



SeaSpider: Automated information gathering on vessel movements in support of Maritime Domain Awareness

Serhan Tatar

David M.F. Chapman

Defence R&D Canada – Atlantic

Technical Memorandum
DRDC Atlantic TM 2007-294
December 2007

This page intentionally left blank.

SeaSpider: Automated information gathering on vessel movements in support of Maritime Domain Awareness

Serhan Tatar
David M.F. Chapman

Defence R&D Canada – Atlantic

Technical Memorandum

TM 2007-294

December 2007

Authors

Original signed by David M.F. Chapman

Serhan Tatar and David M.F. Chapman

Approved by

Original signed by James S. Kennedy

James S. Kennedy

Head, Maritime Information and Combat Systems Section

Approved for release by

Original signed by James L. Kennedy

James L. Kennedy

DRP Chair

© Her Majesty the Queen as represented by the Minister of National Defence, 2007

© Sa majesté la reine, représentée par le ministre de la Défense nationale, 2007

Abstract

SeaSpider is an R&D tool to investigate the development of a software agent that would aid an operator in gathering information about marine vessels from public sources of information published on the internet. This information would supplement sensor information used for Intelligence, Surveillance, and Reconnaissance (ISR) to enhance Maritime Domain Awareness (MDA) and to complete the Recognized Maritime Picture (RMP). Specifically, SeaSpider is fine-tuned to search for, organize, and display information about locations (ports), dates and times, and activities (arrival, in berth, departure). One module manages the Google searches and retrieves the html pages; another module extracts relevant ship movement information and populates a database; a third module retrieves information from the database in response to user-generated queries. In this memorandum, the SeaSpider concept is introduced, the design details of the prototype are presented, and performance is analysed, with a view to future enhancements.

Résumé

SeaSpider est un outil de R et D permettant de mettre au point un agent logiciel qui aiderait l'opérateur à recueillir de l'information sur la circulation maritime en puisant dans les informations publiques diffusées sur Internet. Cette information complètera l'information obtenue à l'aide des capteurs employés pour le RSR (renseignement, surveillance et reconnaissance) dans le but d'améliorer la connaissance du domaine maritime (CDM) et le tableau de la situation maritime (TSM). Plus particulièrement, SeaSpider est un outil de précision qui permet de chercher, d'organiser et d'afficher de l'information sur les emplacements (ports), les horaires (dates et heures), ainsi que les activités (arrivées, à quai, départs). Un des modules de cet outil gère les recherches effectuées à l'aide de Google et récupère les pages html; un autre module extrait l'information pertinente sur les déplacements des navires et enrichit une base de données; un troisième module enfin récupère l'information contenue dans les bases de données pour donner suite aux demandes générées par les utilisateurs. Dans le présent document, nous présentons le SeaSpider, fournissons les détails de conception du prototype et analysons sa performance, en laissant entrevoir d'éventuelles améliorations.

This page intentionally left blank.

Executive summary

SeaSpider: Automated information gathering on vessel movements in support of Maritime Domain Awareness

**Serhan Tatar and David M.F. Chapman; DRDC Atlantic TM 2007-294;
Defence R&D Canada–Atlantic.**

Background

SeaSpider is an R&D tool to investigate the development of a software agent that would aid an operator in gathering information about marine vessels from public sources on the internet. This information would supplement sensor information used for Intelligence, Surveillance, and Reconnaissance (ISR) to enhance Maritime Domain Awareness (MDA) and to complete the Recognized Maritime Picture (RMP).

Implementation

Specifically, SeaSpider is fine-tuned to search for, organize, and display information about locations (ports), dates and times, and activities (arrival, in berth, departure). One module manages the Google searches and retrieves the html pages; another module extracts relevant ship movement information and populates a database; a third module retrieves information from the database in response to user-generated queries. The search module uses an extensive list of keywords specific to marine vessel activity. The extraction module concentrates on recognizing tabular information and re-structuring it for automatic classification. The application runs on an ordinary PC with Windows operating system and access to the internet.

Results

The key innovation in SeaSpider is that the server application runs in the background, continually searches for relevant pages on the internet, extracts the information, and populates a database. When the user has a query, the response is very quick. The results can be imported into Google Earth as a geographic display. In the current (R&D) version the overall Precision is high (that is, the reported information has a high percentage accuracy) but the Recall is only fair (that is, too many relevant “hits” are not found in the first place).

Significance and Future Plans

The SeaSpider work has identified a successful methodology for automated search of unstructured information on web pages; this methodology could be applied to other domains of interest. Much has been learned about specific technical difficulties that stand in the way of automated information gathering (date/time conventions, name misspellings, poor data structure of web pages, and so on).

Before SeaSpider would be ready for operational use, a number of technical issues need to be resolved, for example: reliance on the Google search engine, “missing field” search algorithms, and a more dynamic keyword generator (based on result statistics). It is planned to continue this work in FY 08/09.

Sommaire

SeaSpider : Collecte d'information automatisée sur les mouvements de navires à l'appui de la connaissance du domaine maritime (CDM)

Serhan Tatar et David M.F. Chapman; RDDC Atlantique TM 2007-294; R & D pour la défense –Atlantique

Contexte

SeaSpider est un outil de R et D permettant de mettre au point un agent logiciel qui aiderait l'opérateur à recueillir de l'information concernant la circulation maritime en puisant dans les sources d'information publiques diffusées sur Internet. Cette information complètera l'information obtenue à l'aide des capteurs employés pour le RSR (Renseignement, Surveillance et Reconnaissance) dans le but d'améliorer la connaissance du domaine maritime (CDM) et le tableau de la situation maritime (TSM).

Mise en oeuvre

Plus particulièrement, SeaSpider est un outil de précision qui permet de chercher, d'organiser et d'afficher de l'information sur les emplacements (ports), sur les horaires (dates et heures), ainsi que sur les activités (arrivées, à quai, départs). Un des modules de cet outil gère les recherches effectuées à l'aide de Google et récupère les pages html; un autre module extrait l'information pertinente sur les déplacements des navires et enrichit une base de données; un troisième module enfin récupère l'information contenue dans les bases de données pour donner suite aux demandes générées par les utilisateurs. Le module de recherche comporte une liste élaborée des mots clés propres à la circulation maritime. Le module d'extraction sert essentiellement à reconnaître l'information se présentant sous forme de tableaux et à la restructurer en vue d'une classification automatique. L'application peut être exécutée sur un PC ordinaire possédant le système d'exploitation Windows ainsi qu'un accès Internet.

Résultats

L'innovation clé de SeaSpider est que l'application serveur est exécutée en arrière-plan, et cherche de manière continue les pages pertinentes sur Internet, en extrait l'information et enrichit une base de données. Lorsque l'utilisateur présente une demande, la réponse vient très rapidement. Les résultats peuvent être importés dans *Google Earth* et affichés à titre d'information géographique. Dans la version actuelle (R et D), la fonction *Precision* dans l'ensemble donne de bons résultats (en fait, l'information qui y figure possède un pourcentage de précision élevé), mais la fonction *Recall* est considérée comme étant « acceptable » seulement (un nombre suffisant de « résultats » pertinents n'est pas obtenu dès le premier essai).

Portée et recherches futures

Les travaux portant sur SeaSpider ont permis de mettre au point une méthode de recherche automatique d'information non structurée sur des pages Web; cette méthode pourrait être appliquée à d'autres domaines. On a appris beaucoup sur certaines difficultés techniques qui

entravent la collecte automatisée de l'information (conventions temporelles, mauvaise graphie des noms, structure quelconque des données dans les pages Web, etc. ...).

Avant que SeaSpider soit prêt pour une utilisation opérationnelle, un certain nombre de questions techniques doivent être résolues, notamment la fiabilité du moteur de recherche Google, la question des algorithmes de recherche « champ manquant » et celle du générateur de mots clés plus efficace (basé sur les statistiques de résultats). Nous prévoyons poursuivre ces travaux au cours de l'exercice financier 2008-2009.

Table of contents

Abstract/Resume.....	i
Executive Summary.....	iii
Sommaire.....	v
Table of contents	vi
List of figures	viii
List of tables	ix
Acknowledgements	x
1. Introduction	1
2. SeaSpider.....	2
2.1 SeaSpider- Retrieval Engine.....	4
2.1.1 Query Selection & Query Evaluation.....	6
2.1.2 Page Downloading	8
2.1.3 Result Filtering	8
2.2 SeaSpider- Extractor.....	8
2.2.1 Named Entity Recognition (NER).....	9
2.2.2 Column Recognition & Post-Processing	12
2.2.3 Relation Detection	13
2.2.4 Activity Extraction	16
2.3 SeaSpider- Client.....	17
3. Experimental Evaluation	19
3.1 Example Corpus	19
3.2 Methodology	19
3.3 Results	19
3.4 Discussion of Results	21
4. Conclusion & Future Work	22

References	23
List of Acronyms.....	24
Distribution List.....	25

List of figures

Figure 1. The general flow of SeaSpider	4
Figure 2. The Retrieval Engine.....	5
Figure 3. Timeline of the tasks performed by the Retrieval Engine.....	6
Figure 4. Statistical data collection.....	7
Figure 5. General flow of the tasks performed by the Extractor	9
Figure 6. An excerpt of an example output tree	10
Figure 7. An example itinerary table	11
Figure 8. Example column merge operations	14
Figure 9. A snapshot of the Query Interface.....	17
Figure 10. A snapshot of the Activity Result List	18
Figure 11. A vessel activity shown on Google Earth	18

List of tables

Table 1. Named Entity Classes in SeaSpider System.....	11
Table 2. Column Types defined in SeaSpider	13
Table 3. Field Types defined in SeaSpider	15
Table 4. Activity Extraction rules defined in SeaSpider	16
Table 5. Confusion matrix for result filtering.....	20
Table 6. Precision/Recall values for result filtering	20
Table 7. Confusion matrix for table recognition	20
Table 8. Precision/Recall values for table recognition	20
Table 9. Precision/Recall values for missing field search	21
Table 10. Precision/Recall values for activity extraction	21

Acknowledgements

Many thanks to Dr. Alain Auger of DRDC Valcartier for introducing us to the GATE open source software used in SeaSpider.

This work was supported by the Canadian Defence Research Fellowship Program 2006/7, which supported Lt. Tatar (Turkish Air Force) in this work.

1. Introduction

Although information seems to expand exponentially with time, the information world itself is becoming smaller and more connected, closing the gap between international and domestic security. We no longer feel secure at home from hostilities originating in far-away lands. In this unpredictable world, relevant information about intentions, capabilities, and activities of possible threats becomes paramount. Information superiority plays an important role in making decisions regarding the scope and design of security programs, the allocation of resources, and the deployment of assets [1].

The ability to distinguish the important from the irrelevant and friend from foe is necessary in order to counter security threats. However, when one considers the size of the area¹ to be kept secure, one realizes that significant resources and effort must be committed to gain and maintain this ability [2].

Information filtering and management is among the major goals of C4ISR (Command, Control, Communications, Computers, Intelligence, Surveillance & Reconnaissance). Technology continually provides new systems to decision makers. For example, Automatic Identification System (AIS), a self-reporting navigation and communications system for ships at sea, has been adopted as an information source for Maritime Domain Awareness (MDA). What kind of significance AIS holds for the construction of the Recognized Maritime Picture (RMP), how it might be used for Marine Intelligence, Surveillance, and Reconnaissance (MISR), and what research activities might be conducted in support of AIS for MISR were discussed in [3].

Besides existing sensors and systems, public information located on the World Wide Web (WWW) carries some potential for MDA. Although the Internet is used by many individuals for information-gathering, its potential for day-to-day MDA has not been completely explored, perhaps because of the heterogeneous nature of the Internet or perhaps because of poorly structured data. We believe that WWW utility can be improved by providing software aids to operators.

The SeaSpider project is an attempt to use information sources found on the Internet in order to improve MDA capability. How can we reduce operator overload? How can we make the RMP more meaningful? How can we use public information on the Internet more effectively? Our investigation addressed these questions, identified some implementation challenges, and opened the doors for similar applications.

This memorandum describes the SeaSpider effort. We present the design of the SeaSpider application prototype, some implementation details, measures of performance, and our conclusions, pointing towards future research. The structure of the report is as follows: Section 2 describes the project details: objectives of the project, desired functions, and the details of the solution model with the detailed explanation of each module. Section 3 describes the experimental evaluation of the study. Finally, in the last section we present our conclusions.

¹ For example, it is reported in [2] that there are around 1700 ships in Canada's Atlantic, Pacific and Arctic areas of responsibility on a typical day (even more, if we consider the non-reported contacts further away from the major regulated ports or vessel traffic management systems).

2. SeaSpider

The main objective of the SeaSpider project is to enable automated gathering of public information about vessel movements (ports, arrival & departure times etc.). In this way, it aims to provide comprehensive assistance to ISR operators. Making the RMP more meaningful by supplying past and future information about vessels is another goal of the project.

The project combines several research fields to collect information automatically from the Web. The system focuses on accuracy, speed, and usability. The key requirement is to be able to provide accurate information, with a good presentation, in a reasonable time. Furthermore, it proposes various useful functions:

- Accepting user queries with different search criteria,
- Collecting relevant information about vessels of interest (VOIs) from the Web,
- Extracting relevant information from unstructured documents and HTML pages: ship names, dates, places, activities etc.
- Managing the queries and the summarized information,
- Presenting a timeline of ship motion from the past, through the present, into the future.

SeaSpider was first conceived as an operator aid in September, 2004. The first stone was laid in January, 2005. The work continued during summer, 2005. The result was a basic application capable of sending user queries to the Google search engine (www.google.ca) and of extracting common named entities from the returned result pages by using open-source GATE (General Architecture for Text Engineering) software [4]. However, the developed application was not adequate to fulfill all of the requirements. After summer 2005, work on the project was ceased. Finally, the project was restarted in September, 2006 after a year's break.

During the review of the SeaSpider concept in September, 2006, some deficiencies of the previous approach were recognized. The main problem was searching the WWW only after the user enters a query. This strategy is inefficient, as the elapsed time before response to the query is far too long. Even if the application had used a general-purpose search engine (Google, in our case), it would have taken an extremely long time to distinguish relevant pages², extract the relevant information located in the retrieved documents, and transform the extracted information for presentation. It was decided that the information needed to be found, extracted, and stored *before* the query. The user would be then able to readily access relevant information.

² It may seem unnecessary to re-rank the search engine results. However, the search engine may rank your query results higher in domains other than your actual domain of interest. For example, if you want to find information about port arrivals and departures of ship "Holiday" and if you search Google with the keywords "Holiday ship arrival departure", the top-rank results will probably not be what you are looking for.

As mentioned earlier, the SeaSpider concept involves research challenges in several different fields. The significant challenges are related to information retrieval (IR) and information extraction (IE).

The system has to be capable of finding and extracting relevant information from the WWW. However, there are not many relevant and comprehensive information sources on the Web. In fact, there are a few commercial sources that include considerable information, but they are not open to the public and require a prohibitive license fee. Instead, there are many sites that contain limited amounts of information. SeaSpider was conceived to find, organize, and present this information.

Moreover, web pages are designed for human beings. Typically, they are not structured for automatic information extraction. HTML, the language used for WWW documents, is useful in formatting web information for presentation; however, HTML is not optimal for conveying the meaning of the information. Hence, SeaSpider needs to cope with unstructured documents written in natural language. Furthermore, it should be able to handle ungrammatical sentences, non-unique ship names, variant spellings, and misspellings. Another issue in information extraction is that the relevant information on the Web pages may not be located in sentences. Instead, it may be found in tabular form. This makes applying classical natural language processing (NLP) methods more difficult.

Even if the SeaSpider performs the information extraction task with high accuracy, the extracted information may need additional processing. There are a multitude of date/time conventions and time zone issues. In addition, location identification and the mapping of locations into latitude/longitude coordinates are among the issues waiting to be addressed.

The general flow of SeaSpider is illustrated in Figure 1. The model consists of three principal components that are responsible for performing five main tasks:

- The *SeaSpider-Retrieval Engine* handles Information Retrieval task.
- The *SeaSpider-Extractor* is responsible for Information Extraction and Storing Integrated Information into the SeaSpider Database.
- The *SeaSpider-Client* is a user tool for Answering User Queries.

In order to answer user queries effectively, SeaSpider uses continuous processes to store relevant information into the database in advance. The retrieved, extracted, and processed information is inserted into the SeaSpider database before user search. The client-side application, SeaSpider-Client, sends user queries to the server and presents the returned results to the user in appropriate format. On the other side, the server continuously searches, extracts and stores relevant information on vessel movements.

The structural details and the design rationale of each component in the model are described in the following sections.

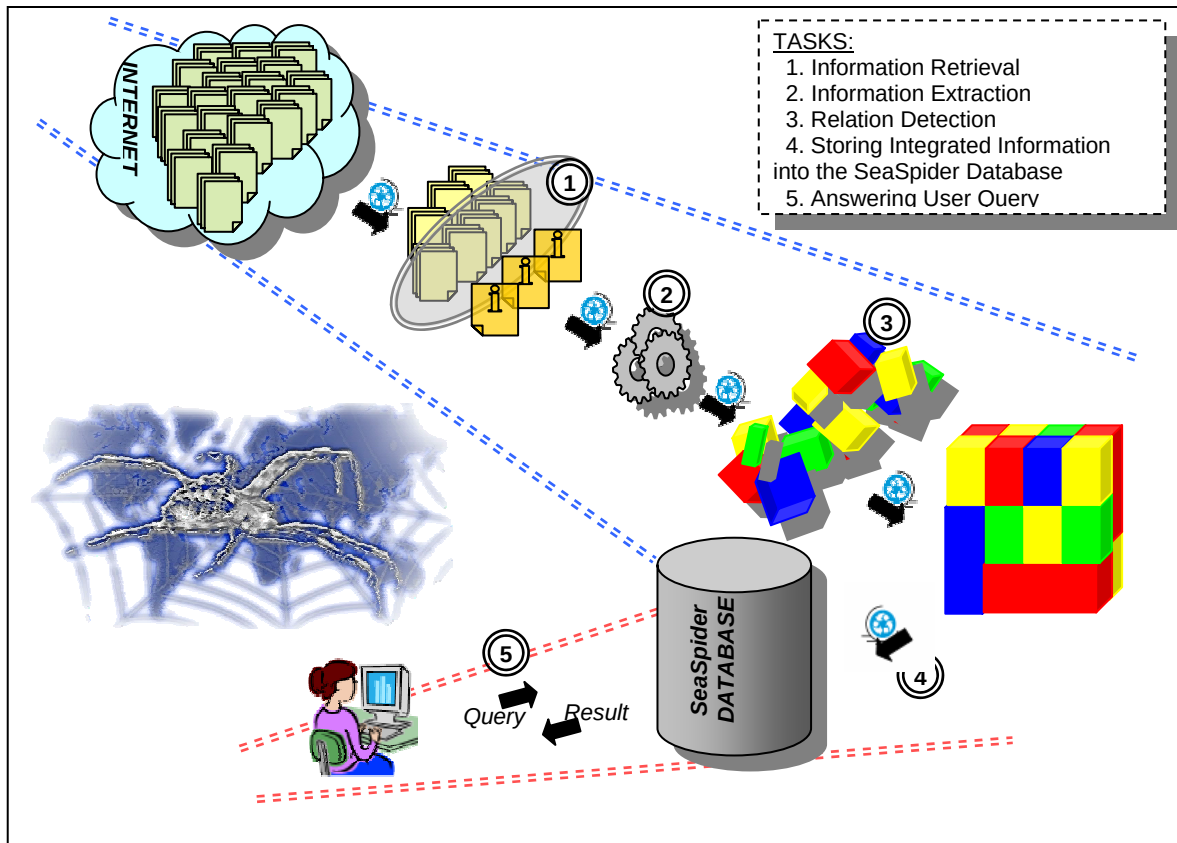


Figure 1. The general flow of SeaSpider. Arrows with wheels denote continuous processes.

2.1 SeaSpider- Retrieval Engine

The Retrieval Engine is the module that is responsible for searching the WWW for relevant information sources, fetching relevant web pages, and passing them to the Extractor module. For a system like SeaSpider, there are several design alternatives for the Retrieval Engine. The first and the simplest method is to use an expandable list of selected pages or documents. Secondly, it is possible to use a general purpose search engine (e.g., Google) for the same purpose by making well-designed search requests. The third option may be developing a focused crawler which is capable of finding information sources related to the domain of interest by crawling the WWW. Although the last option seems to be the one with the best performance, it is also the one which is the hardest to implement and the most expensive to maintain. The advantage of having one's own crawler which selects the pages according to one's own criteria can not be ignored, but by the same token we can not neglect the development and management difficulties of running a focused crawler. On the other hand, using an expandable list of selected pages is simple and easy. A URL list can be constructed from the known relevant information sources. In this case, the main problem is how to update the list. If there were a small number of comprehensive sources³, a manual method of

³ For example, national newspapers with more than certain circulation size could be sufficient for a news-collecting agent. In that case, the number of information sources would be manageable.

updating the list would be manageable. Since we expect a very large and dynamic number of relevant URLs, we consider this option inapplicable in our case.

Before giving the details of the Retrieval Engine, it is useful to give definitions of different structures in the system and the relations between them:

- *Keyword*: SeaSpider has a predefined set of keywords which are used to construct queries. One keyword can be used in many queries.
- *Query*: Queries consist of one or more keywords and are used to search Google. For Google, keyword order in queries is significant. The same queries with different word order may give different results.
- *Result*: Results are the URLs returned by Google search engine in response to queries. Results may contain different information at different times according to update period of the page. One query can return many results.
- *Result Snapshot*: A view of an actual web page associated with a given result at a given instant is called a result snapshot. Accordingly, one result can have many result snapshots.

In our design, the Retrieval Engine uses Google for searching the WWW for relevant information sources. It prepares queries that include keywords from a predefined set. Then, the prepared queries are sent to the search engine using Google Search API [5]. Finally, the returned results serve as input to the Extractor. In this way, it is not needed to develop and

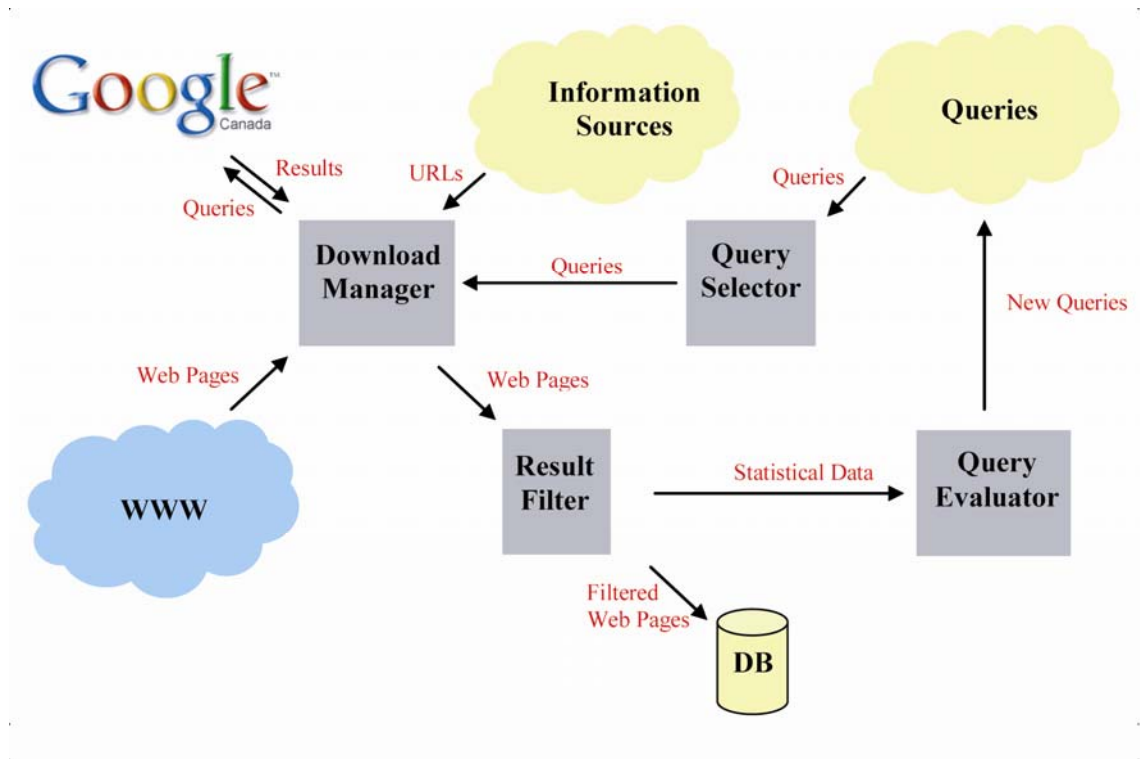


Figure 2: The Retrieval Engine

maintain a crawler to automatically search a large number of information sources. In addition to using Google for searching for relevant information sources, SeaSpider is able to scan a list of user-specified information sources regularly.

As illustrated in Figure 2, the Retrieval Engine has four main sub-modules: Query Selector, Download Manager, Result Filter, and Query Evaluator.

The Query Selector is responsible for selecting the queries with high scores from the query pool and sending selected queries to the Download Manager. The Download Manager sends these queries to Google and gets result pages. After getting result pages, the Download Manager downloads both Google-returned results and user-specified information sources. Downloaded Web pages are sent to the Result Filter. The Result Filter processes downloaded pages and discriminates relevant pages from non-relevant ones according to their context. After result filtering is finished, the Query Evaluator re-evaluates the queries and constructs new queries. A timeline of the tasks performed by the Retrieval Engine is shown in Figure 3.

2.1.1 Query Selection & Query Evaluation

As shown in Figure 2, the Query Selector selects a certain number of queries from a query pool. Each query is assigned a value that indicates its relative relevancy factor. The Query Selector simply selects the queries with high relevancy factors. The Query Evaluator is the sub-module that assigns relevancy factors to the queries. Moreover, it creates new queries and adds them to the query pool by using statistical methods.

The query evaluator performs a query evaluation task in two steps: statistical data collection and query generation. In the first step, it assigns different relevancy factors to the above structures. A result snapshot can be either relevant or irrelevant. The result filter decides whether a given result snapshot is relevant or irrelevant. The ratio of relevant result snapshots to total result snapshots for a result gives the relevancy factor of that result. In the same way, the ratio of relevant result snapshots to total result snapshots for a query gives the relevancy

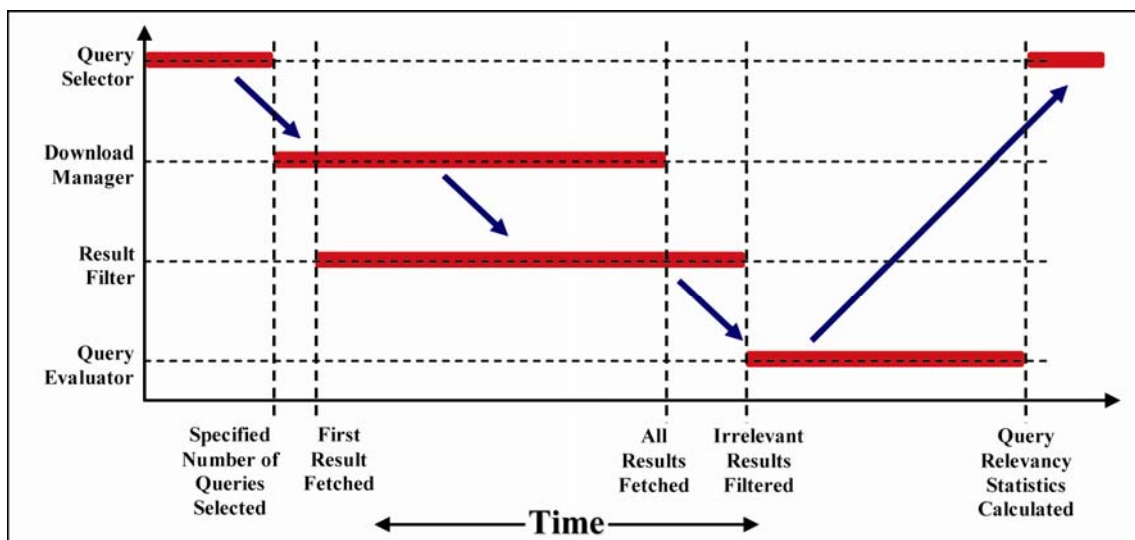


Figure 3: Timeline of the tasks performed by the Retrieval Engine

factor of that query. Finally, the ratio of relevant result snapshots to total result snapshots for a keyword gives the relevancy factor of that keyword. The statistical data collection task is shown in Figure 4.

The query evaluator assigns relevancy factors to the available queries in the system during statistical data collection. The other important task of the query evaluator in the system is query construction. SeaSpider uses a bigram model, a special case of N-gram which is used in various areas of statistical natural language processing, to construct new queries.

Statistical language models date back to Shannon's work on information theory [6]. One of the basic aims of the statistical language models is to predict the probability of the next word, given the previous word sequence:

$$P(w_n|w_1, \dots, w_{n-1})$$

However, there is no easy way to compute the probability of a word given a long sequence of word history. It is not practical to keep the possibility of each word sequence, as this would imply very large sample space. Moreover, when we consider the phrases or sentences with arbitrarily length, it is not possible to observe most of the sequences during training of the language model (since we would need far too large a corpus). This will lead to the data sparseness problem of overfitting. For that reason, we group word sequences according to their last $n-1$ words to obtain an n -gram language model.

An n -gram of size 2 is called bigram. Given a bigram language model, it is straightforward to compute the probability of a word sequence as follows [7]:

$$P(w_1, w_2, w_3, \dots, w_n) = P(w_1) P(w_2|w_1) P(w_3|w_2) \dots P(w_n|w_{n-1})$$

For example, the probability of the query “Ship Arrival Departure” is computed as follows in our model:

$$P(\text{Ship Arrival Departure}) = P(\text{Ship}) P(\text{Arrival} | \text{Ship}) P(\text{Departure} | \text{Arrival})$$

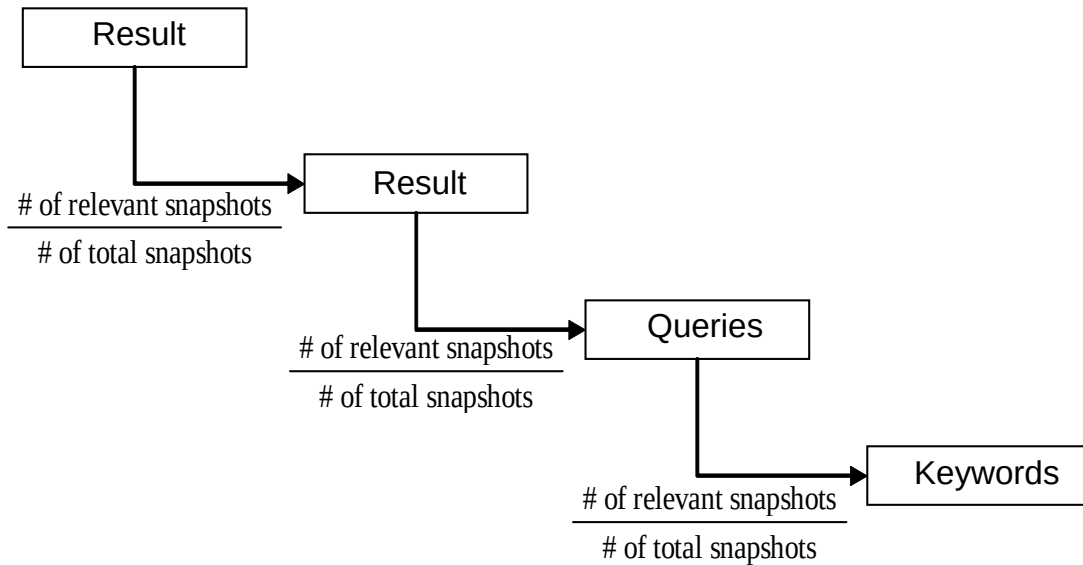


Figure 4. Statistical data collection

We approach the query construction problem as finding the most likely sequence of words. For finding the most likely keyword sequences, we used a dynamic programming technique: the Viterbi algorithm [8, 9]. Each keyword is treated as a state. Moreover, each state can only emit one value which is the keyword value of that state. After applying the Viterbi algorithm to this model, a certain number of the most likely keyword sequences are selected as new queries.

2.1.2 Page Downloading

The page downloading task is performed by a multithreaded download manager. The download manager downloads result pages returned by the Google search engine in response to sent queries and information sources specified by the users. The download manager does not download pages which cannot be parsed by SeaSpider. It simply eliminates files with certain extensions, such as ".pdf", ".doc", ".xls", ".ppt", ".avi" and so on. Moreover, it eliminates non-HTML files. In order for the download manager to download a web document, the document has to contain at least one of the following HTML tags: "<script>", "<html>", "<body>", "<head>", "<title>", "<table>", "".

2.1.3 Result Filtering

The objective of the result filtering is to filter out irrelevant pages (pages which do not contain vessel movement information). Although well-constructed queries filter out most of the irrelevant pages, they may not eliminate all of them. For this reason, the system has to process returned result pages and discriminate relevant pages from non-relevant ones. There are several methods for classifying retrieved documents based on their relevancy. One possible method is to discriminate relevant documents from non-relevant ones according to their context. In this method, the system scores the returned results by using a *term frequency-inverse document frequency* (TF/IDF) weighting scheme. It simply measures the relevancy of the returned result page. If the score given to the page exceeds a certain threshold, the system decides that the page is relevant and sends it to the Extractor. The second alternative is to classify documents according to the frequency of certain word categories (e.g., port names, vessel names, date/time values etc.). During our experiments, neither method provided satisfactory results.

Our observations showed that most of the vessel movement information is found in tabular form. Vessel activities are usually embedded in itinerary tables. From this observation, our approach is to categorize documents into two groups: relevant documents include an itinerary table and irrelevant documents do not include an itinerary table. An itinerary table can be defined as an HTML table with a vessel column or date/time and port columns. Therefore, the result filtering problem is reduced to an itinerary table detection problem. Named entity recognition (NER) and column recognition are two sequential steps in the table detection process. These two steps are also the first two steps of the extraction process. The details of these two tasks are given in section 2.2.1 and 2.2.2.

2.2 SeaSpider- Extractor

The Extractor module is responsible for extracting information existing in the filtered pages. After finding the relevant pages, the Retrieval Engine passes the found pages to the Extractor.

It is the Extractor which locates the relevant information and extracts it from the found pages. From this point of view, the Extractor's task can be seen as a slot-filling problem: filling the fields of a predefined target template. After the information is extracted and become ready for use, it is stored to the target database. The target database is a relational database with tables of ships, ports, countries, cities, and ship movements.

The relevant information which SeaSpider needs is generally found in tabular form on the Internet. The problem with tabular representation is the lack of sentence structure. Therefore, some classical analysis methods used for natural language understanding may not be very helpful in the case of tabular data. On the other hand, there are some features specific to tabular data from which we can benefit. For example, HTML tags in the source WEB pages may be clues of the hidden information in the pages. HTML tags will be more evident and useful in tabular data because of the certain syntax of HTML table structures (<table> <tr> <td> </td> ... </tr> ... </table>).

Processing in this module includes named entity recognition, column recognition, relation detection, date/time conversions, entity identification, and integration of the information pieces coming from different sources. The general flow of the tasks performed by the Extractor is shown in Figure 5.

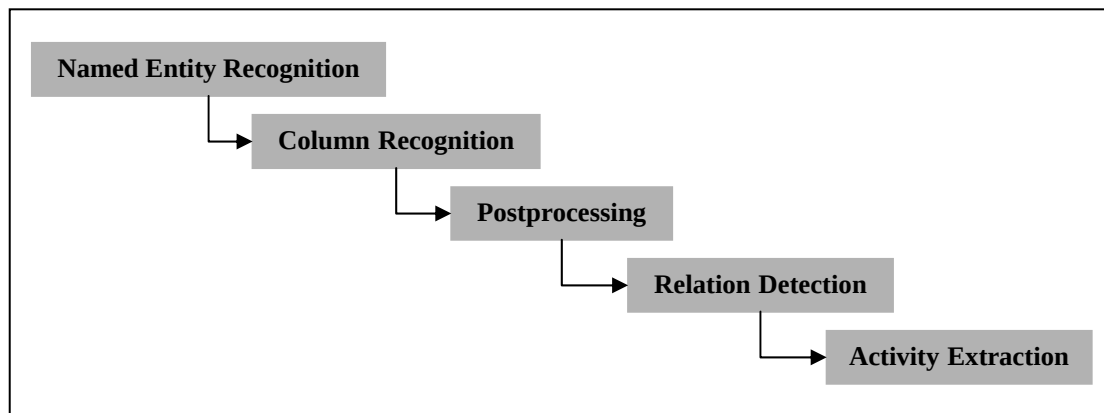


Figure 5. General flow of the tasks performed by the Extractor

2.2.1 Named Entity Recognition (NER)

SeaSpider starts the extraction process off by recognizing named entities. Named entities are predefined entity categories such as the names of persons, organizations, locations and so on. The Named Entity Recognizer (NER) sub-module in SeaSpider takes as input web documents in HTML and outputs found named entities stored in an HTML element tree.

For NER, we used the open-source software General Architecture for Text Engineering (GATE). GATE provides several facilities for developing, evaluating, and embedding Human Language Technology. It can be used as an architecture, framework, and development environment. It is able to handle common natural language tasks (e.g., sentence splitting, POS tagging etc.). The extraction rules for the common named entities (e.g., person or location) are defined in the system. Adding new target entities and defining rules for them are also supported in GATE. Moreover, applications developed within GATE can be deployed outside its Graphical User Interface (GUI), using programmatic access via the GATE API.

For this sub-module, we extended the rules and gazetteer lists defined in GATE. Moreover, new named entity classes and rules which are very special for our domain were defined. The complete list of the named entities used for vessel activity extraction is shown in Table 1.

SeaSpider also makes use of the HTML tag hierarchy embedded in the document to extract target information in the document. Each HTML element has a matching class in SeaSpider. Furthermore, there are some utility classes defined in the system. As mentioned earlier, the output of the NER is a special tree structure which stores named entity information and the matched HTML tags while keeping the HTML tag hierarchy at the same time. An excerpt of an example output tree is shown in Figure 6. A word or phrase in the document can correspond to more than one named entity class, as it can be seen in Figure 7. For instance, the word “Halifax” in Figure 7 can be both a port name and a ship name. The NER does not try to resolve that kind of ambiguity at this level. Final class labeling is performed during post-processing.

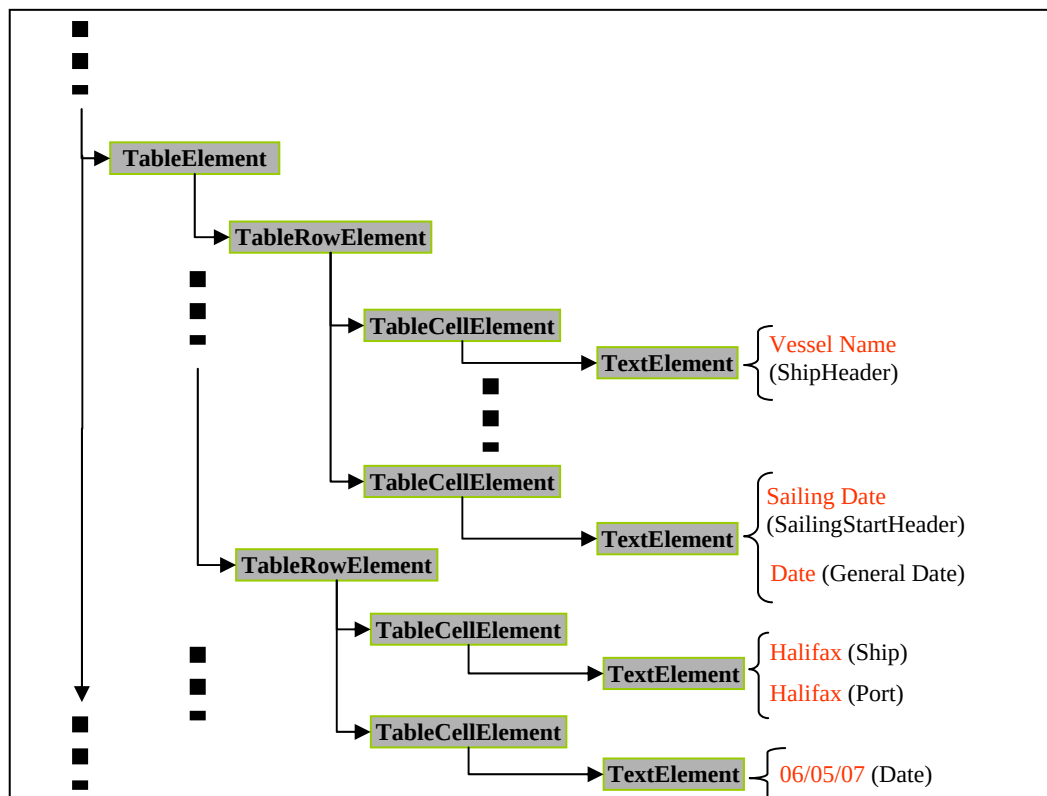


Figure 6. An excerpt of an example output tree

Table 1. Named Entity Classes in SeaSpider

NAMED ENTITY CLASS	DEFINITION
ArrivalDateHeader	Arrival Date column headers (e.g. Arrival Date)
ArrivalDepartureDateHeader	Arrival/Departure Date column headers (e.g. Arrival-Departure Date)
ArrivalPortHeader	Arrival Port column headers (e.g. Port of Arrival)
Date	Date expressions (e.g. 23.05.2006)
DateTime	Date & Time expressions (e.g. 23.05.2006 14:00)
DateTimeInterval	Date & Time interval expressions (e.g. 23.05.2006 – 06.06.2006)
DepartureDateHeader	Departure Date column headers (e.g. Depart Time)
DeparturePortHeader	Departure Port column headers (e.g. Departs From)
GeneralDateHeader	General (Not Arrival or Departure) Date column headers (e.g. Date)
GeneralPortHeader	General Port column headers (e.g. Port)
IgnoreElement	Domain dependent expressions to ignore (e.g.)
IrrelevantHeader	Irrelevant column headers (e.g. Availability)
ItineraryDay	Days in itineraries (e.g. Day 1, Day 2)
Location	Location expressions (e.g. London, UK)
PortCandidate	Expressions which can be a port name (e.g. Halifax)
PortName	Port names (e.g. Toronto)
SailingEndHeader	Expressions which labels an end date of a sailing (e.g. Return Date)
SailingPeriodHeader	Expressions which labels a start date and an end date of a sailing (e.g. Sailing Date)
SailingStartHeader	Expressions which labels a start date of a sailing (e.g. Sailing Date)
ShipCandidate	Expressions which can be a ship name (e.g. Halifax)
ShipHeader	Ship Headers (e.g. Ship Name)
ShipName	Ship Names (e.g. Queen Mary 2)
Time	Time expressions (e.g. 10:30 am)

2.2.2 Column Recognition & Post-Processing

It is essential to recognize itinerary tables to extract relevant vessel activity information from them. However, most tables are designed for human perception, so their layout and semantic meanings are not well-defined in the context of machine interpretation. To interpret these complex yet unstructured tables, SeaSpider needs a procedure for reading them.

SeaSpider performs the table recognition process in two steps. The first step is called column recognition. The column recognizer (the sub-module responsible for column recognition) first tries to reduce table complexity. After table complexity reduction, the system recognizes the table column types. We use majority voting for the column type recognition task. Moreover, to label a column with a specific column type at least one header cell with that specific column type must be found in the column. Headers and some unnecessary cell values (text elements labeled as “IgnoreElement” during NER) are not included in voting. An example itinerary table is shown in Figure 7. The first column in the example is found to be of type “ITINERARY_DAY”. The second column is type “GENERAL_PORT”. While analyzing the second column, the column recognizer does not take the cell values “At Sea” into account. Because they were labeled as “IgnoreElement” during NER, these cells are not included in voting. In the same way, the cell values “---” in the third and fourth column are also not included in voting. The third column is labeled as an “ARRIVAL_TIME” column. The last column is of type “DEPARTURE_TIME”. All column types are shown in Table 2.

The second step in the table recognition process is called postprocessing. The main purpose of postprocessing is to tune up the column recognition task. In the design of the system, the output of a component is input to the next component. From this point of view, an incorrect recognition of a named entity during the NER phase may affect the success of the activity extraction task negatively. Postprocessing is the second and last chance to recover from the errors made at lower levels. For instance, the NER sub-module sometimes fails to find all port names in the document. In such a situation, if the column recognizer marks a column as “GENERAL_PORT” and there is a cell value in the column whose named entity category is neither “PortName” nor “IgnoreElement”, then that cell value is marked as “PortName” during postprocessing. Moreover, some long sentences which can be found in itinerary tables are eliminated during postprocessing. After these, header cells and ignored cells are relabeled. The last step in the postprocessing task is to reassess the relevancy of the table. According to newly assigned labels, itinerary table candidates are re-evaluated. Tables marked as irrelevant are eliminated at this level.

DAY	PORT	ARRIVAL	DEPARTURE
1	Port Canaveral, Florida	---	4:00 p.m.
2	At Sea	---	---
3	Key West, Florida	7:00 a.m.	1:00 p.m.
4	Cozumel, Mexico	11:00 a.m.	7:00 p.m.
5	At Sea	---	---
6	Port Canaveral, Florida	7:00 a.m.	---

Figure 7. An example itinerary table

Table 2. Column Types defined in SeaSpider

COLUMN TYPE	NE CLASS OF NECESSARY HEADER	NE CLASS OF SUITABLE CELL VALUES
ARRIVAL_DATE	ArrivalDateHeader	Date
ARRIVAL_DATETIME	ArrivalDateHeader	DateTime
ARRIVAL_DEPARTURE_DATE	ArrivalDepartureDateHeader; ArrivalDateHeader+DepartureDateHeader	Date
ARRIVAL_DEPARTURE_DATETIME	ArrivalDepartureDateHeader; ArrivalDateHeader+DepartureDateHeader	DateTime
ARRIVAL_DEPARTURE_TIME	ArrivalDepartureDateHeader; ArrivalDateHeader+DepartureDateHeader	Time
ARRIVAL_PORT	ArrivalPortHeader	PortName
ARRIVAL_TIME	ArrivalDateHeader	Time
DEPARTURE_DATE	DepartureDateHeader	Date
DEPARTURE_DATETIME	DepartureDateHeader	DateTime
DEPARTURE_PORT	DeparturePortHeader	PortName
DEPARTURE_TIME	DepartureDateHeader	Time
GENERAL_DATE	GeneralDateHeader	Date
GENERAL_DATETIME	GeneralDateHeader	DateTime
GENERAL_PORT	GeneralPortHeader	PortName
ITINERARY_DAY		ItineraryDay
SHIP	ShipHeader	ShipName

2.2.3 Relation Detection

Itinerary tables detected by the system do not always contain complete information necessary to fill activity templates. For instance, the itinerary table shown in Figure 7 does not have a vessel name column. Moreover, the first column in the table has to be mapped to date values. Information distributed to multiple columns is another issue to be addressed. An itinerary table may contain two different columns which have to be combined. All these issues are handled by the Relation Detector. The raw information pieces are mapped and integrated in this phase. Furthermore, processing in this module includes date/time conversions, and integration of the information pieces coming from different sources.

Relation detection is the most crucial phase in the extraction process as it shapes the future course of the process. The Relation Detector takes as input itinerary tables with column types labeled and outputs field sets. As stated above, we treat the vessel activity extraction problem as a slot filling problem: filling the fields of a predefined target template. Field sets are

constructed during the relation detection process. The actual field filling process is performed at this level. What is done in higher levels is the classification of the field sets and extraction of the activities.

As mentioned earlier, one of the main tasks of the Relation Detector is to find missing fields in the itinerary tables. In a field set, there are three compulsory fields: a date field, a port field, and a ship field. Therefore, the system performs a missing field search, if it detects one of these fields are missing. This task also includes itinerary day mapping. We used different approaches for different tasks. For itinerary day mapping, we followed a rule-based approach. Moreover, for searching missing date and port fields, the best results are obtained by using rule-based fields. On the other hand, we use a statistical approach to search for missing ship names. Probability of being a ship name, distance from a ship header, and distance from the itinerary table are the features used while searching for missing ship names.

Relations between the different columns of the itinerary tables are also detected during the relation detection phase. We defined rules to find and combine related fields to form new columns. An example case is illustrated in Figure 8. In the first merge operation, the first column, of “GENERAL_DATE” type, and the fifth column, of “ARRIVAL_TIME” type, are combined to form a new column which is of “ARRIVAL_DATETIME” type. In the second merge operation, a new column which is of “DEPARTURE_DATETIME” type is constructed by combining a column of “GENERAL_DATE” type and a column of “DEPARTURE_TIME”. All merge operations in the system are performed according to the column combination rules defined in the system.

The last and the decisive step in the relation detection process is to convert columns to fields. All operations (mappings, merges etc.) until this point are performed over columns. However, from this point further, it is necessary to obtain final fields to construct output field sets. The conversion operation is performed according to conversion rules defined in the system. The complete list of the target fields is given in Table 3.

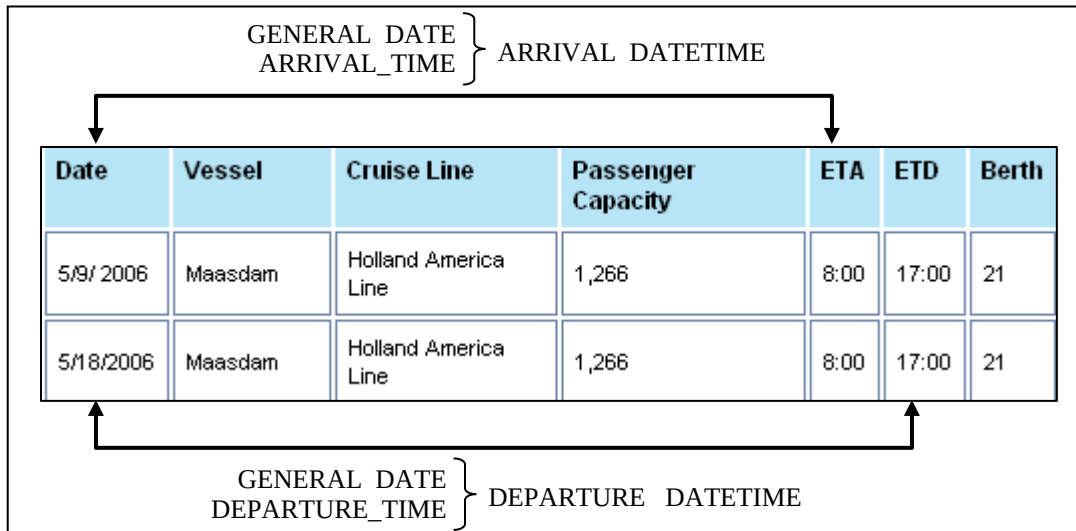


Figure 8. Example column merge operations

Table 3. Field Types defined in SeaSpider

FIELD TYPE	COLUMN TYPES TO FORM
ARRIVAL_DATETIME	ARRIVAL_DATE; ARRIVAL_DATE + ARRIVAL_TIME; GENERAL_DATE + ARRIVAL_TIME; ARRIVAL_DATETIME; ITINERARY_DAY + ARRIVAL_TIME; ARRIVAL_DEPARTURE_DATETIME; ARRIVAL_DEPARTURE_DATE + ARRIVAL_TIME; ARRIVAL_DEPARTURE_DATE + ARRIVAL_DEPARTURE_TIME;
ARRIVAL_PORT	ARRIVAL_PORT
DEPARTURE_DATETIME	DEPARTURE_DATE; DEPARTURE_DATE + DEPARTURE_TIME; GENERAL_DATE + DEPARTURE_TIME; DEPARTURE_DATETIME; ITINERARY_DAY + DEPARTURE_TIME; ARRIVAL_DEPARTURE_DATETIME; ARRIVAL_DEPARTURE_DATE + DEPARTURE_TIME; ARRIVAL_DEPARTURE_DATE + ARRIVAL_DEPARTURE_TIME;
DEPARTURE_PORT	DEPARTURE_PORT
GENERAL_DATETIME	GENERAL_DATETIME; ITINERARY_DAY; GENERAL_DATE;
GENERAL_PORT	GENERAL_PORT
SHIP	SHIP

2.2.4 Activity Extraction

Activity extraction is the last step in the process of extraction. Input field sets are processed and converted to vessel activities during this phase. For conversion, at least a Ship field, a Port field and a DateTime field are necessary. Field Sets are converted to activities according to activity extraction rules. There are three different types of activities defined in the system: arrival, departure, and uncategorized. As a result of applying conversion rules, the Activity Extractor sub-module outputs classified activities and inserts extracted activities into the SeaSpider database. Activity extraction rules defined in the system is shown in Table 4.

Table 4. Activity Extraction rules defined in SeaSpider

RULES (IN THE ORDER OF PRIORITY)	ACTIVITIES
SHIP + ARRIVAL_DATETIME + DEPARTURE_DATETIME + ARRIVAL_PORT + DEPARTURE_PORT	ARRIVAL + DEPARTURE
SHIP + ARRIVAL_DATETIME + DEPARTURE_DATETIME + ARRIVAL_PORT	ARRIVAL
SHIP + ARRIVAL_DATETIME + DEPARTURE_DATETIME + DEPARTURE_PORT	DEPARTURE
SHIP + ARRIVAL_DATETIME + DEPARTURE_DATETIME + GENERAL_PORT	ARRIVAL + DEPARTURE
SHIP + ARRIVAL_DATETIME + ARRIVAL_PORT + DEPARTURE_PORT	ARRIVAL
SHIP + ARRIVAL_DATETIME + ARRIVAL_PORT	ARRIVAL
SHIP + ARRIVAL_DATETIME + GENERAL_PORT	ARRIVAL
SHIP + DEPARTURE_DATETIME + ARRIVAL_PORT + DEPARTURE_PORT	DEPARTURE
SHIP + DEPARTURE_DATETIME + DEPARTURE_PORT	DEPARTURE
SHIP + DEPARTURE_DATETIME + GENERAL_PORT	DEPARTURE
SHIP + GENERAL_DATETIME + ARRIVAL_PORT + DEPARTURE_PORT	ARRIVAL + DEPARTURE
SHIP + GENERAL_DATETIME + ARRIVAL_PORT	ARRIVAL
SHIP + GENERAL_DATETIME + DEPARTURE_PORT	DEPARTURE
SHIP + GENERAL_DATETIME + GENERAL_PORT	UNCATEGORIZED

2.3 SeaSpider- Client

The SeaSpider-Client is a client-side application which sends user queries to the database and presents the returned results to the user in an appropriate format. The principal function of this application is to answer the user queries. It enables users to query the SeaSpider database for vessel activities. Moreover, it can export data into Google Earth and enables users to display vessel activities on Google Earth. Another function supported in the client application is to access the result snapshots stored in the SeaSpider database. The ability of accessing the source of each individual vessel activity is extremely relevant to operators. Furthermore, it is possible to make a keyword search on the stored documents.

A snapshot of SeaSpider-Client's query interface is shown in Figure 9. A user can prepare a query by filling out the query form and restrict his/her search by entering/changing the values of the fields found in the form. Figure 10 shows how the vessel activities found for the search are presented by the SeaSpider-Client. The user can select several activities and export them into Google Earth. Figure 11 shows how vessel activities are shown on Google Earth.

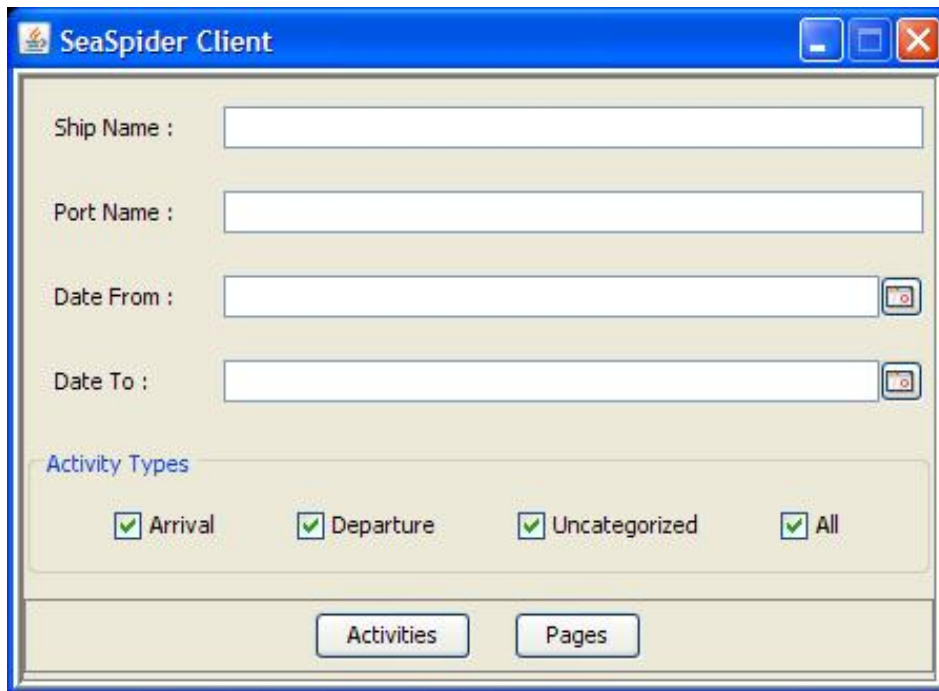


Figure 9. A snapshot of the Query Interface

Search Results - Activities					
☐ Select All		7809 records found...			Export Selected
	Type	Ship Name	Port Name	Date	Time
☐	A	YATCH COSMOS	Jebel Ali	2006-04-12	09:00:00
☐	A	SUPER USA	Jebel Ali	2006-04-15	08:00:00
☐	A	Maasdam	Halifax	2006-05-13	08:00:00
☐	D	Maasdam	Halifax	2006-05-13	17:00:00
☐	D	SAGA RUBY	Zeebrugge	2006-05-19	18:00:00
☐	A	MV LUCIEN GA	Mundra	2006-06-05	00:00:00
☐	A	MY DUBAI	Jebel Ali	2006-07-01	00:01:00
☒	A	Grandeur of the Seas	Halifax	2006-07-04	08:00:00
☐	D	Grandeur of the Seas	Halifax	2006-07-04	18:00:00
☐	A	Maasdam	Halifax	2006-07-11	08:00:00
☐	D	Maasdam	Halifax	2006-07-11	17:00:00
☐	A	MARLIN	Amsterdam	2006-07-19	13:00:00
☐	D	NANCOWRY	Chennai	2006-08-04	
☐	A	SWARAJ DWEEP	Kolkata	2006-08-05	
☐	A	NANCOWRY	Port Blair	2006-08-07	
☐	D	SWARAJ DWEEP	Kolkata	2006-08-08	

Figure 10. A snapshot of the Activity Result List

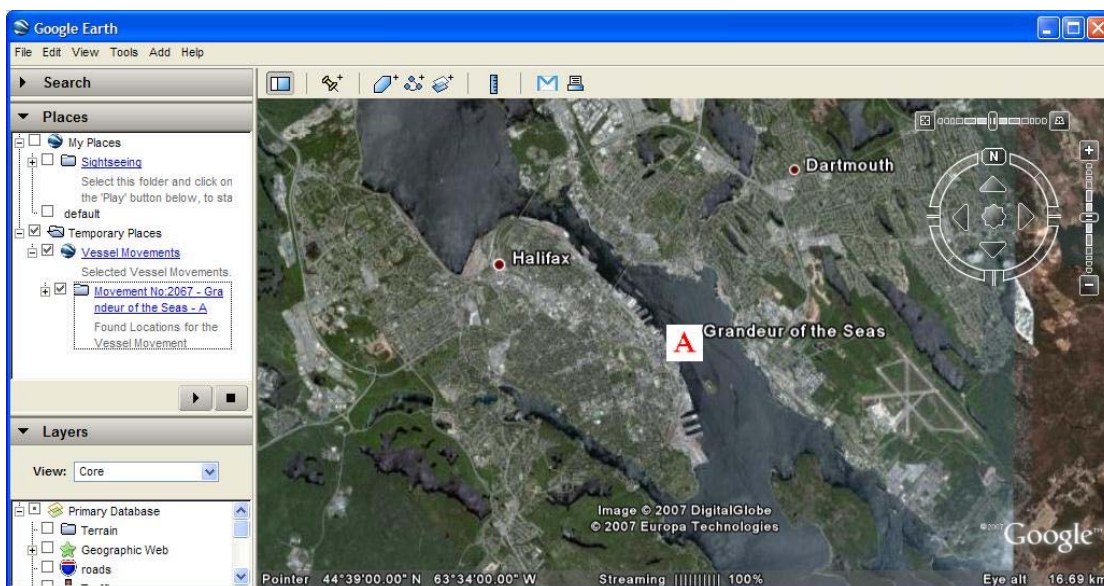


Figure 11. A typical vessel activity shown on Google Eart: A = arrival.

3. Experimental Evaluation

3.1 Example Corpus

In this section, we present how our model performs different tasks in terms of precision and recall. We begin this section by explaining the corpus used in our experiments. In order to conduct experiments, we built a corpus which contains 496 documents. 113 documents in the set are relevant and the other 383 documents are irrelevant. Documents were randomly selected. The total number of tables in the documents is 2097. 1860 of these tables are irrelevant and 237 tables contain ship activity information. There are 384 missing fields (port names, ship names, date/time values) to find in the corpus. The corpus contains 3588 arrival, 3243 departure and 557 uncategorized activities.

3.2 Methodology

We are using the most conservative approach to determine the truth-value of the matching in which target sequences should be matched exactly. For example if the ship name is “Seven Seas Mariner” and “Seas Mariner” is extracted, we do not count this extraction as a correct match.

We measured precision (percentage of extracted names that are correct), recall (percentage of correct names that are found), and F-measure (harmonic mean of precision and recall) as is commonly done in the Message Understanding Conference (MUC) evaluations [10, 11, 12, 13, 14]. The metrics and their calculation methods are shown below:

$$\text{Precision : } P = \frac{\# \text{ of correct matches}}{\# \text{ of extracted instances}}$$

$$\text{Recall : } R = \frac{\# \text{ of correct matches}}{\# \text{ of instances to extract}}$$

$$\text{F-measure : } F = \frac{(1 + \beta^2)PR}{\beta^2 P + R}$$

The parameter β determines how much to favour recall over precision. We set $\beta = 1$ to equally weight precision and recall.

3.3 Results

We performed several experiments to analyze the success of SeaSpider on different tasks: result filtering, table recognition, missing field search and activity extraction. In this section, we present the quantitative results. Results show how successful our approach has been.

- *Result Filtering*: Table 5 and Table 6 show how our result filter classifies relevant (R) and irrelevant (I) web documents retrieved by the download manager. The confusion matrix for the task is shown in Table 5. Moreover, calculated precision and recall values for the same task are presented in Table 6.

Table 5. Confusion matrix for result filtering

	R'	I'
R	108	5
I	9	374

Table 6. Precision/Recall values for result filtering

Precision	92.3%
Recall	95.6%
F-Value	93.9%

- *Table Recognition*: In this part, we present the combined results of named entity recognition and column recognition tasks. As explained earlier, at the end of these two serial processes, the system decides on whether or not a table is an itinerary table. The confusion matrix for table recognition task is shown in Table 7. Table 8 shows the calculated precision and recall values for the same task.

Table 7. Confusion matrix for table recognition

	R'	I'
R	229	8
I	1	1859

Table 8. Precision/Recall values for table recognition

Precision	99.6%
Recall	96.6%
F-Value	98.1%

- *Missing Field Search:* Table 9 shows the calculated precision and recall values for missing field search. These results show the performance of the system on finding different fields (date expressions, port names, vessel names and so on) which are located out of the itinerary tables.

Table 9. Precision/Recall values for missing field search

Precision	70.2%
Recall	60.7%
F-Value	65.1%

- *Activity Extraction:* Finally, Table 10 shows the general extraction performance of the SeaSpider system. Calculated precision and recall values for the activity extraction process are presented in Table 10.

Table 10. Precision/Recall values for activity extraction

Precision	94.7%
Recall	66.5%
F-Value	78.2%

3.4 Discussion of Results

As seen in Tables 6 and 8, the success rates of the result filtering and table recognition tasks are quite satisfactory. On the other hand, it is obvious that missing field search algorithms need improvement, though satisfying results are obtained for many of the implemented methods. Another observation arising from the results is that general flow of the system makes components interdependent. The output of a component is input to the next component. Therefore the success of a component affects the success of the others. Errors can propagate from one module to another. That is the main reason why performance of the activity extraction task declined compared to result filtering and table recognition tasks.

4. Conclusion & Future Work

This report introduces the SeaSpider project which aims to enable automated gathering of public information about vessel movements. SeaSpider is intended primarily as an operator aid to assist the conventional systems and applications. Moreover, the project is an attempt to use information sources found on the Internet in order to improve MISR capability. The study shows that public information located on the WWW carries some potential for Maritime Domain Awareness. SeaSpider also carries immense importance as being one of the first projects in the field.

In the frame of this study, a prototype system was implemented. Different rule-based and statistical methods were explored and satisfying results obtained for many of the implemented methods. Although SeaSpider combines several research fields to collect information automatically from the Web, it is mainly focused on information extraction. While the existing application areas for IE are broad and varied, our belief is that SeaSpider is an addition to the field.

Future work would include further refinements of the hand-crafted rules for expanding named entity boundaries and improvements of the generalization capability. Moreover, the system needs a rule engine to ease the task of rule definition and interpretation. The overall performance achieved for the system is quite satisfying. However, results also show that better performance could be achieved by improving missing field search algorithms. The system uses keywords from a predefined set of keywords to construct new queries. However, the keyword set is static in the current design. We believe that better queries can be obtained during the retrieval process by addition of a keyword extraction facility, which can update the keyword set according to result filter statistics.

5. References

1. Canada. Privy Council Office. (April, 2004). Securing an Open Society: Canada's National Security Policy. Retrieved from http://www.pco-bcp.gc.ca/docs/InformationResources/Publications/NatSecurnat/natsecurnat_e.pdf
2. Canada. Directorate of Maritime Strategy. (May, 2005). Securing Canada's Ocean Frontiers: Charting the Course from Leadmark. Retrieved from http://www.navy.forces.gc.ca/mspa_video-media/Securing_Canada_E.pdf
3. Hammond, T.R. and Kessel, R.T. 2003. The implications of the universal shipborne automatic identification system (AIS) for maritime intelligence, surveillance and reconnaissance, DRDC Atlantic TM 2003-143
4. H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02). Philadelphia, July 2002.
5. Google SOAP Search API. <http://code.google.com/apis/soapsearch/>.
6. Shannon, Claude E. 1948. A Mathematical Theory of Communication. Bell System Technical Journal, vol. 27, pp. 379-423 and 623-656.
7. Jurafsky, D. and Martin, J.H. 2000. Speech and Language Processing. Prentice Hall.
8. Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Transactions on Information Theory 13(2):260-269, April 1967.
9. L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2):257-286, February 1989.
10. Proceedings of the Third Message Understanding Conference (MUC-3). Morgan Kaufmann, 1991.
11. Proceedings of the Fourth Message Understanding Conference (MUC-4). Morgan Kaufmann, 1992.
12. Proceedings of the Fifth Message Understanding Conference (MUC-5). Morgan Kaufmann, 1993.
13. Proceedings of the Sixth Message Understanding Conference (MUC-6). Morgan Kaufmann, 1995.
14. Proceedings of the Seventh Message Understanding Conference (MUC-7). 1998.

List of acronyms

AIS	Automatic Identification System
API	Application Programming Interface
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance & Reconnaissance
GATE	General Architecture for Text Engineering
GUI	Graphical User Interface
HTML	HyperText Markup Language
IE	Information Extraction
IR	Information Retrieval
ISR	Intelligence, Surveillance & Reconnaissance
MDA	Maritime Domain Awareness
MISR	Marine Intelligence, Surveillance and Reconnaissance
MUC	Message Understanding Conference
NER	Named Entity Recognition
NLP	Natural Language Processing
POS	Part of Speech
RMP	Recognized Maritime Picture
URL	Uniform Resource Locator
WWW	World Wide Web
VOI	Vessel of Interest

Distribution list

Document No.: DRDC Atlantic TM 2007-294

LIST PART 1: Internal Distribution by Centre:

- 2 DRDC ATLANTIC LIBRARY FILE COPIES
- 3 DRDC ATLANTIC LIBRARY (SPARES)
- 1 CSci
- 1 SMO
- 1 H/MICS
- 1 GL/MIKM
- 1 A.-L. LAPINSKI
- 3 D. CHAPMAN
- 1 S. WEBB
- 1 M. HAZEN
- 1 B. MACARTHUR
- 1 T. HAMMOND
- 2 A. ISENR
- 1 B. THWAITES
- 1 F. DESHARNAIS

21 TOTAL LIST PART 1

LIST PART 2: External Distribution by DRDKIM

- 3 ADM(S&T)
 - DRDKIM 3
 - COS Chief Scientist
 - DSTM
- 4 DRDC Ottawa
 - 3701 Carling Avenue, Ottawa, Ontario, K1A 0Z4
 - Attn: Chris Helleur, Dan Brookes, Richard Brown, Gary Geling
- 5 DRDC Valcartier
 - 2459 Pie-XI Blvd North, Québec, Quebec G3J 1X5
 - Attn: Alain Auger, Jean Roy, Alain Bouchard, Regine Lecocq, Catherine Daigle
- 2 JTF(P) Commanding Officer
 - P.O. Box 17000 Stn Forces
 - Victoria, BC,

- V9A 7N2 Attn: CO and LCdr A. Gyorkos
- 2 TRINITY Commanding Officer
P.O. Box 99000 Stn Forces
Halifax, NS ,
B3K 5X5 Attn: CO and Lt(N) J. Warwick
- 1 NDHQ/CMS
101 Col. By Dr., Ottawa ON,
K1A 0K2
Attn: Cdr DP Langlais, D Mar Strat Int.
- 1 DMRS 2
NDHQ / DMRS 2, 101 Col. By Dr., Ottawa ON, K1A 0K2
- 1 PM MSOC
NDHQ / DGDIMPD, 101 Col. By Dr., Ottawa ON, K1A 0K2
- 1 PD MSOC
NDHQ / DGDIMPD, 101 Col. By Dr., Ottawa ON, K1A 0K2
- 1 TTCP MAR AG-8 CNL
DRDC Ottawa, 3701 Carling Avenue, Ottawa, Ontario, K1A 0Z4
c/o Gary Geling
- 1 TTCP MAR AG-8 USNL
Office of Naval Research, Suite 1425, 875 North Randolph Street, Arlington,
Virginia, 22203-1995, USA
Attn: Dr. Rebecca Goolsby
- 1 TTCP MAR AG-8 ANL
DSTO, Department of Defence, PO Box 1500, Edinburgh, SA 5111, Australia
Attn: Anthony Antoniou
- 1 TTCP MAR AG-8 UKNL
DSTL, Room 203, Building A33, DSTL Winfrith, Dorset, DT2 8XJ, United Kingdom
Attn: Neal Freeman
- 24 TOTAL LIST PART 2**
- 45 TOTAL COPIES REQUIRED**

UNCLASSIFIED

DOCUMENT CONTROL DATA (Security classification of the title, body of abstract and indexing annotation must be entered when the overall document is classified)		
1. ORIGINATOR (The name and address of the organization preparing the document, Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's document, or tasking agency, are entered in section 8.) Publishing: DRDC PO Box 1012, Dartmouth, NS, B2Y 3Z7, Atlantic Canada Performing: DRDC PO Box 1012, Dartmouth, NS, B2Y 3Z7, Atlantic Canada Monitoring: Contracting:		2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.) UNCLASSIFIED
3. TITLE (The complete document title as indicated on the title page. Its classification is indicated by the appropriate abbreviation (S, C, R, or U) in parenthesis at the end of the title) SeaSpider: Automated information gathering on vessel movements in support of maritime domain awareness (U) Collecte d'information automatisée sur les mouvements de navires à l'appui de la connaissance du domaine maritime (CDM)		
4. AUTHORS (First name, middle initial and last name. If military, show rank, e.g. Maj. John E. Doe.) Serhan Tatar; David M. F. Chapman		
5. DATE OF PUBLICATION (Month and year of publication of document.) December 2007	6a NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.) 26	6b. NO. OF REFS (Total cited in document.) 14
7. DESCRIPTIVE NOTES (The category of the document, e.g. technical document, technical note or memorandum. If appropriate, enter the type of document, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.) Technical Memorandum This work was supported by a NATO Research Fellowship 2006–2007		
8. SPONSORING ACTIVITY (The names of the department project office or laboratory sponsoring the research and development – include address.) Sponsoring: Tasking:		
9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant under which the document was written. Please specify whether project or grant.) 11hg		9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document) DRDC Atlantic TM 2007–294		10b. OTHER DOCUMENT NO(s). (Any other numbers under which may be assigned this document either by the originator or by the sponsor.)
11. DOCUMENT AVAILABILITY (Any limitations on the dissemination of the document, other than those imposed by security classification.) Unlimited distribution		
12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, when further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.)) Unlimited announcement		

UNCLASSIFIED

UNCLASSIFIED

DOCUMENT CONTROL DATA

(Security classification of the title, body of abstract and indexing annotation must be entered when the overall document is classified)

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)
- (U) SeaSpider is an R&D tool to investigate the development of a software agent that would aid an operator in gathering information about marine vessels from public sources on the internet. This information would supplement sensor information used for Intelligence, Surveillance, and Reconnaissance (ISR) to enhance Maritime Domain Awareness (MDA) and to complete the Recognized Maritime Picture (RMP). Specifically, SeaSpider is fine-tuned to search for, organize, and display information about locations (ports), dates and times, and activities (arrival, in berth, departure). One module manages the Google searches and retrieves the html pages; another module extracts relevant ship movement information and populates a database; a third module retrieves information from the database in response to user-generated queries. In this memorandum, the SeaSpider concept is introduced, the design details of the prototype are presented, and performance is analysed, with a view to future enhancements.
- (U) SeaSpider est un outil de R et D permettant de mettre au point un agent logiciel qui aiderait l'opérateur à recueillir de l'information sur la circulation maritime en puisant dans les informations publiques diffusées sur Internet. Cette information complètera l'information obtenue à l'aide des capteurs employés pour le RSR (renseignement, surveillance et reconnaissance) dans le but d'améliorer la connaissance du domaine maritime (CDM) et le tableau de la situation maritime (TSM). Plus particulièrement, SeaSpider est un outil de précision qui permet de chercher, d'organiser et d'afficher de l'information sur les emplacements (ports), les horaires (dates et heures), ainsi que les activités (arrivées, à quai, départs). Un des modules de cet outil gère les recherches effectuées à l'aide de Google et récupère les pages html; un autre module extrait l'information pertinente sur les déplacements des navires et enrichit une base de données; un troisième module enfin récupère l'information contenue dans les bases de données pour donner suite aux demandes générées par les utilisateurs. Dans le présent document, nous présentons le SeaSpider, fournissons les détails de conception du prototype et analysons sa performance, en laissant entrevoir d'éventuelles améliorations.
14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)
- (U) Maritime Domain Awareness; MDA; Intelligence, Reconnaissance, and Surveillance; ISR; information extraction; web crawler; Google; ships; Recognized Maritime Picture; RMP; marine vessels; internet; WWW; confusion matrix

UNCLASSIFIED

This page intentionally left blank.

Defence R&D Canada

Canada's leader in defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale



www.drdc-rddc.gc.ca