Text Mining/IR Techniques for Question Answering System

Project Report, CMSC 698

Sandipan Dey

Graduate Student, UMBC CSEE

Intern, Siemens Corporate Research

May 25, 2011

Abstract

A lot of works have already been done on the application of pure statistical techniques in open-domain question-answering systems [1, 2, 3, 4]. In this paper, we restrict our domain to a set of multiple-choice-type questions for USC DDS Student Board Preparation Sessions (practice mock board questions in Pharmacology). Our goal is to build a question-answering system, by applying a set of searching/text-mining techniques to find answers to the multiple choice questions and compare the performance of the system in terms of precision/recall. Different distance measures and ranking methods are used (and their performances are compared) to match the answer-choices of a multiple-choice question with the text retrieved from the document corpus by searching with the question-concept.

Keywords

Question Answering, Tf-Idf, Jaccard, Ontology, PMI, MSR

Introduction

A multiple-choice question-answering system is developed and the performance of the performance of the system is measured in terms of correctness of the answer option chosen by the system. The schematic diagram of the system is shown in figure 1. As question repository, we use USC DDS Student Board

Preparation Sessions (practice mock board questions in Pharmacology), where each question has multiple options along with the correct answer choice. As shown in the figure 1, there are 3 main components of the system, namely, (1)Question Parsing (2)Answer Searching (3) Ranking the answer-choices.

First, the question asked to the system is parsed and the "*concept*" asked-for is identified. Then the concept is searched in different corpus, e.g. NDF, Wikipedia, different NCBI databases (e.g., PubMed, MeSH etc.) are used as knowledge-bases. The answer-choices corresponding to the question are then compared with the retrieved support-text (we shall refer to the text retrieved from the corpus by searching with question-concept as "*support-text*", i.e., the text supporting the concept) and ranked on the basis of tf-idf / PMI / semantic relatedness scores. Different distance measures (e.g., edit distance, longest common substring etc.) are used to find matches ("*evidences*") while computing the scores for each of the answer-choices. The answer-choice with the highest score is returned as the winner, the potential correct answer predicted by the system. A rigorous set of experiments are run and the performance of the system was compared using (1) Different corpus (2) Different distance measures (3) Different ranking methods.

The Approach

The following describes the basic steps the system takes to answer a multiple-choice question:

Steps the system performs to answer a question

As described in the algorithm 1 and in the figures [1, 4], there are the following gross steps performed by the system to answer a question asked:

- Question parsing and concept retrieval.
- Searching the concept from the corpus and retrieve relevant information as text.
- Matching the answer-choices with the retieved text.
- Ranking the answer-choices.
- Predicting the answer of the question.
- Finding the precision of the system by comparing the system-predicted answer with the correct answer.

Question Parsing

This is the first step that is performed by the system.

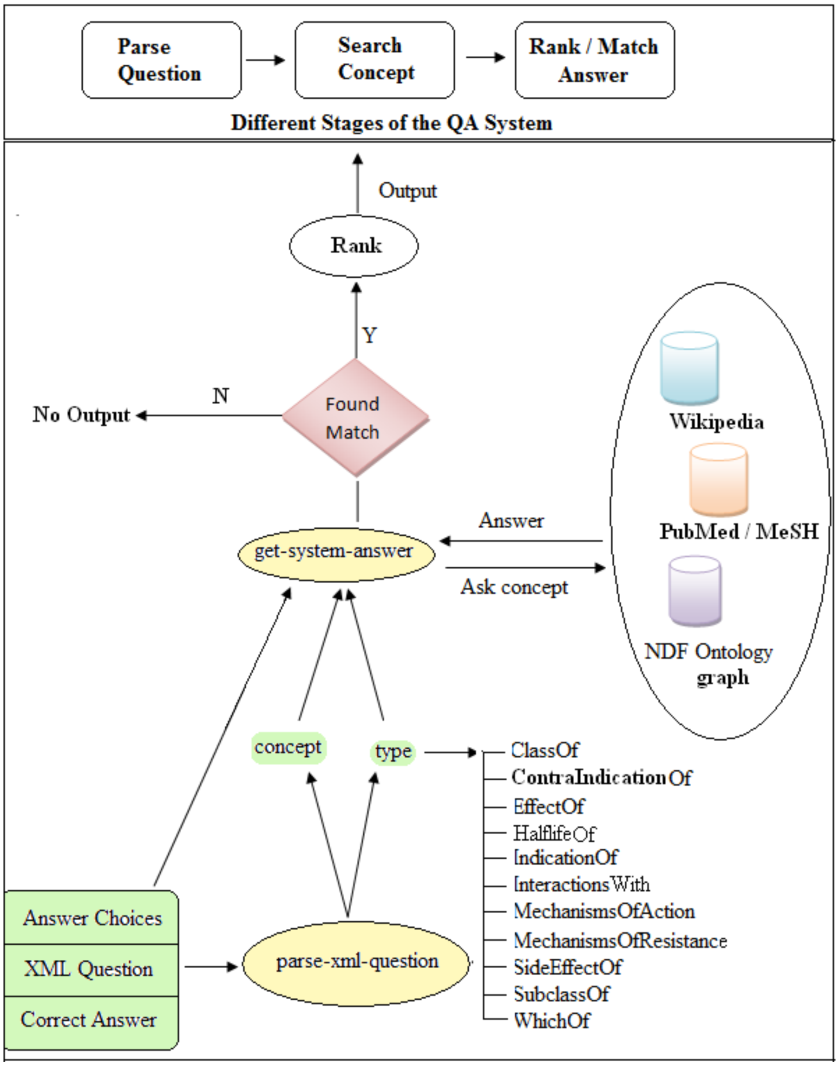


Figure 1: System Architecture

Question repository

Currently a set of (594) multiple-choice-type questions for USC DDS Student Board Preparation Sessions (practice mock board questions in Pharmacology) are used as question repository.

Format of a question

Each question has 4/5 answer choices and also contain the correct answer-choice along with it.

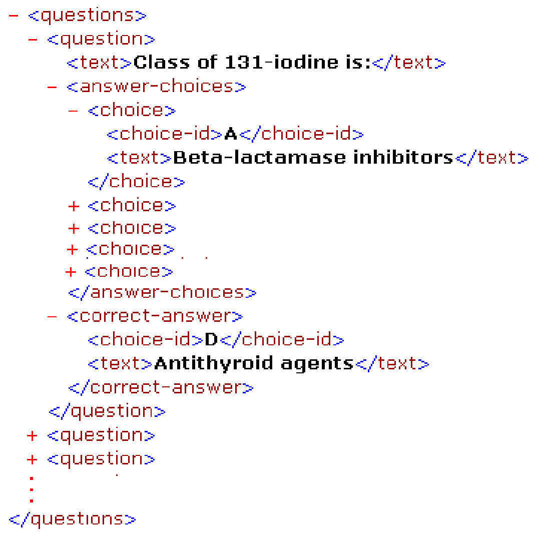
Example question:

Class of 131-iodine is:

(A) Beta-lactamase inhibitors
(B) Beta blockers
(C) Antivirals
(D) Antithyroid agents
(E) Antiarrhythmic agents

The correct answer is: (E) Antithyroid agents.

The question repository (expected to grow) is kept in a structured XML format to facilitate retrieval, as shown in the following figure . Utilities are written to convert an html questions file (with a specific format) to an XML file.



```
- <questions>
  - <question>
      <text>Class of 131-iodine is:</text>
    - <answer-choices>
      - <choice>
          <choice-id>A</choice-id>
          <text>Beta-lactamase inhibitors</text>
        </choice>
      + <choice>
      + <choice>
      + <choice>
      + <choice>
      </answer-choices>
    - <correct-answer>
        <choice-id>D</choice-id>
        <text>Antithyroid agents</text>
      </correct-answer>
    </question>
  + <question>
  + <question>
  :
</questions>
```

Figure 2: XML Question Repository

Concept retrieval

First the concept asked for in the question is retrieved from the question. The question parsing can easily be done using simple regular expressions, since right now there are a fixed number (namely 10) of question types (as shown in the figure 4) with fixed formats in the repository used. For instance, the "*concept*" retrieved from the example question "Class of 131-iodine is:" will be "*131-iodine*".

Searching the concept

The concept retrieved in the question-parsing stage is then searched in the (local) knowledge-base (corpus). The information relevant to the question-concept are retrieved as text.

Corpus/knowledgebases used

Different corpus used to search answer to a question:

1. Nation Drug File (NDF) Ontology: The RDF owl file containing drug concepts is loaded as a graph, with each concept as a node in the graph. Two given nodes in the graph are connected by an edge iff the corresponding RDF elements have a relation in between them.

2. For external corpus, the system can operate on couple of modes:

    - Online searching mode: concepts (present in the questions) are searched dynamically on-the-fly using the web client
    - Offline searching mode: a 2-step process: (1) Question-concepts are first downloaded and saved locally in files that have same names as the concepts (one-time process). (2) Later these locally-saved corpus is indexed to search the question-concepts (assumption in the searching phase: all concepts are present in the local corpus).

    External corpus used:

    - Webpages crawled using Wikipedia.
    - Entrez *esearch* interface is used to query the concept in an NCBI database and get a list of DB record indices (ids) matching the concept.
      Example of the databases searched are PubMed and MeSH. Example query to the esearch interface looks like the following: *http://www.ncbi.nlm.nih.gov/entrez/eutils/ esearch.fcgi?db=PubMed & term="aspirin"&retmode=html.*
      A subsequent query to the *efetch* interface with the record ids actually brings corresponding text. Example query to Entrez efetch interface looks like the following: *http://www.ncbi.nlm.nih.gov/entrez/ eutils/efetch.fcgi?id=21510749,21272686,18426828&db=PubMed*

Save the text retrieved locally in a file named “aspirin” in the corresponding corpus directory (pubcorpus).

- Web pages crawled using Google/Yahoo/Bing using their AJAX search interface.

For each corpus, we have a client that retrieves the relevant information about a “concept” from the corpus, when asked for.

Retrieving the result of search

- NDF Graph Node: All reachable nodes (by some path, using BFS) from the concept node are considered “relevant” and the corresponding text descriptions (expanded node) are returned as the result of search.
- Wikipedia/PubMed/Yahoo/Google/Bing: Search the corresponding concept file in the corresponding directory (offline mode) or search online using the search-engine client and crawl the webpages returned.

Matching

First, each of the answer-choices in a question is split into a series of “*term*”s (split-characters used: $\{, -.()[]$” $<>:=\}$ along with space). Then a stop words’ list is used to get rid of the insignificant terms. The stop words’ list is built by preliminary examination on data, by finding very frequently-occurring unimportant words.

Stop words’ list used:
{“a”, “an”, “the”, “in”, “at”, “to”, “for”, “with”, “of”, “up”, “by”, “may”, “can”, “shall”, “will”, “or”, “and”, “is”, “are”, “system”, “drug”, “drugs”, “disease”, “effect”, “effects”, “loop”, “agent”, “agents”, “against”, “type”, “factor”, “factors” }.

Next, the support-text retrieved from the corpus is also split into a series of “*word*”s. Finally, one of the following match-algorithms (exact or approximate match technique) is used to “*match*” every answer-choice with the “*support-text*” retrieved from the corpus (by matching “*term*”s from an answer-choice with “*word*”s from the “*support-text*”).

Similarity/distance measures used

The following similarity measures are used to match an answer-choice with the support-text and rank the answer-choice.

Keyword-Based match

In this case the terms are considered to be keywords and an exact match is tried to be obtained inside the concept text retrieved. A match occurs if a given term is present as a substring of the concept text.

Edit-Distance match

This is an approximate match technique. A match occurs if a given term has less normalized-edit-distance than a threshold (e.g., $MAX\_EDIT\_MATCH\_FRAC$ $= 0.2 \Rightarrow$ the term has at most 20% difference) from a word ($\exists$ at least one such word) in the support-text, where the normalized-edit-distance is defined as follows:

$$\text{normalized-edit-distance(term, word)} = \frac{\text{edit-distance(term, word)}}{\text{term-length}}$$

where the edit-distance (Levenshtein distance) $d(m,n)$ between a $term[1 \ldots m]$ and a $word[1 \ldots n]$ is given by the following well-known dynamic programming

$$d(i,j) = \left\{ \begin{array}{cc} d(i-1,j-1) & \text{, term[i]=word[j]} \\ min \left\{ \begin{array}{c} d(i-1,j)+1 \\ d(i,j-1)+1 \\ d(i-1,j-1)+1 \end{array} \right\} & \text{, otherwise} \end{array} \right\}$$

Longest Common Substring (LCSubstr) match

This is another approximate match technique.

A match occurs if a given term has an LCSubtr with the support-text that has more normalized-LCS-length than a given threshold with a word in the support-text (e.g., $MIN\_LCS\_MATCH\_FRAC = 0.8 \Rightarrow$ the term has at least 80% common with at least one word in the support-text), where the normalized-LCS-length is defined as follows (the longest common suffix between all possible prefixes):

$$\text{normalized-LCS-length(term, word)} = \frac{\text{LCS-length(term, word)}}{\text{term-length}}$$

where $LCS - length$ is defined by the length of the largest common substring ($LCSubStr$) in between the $term[1 \ldots m]$ and $word[1 \ldots n]$, with LCSubStr being defined by the following dynamic programming

$$LCSubstr(term, word) = \max_{1 \le i \le m, 1 \le j \le n} LCSuff(term_{1..i}, word_{1..j})$$

where the longest common suffix LCSuff is defined by the dynamic programming

$$LCSuff(term_{1..i}, word_{1..j}) = \begin{cases} LCSuff(term_{1..i-1}, word_{1..j-1}) + 1 & \text{if } term[i] = word[j] \\ 0 & \text{otherwise} \end{cases}$$

Jaccard-Distance match

This is another approximate match technique. A match occurs if a given term has less normalized-Jaccard-distance than a threshold (e.g., $MAX\_JACCARD$ $\_MATCH\_FRAC = 0.1 \Rightarrow$ the term has at most 10% difference) from a word ($\exists$

at least one such word) in the support-text, where the normalized-edit-distance is defined as follows:

$$\text{normalized-J-distance(term, word)} = \frac{\text{J-distance(term, word)}}{\text{term-length}}$$

where the J-distance (Jaccard-Distance) between a term and a word is defined as follows:

$$Jaccard - Distance(term, word) = 1 - \frac{|S_{term} \bigcap S_{word}|}{|S_{term} \bigcup S_{word}|}$$

where S_{term} = set of the alphabets in the term and S_{word} = set of the alphabets in the word.

Intuition behind the distance measures used

As can be seen from above, the keyword match tries to find an exact match (hence the strongest match) of the term inside the text, where LCS-match tries to find the largest common match (with contiguous characters) in between the term and a word in the text, edit-distance match find the match (maintaining the order, but may not be contiguous characters) and Jaccard-match only considers the occurrence of characters (without maintaining the order, hence the weakest match).

If the concept asked in the question is not found or none of the answer-choices match with retrieved support-text from the underlying knowledge-base then the system will

1. choose not to answer (random-answering disabled mode)

2. will select a random answer from the answer-choices (random-answering enabled mode).

Ranking

For each of the answer-choices, a score is computed that enables the system to choose the most probable (potential) answer-choice (corresponding to the highest score). The following techniques are used to compute the score.

Tf-Idf score

- Tf(term) = Number of times the "*term*" is found in the "*support-text*" (term frequency).

- Df(term) = Number of documents in which the term is found in the entire corpus (document frequency, e.g., for NDF Ontology Client, each node in the NDF graph is considered to be a document)

- Idf(term) $= \frac{1}{1+Df(term)}$ (inverse document frequency, finds how rare a term is in a document corpus)
- $Score(term) = Tf(term) \times Idf(term)$.

Whether a term is "found" or not is to be guided by the match-algorithm as described in the earlier section.

Issues in computing Idf

There exist performance issues in computing the Idf of a term both for online (Idf needs to search each document in the corpus for the presence of the term, hence time consuming) and offline (number of terms are very huge, hence problem in loading into memory) computation. Here, the Idf for all the terms in the answer-choices corresponding to a given question are computed at the same time on-the-fly.

PMI/Semantic Similarity score

The pointwise mutual information (PMI) can be used to compute semantic relatedness between two concepts. There are several variations for the pointwise mutual information (PMI) score computation [5]. We used PMI to find the semantic relatedness in between the question "*concept*" & the answer-choices, and used it for scoring/ranking the answer-choices.

PMI score is computed in between a "*term*" (in answer-choice) and the "*concept*" (from the question) in the following manner:

$$Score(term) = \frac{P(term \ \& \ concept)}{P(term)} \text{ (from [5])}$$

where

$$P(term) = \frac{\text{\# documents where term occurs at least once}}{\text{total \# documents in the corpus}}$$
$$P(term \ \& \ concept) = \frac{\text{\# documents where term \& concept occur together}}{\text{total \# documents in the corpus}}$$

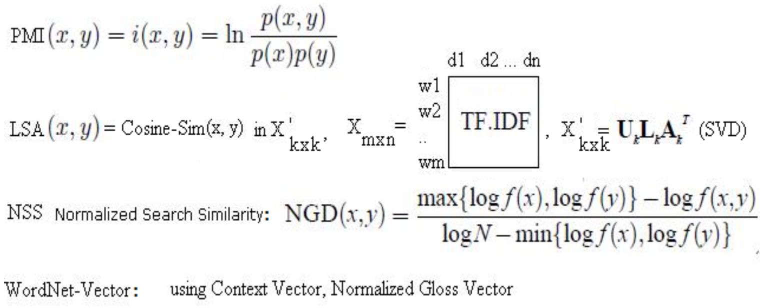
Alternatively, PMI between a "*term*" (in answer-choice) and a "*word*" (from the "*support-text*") can be defined as:

$$Score(term) = \max_{\text{word} \in \text{support-text}} \frac{P(term \ \& \ word)}{P(term)}$$

where

$$P(term) = \frac{\text{\# documents where term occurs at least once}}{\text{total \# documents in the corpus}}$$
$$P(term \ \& \ word) = \frac{\text{\# documents where term \& word occur together}}{\text{total \# documents in the corpus}}$$

Some of the popular semantic relatedness measures that are used by the Renseller MSR system are shown in the figure 3.



$$\mathrm{PMI}(x,y) = i(x,y) = \ln \frac{p(x,y)}{p(x)p(y)}$$

$$\mathrm{LSA}(x,y) = \text{Cosine-Sim(x, y)} \text{ in } \mathrm{X}'_{\mathrm{kxk}}, \quad \mathrm{X}_{\mathrm{mxn}} = \begin{array}{c|c|} & \mathrm{d1}\ \ \mathrm{d2} \dots \mathrm{dn} \\ \mathrm{w1} & \\ \mathrm{w2} & \mathrm{TF.IDF} \\ .. & \\ \mathrm{wm} & \end{array}, \quad \mathrm{X}'_{\mathrm{kxk}} = \mathbf{U}_k \mathbf{L}_k \mathbf{A}_k^T \text{ (SVD)}$$

$$\text{NSS Normalized Search Similarity: } \mathrm{NGD}(x,y) = \frac{\max\{\log f(x), \log f(y)\} - \log f(x,y)}{\log N - \min\{\log f(x), \log f(y)\}},$$

WordNet-Vector: using Context Vector, Normalized Gloss Vector

Figure 3: Semantic Relatedness measures

Using DISCO and MSR

DISCO (extracting Distributionally related words using CO-occurrences: *http://www.linguatools.de/disco/disco_en.html*) class (along with the en-BNC-20080721 corpus) is also tried to compute the 1st order and 2nd order similarity between terms and words.

Also, the Renseller MSR (Measures of Semantic Relatedness) demo Server(*http://cwl-projects.cogsci.rpi.edu/msr/*) APIs were also used to find semantic similarity between terms and words (e.g., *http://cwl-projects.cogsci.rpi.edu/cgi-bin/msr/msr.cgi?msr=NSS-Gwikipedia&terms= [('beta','131-iodine'),('lactamase','131-iodine'), ('inhibitors','131-iodine')]*).

Ranking Strategy

There are couple of different ranking strategies:

- The answer-choice that has the term that gets the highest score is chosen as the predicted answer. In other words,

  $$\text{the score of an answer-choice} = \max_{\text{term} \in \text{answer-choice}} score(\text{term}),$$

  i.e.,

  $$\text{predicted-answer} = \underset{\text{term} \in \text{some answer-choice}}{argmax} score(\text{term})$$

- Alternatively, the answer-choice that has the highest normalized sum of score of its terms is chosen as the predicted answer. In other words,

  $$\text{the score of an answer-choice} = \sum_{\text{term} \in \text{answer-choice}} \frac{score(\text{term})}{\text{length}(\text{term})}.$$

The score computed is also used as *confidence threshold* when running experiments, in the sense that system chooses to not answer the questions (not confident enough) having the most potential answer-choice score below the confidence threshold. The algorithm 1 describes the consolidated steps the system takes to choose the multiple choice questions.

Algorithm 1 AnswerQuestion

```
Download external corpus (e.g., Wikipedia, Entrez etc.) and save them
locally, indexing by concept names (one-time job).
For each corpus, load the corpus (e.g., for NDF, load NDF as graph) into
memory (done once at the start of each run).
for each question in repository do
    Parse the question to retrieve the concept asked.
    Search the corpus using the concept name and retrieve the text corresponding to the concept.
    For each of the alternative answer-choices, break the answer-choice into series of terms.
    Eliminate the stops words from the set of terms obtained.
    Break the support-text obtained into a series of words.
    Match the terms (using various distance measures, e.g., Edit Distance, Jaccard etc.) with the words from the retrieved text.
    Rank them using some ranking algorithm (e.g. Tf-Idf or PMI).
    Choose the answer-choice that has a term that gets the highest score (figure 4).
end for
```

Figure (4) shows how the system works when using NDF Ontology as corpus and Tf-Idf as ranking algorithm.

System Performance Measures

Precision and Recall are used to find the performance of the system in terms of the correctness of the predicted answer.

$$Precision = \frac{\text{\# Questions System Answered Correctly}}{\text{\# Questions System Answered}}$$

$$Recall = \frac{\text{\# Questions System Answered Correctly}}{\text{\# Total Questions}}$$

$$F - Measure = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Developing the Question-Answering System

Technology

Spring MVC framework / JSTL is used to along with Tomcat server to host the web interfaces for the question answering system. The system has 3 different views: (1) askxml: asking question/retrieving answer from the system (2)

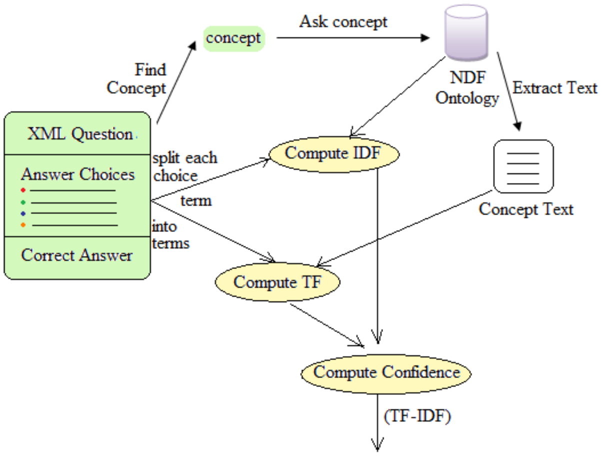


Figure 4: How System works for NDF Ontology Corpus and with Tf-Idf scoring

askxmlstat: batch mode (3) instant: for instant search. Initially the system was developed using LISP / Sparql queries but later we moved to java for the data driven track.

Modes

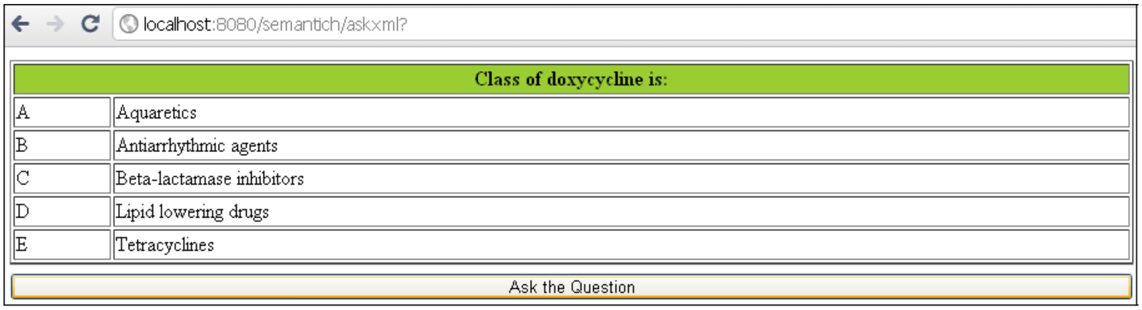
The system can primarily be run in couple of different modes: (1) Question-Anwering Mode (2) Batch Mode.

Question-Answering Mode

The interface ***http://pccs000222ws:8080/semantich/askxml*** is the http interface hosted on Tomcat to serve the user's request for the question-answering mode of the system. A question is randomly selected from the repository and displayed to the user.

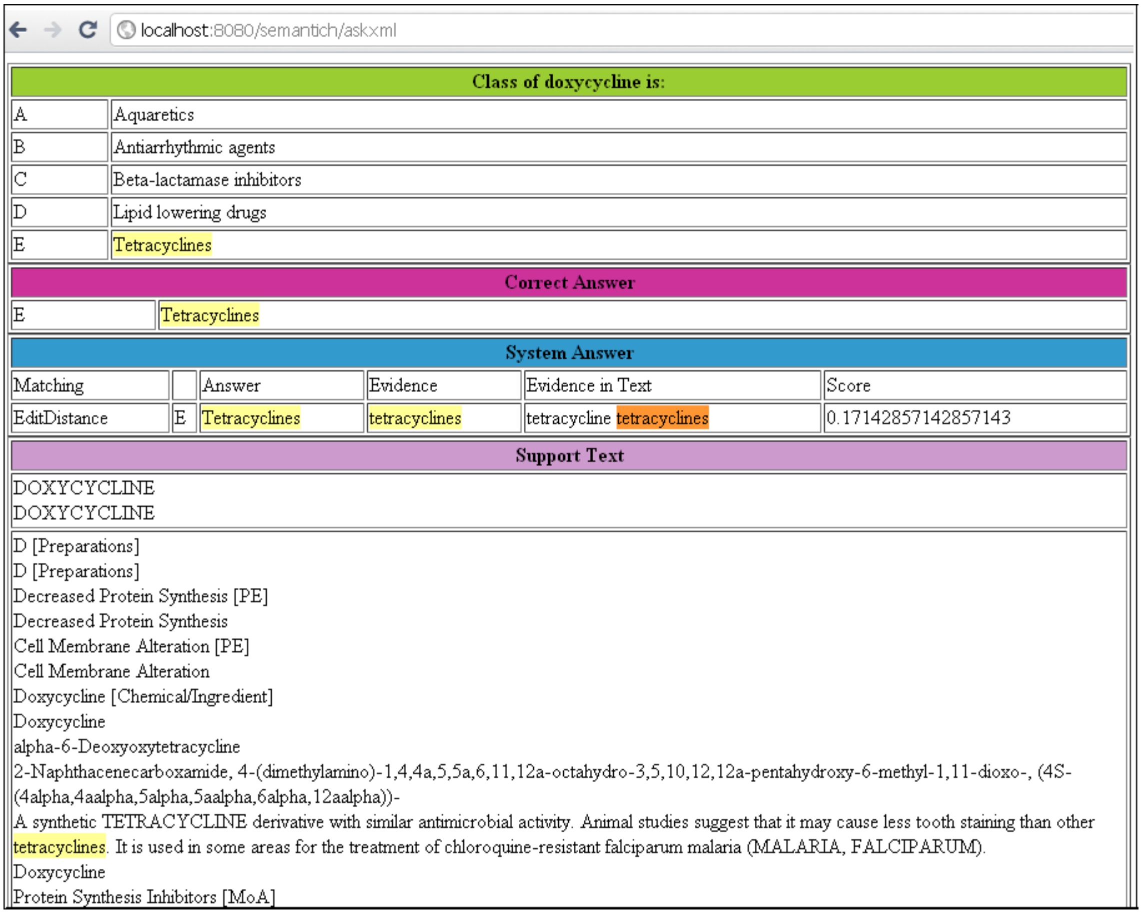
When the user clicks "Ask the Question", the system predicts the answer to the question by picking the answer-choice corresponding to the highest score as described earlier and shows the answer back to the user (see figure 6). User can compare the system-predicted answer with the correct answer to the question. The "*confidence*" of the system in answering the question (in terms of tf-idf / PMI score) is shown to the user.

As shown in the figures 5 & 6, using NDF Ontology Client, the answer texts retrieved from the RDF nodes are shown to the user after ordering them according to the increasing distance from the concept node in the graph.



Class of doxycycline is:	
A	Aquaretics
B	Antiarrhythmic agents
C	Beta-lactamase inhibitors
D	Lipid lowering drugs
E	Tetracyclines

Figure 5: Question-Answering Interface: Display a question randomly chosen & prompt user to ask the question to the system



Class of doxycycline is:	
A	Aquaretics
B	Antiarrhythmic agents
C	Beta-lactamase inhibitors
D	Lipid lowering drugs
E	Tetracyclines

Correct Answer	
E	Tetracyclines

System Answer

Matching		Answer	Evidence	Evidence in Text	Score
EditDistance	E	Tetracyclines	tetracyclines	tetracycline tetracyclines	0.17142857142857143

Support Text

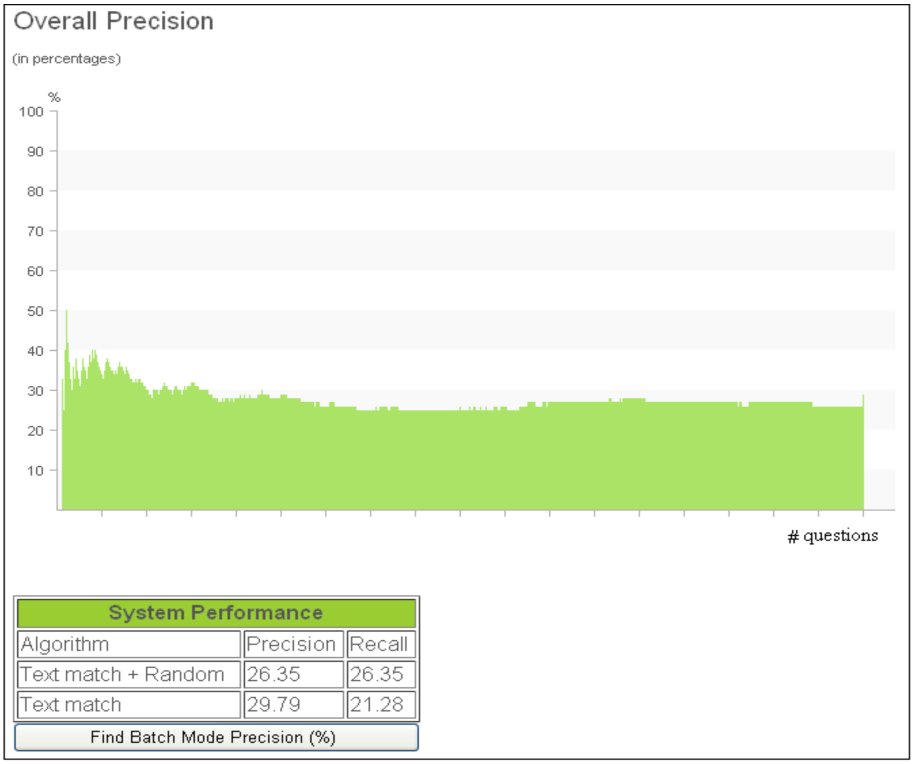
DOXYCYCLINE
DOXYCYCLINE

D [Preparations]
D [Preparations]
Decreased Protein Synthesis [PE]
Decreased Protein Synthesis
Cell Membrane Alteration [PE]
Cell Membrane Alteration
Doxycycline [Chemical/Ingredient]
Doxycycline
alpha-6-Deoxyoxytetracycline
2-Naphthacenecarboxamide, 4-(dimethylamino)-1,4,4a,5,5a,6,11,12a-octahydro-3,5,10,12,12a-pentahydroxy-6-methyl-1,11-dioxo-, (4S-(4alpha,4aalpha,5alpha,5aalpha,6alpha,12aalpha))-
A synthetic TETRACYCLINE derivative with similar antimicrobial activity. Animal studies suggest that it may cause less tooth staining than other tetracyclines. It is used in some areas for the treatment of chloroquine-resistant falciparum malaria (MALARIA, FALCIPARUM).
Doxycycline
Protein Synthesis Inhibitors [MoA]

Figure 6: Question-Answering Interface: Display the system-predicted answer to the question along with the evidence text and the confidence (score)

Batch Mode

In this mode, all the questions from the repository are asked to the system one by one. The overall performance of the system for each question type is computed in terms of precision and recall measures. The figure 7 shows how the system precision varies as it tries to answer more and more questions.



System Performance		
Algorithm	Precision	Recall
Text match + Random	26.35	26.35
Text match	29.79	21.28

Figure 7: Question-Answering Interface Batch Mode

The Instant Search Interface

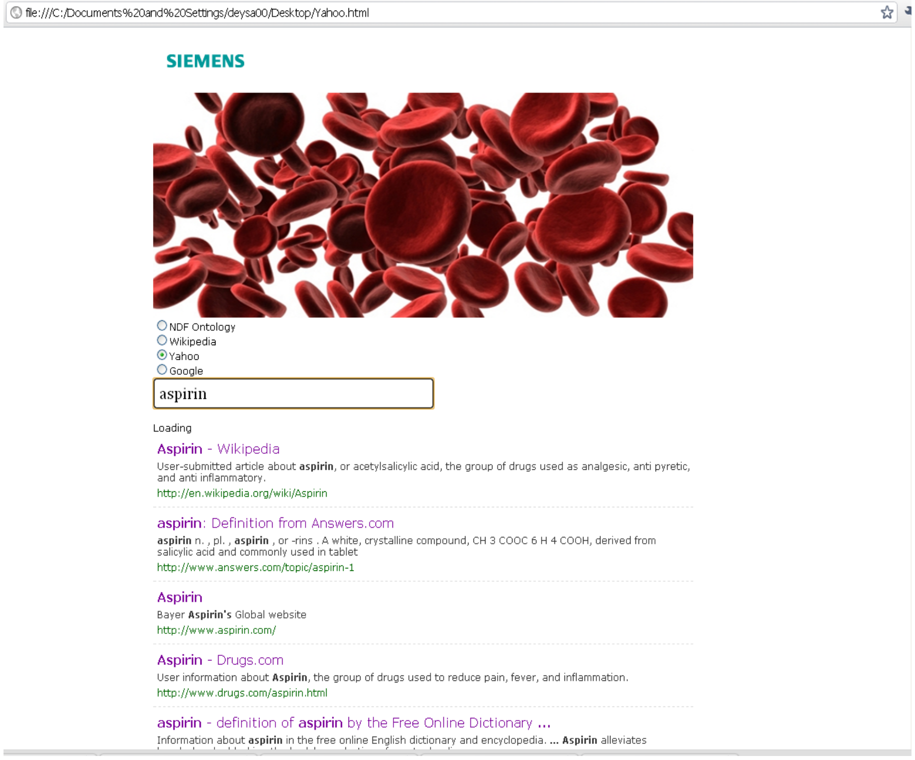
An instant search interface is implemented for Yahoo (figure 8).

System Design and Implementation

The following figure (9) shows the design of the system:

Corpus Clients

Any client implementing the CorpusClient interface must implement getSystemAnswer and rankAnswer methods, implementation of methods are specific to



file:///C:/Documents%20and%20Settings/deysa00/Desktop/Yahoo.html

SIEMENS

○ NDF Ontology
○ Wikipedia
● Yahoo
○ Google

aspirin

Loading

Aspirin - Wikipedia
User-submitted article about **aspirin**, or acetylsalicylic acid, the group of drugs used as analgesic, anti pyretic, and anti inflammatory.
http://en.wikipedia.org/wiki/Aspirin

aspirin: Definition from Answers.com
aspirin n. , pl. , **aspirin** , or -rins . A white, crystalline compound, CH 3 COOC 6 H 4 COOH, derived from salicylic acid and commonly used in tablet
http://www.answers.com/topic/aspirin-1

Aspirin
Bayer **Aspirin's** Global website
http://www.aspirin.com/

Aspirin - Drugs.com
User information about **Aspirin**, the group of drugs used to reduce pain, fever, and inflammation.
http://www.drugs.com/aspirin.html

aspirin - definition of aspirin by the Free Online Dictionary ...
Information about **aspirin** in the free online English dictionary and encyclopedia. **...** **Aspirin** alleviates

Figure 8: Yahoo Instant Search

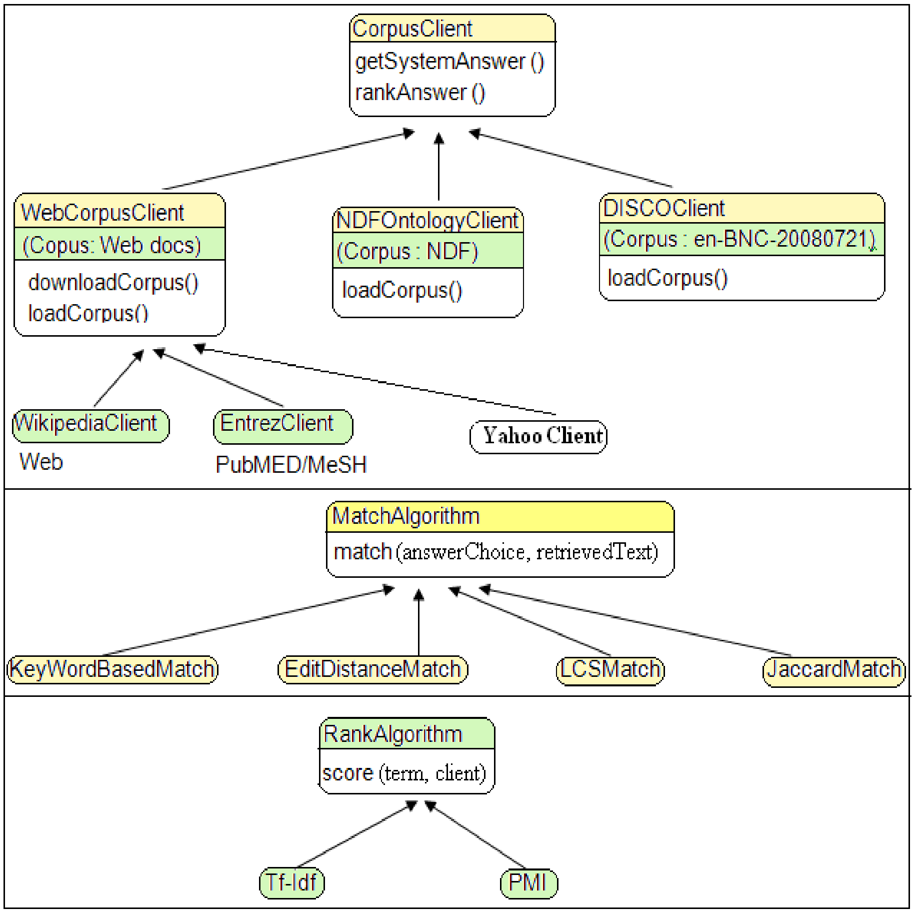
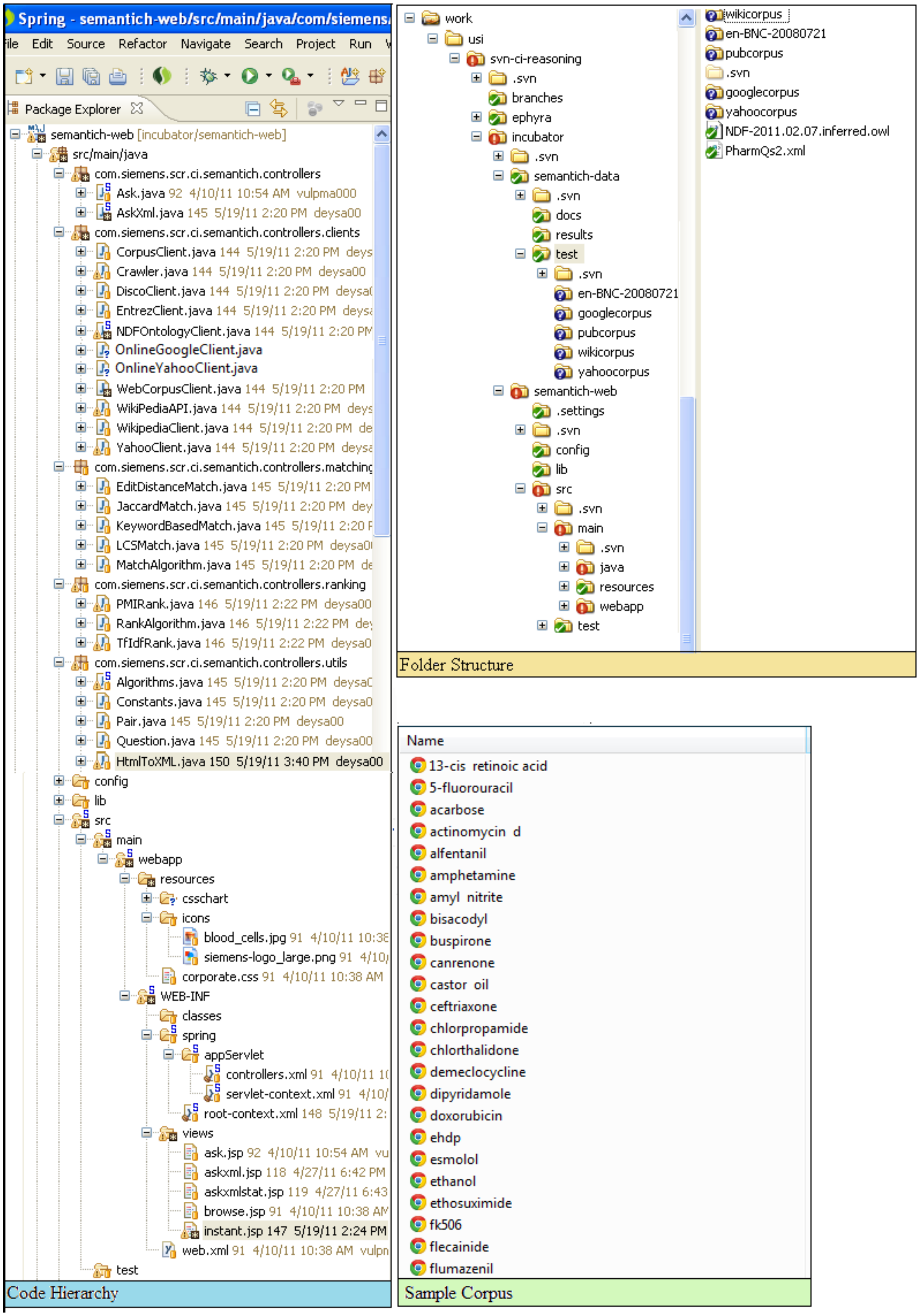


Figure 9: Design of the Question-Answering System


Spring - semantich-web/src/main/java/com/siemens
ile Edit Source Refactor Navigate Search Project Run
Package Explorer

```
semantich-web [incubator/semantich-web]
  src/main/java
    com.siemens.scr.ci.semantich.controllers
      Ask.java 92  4/10/11 10:54 AM  vulpma000
      AskXml.java 145  5/19/11 2:20 PM  deysa00
    com.siemens.scr.ci.semantich.controllers.clients
      CorpusClient.java 144  5/19/11 2:20 PM  deys
      Crawler.java 144  5/19/11 2:20 PM  deysa00
      DiscoClient.java 144  5/19/11 2:20 PM  deysa(
      EntrezClient.java 144  5/19/11 2:20 PM  deysa
      NDFOntologyClient.java 144  5/19/11 2:20 PM
      OnlineGoogleClient.java
      OnlineYahooClient.java
      WebCorpusClient.java 144  5/19/11 2:20 PM
      WikiPediaAPI.java 144  5/19/11 2:20 PM  deys
      WikipediaClient.java 144  5/19/11 2:20 PM  de
      YahooClient.java 144  5/19/11 2:20 PM  deysa
    com.siemens.scr.ci.semantich.controllers.matching
      EditDistanceMatch.java 145  5/19/11 2:20 PM
      JaccardMatch.java 145  5/19/11 2:20 PM  dey
      KeywordBasedMatch.java 145  5/19/11 2:20 F
      LCSMatch.java 145  5/19/11 2:20 PM  deysa0
      MatchAlgorithm.java 145  5/19/11 2:20 PM  de
    com.siemens.scr.ci.semantich.controllers.ranking
      PMIRank.java 146  5/19/11 2:22 PM  deysa00
      RankAlgorithm.java 146  5/19/11 2:22 PM  dey
      TfIdfRank.java 146  5/19/11 2:22 PM  deysa0
    com.siemens.scr.ci.semantich.controllers.utils
      Algorithms.java 145  5/19/11 2:20 PM  deysa0
      Constants.java 145  5/19/11 2:20 PM  deysa0
      Pair.java 145  5/19/11 2:20 PM  deysa00
      Question.java 145  5/19/11 2:20 PM  deysa00
      HtmlToXML.java 150  5/19/11 3:40 PM  deysa00
    config
    lib
    src
      main
        webapp
          resources
            csschart
            icons
              blood_cells.jpg 91  4/10/11 10:38
              siemens-logo_large.png 91  4/10/
            corporate.css 91  4/10/11 10:38 AM
          WEB-INF
            classes
            spring
              appServlet
                controllers.xml 91  4/10/11 1(
                servlet-context.xml 91  4/10/
              root-context.xml 148  5/19/11 2:
            views
              ask.jsp 92  4/10/11 10:54 AM  vu
              askxml.jsp 118  4/27/11 6:42 PM
              askxmlstat.jsp 119  4/27/11 6:43
              browse.jsp 91  4/10/11 10:38 AM
              instant.jsp 147  5/19/11 2:24 PM
            web.xml 91  4/10/11 10:38 AM  vulpn
      test
```

Code Hierarchy

```
work
  usi
    svn-ci-reasoning
      .svn
      branches
      ephyra
      incubator
        .svn
        semantich-data
          .svn
          docs
          results
          test
            .svn
            en-BNC-20080721
            googlecorpus
            pubcorpus
            wikicorpus
            yahoocorpus
        semantich-web
          .settings
          .svn
          config
          lib
          src
            .svn
            main
              .svn
              java
              resources
              webapp
            test
```

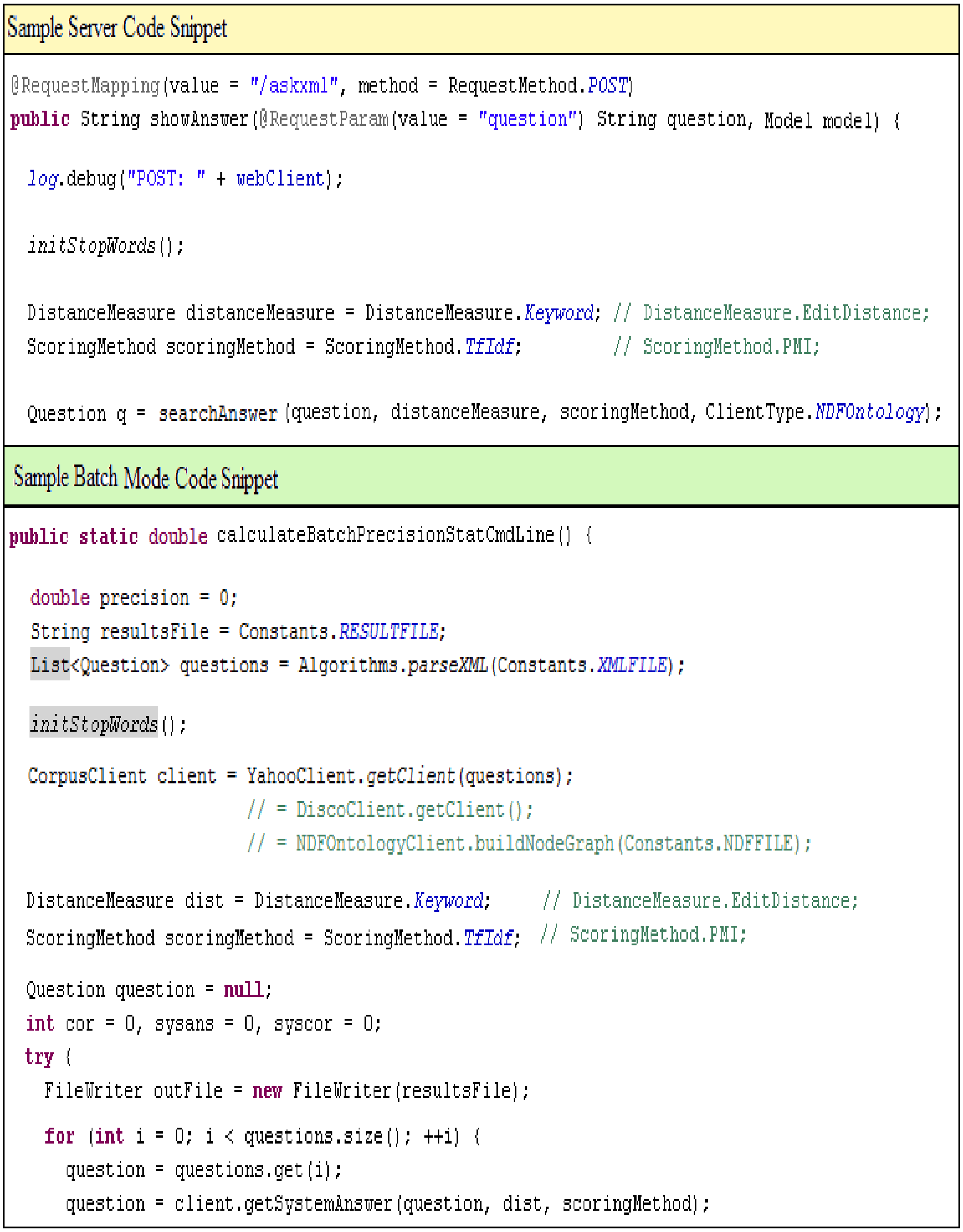
```
wikicorpus
en-BNC-20080721
pubcorpus
.svn
googlecorpus
yahoocorpus
NDF-2011.02.07.inferred.owl
PharmQs2.xml
```

Folder Structure

```
Name
13-cis  retinoic acid
5-fluorouracil
acarbose
actinomycin  d
alfentanil
amphetamine
amyl  nitrite
bisacodyl
buspirone
canrenone
castor  oil
ceftriaxone
chlorpropamide
chlorthalidone
demeclocycline
dipyridamole
doxorubicin
ehdp
esmolol
ethanol
ethosuximide
fk506
flecainide
flumazenil
```

Sample Corpus

Figure 10: Code Hierarchy



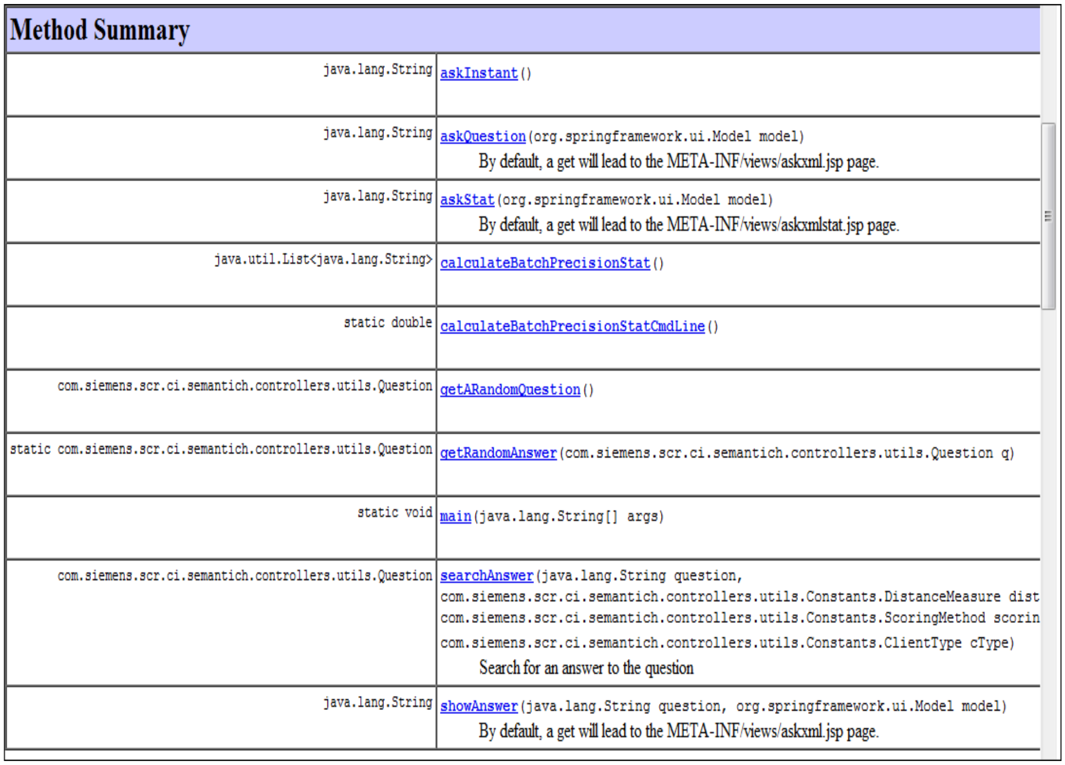
Sample Server Code Snippet

```
@RequestMapping(value = "/askxml", method = RequestMethod.POST)
public String showAnswer(@RequestParam(value = "question") String question, Model model) {

  log.debug("POST: " + webClient);

  initStopWords();

  DistanceMeasure distanceMeasure = DistanceMeasure.Keyword; // DistanceMeasure.EditDistance;
  ScoringMethod scoringMethod = ScoringMethod.TfIdf;         // ScoringMethod.PMI;

  Question q = searchAnswer(question, distanceMeasure, scoringMethod, ClientType.NDFOntology);
```

Sample Batch Mode Code Snippet

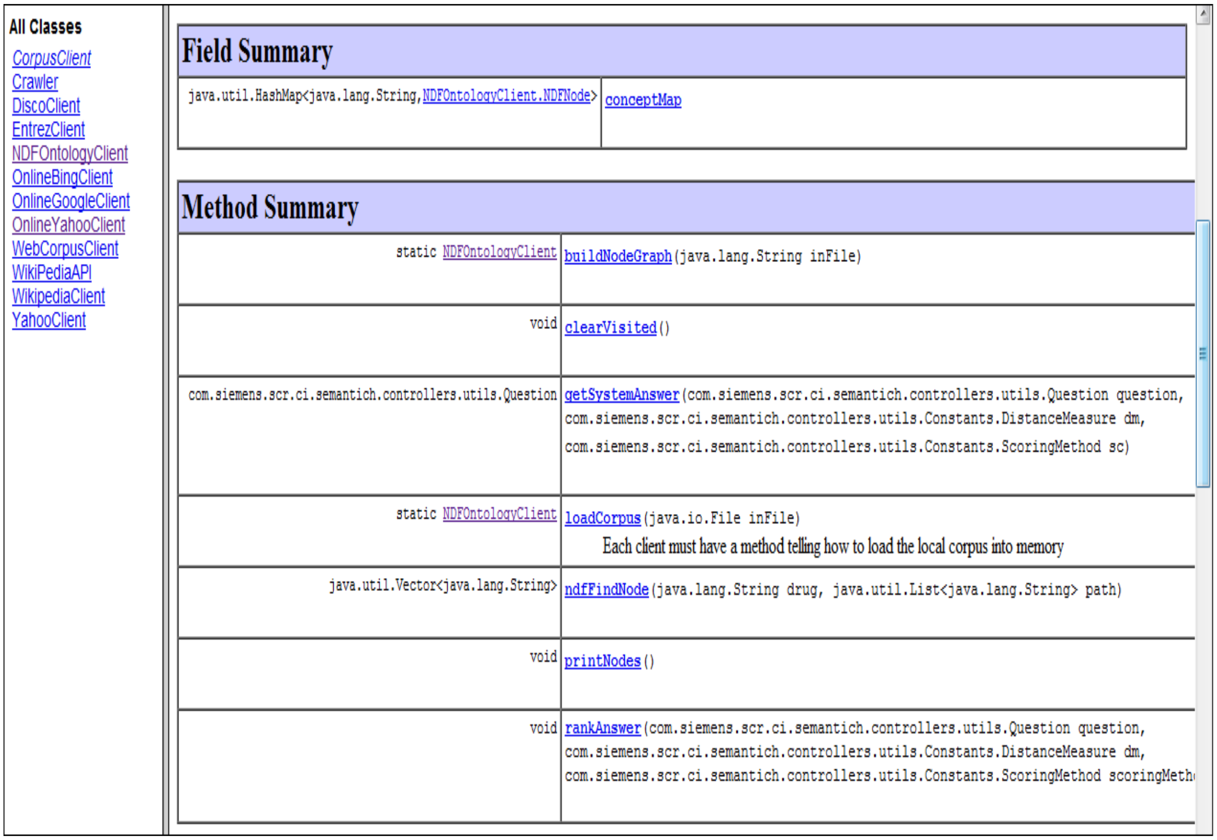
```
public static double calculateBatchPrecisionStatCmdLine() {

  double precision = 0;
  String resultsFile = Constants.RESULTFILE;
  List<Question> questions = Algorithms.parseXML(Constants.XMLFILE);

  initStopWords();

  CorpusClient client = YahooClient.getClient(questions);
                     // = DiscoClient.getClient();
                     // = NDFOntologyClient.buildNodeGraph(Constants.NDFFILE);

  DistanceMeasure dist = DistanceMeasure.Keyword;     // DistanceMeasure.EditDistance;
  ScoringMethod scoringMethod = ScoringMethod.TfIdf;  // ScoringMethod.PMI;

  Question question = null;
  int cor = 0, sysans = 0, syscor = 0;
  try {
    FileWriter outFile = new FileWriter(resultsFile);

    for (int i = 0; i < questions.size(); ++i) {
      question = questions.get(i);
      question = client.getSystemAnswer(question, dist, scoringMethod);
```

Figure 11: Sample Code for Demonstration



Method Summary	
java.lang.String	askInstant()
java.lang.String	askQuestion(org.springframework.ui.Model model) By default, a get will lead to the META-INF/views/askxml.jsp page.
java.lang.String	askStat(org.springframework.ui.Model model) By default, a get will lead to the META-INF/views/askxmlstat.jsp page.
java.util.List<java.lang.String>	calculateBatchPrecisionStat()
static double	calculateBatchPrecisionStatCmdLine()
com.siemens.scr.ci.semantich.controllers.utils.Question	getARandomQuestion()
static com.siemens.scr.ci.semantich.controllers.utils.Question	getRandomAnswer(com.siemens.scr.ci.semantich.controllers.utils.Question q)
static void	main(java.lang.String[] args)
com.siemens.scr.ci.semantich.controllers.utils.Question	searchAnswer(java.lang.String question, com.siemens.scr.ci.semantich.controllers.utils.Constants.DistanceMeasure dist com.siemens.scr.ci.semantich.controllers.utils.Constants.ScoringMethod scorin com.siemens.scr.ci.semantich.controllers.utils.Constants.ClientType cType) Search for an answer to the question
java.lang.String	showAnswer(java.lang.String question, org.springframework.ui.Model model) By default, a get will lead to the META-INF/views/askxml.jsp page.

Figure 12: AskXml Documentation



All Classes

CorpusClient
Crawler
DiscoClient
EntrezClient
NDFOntologyClient
OnlineBingClient
OnlineGoogleClient
OnlineYahooClient
WebCorpusClient
WikiPediaAPI
WikipediaClient
YahooClient

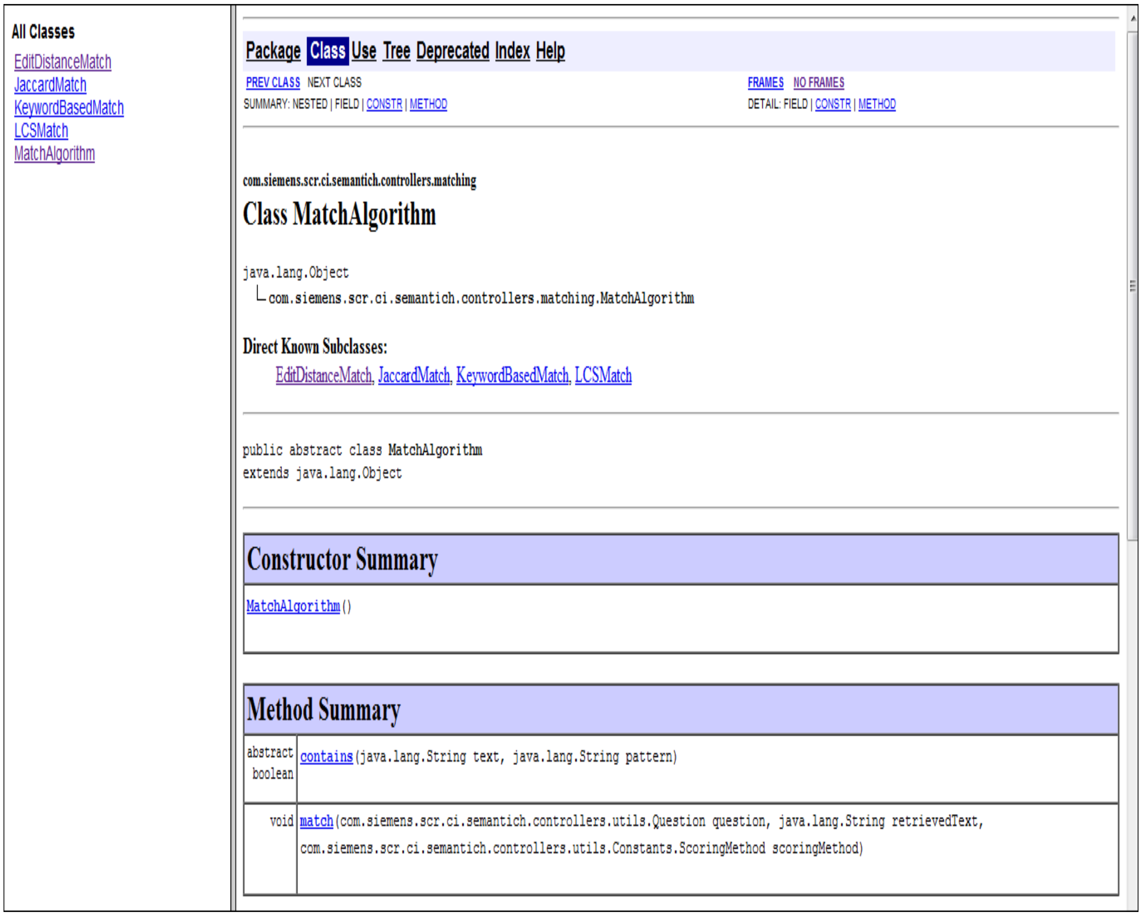
Field Summary

java.util.HashMap<java.lang.String,NDFOntologyClient.NDFNode>	conceptMap

Method Summary

static NDFOntologyClient	buildNodeGraph(java.lang.String inFile)
void	clearVisited()
com.siemens.scr.ci.semantich.controllers.utils.Question	getSystemAnswer(com.siemens.scr.ci.semantich.controllers.utils.Question question, com.siemens.scr.ci.semantich.controllers.utils.Constants.DistanceMeasure dm, com.siemens.scr.ci.semantich.controllers.utils.Constants.ScoringMethod sc)
static NDFOntologyClient	loadCorpus(java.io.File inFile) Each client must have a method telling how to load the local corpus into memory
java.util.Vector<java.lang.String>	ndfFindNode(java.lang.String drug, java.util.List<java.lang.String> path)
void	printNodes()
void	rankAnswer(com.siemens.scr.ci.semantich.controllers.utils.Question question, com.siemens.scr.ci.semantich.controllers.utils.Constants.DistanceMeasure dm, com.siemens.scr.ci.semantich.controllers.utils.Constants.ScoringMethod scoringMeth

Figure 13: Documentation for Clients (NDFOntologyClient)



All Classes

EditDistanceMatch
JaccardMatch
KeywordBasedMatch
LCSMatch
MatchAlgorithm

Package Class Use Tree Deprecated Index Help
PREV CLASS NEXT CLASS
FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD

com.siemens.scr.ci.semantich.controllers.matching

Class MatchAlgorithm

```
java.lang.Object
  └com.siemens.scr.ci.semantich.controllers.matching.MatchAlgorithm
```

Direct Known Subclasses:

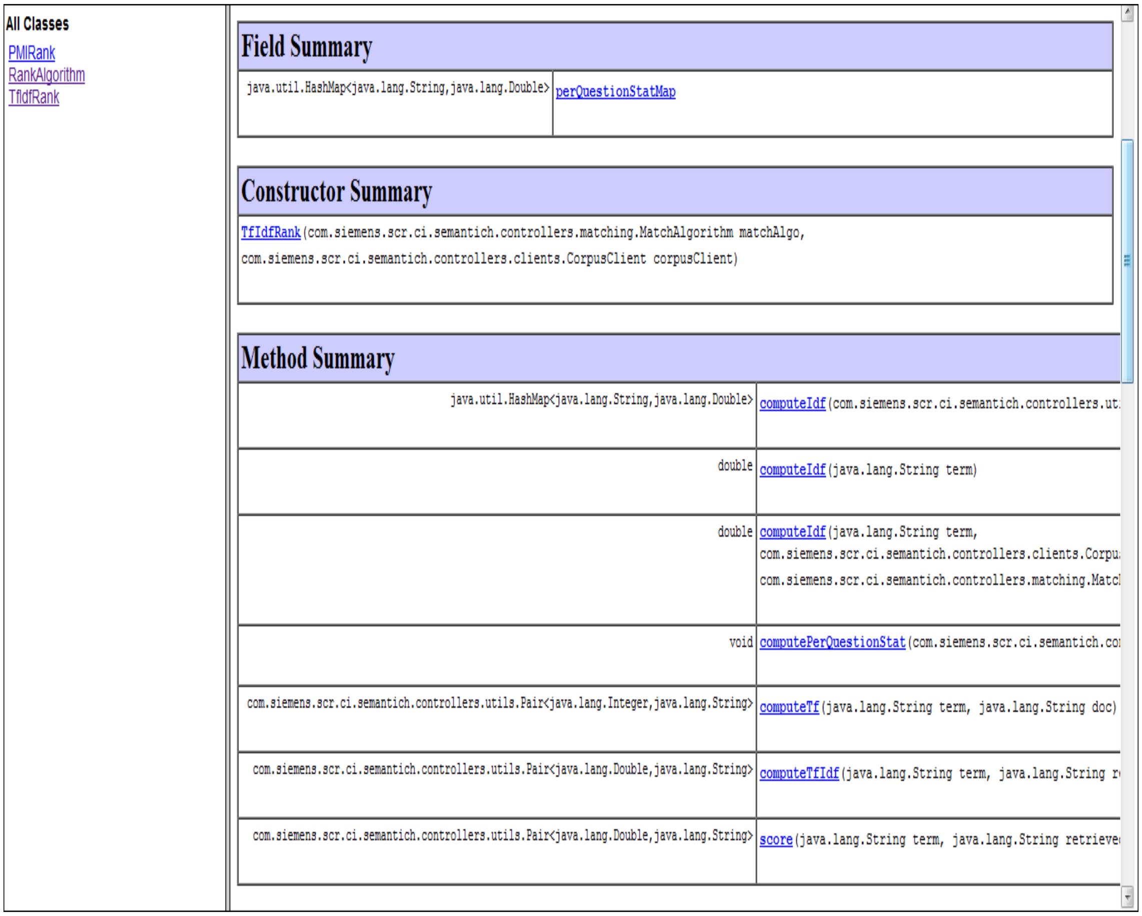
EditDistanceMatch, JaccardMatch, KeywordBasedMatch, LCSMatch

```
public abstract class MatchAlgorithm
extends java.lang.Object
```

Constructor Summary
MatchAlgorithm()

Method Summary	
abstract boolean	contains(java.lang.String text, java.lang.String pattern)
void	match(com.siemens.scr.ci.semantich.controllers.utils.Question question, java.lang.String retrievedText, com.siemens.scr.ci.semantich.controllers.utils.Constants.ScoringMethod scoringMethod)

Figure 14: Documentation for Match Algorithms



All Classes

- PMIRank
- RankAlgorithm
- TfIdfRank

Field Summary

java.util.HashMap<java.lang.String,java.lang.Double>	perQuestionStatMap

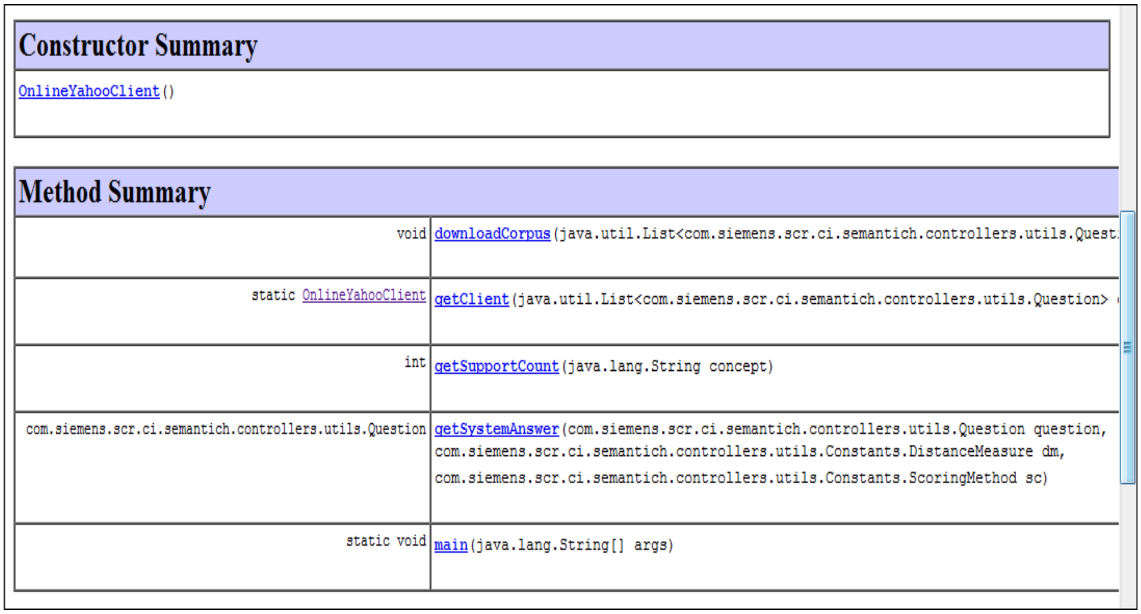
Constructor Summary

```
TfIdfRank(com.siemens.scr.ci.semantich.controllers.matching.MatchAlgorithm matchAlgo,
com.siemens.scr.ci.semantich.controllers.clients.CorpusClient corpusClient)
```

Method Summary

java.util.HashMap<java.lang.String,java.lang.Double>	computeIdf(com.siemens.scr.ci.semantich.controllers.ut
double	computeIdf(java.lang.String term)
double	computeIdf(java.lang.String term, com.siemens.scr.ci.semantich.controllers.clients.Corpu com.siemens.scr.ci.semantich.controllers.matching.Matc
void	computePerQuestionStat(com.siemens.scr.ci.semantich.co
com.siemens.scr.ci.semantich.controllers.utils.Pair<java.lang.Integer,java.lang.String>	computeTf(java.lang.String term, java.lang.String doc)
com.siemens.scr.ci.semantich.controllers.utils.Pair<java.lang.Double,java.lang.String>	computeTfIdf(java.lang.String term, java.lang.String r
com.siemens.scr.ci.semantich.controllers.utils.Pair<java.lang.Double,java.lang.String>	score(java.lang.String term, java.lang.String retrieve

Figure 15: Documentation for Rank Algorithms (TfIdfRank)



Constructor Summary
OnlineYahooClient()

Method Summary	
void	downloadCorpus(java.util.List<com.siemens.scr.ci.semantich.controllers.utils.Quest
static OnlineYahooClient	getClient(java.util.List<com.siemens.scr.ci.semantich.controllers.utils.Question>
int	getSupportCount(java.lang.String concept)
com.siemens.scr.ci.semantich.controllers.utils.Question	getSystemAnswer(com.siemens.scr.ci.semantich.controllers.utils.Question question, com.siemens.scr.ci.semantich.controllers.utils.Constants.DistanceMeasure dm, com.siemens.scr.ci.semantich.controllers.utils.Constants.ScoringMethod sc)
static void	main(java.lang.String[] args)

Figure 16: Online Search Client

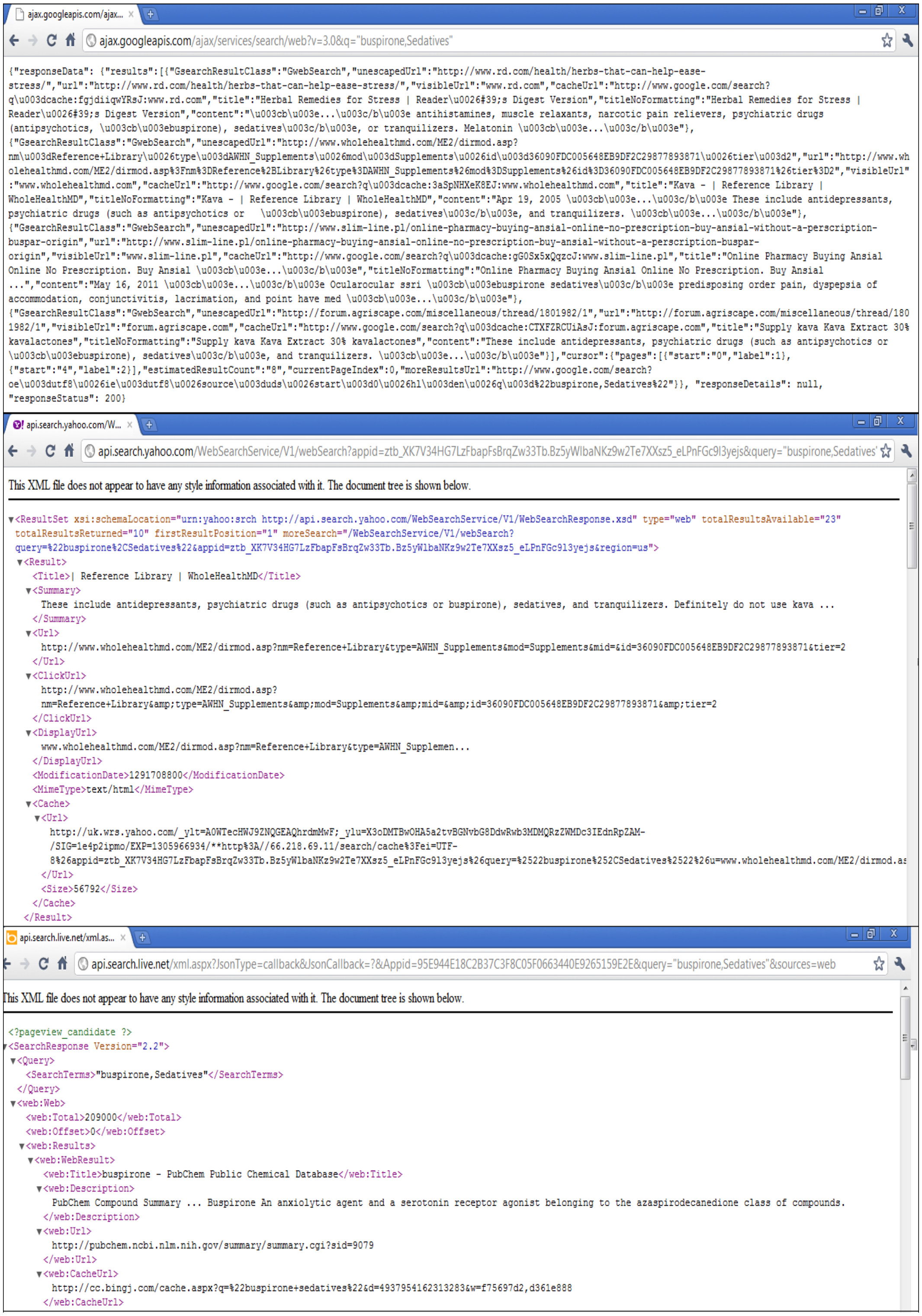
the corresponding client (e.g., NDFOntologyClient will have to traverse its node graph to find an answer when WebCorpusClient has to search for the concept in its downloaded corpus). Each client must have a method telling how to load the local corpus into memory (e.g., as graph, map etc.). Additionally, a web corpus client must provide a means of downloading the corpus from the web and storing it locally. Each client is implemented as singleton interfaces so that they are instantiated and the corresponding corpus is loaded in memory only once. Any client to be added in future should implement any of CorpusClient or WebCorpusClient interface.

Matching with distance measures

Algorithms for matching a term from the alternative answer option (pattern) with a word in the answer retrieved (text) are abstracted using strategy-like design pattern. Any algorithm for finding a match (by using some similarity/distance measure) should implement the MatchAlgorithm interface that has a method match that basically tells how to match. Any similarity / distance measure to be added in future should implement MatchAlgorithm.

Ranking answers

Algorithms for ranking an answer-alternative with score are abstracted in a similar manner. Any ranking algorithm must implement the score method. Note


ajax.googleapis.com/ajax...
api.search.yahoo.com/W...
api.search.live.net/xml.as...

ajax.googleapis.com/ajax/services/search/web?v=3.0&q="buspirone,Sedatives"

```
{"responseData": {"results":[{"GsearchResultClass":"GwebSearch","unescapedUrl":"http://www.rd.com/health/herbs-that-can-help-ease-
stress/","url":"http://www.rd.com/health/herbs-that-can-help-ease-stress/","visibleUrl":"www.rd.com","cacheUrl":"http://www.google.com/search?
q\u003dcache:fgjdiiqwYRsJ:www.rd.com","title":"Herbal Remedies for Stress | Reader\u0026#39;s Digest Version","titleNoFormatting":"Herbal Remedies for Stress |
Reader\u0026#39;s Digest Version","content":"\u003cb\u003e...\u003c/b\u003e antihistamines, muscle relaxants, narcotic pain relievers, psychiatric drugs
(antipsychotics, \u003cb\u003ebuspirone), sedatives\u003c/b\u003e, or tranquilizers. Melatonin \u003cb\u003e...\u003c/b\u003e"},
{"GsearchResultClass":"GwebSearch","unescapedUrl":"http://www.wholehealthmd.com/ME2/dirmod.asp?
nm\u003dReference+Library\u0026type\u003dAWHN_Supplements\u0026mod\u003dSupplements\u0026id\u003d36090FDC005648EB9DF2C29877893871\u0026tier\u003d2","url":"http://www.wh
olehealthmd.com/ME2/dirmod.asp%3Fnm%3DReference%2BLibrary%26type%3DAWHN_Supplements%26mod%3DSupplements%26id%3D36090FDC005648EB9DF2C29877893871%26tier%3D2","visibleUrl"
:"www.wholehealthmd.com","cacheUrl":"http://www.google.com/search?q\u003dcache:3aSpNHXeK8EJ:www.wholehealthmd.com","title":"Kava - | Reference Library |
WholeHealthMD","titleNoFormatting":"Kava - | Reference Library | WholeHealthMD","content":"Apr 19, 2005 \u003cb\u003e...\u003c/b\u003e These include antidepressants,
psychiatric drugs (such as antipsychotics or   \u003cb\u003ebuspirone), sedatives\u003c/b\u003e, and tranquilizers. \u003cb\u003e...\u003c/b\u003e"},
{"GsearchResultClass":"GwebSearch","unescapedUrl":"http://www.slim-line.pl/online-pharmacy-buying-ansial-online-no-prescription-buy-ansial-without-a-perscription-
buspar-origin","url":"http://www.slim-line.pl/online-pharmacy-buying-ansial-online-no-prescription-buy-ansial-without-a-perscription-buspar-
origin","visibleUrl":"www.slim-line.pl","cacheUrl":"http://www.google.com/search?q\u003dcache:gG0Sx5xQqzcJ:www.slim-line.pl","title":"Online Pharmacy Buying Ansial
Online No Prescription. Buy Ansial \u003cb\u003e...\u003c/b\u003e","titleNoFormatting":"Online Pharmacy Buying Ansial Online No Prescription. Buy Ansial
...","content":"May 16, 2011 \u003cb\u003e...\u003c/b\u003e Ocularocular ssri \u003cb\u003ebuspirone sedatives\u003c/b\u003e predisposing order pain, dyspepsia of
accommodation, conjunctivitis, lacrimation, and point have med \u003cb\u003e...\u003c/b\u003e"},
{"GsearchResultClass":"GwebSearch","unescapedUrl":"http://forum.agriscape.com/miscellaneous/thread/1801982/1","url":"http://forum.agriscape.com/miscellaneous/thread/180
1982/1","visibleUrl":"forum.agriscape.com","cacheUrl":"http://www.google.com/search?q\u003dcache:CTXFZRCUiAsJ:forum.agriscape.com","title":"Supply kava Kava Extract 30%
kavalactones","titleNoFormatting":"Supply kava Kava Extract 30% kavalactones","content":"These include antidepressants, psychiatric drugs (such as antipsychotics or
\u003cb\u003ebuspirone), sedatives\u003c/b\u003e, and tranquilizers. \u003cb\u003e...\u003c/b\u003e"}],"cursor":{"pages":[{"start":"0","label":1},
{"start":"4","label":2}],"estimatedResultCount":"8","currentPageIndex":0,"moreResultsUrl":"http://www.google.com/search?
oe\u003dutf8\u0026ie\u003dutf8\u0026source\u003duds\u0026start\u003d0\u0026hl\u003den\u0026q\u003d%22buspirone,Sedatives%22"}}, "responseDetails": null,
"responseStatus": 200}
```

api.search.yahoo.com/WebSearchService/V1/webSearch?appid=ztb_XK7V34HG7LzFbapFsBrqZw33Tb.Bz5yWlbaNKz9w2Te7XXsz5_eLPnFGc9l3yejs&query="buspirone,Sedatives'

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<ResultSet xsi:schemaLocation="urn:yahoo:srch http://api.search.yahoo.com/WebSearchService/V1/WebSearchResponse.xsd" type="web" totalResultsAvailable="23"
 totalResultsReturned="10" firstResultPosition="1" moreSearch="/WebSearchService/V1/webSearch?
 query=%22buspirone%2CSedatives%22&appid=ztb_XK7V34HG7LzFbapFsBrqZw33Tb.Bz5yWlbaNKz9w2Te7XXsz5_eLPnFGc913yejs&region=us">
  ▼<Result>
     <Title>| Reference Library | WholeHealthMD</Title>
    ▼<Summary>
       These include antidepressants, psychiatric drugs (such as antipsychotics or buspirone), sedatives, and tranquilizers. Definitely do not use kava ...
     </Summary>
    ▼<Url>
       http://www.wholehealthmd.com/ME2/dirmod.asp?nm=Reference+Library&type=AWHN_Supplements&mod=Supplements&mid=&id=36090FDC005648EB9DF2C29877893871&tier=2
     </Url>
    ▼<ClickUrl>
       http://www.wholehealthmd.com/ME2/dirmod.asp?
       nm=Reference+Library&amp;type=AWHN_Supplements&amp;mod=Supplements&amp;mid=&amp;id=36090FDC005648EB9DF2C29877893871&amp;tier=2
     </ClickUrl>
    ▼<DisplayUrl>
       www.wholehealthmd.com/ME2/dirmod.asp?nm=Reference+Library&type=AWHN_Supplemen...
     </DisplayUrl>
     <ModificationDate>1291708800</ModificationDate>
     <MimeType>text/html</MimeType>
    ▼<Cache>
      ▼<Url>
         http://uk.wrs.yahoo.com/_ylt=A0WTecHWJ9ZNQGEAQhrdmMwF;_ylu=X3oDMTBwOHA5a2tvBGNvbG8DdwRwb3MDMQRzZWMDc3IEdnRpZAM-
         /SIG=1e4p2ipmo/EXP=1305966934/**http%3A//66.218.69.11/search/cache%3Fei=UTF-
         8%26appid=ztb_XK7V34HG7LzFbapFsBrqZw33Tb.Bz5yWlbaNKz9w2Te7XXsz5_eLPnFGc913yejs%26query=%2522buspirone%252CSedatives%2522%26u=www.wholehealthmd.com/ME2/dirmod.as
       </Url>
       <Size>56792</Size>
     </Cache>
   </Result>
```

api.search.live.net/xml.aspx?JsonType=callback&JsonCallback=?&Appid=95E944E18C2B37C3F8C05F0663440E9265159E2E&query="buspirone,Sedatives"&sources=web

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?pageview_candidate ?>
▼<SearchResponse Version="2.2">
 ▼<Query>
    <SearchTerms>"buspirone,Sedatives"</SearchTerms>
  </Query>
 ▼<web:Web>
    <web:Total>209000</web:Total>
    <web:Offset>0</web:Offset>
   ▼<web:Results>
     ▼<web:WebResult>
        <web:Title>buspirone - PubChem Public Chemical Database</web:Title>
       ▼<web:Description>
          PubChem Compound Summary ... Buspirone An anxiolytic agent and a serotonin receptor agonist belonging to the azaspirodecanedione class of compounds.
        </web:Description>
       ▼<web:Url>
          http://pubchem.ncbi.nlm.nih.gov/summary/summary.cgi?sid=9079
        </web:Url>
       ▼<web:CacheUrl>
          http://cc.bingj.com/cache.aspx?q=%22buspirone+sedatives%22&d=4937954162313283&w=f75697d2,d361e888
        </web:CacheUrl>
```

Figure 17: Online Searching: searching the clients with “buspirone,sedatives”

that score method requires the client (unlike match) because scoring requires searching in the corpus as well (e.g. tf-idf).

Experimental Results

Using distance measures to compute similarity of concepts

The following table shows the performance of the System. Total number of questions in the repository is 594. Distance measure used is edit-distance. Ranking algorithm used is Tf-Idf.

Corpus	Answered	Correct	Precision	Recall	F-Measure
NDF	406	210	51.72	40.57	45.47
Wiki	541	323	59.70	54.38	56.92
PubMed	569	296	52.02	49.83	50.90

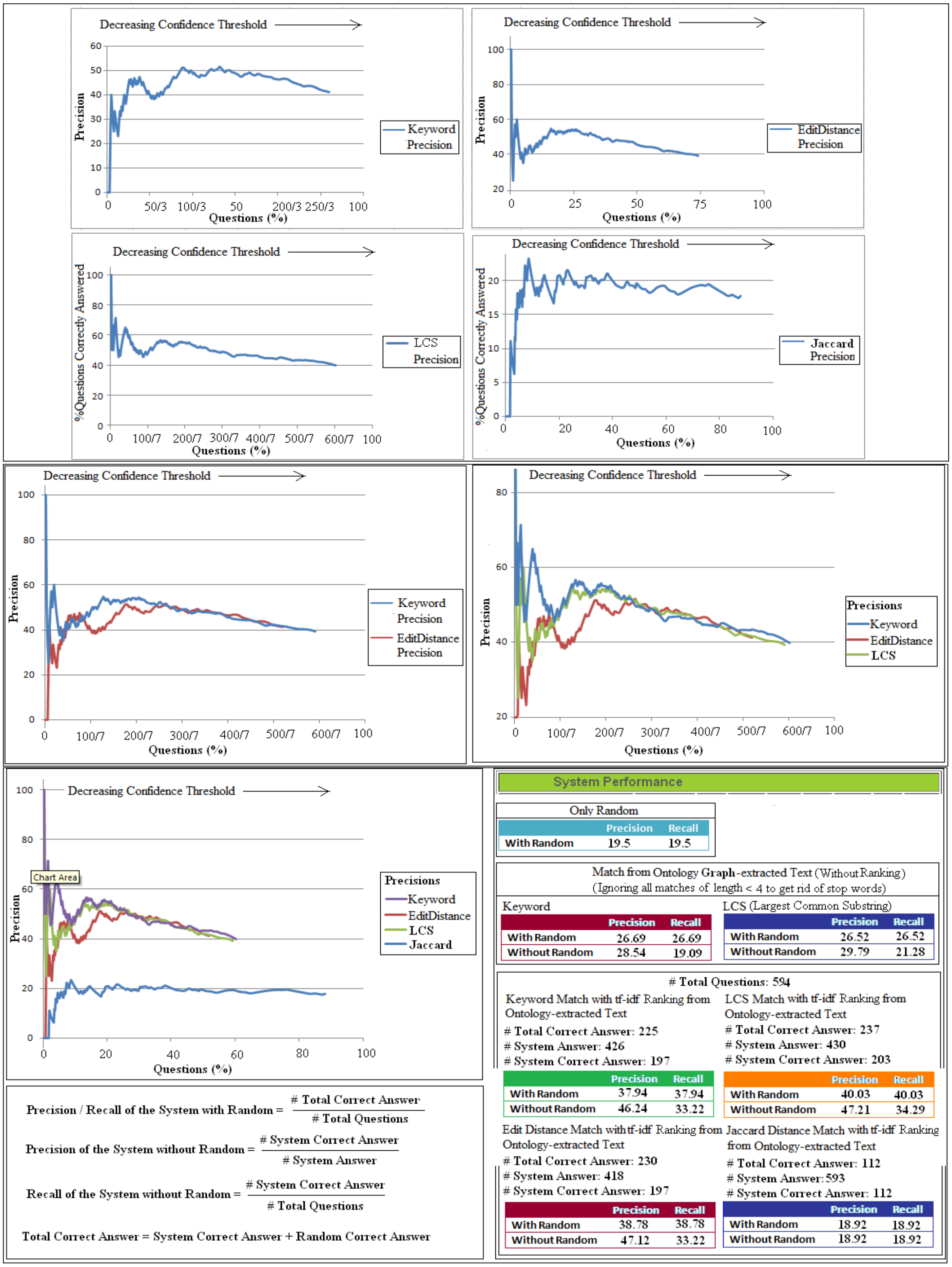
If the system is made to guess the answer of the multiple-choice questions purely randomly (by selecting any one of 5 answer-choices), the precision of the system comes to be around 20%. As different experimental results show, we got more than 30% improvement in terms of precision by applying different ranking/matching techniques and using different corpus.

A set of ROC-like plots are generated as experimental results (shown in the figures 18, 19, 20, 21). In each of these plots, the "*confidence threshold*" (e.g. Tf-Idf score for the answer-choice to be predicted) is decreased gradually from left to right. The system chooses to answer a question iff the score of the predicted answer-choice is greater than the confidence threshold, otherwise it does not answer the question at all. Hence, as the threshold is decreased, more and more questions become candidates to be answered by the system. But this also changes the precision that gives us the following ROC-like curves (figures 20, 21).

The precision curves corresponding to different distance measures (used in matching) are compared, as can be seen Keyword, LCS and EditDistance measure give similar precision, where Jaccard gives much lower precision (since it captures very weak match, as shown in the figure 19).

Also, the precision of the system is measured in terms of different question types (figures 19). As can be seen, the simpler question types like "*class of*", "*indication of*" and "*mechanisms of action/resistence*" questions result in higher precision, while complex question types like "*side effect of*" and "*which of lead to poor precision values*", which is an intuitively expected result.

Finally, the precision of the system is compared using different corpus (NDFOntology, PubMed, Wikipedia etc.). As can be seen, wikipedia/pubmed corpus outperforms others in most of the cases (figures 20, 21). For medical corpus like PubMed/MeSH the result is expected to be better than others, but ranking using tf-idf performs a little worse in this case.



System Performance

Only Random

	Precision	Recall
With Random	19.5	19.5

Match from Ontology Graph-extracted Text (Without Ranking)
(Ignoring all matches of length < 4 to get rid of stop words)

Keyword

	Precision	Recall
With Random	26.69	26.69
Without Random	28.54	19.09

LCS (Largest Common Substring)

	Precision	Recall
With Random	26.52	26.52
Without Random	29.79	21.28

Total Questions: 594

Keyword Match with tf-idf Ranking from Ontology-extracted Text

Total Correct Answer: 225
System Answer: 426
System Correct Answer: 197

	Precision	Recall
With Random	37.94	37.94
Without Random	46.24	33.22

LCS Match with tf-idf Ranking from Ontology-extracted Text

Total Correct Answer: 237
System Answer: 430
System Correct Answer: 203

	Precision	Recall
With Random	40.03	40.03
Without Random	47.21	34.29

Edit Distance Match with tf-idf Ranking from Ontology-extracted Text

Total Correct Answer: 230
System Answer: 418
System Correct Answer: 197

	Precision	Recall
With Random	38.78	38.78
Without Random	47.12	33.22

Jaccard Distance Match with tf-idf Ranking from Ontology-extracted Text

Total Correct Answer: 112
System Answer: 593
System Correct Answer: 112

	Precision	Recall
With Random	18.92	18.92
Without Random	18.92	18.92

$$\text{Precision / Recall of the System with Random} = \frac{\text{\# Total Correct Answer}}{\text{\# Total Questions}}$$

$$\text{Precision of the System without Random} = \frac{\text{\# System Correct Answer}}{\text{\# System Answer}}$$

$$\text{Recall of the System without Random} = \frac{\text{\# System Correct Answer}}{\text{\# Total Questions}}$$

Total Correct Answer = System Correct Answer + Random Correct Answer

Figure 18: Performance comparison w.r.t. different distance-measures used

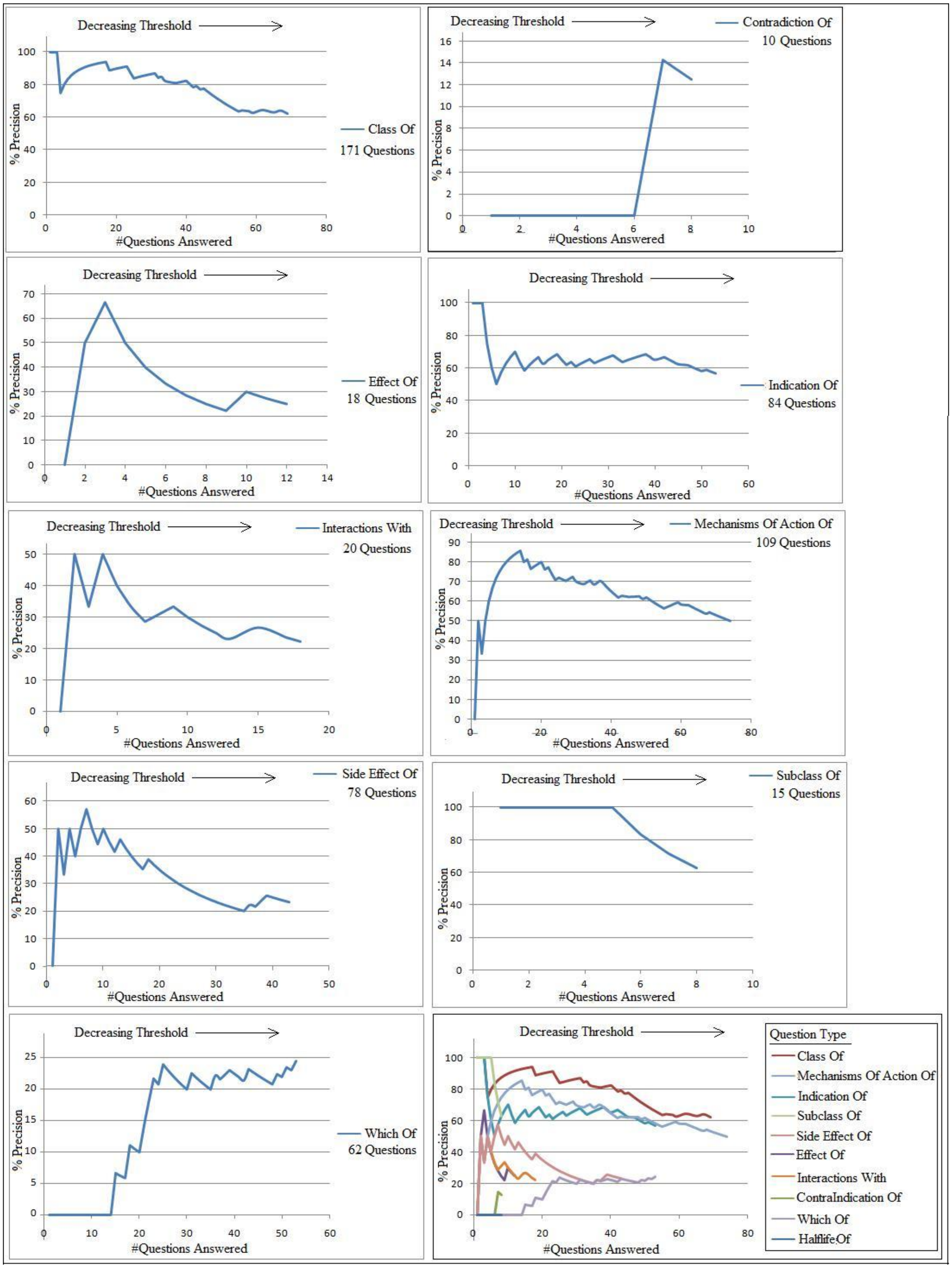


Figure 19: Performance comparison w.r.t. different question types using NDF-Ontology corpus and Keyword-based match using Tf-Idf Scoring (without eliminating stopwords)

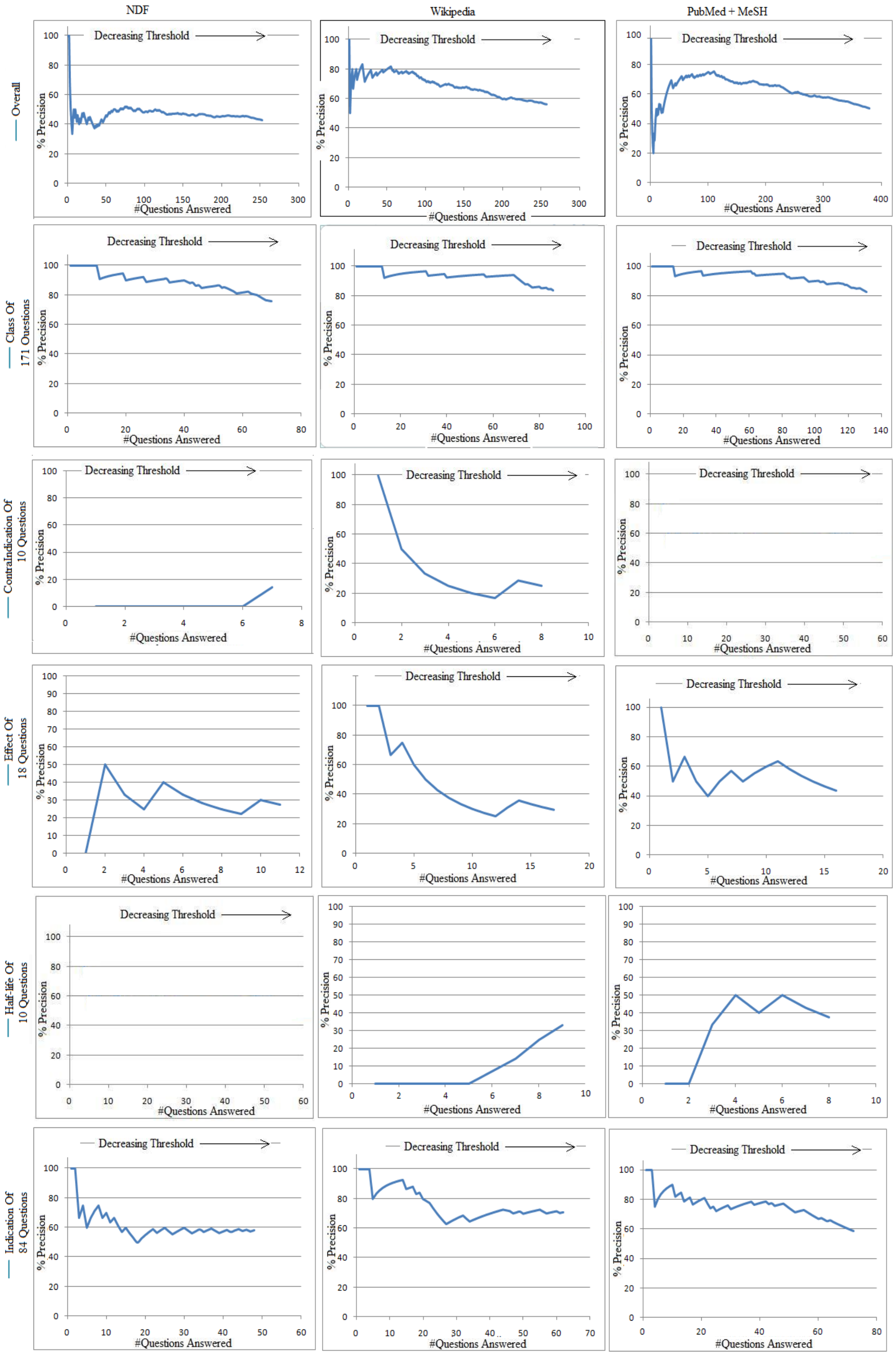


Figure 20: Performance comparison w.r.t. different question types and corpus using Edit-Distance match using Tf-Idf Scoring (after eliminating stopwords)

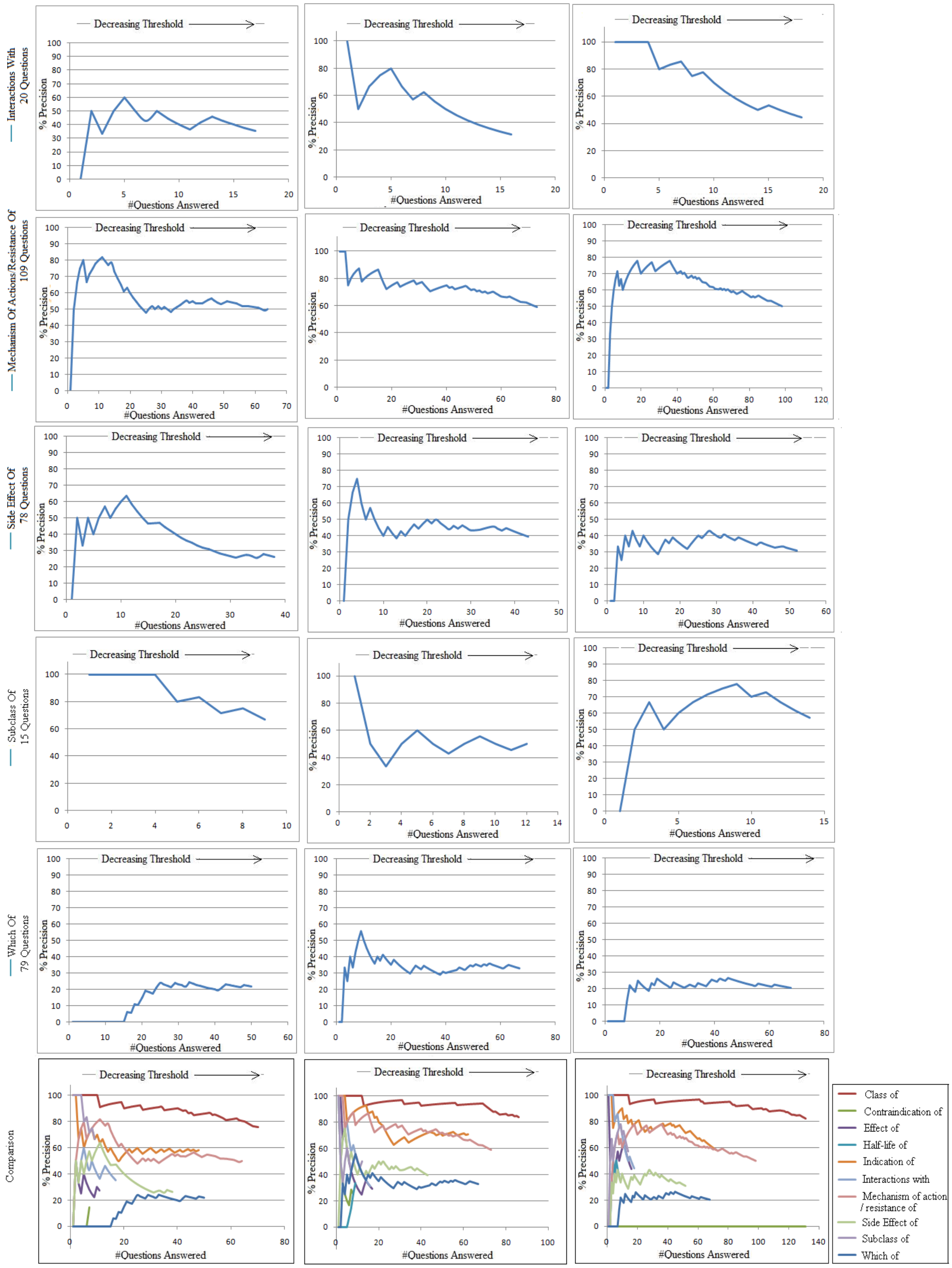


Figure 21: Performance comparison w.r.t. different question types and corpus using Edit-Distance match using Tf-Idf Scoring (after eliminating stopwords)

Using Semantic Similarity between concepts

Similarity	Precision	Recall
Disco (2nd Order)	21.43	18.52
PMI (local)	16.33	16.33

As can be seen, the performance of the system using semantic similarity-based ranking is much poorer (due to lack of medical terms in the DISCO corpus and also for not considering the "importance" of a particular term in an answer-choice) than when using tf-idf based ranking. The definition of the PMI similarity can be modified to take the importance of a term into account to improve the performance of the system.

Two new approaches and results

Query Reformulation

In this case, a given question is reformulated (to get multiple queries) by identifying the relation information in the question. Steps:

- First, retrieve the concept and the question-type from the question.
- Maintain a set of additional "relation keywords" associated with each question type, e.g., for "class of" questions we have similar phrases like "example of", "type of" etc. having the same meaning.
- Eliminate the stop-words / terms with low scores (using some threshold?) from the answer-choices to get concepts.
- e.g., for the question
  Class of 131-iodine is:
  (A)Beta-lactamase inhibitors
  (B)Beta blockers
  (C)Antivirals
  (D)Antithyroid agents
  (E)Antiarrhythmic agents
  the questions formed are: "class of 131-iodine Beta-lactamase inhibitors", "example of 131-iodine Beta blockers" etc. Match these queries with the retrieved text and choose the answer choice getting the highest score.
- Steps:
  1. Get an AppID to Search
  2. Reformulate QUeries: get a list of queries as English sentences formed out of the combination of question concept, question relation and concepts from an answer-choice. Search the search engine client (online) with all expanded queries.

e.g., "*class of 131-iodine*" is reformulated as the following expanded set of queries:

"Beta-lactamase inhibitors is the class of 131-iodine"
"131-iodine is a Beta-lactamase inhibitors"
"131-iodine,Beta-lactamase inhibitors"
"Beta blockers is the class of 131-iodine"
"131-iodine is a Beta blockers"
"131-iodine,Beta blockers"

3. Search different clients (yahoo/bing/google) and get "*Support Count*" as score from the returned XML/JSon result.
4. The query having the maximum support count is the winner
5. Find Precision of the system with reformulated queries.
6. Got only around 16% precision even in case of "class of" questions (Hence our assumption of co-occurrence of the concept and relation together in an English sentence does not work most of the times).

Classification using the Stanford NER Classifier

1. Prepare the training dataset:

   - Generate a dictionary of drugs from the NDF Ontology (by putting all the elements $< owl : Class\ rdf : ID = “N???????'' > \ldots < /owl : Class >$ that have their $< Level >$ attributes as *"VA Class"*, *"Ingredient"* or *"VA Product"*).
   - Start with a plain text containing some drug names (as positive data tuple) and some non-drug names (as negative data tuple) both, e.g. the text of 594 Pharmacology Board exam questions.
   - Generate the *annotated file* from the plain text as training file by assigning a label "DRUG" to a word if it belongs to the dictionary, assigning the lable "O" otherwise. Use this file to train a binary classifier.

2. Train: Learn a binary classifier (using the stanford-ner package) from the annotated training file.

3. Test: Tested on the drugs file obtained from the question hyperlinks (that are sure to be drugs) of the 594 Pharmacology Board exam questions.

4. Result: Got only around 18% accuracy (due to poor training data).

5. Cross-validation: to be done.

Conclusion

As explained earlier, if the QA system is made to guess the answer of a multiple-choice question purely randomly (by selecting any one of 5 answer-choices), the precision of the system will be around 20%. As different experimental results show, we got more than 30% improvement in terms of precision by applying different ranking/matching techniques and using different corpus. Edit distance/keyword/LCS distance measures give better performances while tf-idf outperforms semantic similarity and Wikipedia outperforms other corpus.

References

[1] A. L. Berger, S. D. Pietra, and V. J. D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[2] E. Brill, J. J. Lin, M. Banko, S. T. Dumais, and A. Y. Ng. Data-intensive question answering. In *TREC*, 2001.

[3] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, and C. A. Welty. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79, 2010.

[4] A. Ittycheriah, M. Franz, and S. Roukos. Ibm's statistical question answering system - trec-10. In *TREC*, 2001.

[5] P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *ECML*, pages 491–502, 2001.