Analyzer implements the Command design pattern, this object has one major method that executes an algorithm.

Analyzer's role is to analyze and tag an article before it enters to Full Text Index.

Analyzer has a pointer to TopicModel. All the articles in the Full Text Index got the topics by this TopicModel.

Main method:

analyze(article) –

- analyzes the given article with the MorphologicalAnalyzer.
- tags the given article with the TopicModel.

@invariant: the given article only has metadata and text, i.e. it doesn't have morphological analysis and topics list.

ServerManager

ServerManager is the main object in the server.

ServerManager has:

- HTTP, through it the ServerManager communicates with the variant clients.
- a jar file that includes the LDA algorithm
- a pointer to the archives, that the SolR built by it
- StatisticManager that ServerManager uses it to insert and get the statistics.

Main methods:

insertQuery(query) –

- gets a result from SolR to the given query, by sending it to SolR.
- updates the StatisticMeneger
- returns the result

getSubject(index) –

- gets the wanted topic by calling the Archives' method getSubject(index) .
- if the topic is a Topic (i.e. a leaf in the tree) – returns a html page with summaries of the texts which deals with topic.
- if the topic is a TopicFather (i.e. not a leaf in the tree) – returns a html page with the topic's subtree.

@invariant: the given index must be ≥ 0 and \leq the number of the topics in the Archives.

@pre: the requested topic is in the Archive's topics collection.

getText(index) –

- gets the wanted article by calling the Archives' method getText(index) .
- updates this user selection in the StatisticManager
- returns a html page with article's text, metadata and the topics

@invariant: the given index must be ≥ 0 and \leq the number of the articles in the Archives.

@pre: the requested article is in the Archive's articles collection.

deleteText(index) –

- removes the wanted article from the Archive by calling the Archives' method remove(index).
- deletes the wanted article from the Full Text Index by calling the SolR's method deleteText(index).

@invariant: the given index must be ≥ 0 and \leq the number of the articles in the Archives.

@pre: the requested article is in the Archive's the SolR's articles collections.

@post: the requested article is no longer in the Archives and the SolR.

insertText(file,metadata) –

- makes a new article by calling the Archives' method add(text,metadata).
- enters the new article to the Full Text Index.

@post: the new article is in the Archives' and the SolR's articles collections.

updateAlgorithm(algorithm) – replaces the LDA algorithm's jar with the given one.

@invariant: the given jar file is proper and has an algorithm that implements the LDA's interface.

@post: the LDA's algorithm is the given one.

getStatistics() – returns the statistics from StatisticManager.

rebuildTree() –

- executesOfflineAlgorithm's algorithm in order to build a new SolR, to update the Archives and the TopicModel.
- replaces the old SolR with the new one by calling to the ServerManager's method updateSolR(newSolR).

@pre: the system has a proper jar with the LDA algorithm.

@post: the SolR is the new one.

updateSolR(newSolR) – replaces the old SolR with the given one.

@post: the SolR is the new one.

HTTP

HTTP class is responsible for the communication with the clients.

HTTP has the ServerManager.

HTTP gets requests from the web and transfers them to the ServerManager.

Main methods:

GET(): The Get method is used to getting the data from the server. Get method appends the parameters passed as query string to a URL, in the form of key- value pairs. for example, if a parameter is name = Williams, then this string will be appended in the URL. By default the method is Get.

POST(): The post method is used for sending data to the server. In post method the query string is appended along the request object, they do not get appended in the URL, so parameters transfer in hidden form.

4.3 Packages

Package	Classes in the Package	PackageDescription
Topics	TopicNode, Topic, TopicFather, WrodDistribution, TopicModel	Centralizes everything about the hierarchy topic tree. Includes the data object, and the topic model that tags articles in the hierarchy topic tree.
Articles	Article, MetaData, MetaDataPair, MorphologicalAnalysis, Archives	Includes the data objects that have information about the articles: the article, the metadata and the morphological analysis. It's also include the Archives, but not the hierarchy topics tree.
Server HTTP SolR	HTTP (and maybe more)	Centralizes the connection with the Clients
Offline Algorithm	SolR, Search, FullTextIndex, QueryProcessor, Analyzer OfflineAlgorithm, LDA, MorphologicalAnalyzer	Includes the SolR and his plugins Contains the Offline Algorithm that builds the SolR (SolR used to search) and every algorithm the that the Offline Algorithm uses.
Server Manager	ServerManager, StatisticManager	Connects the various parts of the server and responsible for the statistic

4.4 Unit Testing

ServerManager

getSubject

Test Number	Description	Expected result
1.	We will build the hierarchy topics tree so will we know about a specific topic node's index. We will ask to get this topic by its index.	The requested topic node's details.
2.	We will build the hierarchy topics tree so will we know about a specific non-exists topic node's index. We will ask to get this topic by its index.	We will get an error message.

ServerManager

getText

Test Number	Description	Expected result
1.	We will build the archives so will we know about a specific article's index. We will ask to get this article by its index.	The requested article's details.
2.	We will build the archives so will we know about a specific non-exists article's index. We will ask to get this article by its index.	We will get an error message.

ServerManager

deleteText

Test Number	Description	Expected result
1.	We will build the archives so will we know about a specific article's index. We will ask to delete this article by its index.	The article will be deleted from the archive and the Full Text Index. We will get a message that the text has been deleted.
2.	We will build the archives so will we know about a specific non-exists article's index. We will ask to delete this article by its index.	We will get an error message.

ServerManager

addText

Test Number	Description	Expected result
1.	We will ask to add a new text to the system.	A new article will be created and added to the Archives and the Full Text Index. We will get a message that the text has been added.

ServerManager

updateAlgorithm/rebuildTree

Learning Content Models for Semantic Search

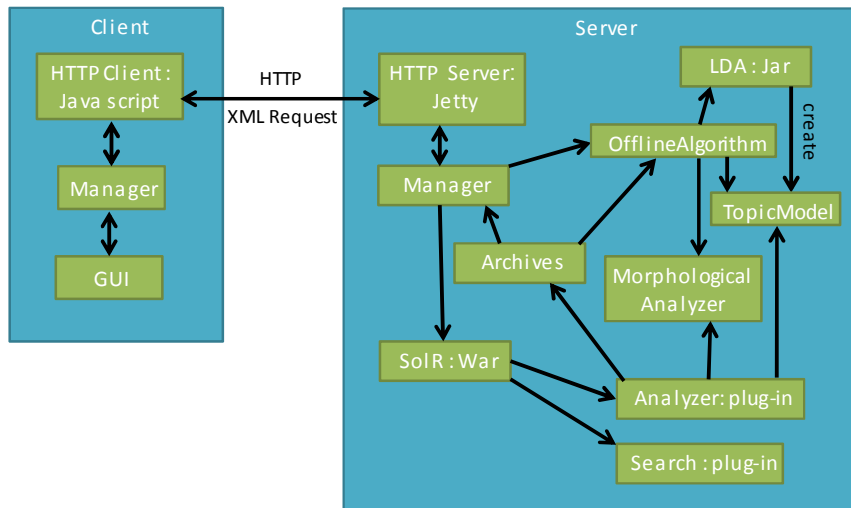
Test Number	Description	Expected result
1.	We will build a LDA algorithm that returns constant output and then we will rebuild the hierarchy topics tree. Next, we will change the LDA algorithm to another that returns other constant output and then we will rebuild the hierarchy topics tree.	At first we will get the first constant answer to each search request. After the chances will get a message that the algorithm has been changed. After the chances will get the second constant answer to each search request.

TopicModel

tagText

Test Number	Description	Expected result
1.	We will build the topicModel so it would hasa specific hierarchy topics tree. In that way we will know to each article, what are the topics it talks about.	The articles would be tagged by the topics that we think they talks about.

5 System Architecture



6 User Interface Draft

End User

Output: The main screen displays a box, where the user can enter his query. Next to the box there is a button "search".

Input: All the users can enter to the box a text as query and that way to search texts and topics at the repository. After typing the text, the user clicks on the "search" button, in order to enter his text as query.

Output: After the user entered a text as query, the system presents to the user a list of the relevant texts' summary. Each summary attached with the subjects/topics that the text deals with. In addition, the system shows to the user a topics subtree with all the topics that might interested the user, each topic with how many documents are relevant to the search.

At the head of the screen the system displays the search box and to buttons:

- "new search" – the user can enter a new query and start a new search.
- "search in the results" – the user can enter a query and search in the results.

At the bottom of the screen the system presents how many documents have been found.

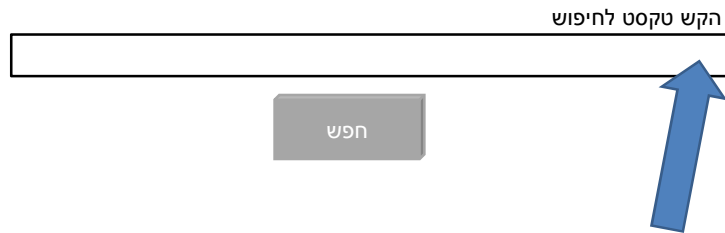
Input: After getting the results, the user can choose, by clicking on it, the topic that he thinks that he wants.

Output: After choosing a topic at the subtree, the system shows to the user the topic's subtree with all the topics that are more specific than the chosen topic, and the topic's fathers at the hierarchical topics tree. If the chosen topic has no subtree, i.e. it is a leaf - the system displays to the user a list of the relevant texts. Each text attached with the subjects/topics that the text deals with.

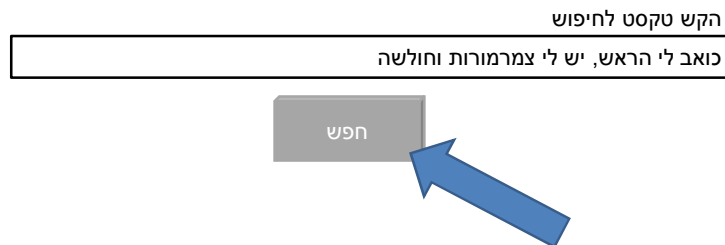
Input: The user can choose, by clicking on it, the text's summary that interested him.

Output: After choosing a text's summary, the system shows to the user the full text with its topics and metadata, next to them, displays links to articles, which have one or more identical metadata's criterion.

A Visual Example



A Visual Example



A Visual Example

תוצאות עבור החיפוש "כאב לי הראש, יש לי צמרמורות וחולשה":

חפש בתוצאות

חיפוש חדש

כאב ראש חזק ומתמשך שמצריך בירור

סרציתם לדעת על: אני בת 29, בריאה בדרך כלל, למעט ניתוחים אורתופדיים שעברתי ביד בעקבות שבר. בחודש האחרון התחילו להופיע אצלי כאבי ראש בעוצמה מאוד חזקה.

נושא: [כאבי ראש כרוניים](#), [ניתוחים אורתופדיים בגפיים](#)

כאבי ראש בילדים

התלונה על כאבי ראש שכיחה אצל ילדים והיא עלולה להופיע כבר בגיל שנתיים, ללא קשר עם הופעת חום או מחלה נלווית אחרת. כאב ראש מתמשך אצל ילדים...

נושא: [כאבים שונים אצל ילדים](#), [תופעות לוואי של תרופות לילדים](#), [כאבי ראש כרוניים](#)

נמצאו 102 מאמרים
עמוד 1 : מאמרים 1 עד 2
[ללכת לעמוד הבא](#) [ללכת לעמוד האחרון](#)

עץ נושאים:

- [רפואת ילדים \(20\)](#)
- [כאבים שונים אצל ילדים \(10\)](#)
- [מחלות נגיפיות בקרב ילדים \(7\)](#)
- [חום אצל ילדים \(3\)](#)
- [כאבים כרוניים \(12\)](#)
- [כאבים שונים אצל ילדים \(10\)](#)
- [כאבי ראש כרוניים \(1\)](#)
- [כאבי בטן חוזרים \(1\)](#)
- [מחלות נפש \(8\)](#)
- [סכיזופרניה \(4\)](#)
- [מחלות נפש בעקבות טראומה \(2\)](#)
- [פסיכואנליזה \(2\)](#)

A Visual Example

תוצאות עבור החיפוש "כאב לי הראש, יש לי צמרמורות וחולשה":

דברים שקשורים ל**כאבים שונים אצל ילדים**

חפש בתוצאות

חיפוש חדש

על דלקת עיניים אצל ילדים

דלקת עיניים אצל ילדים עלולה להופיע בלחמית, בקרנית או בעפעפיים. דלקת עיניים יכולה להיגרם כתוצאה מחיידק, וירוס או מאלרגיה. אבחון מוקדם של דלקת עיניים עשוי לפתור את הבעיות ביעילות

נושא: [כאבים שונים אצל ילדים](#), [רפואת עיניים](#)

כאבי ראש בילדים

התלונה על כאבי ראש שכיחה אצל ילדים והיא עלולה להופיע כבר בגיל שנתיים, ללא קשר עם הופעת חום או מחלה נלווית אחרת. כאב ראש מתמשך אצל ילדים...

נושא: [כאבים שונים אצל ילדים](#), [תופעות לוואי של תרופות לילדים](#), [כאבי ראש כרוניים](#)

נמצאו 41 מאמרים
עמוד 1 : מאמרים 1 עד 2
[ללכת לעמוד הבא](#) [ללכת לעמוד האחרון](#)

נושאים מרחיבים

- [רפואת ילדים \(20\)](#)
- [כאבים כרוניים \(12\)](#)

נושאים מצומצמים

- [כאבי ראש אצל ילדים \(4\)](#)
- [כאבי שיניים אצל ילדים \(3\)](#)
- [כאבים אצל ילדים בגיל הרך \(2\)](#)
- [כאבים לא ברורים אצל ילדים \(2\)](#)
- [כאבי עיניים אצל ילדים \(1\)](#)

A Visual Example

על דלקת עיניים אצל ילדים

מאת: ד"ר חיים סטולוביץ'

תאריך: 11.05.06

נושא: [כאבים שונים אצל ילדים](#), [רפואת עיניים](#)

דלקת עיניים אצל ילדים עלולה להופיע בלחמית, בקרנית או בעפעפיים. דלקת עיניים יכולה להיגרם כתוצאה מחיידק, וירוס או מאלרגיה. אבחון מוקדם של דלקת עיניים עשוי לפתור את הבעיות ביעילות

כאשר העין של התינוק או הילד מציקה, כואבת, נפוחה, דביקה ומפרישה הפרשות, סביר להניח כי הוא סובל מ"דלקת עיניים".

ישנם סוגים רבים של דלקות עיניים הנבדלות בגורמים להן, בסימפטומים שלהן, באופן הטיפול ובמשך זמן ההחלמה.

דלקת עיניים יכולה להיות ממוקמת בלחמית, בקרנית ובעפעפיים והגורם לה יכול להיות חיידק, וירוס או אלרגיה. ישנה חשיבות רבה לאבחון נכון של סוג הדלקת על ידי רופא עיניים ולטיפול מתאים, כדי למנוע סיבוכים ונזקים בלתי הפיכים.

ראה עוד מסמכים:

[מאת ד"ר חיים סטולוביץ' משנת 2006](#)

בנושא: [כאבים שונים אצל ילדים](#), [רפואת עיניים](#)

Context Manager

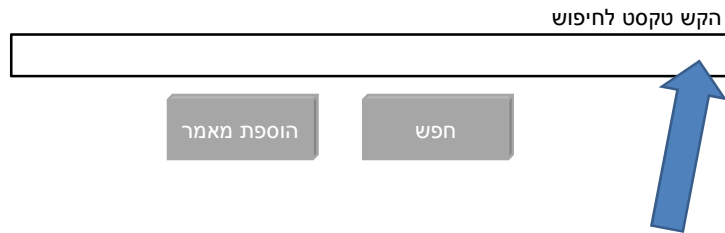
The context manager can use the same inputs and get the same output as the end user.

Output: When getting his query's results, in addition to what presented to the end user, a "delete" button is attached to each text's summary.

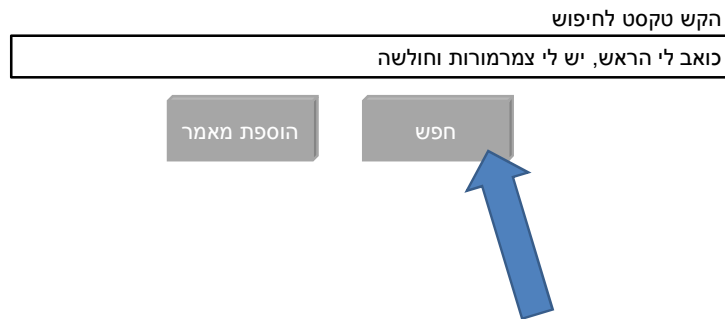
Input: After entering a text as query and getting the results, the context manager can choose, by clicking on its "delete" button, to delete a text.

Output: After choosing to delete a text, the system displays to the manager a message that the text has been deleted successfully. If the deletion failed, the system displays to the manager a message that the text can't be deleted.

A Visual Example



A Visual Example



A Visual Example

תוצאות עבור החיפוש "כואב לי הראש, יש לי צמרמורות וחולשה":

חפש בתוצאות

חיפוש חדש

מחק

ראש חזק ומתמשך שמצריך בירור

כל מה שרציתם לדעת על: אני בת 29, בריאה בדרך כלל, למעט ניתוחים אורתופדיים שעברתי ביד בעקבות שבר. בחודש האחרון התחילו להופיע אצלי כאבי ראש בעוצמה מאוד חזקה.

נושא: [כאבי ראש כרוניים](#), [ניתוחים אורתופדיים בגפיים](#)

[כאבי ראש בילדים](#)

התלונה על כאבי ראש שכיחה אצל ילדים והיא עלולה להופיע כבר בגיל שנתיים, ללא קשר עם הופעת חום או מחלה נלווית אחרת. כאב ראש מתמשך אצל ילדים...

נושא: [כאבים שונים אצל ילדים](#), [תופעות לוואי של תרופות לילדים](#), [כאבי ראש כרוניים](#)

נמצאו 102 מאמרים
עמוד 1 : מאמרים 1 עד 2
[ללכת לעמוד הבא](#) [ללכת לעמוד האחרון](#)

עץ נושאים:

- [רפואת ילדים \(20\)](#)
 - [כאבים שונים אצל ילדים \(10\)](#)
 - [מחלות נגיפיות בקרב ילדים \(7\)](#)
 - [חום אצל ילדים \(3\)](#)
 - [כאבים כרוניים \(12\)](#)
 - [כאבים שונים אצל ילדים \(10\)](#)
 - [כאבי ראש כרוניים \(1\)](#)
 - [כאבי בטן חוזרים \(1\)](#)
 - [מחלות נפש \(8\)](#)
 - [סכיזופרניה \(4\)](#)
 - [מחלות נפש בעקבות טראומה \(2\)](#)
 - [פסיכואנליזה \(2\)](#)

A Visual Example

תוצאות עבור החיפוש "כואב לי הראש, יש לי צמרמורות וחולשה":

חפש בתוצאות

חיפוש חדש

מחק

כאב ראש חזק ומתמשך שמצריך בירור

כל מה שרציתם לדעת על: אני בת 29, בריאה בדרך כלל, למעט ניתוחים אורתופדיים שעברתי ביד בעקבות שבר. בחודש האחרון התחילו להופיע אצלי כאבי ראש בעוצמה מאוד חזקה.

נושא: [כאבי ראש כרוניים](#), [ניתוחים אורתופדיים בגפיים](#)

[כאבי ראש בילדים](#)

התלונה על כאבי ראש שכיחה אצל ילדים והיא עלולה להופיע כבר בגיל שנתיים, ללא קשר עם הופעת חום או מחלה נלווית אחרת. כאב ראש מתמשך אצל ילדים...

נושא: [כאבים שונים אצל ילדים](#), [תופעות לוואי של תרופות לילדים](#), [כאבי ראש כרוניים](#)

נמצאו 102 מאמרים
עמוד 1 : מאמרים 1 עד 2
[ללכת לעמוד הבא](#) [ללכת לעמוד האחרון](#)

עץ נושאים:

- [רפואת ילדים \(20\)](#)
 - [כאבים שונים אצל ילדים \(10\)](#)
 - [מחלות נגיפיות בקרב ילדים \(7\)](#)
 - [חום אצל ילדים \(3\)](#)
 - [כאבים כרוניים \(12\)](#)
 - [כאבים שונים אצל ילדים \(10\)](#)
 - [כאבי ראש כרוניים \(1\)](#)
 - [כאבי בטן חוזרים \(1\)](#)
 - [מחלות נפש \(8\)](#)
 - [סכיזופרניה \(4\)](#)
 - [מחלות נפש בעקבות טראומה \(2\)](#)
 - [פסיכואנליזה \(2\)](#)

Output: At the main screen, in addition to the search zone, there is a "add new text" button.

Input: The manager can choose to add a new text to the repository by clicking the "add new text" button.

Output: After clicking the "add new text" button, the system displays to the manager a new window with a box, where the manager can enter the new text's path. Next to the box there is a button "add".

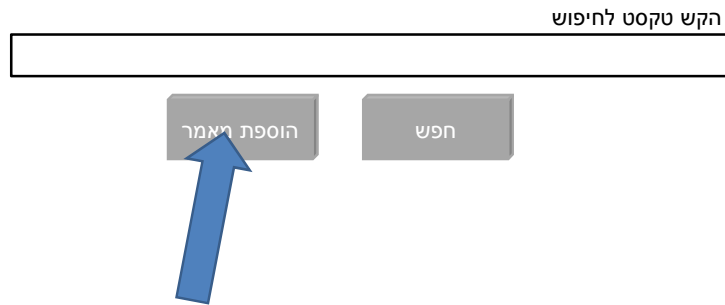
Input: The manager can enter into the box the new text's file path, and by clicking the "add" button, to add this text to the repository.

Output: After choosing to add a text, the system displays to the manager a new window with the metadata's criterions and next to each one a box, where the manager can enter the criterion's value. At the bottom of the window there is "ok" button.

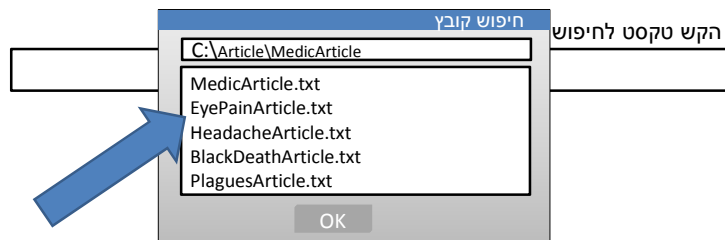
Input: The manager can enter into the box the criterion's value, and by clicking the "ok" button, to add the text with its metadata to the repository.

Output: After choosing to add the text's metadata, the system displays to the manager a message that the text has been added successfully. If the addition failed, the system displays to the manager a message that the text can't be added.

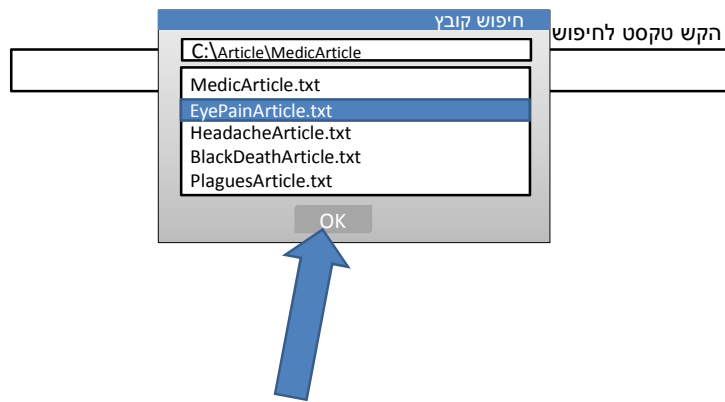
A Visual Example



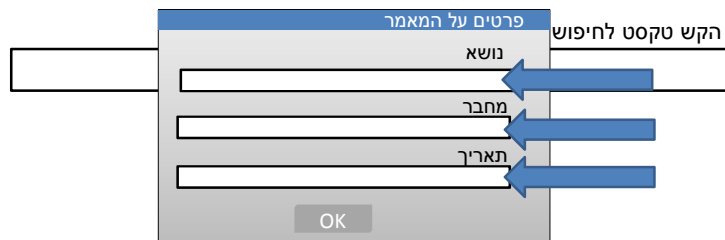
A Visual Example



A Visual Example



A Visual Example



A Visual Example

A visual example of a form. The form has a title bar with the text "פרטים על המאמר" (Details about the article). Below the title bar, there are four input fields with the following labels and values: "נושא" (Subject) with the value "כאבי עניים בראשית העת החדשה" (Eye pain in the beginning of the modern era), "מחבר" (Author) with the value "ד"ר כלשהוא" (Dr. Somebody), "תאריך" (Date) with the value "1/4/2/07", and an "OK" button. A blue arrow points to the "OK" button. To the right of the form, there is a label "הקש טקסט לחיפוש" (Click text to search).

A Visual Example

A visual example of a form. The form has a title bar with the text "הערה" (Note). Below the title bar, there is a single input field with the value "הוסף בהצלחה" (Added successfully) and an "OK" button. To the right of the form, there is a label "הקש טקסט ל" (Click text to). Below the form, there is a label "אמר" (Said).

System Manager (Administrator)

The system manager can use the same inputs and get the same output as the end user.

Output: At the main screen, in addition to the search zone, there are "produce statistics", "update the topic model" and "rearrange the hierarchy topics tree" buttons.

Input: The system manager can click the "produce statistics" button to produce statistics about the system. The statistics are:

- How many times a topic has been selected in searches.
- How many times a document has been selected in searches.
- How many times a document and a topic have been selected together in searches.

Output: The system creates a file that:

For each topic shows how many times it has been selected in searches.

For each document shows how many times it has been selected in searches, and how many times it has been selected in its specific topic.

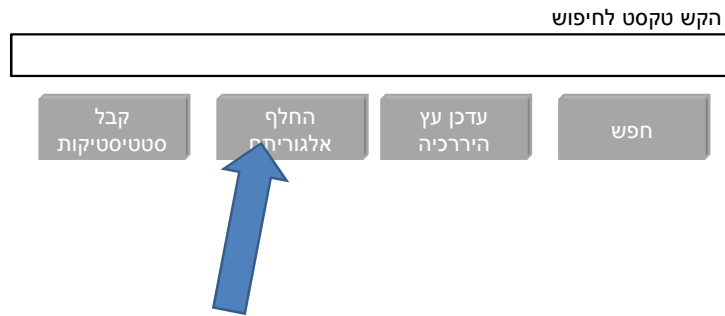
Input: The system manager can click the "update the topic model" button to enter the new LDA algorithm as jar file.

Output: After clicking the "update the topic model" button, the system displays to the manager a new window with a box, where the manager can enter the new LDA's jar file's path. Next to the box there is a button "update".

Input: The manager can enter to the box the jar file path, and by clicking the "update" button, to update the LDA algorithm, and as a consequence the topic model.

Output: After choosing to update the topic model, the system displays to the manager a message that the topic model has been updated successfully. If the update failed, the system displays to the manager a message that the topic model can't be updated.

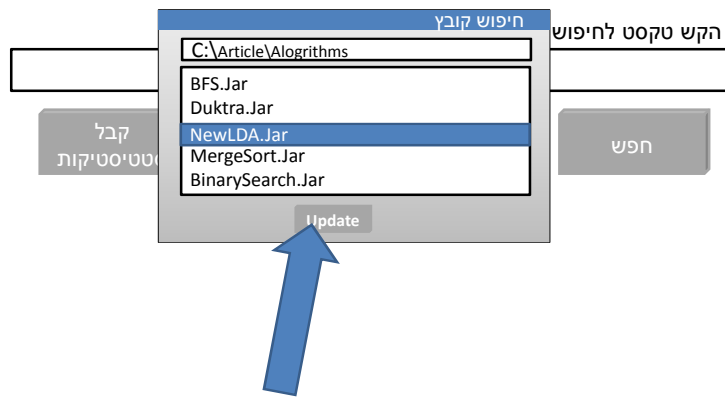
A Visual Example



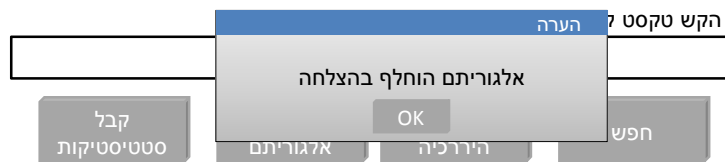
A Visual Example



A Visual Example



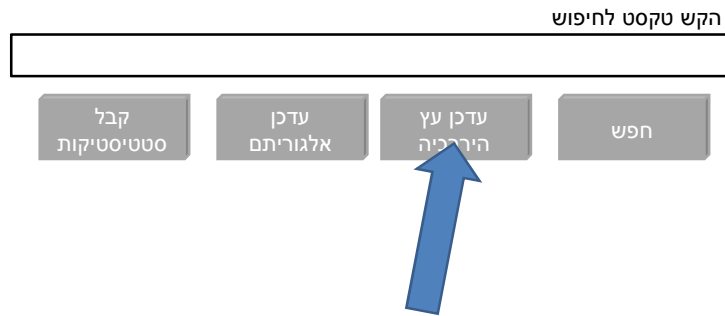
A Visual Example



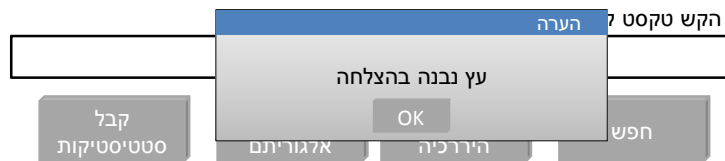
Input: The system manager can click the "rearrange the hierarchy topics tree" button to rebuild the hierarchy topics tree.

Output: After choosing to rearrange the hierarchy topics tree, the system displays to the manager a message that the hierarchy topics tree has been rearranged successfully. If the rearrange failed, the system displays to the manager a message that the hierarchy topics tree can't be rearranged.

A Visual Example



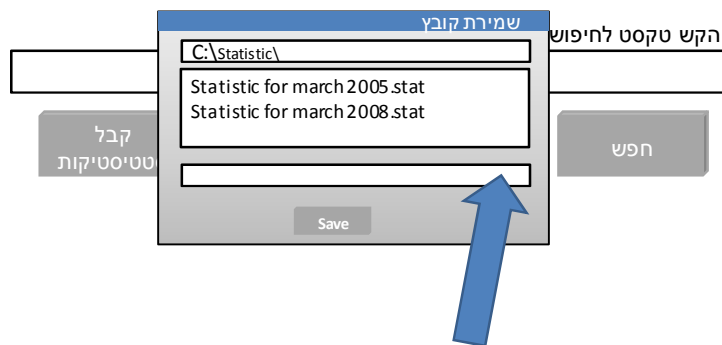
A Visual Example



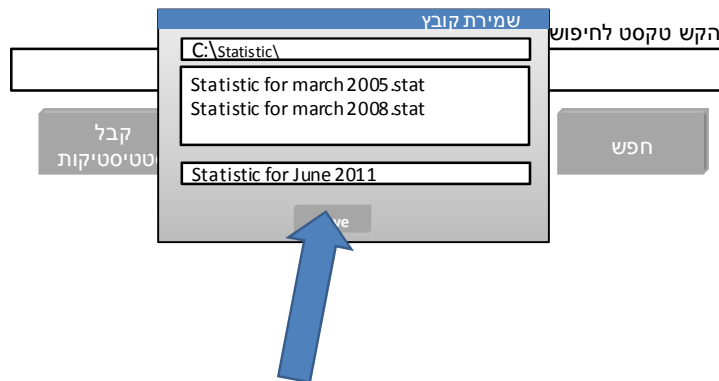
A Visual Example



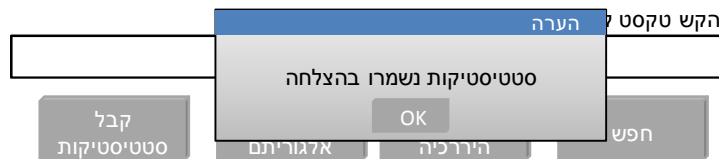
A Visual Example



A Visual Example



A Visual Example



7 Testing

| Test number | Non-functional constraint | Test |
|-------------|--|---|
| 1 | Every transaction must be processed in no more than 2 seconds. | A test that measures time and starts transactions at the system. The test will measure how many time passed from starting the transaction until getting an answer.
We will executes this check several times, and will always make sure that the passed time won't be more than 2 seconds. |
| 2 | 10,000 users can use the system simultaneously. | We will create 10,000 mockup end users, and we will run the 1 st test with each one of the mockup user. |
| 3 | The system can run on every operation system. | We will check our system on the operation systems Windows and Linux. |
| 4 | The system is Web service, but it doesn't depend on specific Web browser. | We will check our system on the Web browsers Firefox, Chrome and Explorer. |
| 5 | The system should support the text in English and in Hebrew. | We will enter Hebrew and English texts and queries. |
| 6 | The system should be available 100% of the time, depends on the availability of the hardware and the network. | We will check the system with extreme situations like will the system be activate while rebuilding a hierarchy topics tree. |
| 7 | <ul style="list-style-type: none"> At 95% cases the system can find the relevant data the user is looking for. The user should understand how to use the system in 10 minutes. The user can prepare the input in 10 seconds. The user can interpret the output in 2.5 minutes. | With beta-users. |
| 8 | All the SE Project constraints | With beta-users. |

8 Task List

| Task ID | Brief title | An estimate of the duration of the task (days) | An estimate of the starting date |
|----------|---|--|----------------------------------|
| 1 | Build Basic Client (End User) | 30 | 20/2/11 |
| 1.1 | Build client manager (only for end user) | 4 | 20/2/11 |
| 1.2 | Build server manager (only connection all other is mockups) | 4 | 24/2/11 |
| 1.3 | Build client GUI (only for end user) | 20 | 28/2/11 |
| 1.3.1 | Build the "enter query" GUI | 1 | 28/2/11 |
| 1.3.2 | Build the "result set (with paging)" GUI | 12 | 1/3/11 |
| 1.3.3 | Build the "hierarchical topics tree" GUI | 6 | 13/3/11 |
| 1.3.4 | Build the "a document" GUI | 1 | 19/3/11 |
| 1.4 | Test connection | 2 | 20/3/11 |
| 2 | Build Basic SolR | 15 | 22/3/11 |
| 2.1 | Build SolR (filled information mockup's information) | 3 | 22/3/11 |
| 2.2 | Make the manager work with SolR | 10 | 25/3/11 |
| 2.3 | Test the connection with SolR | 2 | 4/4/11 |
| 3 | First prototype of the Offline Algorithm | 69 | 6/4/11 |
| | Collect documents in one domain
Domains are: Halacha, Healthcare, Wikipedia (a subset) | 4 | 6/4/11 |
| | Test the documents | 1 | 10/4/11 |
| 3.1 | Build the Article Class, MetaData and MetaDataPair Class | 2 | 11/4/11 |
| 3.2 | Build the MorphologicalAnalysis Class | 5 | 13/4/11 |
| 3.3 | Build the MorphologicalAnalyzer class | 4 | 18/4/11 |
| 3.4 | Test the MorphologicalAnalyzer | 5 | 22/4/11 |
| 3.5 | Build the TopicNode, Topic, TopicFather, WordDistribution classes | 3 | 27/4/11 |
| 3.6 | Build the TopicModel class | 10 | 30/4/11 |
| 3.7 | Build the Archives class | 5 | 10/5/11 |
| 3.8 | Build the prototypeLDA class | 5 | 15/5/11 |
| 3.9 | Build the OfflineAlgorithm LDA class | 5 | 20/5/11 |
| 3.10 | Test the OfflineAlgorithm | 20 | 25/5/11 |
| 4 | Improve the LDA algorithm | 30 | 14/6/11 |
| 4.1 | Improve the LDA algorithm | 20 | 14/6/11 |
| 4.2 | Test the new LDA algorithm | 10 | 4/7/11 |
| 5 | Build All clients | 13 | 14/7/11 |
| 5.1 | Build in the ServerManager the part that work with Context Manager | 2 | 14/7/11 |
| 5.2 | Build the client's GUI (Context Manager) | 1 | 16/7/11 |
| 5.2.1 | Add the "delete" option to "result set (with paging)" GUI | 0.2 | 17/7/11 |
| 5.2.2 | Add the "add article" option to the "enter query" GUI | 0.8 | 17/7/11 |
| 5.3 | Build the ClientManager for Context Manager | 1 | 18/7/11 |
| 5.4 | Test the Context Manager client | 1 | 19/7/11 |
| 5.5 | Build in the ServerManager the part that work with System Manager | 2 | 20/7/11 |
| 5.6 | Build the client's GUI (System Manager) | 1 | 22/7/11 |
| 5.6.1 | Add the "change algorithm" option to the "enter query" GUI | 0.4 | 23/7/11 |
| 5.6.2 | Add the "update the tree" option to the "enter query" GUI | 0.2 | 23/7/11 |
| 5.6.3 | Add the "get statistics" option to the "enter query" GUI | 0.4 | 23/7/11 |

| | | | |
|-----|--|-----------|----------------|
| | GUI | | |
| 5.7 | Build the ClientManager for System Manager | 1 | 24/7/11 |
| 5.8 | Test the System Manager client | 1 | 25/7/11 |
| 5.9 | Perform usability tests on GUI | 3 | 26/7/11 |
| 6 | Finish the project | 15 | 29/7/11 |
| 6.1 | Collect documents in the 2 other domains
Domains are: Halacha, Healthcare, Wikipedia (a subset) | 8 | 29/7/11 |
| 6.2 | Test the documents | 2 | 31/7/11 |
| 6.3 | Black box tests | 5 | 2/8/11 |

9 Appendix

The Gibbs Sampling algorithm. Alfa=50/T, Beta=0.01

$$P(z_i = j | \mathbf{z}_{-i}, w_i, d_i, \cdot) \propto \frac{C_{w_i j}^{WT} + \beta}{\sum_{w=1}^W C_{w j}^{WT} + W\beta} \frac{C_{d_i j}^{DT} + \alpha}{\sum_{t=1}^T C_{d_i t}^{DT} + T\alpha}$$

W – The number of word tokens.

D – The number of documents.

T – The number of topics.

C^{WT} and C^{DT} are matrices of counts with dimensions W x T and D x T respectively; $C_{w j}^{WT}$

contains the number of times word w is assigned to topic j, not including the current instance i and the $C_{d j}^{DT}$ contains the number of times topic j is assigned to some word token in document d, not including the current instance i.

The left part of the equation is the probability of word w under topic j whereas the right part is the probability that topic j has under the current topic distribution for document d. Words are assigned to topics depending on how likely the word for a topic, as well as how dominant a topic is in a document.

In order to get the topics and to tag each document with its relevant topics, we will use the Gibbs sampling algorithm. This algorithm starts by assigning each word token to a random topic in [1...T] (where T is the number of topics). For each word token, the count matrices C^{WT} and C^{DT} are first decremented by one for the entries that correspond to the current topic assignment. Then, a new topic is sampled from the distribution in the equation above and the count matrices C^{WT} and C^{DT} are incremented with the new topic assignment. Each Gibbs sample consists of the set of topic assignments to all N word tokens in the corpus, achieved by a single pass through all documents. To get a representative set of samples from this distribution, a number of Gibbs samples are saved at regularly spaced intervals, to prevent correlations between samples.

Exchangeability of topics. There is no priori ordering on the topics that will make the topics identifiable between or even within runs of the algorithm. Topic j in one Gibbs sample is theoretically not constrained to be similar to topic j in another sample (i.e. samples spaced apart that started with the same random assignments or samples from different random assignments).

Determining the Number of Topics. The idea is to estimate the posterior probability of the model while integrating over all possible parameter settings (i.e. all ways to assign words to topics). The number of topics is then based on the model that leads to the highest posterior probability. Another approach is to choose the number of topics that lead to best generalization performance to new tasks. Recently, researchers have used methods from non-parametric Bayesian statistics to define models that automatically select the appropriate number of topics.