Asian Journal of Information Technology 4 (12): 1159-1169, 2005

Semantic Retrieval of Heterogeneous Data

[1]Reshmy.K.R,[2]S.K.Srivatsa and [3]Rajeevulla Mohammed
[1]Department of IT, Sathyabama Deemed University
[2]Department of ECE, Madras Institute of Technology
[3]Department of CSE, Sathyabama Deemed University, India

Abstract: Our semantic web application integrates images, structured and semi-structured data from various web-pages (HTML, XML) using semantic web technologies. Users can query an integrate view of these sources. The data sources integrated are heterogeneous not only in terms of data but also in terms of how the applications analyze the pages. Before achieving this vision, however, we must address several challenges. We need technologies to extract the data from different web pages (using machine learning techniques), a record linkage system for integrating data (images or structured data) from multiple web-pages to a single entity , a mediator system providing the uniform access to various web-pages, sophisticated analysis of the individual web-pages and efficient query reformulation and execution . In addition a semantic web-based system must recognize when different objects at different pages denote the same real world entity.

Key words: Semantic web, heterogeneous data and mediator system, semantic integrator

INTRODUCTION

The Semantic Web promises seamless integration of heterogeneous data from distributed sources, letting agents (human users or automated programs) perform sophisticated and detailed analyses of this data. An agent would send a query, expressed in terms of its preferred ontology (schema), to a system that would then find and integrate the relevant data from multiple sources and return it using the agents ontology. Before achieving this vision, however, we must address several challenges. We need technologies to integrate data described in different ontologys, for example, as well as different types of data, such as images or structured data. In addition, a Semantic Web-based system must recognize when different objects at different sites denote the same real-world entity. Other challenges include efficiently querying distributed information and converting legacy data in traditional databases and Web sites (HTML) into more semantic representations such as RDF.

semantic integrator is a running application that showcases our approach to meeting these challenges. The application integrates satellite imagery, geospatial data and structured and semi structured data from various online data sources using Semantic Web technologies. Users can query an integrated view of these sources and request semantic integrator to accurately superimpose relevant data (images, text) from various sources on satellite imagery. The data sources integrated by semantic integrator are heterogeneous not only in terms of the data, but also in terms of how the application accesses the sources.

Semantic integrator overview: Semantic integrator helps users obtain satellite imagery, street information and building information about an area. What makes semantic integrator even more attractive is that users can interface manually or have agents query the application using RDF Data Query Language (RDQL) queries and obtain results in RDF. Our GUI consists of an input form and an image. The input form lets users specify attributes the system will use to formulate RDQL queries and retrieve the URL as well as additional information, which the GUI displays. There are three ways to use the applications GUI

- The user retrieves a specific image by filling in its coordinates and clicking update.
- The user navigates to nearby images by clicking one of four white arrows around the image.
- The user clicks on the box on the image that corresponds to a specific house the user wants information (owner and address) about.

Technologies for efficient data interaction: To efficiently integrate semantically heterogeneous information from multiple data technologies:[1]

- Machine-learning techniques for converting traditional legacy Web sources and databases into Web services
- Sources, Building Finder uses several A record linkage system for integrating data from multiple sources referring to a single entity
- A mediator system providing uniform access to data from various Web services

Corresponding Author: S.K. Srivatsa, Department of. ECE, Madras Institute of Technology, India

- An efficient execution system for information-gathering agents
- RDQL and RDF formalisms for representing queries and query results

Online source wrapping: Much information on the Web is formatted for human readers, not machines. Software wrappers let programs or agents retrieve and translate data from Web sources into a format the software can easily manipulate. semantic integrator relies on wrappers to provide the needed data on a per-query basis because the breadth and depth of queries prevents us from storing or caching all data locally. Given a name a city and a state, the wrapper queries the Yahoo site by binding inputs to the wrapper to the corresponding variables in the search form on that site. Yahoo White Pages returns the results page, which lists names, addresses and phone numbers and the wrapper uses extraction rules to extract the data. If Yahoo returns more than one results page, the wrapper extracts the data from all pages. Building Finder filters the data, formats it into a table form and sends it to the mediator, which chooses how to integrate the data to answer queries formulated according to user inputs. Because most Web pages are written in HTML, extracting data is a complicated task. HTML is a language with which Web browsers specify a Web pages format; its not intended to help locate specific data on a page. HTML tags have no semantics with respect to a pages content (Fig .1). Consequently, a wrapper must learn how to extract each data key on each Web page by searching for an HTML tag pattern leading to the data. For example, if we want to extract the persons name ,we might notice that the tag always preceding the name data is <html>Name: and the tag following the name data is . Thus, we would create a rule for the wrapper to extract the keyword A Smith by starting extraction after the tag <html>Name: and stopping at the tag . This process, called wrapper rule learning, uses machine-learning techniques for extracting data automatically from semi structured Web pages[2]. We used the Fetch Agent Platform (www.fetch.com) to create the wrappers used to extract data used in Building

EXISTING SYSTEM

limitation of search engines:

- One of such problems concerns the percentage of pages effectively indexed by search engines
- Another well known limitation of existing search engines are engines are link to pages that are not available anymore ,or in some cases to pages that are not related to the keywords

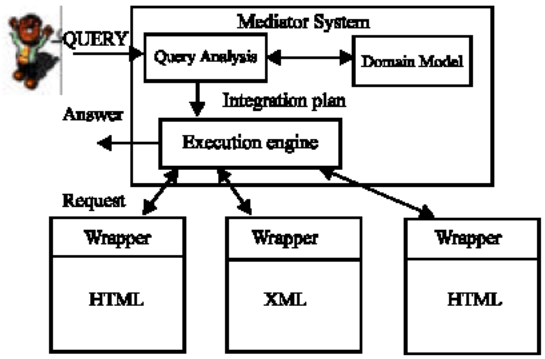


Fig .1: Architecture design

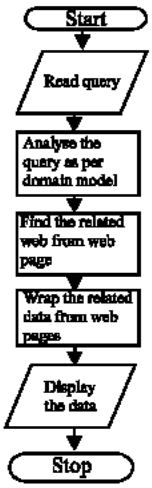


Fig. 2: Data flow diagram

- The redundancy of results is another problem of available search engines.

Our application solves 2nd and 3rd limitations up to 60% by using the Record Linkage System, synonyms of the words and also removes the redundancy up to 60%.

Statement of the problem: To integrate semi-semantically Heterogeneous data (text, pie-charts bar-charts) from different web pages (XML,HTML) and performing sophisticated detail analysis to provide the relevant data ,so that a user can retrieve the data easily as per his query (Fig .2).

Proposed system

Semi-semantic implementation approach:

if data is text:

- We have developed a record linkage system which compares the objects shard attributes in order

identify matching object. Certain attributes are more important for deciding if mapping should exist between two objects. Our approach is to select a primary source for an entity's name and then provide a mapping from that source to each of the other sources where a different naming scheme is used. One way to do this is to create a *mapping table*, which specifies for each entry in one data source what the equivalent entity is called in another data source. Alternatively, if the mapping is computable, it can be represented by a *mapping function*, which is a program that converts one form into another form.

- And also applying the dictionary synonyms to each and every word we can get semi-semantic additionally.

if data is image (pie charts, bar charts):

- checking Whether the two images are equal by shape (circles and squares)
- if equal by shape then extracting the text on the images , comparing whether both text using Record Linkage System

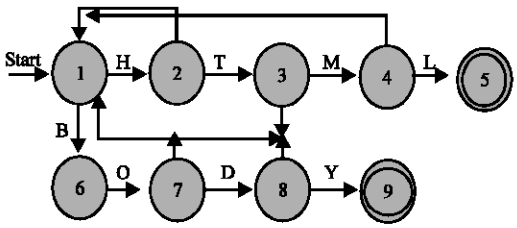
How to implement data integration: As per the query extracting the data from the relevant Web-pages using the record linkage system and placing the relevant info in a separate HTML place by removing the redundant information[3].

Implementation phase:

- retrieve phase or retrieving web pages
- extract phase or source wrapping phase
- analyzing the wrapped data
- record linkage system or mapping system
- integration phase or transformation phase

Retrieving phase: Selecting the corresponding web-pages as per the user specification from the web. Using the syntactical matching between the web paged title and user query

Source wrapping phase or information extract phase
Lexical analysis: Lexical analysis involves scanning the webpage and recognizing the tokens that make up the structure of webpage. Using the lexical analysis[4] finding the tokens (reserved words) of the HTML and XML.



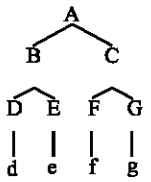
Sample program to identify tokens:

```
getchar()
If(getchar==H)
    Goto label 2
    else goto label 10
Label 2: if(getchar==t)
    goto label 3
    else
    goto label 10
Label 3: if(getchar==m)
    goto label 4
    else
    goto label 10
Label 4: if(getchar==l)
    write token is HTML
```

Syntax analysis:
Parsing: Strings to Trees

d e f g ⇒ (A (B (D d) (E e)) (C (F f) (G g)))

d e f g ⇒



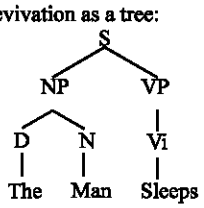
Left-most derivations:

A left-most derivation is a sequence of string $s_1 \ldots s_n$

- S1=S, the start symbol
- Sn $\in \sum^*$, i.e. sn is made up of terminal symbol only
- Each si for i = 2... n is derived from si-1 and by Picking the left-most non-terminal X in si ...1 and replacing it by some β where $X \rightarrow \beta$ is a rule in R

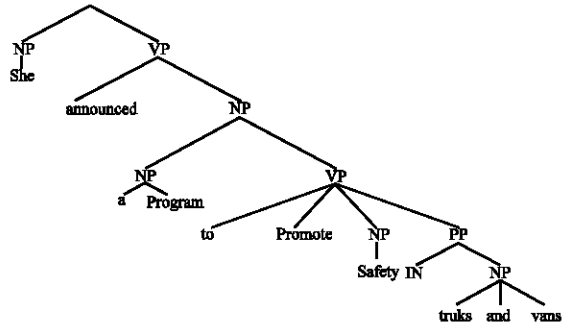
For exmple: [S], [NP VP], [D N VP], [the N VP], [the man VP], [the man Vi], [the man sleeps]

Representatation of a devivation as a tree:



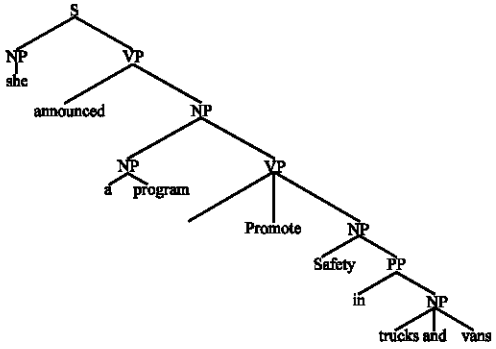
The problem with parsing: ambiguity Input: She announced a program to promote safety in trucks and vans

Possible



Outputs

And there are more…



Named Entity Recognition input: Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

output: Profits soared at [Company Boeing Co.], easily topping forecasts on [Location Wall Street], as their CEO [Person Alan Mulally] announced first quarter results.

Relationships between Entities input: Boeing is located in Seattle. Alan Mulally is the CEO.

output:

{Relationship = Company-Location Company = BoeingLocation = Seattle}

{Employer =Boeing Co.Relationship = Employer-Employee Employee = Alan Mulally}

Part-of-Speech Tagging input: Profits soared at Boeing Co., easily topping forecasts on Wall
Street, as their CEO Alan Mulally announced first quarter results.

output:

Profits/N soared/V at/P Boeing/N Co./N ,/, easily/ADV topping/V
forecasts/N on/P Wall/N Street/N ,/, as/P their/POSS CEO/N
Alan/N Mulally/N announced/V first/ADJ quarter/N results/N ./.
N = Noun
V = Verb
P = Preposition
Adv = Adverb
Adj = Adjective

Named Entity Extraction a Tagging

Input:

Profits soared at Boeing Co., easily topping forecasts on Wall
Street, as their CEO Alan Mulally announced first quarter results.

Output:

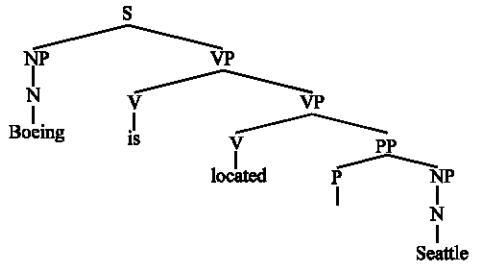
Profits/NA soared/NA at/NA Boeing/SC Co./CC ,/NA easily/NA
topping/NA forecasts/NA on/NA Wall/SL Street/CL ,/NA as/NA
their/NA CEO/NA Alan/SP Mulally/CP announced/NA first/NA
quarter/NA results/NA ./NA
NA = No entity
SC = Start Company
CC = Continue Company
SL = Start Location
CL = Continue Location

Parsing (Syntactic Structure)

Input:

Boeing is located in Seattle.

Output:



Machine translation:

Input:

Boeing is located in Seattle. Alan Mulally is the CEO.

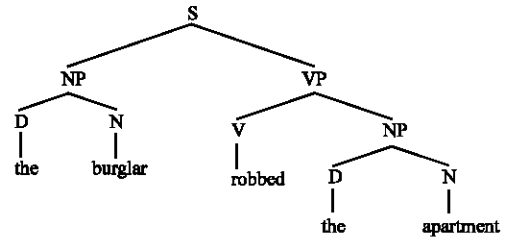
Output:

Boeing ist in Seattle. Alan Mulally ist der CEO.

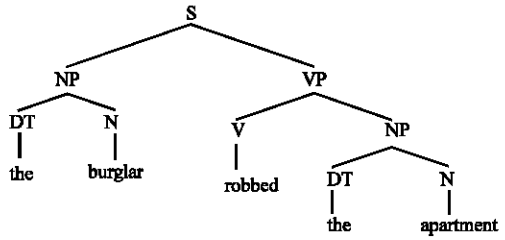
The information conveyed by parse trees

- (1)Part of speech for each word

(N = noun, V = verb, D = determiner)

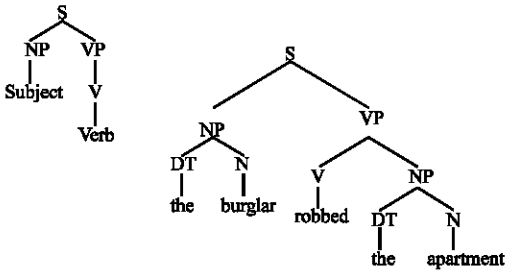


- (2)Phrases



Noun Phrases (NP): the burglar, the apartment
Verb Phrases (VP): robbed the apartment
Sentences (S): the burglar robbed the apartment

- (3)Useful Relationships



→ the burglar is the subject of robbed

An Example Application: Machine Translation

English word order is *subject – verb – object*_
Japanese word order is *subject – object – verb*

English: IBM bought Lotus
Japanese: IBM Lotus bought
English: Sources said that IBM bought Lotus yesterday
Japanese: Sources yesterday IBM Lotus bought that said

Context Free Grammers

A context free grammar G = (N,∑, R, S) where:

- N is a set of non-terminal symbol
- ∑ is a set of terminal symbols
- R is a set of rules of the form $X \rightarrow Y_1 Y_2 \ldots Y_n$ for $n \geq 0$, $X \in N$, $Y_i \in (N \cup \sum)$
- S ∈ N is a distinguished start symbol

A context-free grammar for english:

N = { S, NP, VP, PP, D, Vi, Vt, NP}
S = S
∑ = {sleeps, saw, man, woman, telescope, the, with, in}

R =

S	⇒	NP	VP
VP	⇒	Vi	
VP	⇒	Vt	NP
VP	⇒	VP	PP
NP	⇒	D	N
NP	⇒	NP	PP
PP	⇒	P	NP

Vi	⇒	sleeps
Vt	⇒	saw
N	⇒	man
N	⇒	woman
N	⇒	telescope
D	⇒	the
P	⇒	with
P	⇒	in

Note: S=sentence, VP=verb phrase, NP=noun phrase, PP=prepositional phrase,
D=determiner, Vi=intransitive verb, Vt=transitive verb, N=noun, P=preposition

Operator precedence arsing:

Operator precedence grammar: The grammer that has no productions right hand side is epsilon or has two adjacent non terminals[5].

Using operator precedence parsing removing that tags and getting the required content.

Token Stream	Stack	Data
<HTML>	HTML	
<HEAD>	HTML, HEAD	
<TITLE>	HTML, HEAD, TITLE	
</TITLE>	TITLE	
</HEAD>	HTML, HEAD	
<BODY>	HTML, BODY	
<H1>	HTML, BODY and Hellohowru	
</H1>		
</BODY>	HTML, BODY	
</HTML>	HTML	

Analysing the wrapped data:

- Using the PORTER STEMMER ALGORITHM , finding the basic keywords by removing the suffix from the data[6].
- calculating the number of occurrence of each keyword in each web page , storing the result in the database and subjecting it to necessary calculation to get the matched relevant data as per the user query

Begin:

- For each document
  - Stem all of the words and throw away any common 'noise' words (porter stemmer algorithm)
  - For each of the words
    - Visit and remembereach document that has a direct relationship to this word
    - Score each document based on a distance function from the original document and the relative scarcity of the word in common.
  - For each of the as-of-yet-unvisited new related documents now being tracked
    - Recursively performs the same operation as above.

End Removing suffixes (porter stemmer algorithm):A document is represented by a vetor of words, or \terms\[7].Terms with a common stem will usually have similar meanings, for example:

- CONNECT
- CONNECTED
- CONNECTING
- CONNECTION
- CONNECTIONS

Frequently, the performance of an IR system will be improved if term groups such as this are conflated into a single term. This may be done by removal of the various suffixes -ED, -ING, -ION, IONS to leave the single term CONNECT. In addition, the suffix stripping process will reduce the total number of terms in the IR system and hence reduce the size and complexity of the data in the system, which is always advantageous.

The words of the queries were reduced to stems in the same way and the documents were ranked for each query using term coordination matching of query against document

The algorithm: To present the suffix stripping algorithm in its entirety we will need a few.

Definitions:
A \consonant\ in a word is a letter other than A, E, I, O or U and other than Y preceded by a consonant. (The fact that the term `consonant' is defined to some extent in terms of itself does not make it ambiguous.) So in TOY the consonants are T and Y and in SYZYGY they are S, Z and G. If a letter is not a consonant it is a \vowel\.
A consonant will be denoted by c, a vowel by v. A list ccc... of length greater than 0 will be denoted by C and a list vvv... of length greater than 0 will be denoted by V. Any word, or part of a word, therefore has one of the four forms:

- CVCV ... C
- CVCV ... V
- VCVC ... C
- VCVC ... V

These may all be represented by the single form

[C]VCVC ... [V]

where the square brackets denote arbitrary presence of their contents. Using (VC) {m} to denote VC repeated m times, this may again be written as

[C](VC){m}[V].

m will be called the \measure\ of any word or word part when represented in this form. The case m = 0 covers the null word. Here are some examples:

m=0 TR, EE, TREE, Y, BY.
m=1 TROUBLE, OATS, TREES, IVY.
m=2 TROUBLES, PRIVATE, OATEN, ORRERY.

The \rules\ for removing a suffix will be given in the form

(condition) S1 -> S2

This means that if a word ends with the suffix S1 and the stem before S1 satisfies the given condition, S1 is replaced by S2. The condition is usually given in terms of

m, e.g. (m > 1) EMENT ->

Here S1 is `EMENT' and S2 is null. This would map REPLACEMENT to REPLAC, since REPLAC is a word part for which m = 2.

The `condition' part may also contain the following:
*S - the stem ends with S (and similarly for the other letters).

v - the stem contains a vowel.

*d - the stem ends with a double consonant (e.g. -TT, -SS).

*o - the stem ends cvc, where the second c is not W, X or Y (e.g. -WIL, -HOP).

And the condition part may also contain expressions with \and\, \or\ and \not\, so that

(m>1 and (*S or *T))

tests for a stem with m>1 ending in S or T, while

(*d and not (*L or *S or *Z))

tests for a stem ending with a double consonant other than L, S or Z.

Elaborate conditions like this are required only rarely.

In a set of rules written beneath each other, only one is obeyed and this will be the one with the longest

matching S1 for the given word. For example, with

SSES -> SS
IES -> I
SS -> SS
S ->

(here the conditions are all null) CARESSES maps to CARESS since SSES is the longest match for S1. Equally CARESS maps to CARESS (S_1=` SS') and CARES to CARE ($S_1$=` S').

In the rules below, examples of their application, successful or otherwise, are given on the right in lower case. The algorithm now follows:

Step 1a:

SSES -> SS caresses -> caress
IES -> I ponies -> poni
Ties -> ti
SS -> SS caress -> caress
S ->cats -> cat

Step 1b:

(m>0) EED -> EE feed -> feed
Agreed -> agree
(*v*) ED -> Plastered -> plaster
Bled -> bled
(*v*) ING -> Motoring -> motor
Sing -> sing

If the second or third of the rules in Step 1b is successful, the following is done:

AT -> ATE conflat(ed) -> conflate
BL -> BLE troubl(ed) -> trouble
IZ -> IZE siz(ed) -> size
(*d and not (*L or *S or *Z))
-> single letter
hopp(ing) -> hop
tann(ed) -> tan
fall(ing) -> fall
hiss(ing) -> hiss
fizz(ed) -> fizz
(m=1 and *o) -> E fail(ing) -> fail
fil(ing) -> file

The rule to map to a single letter causes the removal of one of the double letter pair. The -E is put back on -AT, -BL and -IZ, so that the suffixes -ATE, -BLE and -IZE can be recognised later. This E may be removed in
step 4:

Step 1c: (*v*) Y -> I and happy -> happi
sky -> sky

Step 1 deal with plurals and past participles. The subsequent steps are much more straightforward.

Step 2

(m>0) ATIONAL -> ATE relational -> relate
(m>0) TIONAL -> TION conditional -> condition rational -> rational
(m>0) ENCI -> ENCE valenci -> valence
(m>0) ANCI -> ANCE hesitanci -> hesitance
(m>0) IZER -> IZE digitizer -> digitize
(m>0) ABLI -> ABLE conformabli -> conformable
(m>0) ALLI -> AL radicalli -> radical
(m>0) ENTLI -> ENT differentli -> different
(m>0) ELI -> E vileli -> vile
(m>0) OUSLI -> OUS analogousli -> analogous
(m>0) IZATION -> IZE vietnamization -> vietnamize
(m>0) ATION -> ATE predication -> predicate
(m>0) ATOR -> ATE operator -> operate
(m>0) ALISM -> AL feudalism -> feudal
(m>0) IVENESS -> IVE decisiveness -> decisive
(m>0) FULNESS -> FUL hopefulness -> hopeful
(m>0) OUSNESS -> OUS callousness -> callous
(m>0) ALITI -> AL formaliti -> formal
(m>0) IVITI -> IVE sensitivity -> sensitive
(m>0) BILITI -> LE sensibiliti -> sensible

The test for the string S_1 can be made fast by doing a program switch on the penultimate letter of the word being tested. This gives a fairly even breakdown of the possible values of the string S_1. It will be seen in fact that the S_1-strings in step 2 are presented here in the alphabetical order of their penultimate letter. Similar techniques may be applied in the other steps.

Step 3:

(m>0) ICATE -> ICtriplicate -> triplic
(m>0) ATIVE ->formative -> form
(m>0)ALIZE ->ALformalize -> formal
(m>0) ICITI -> ICelectricity-> electric
(m>0) ICAL -> IC and electrical -> electric
(m>0) FUL -> hopeful -> hope
(m>0) NESS ->goodness -> good

Step 4:

(m>1) AL ->revival -> reviv
(m>1) ANCE ->allowance -> allow
(m>1) ENCE -> and inference -> infer
(m>1) ER -> airliner -> airlin
(m>1) IC -> gyroscopic -> gyroscop
(m>1) ABLE ->adjustable -> adjust
(m>1) IBLE -> and defensible -> defens
(m>1) ANT -> irritant -> irrit
(m>1)EMENT->replacement -> replac
(m>1)MENT -> and adjustment -> adjust
(m>1) ENT ->dependent -> depend
(m>1 and (*S or *T)) ION -> adoption -> adopt

(m>1) OU -> homologou -> homolog
(m>1) ISM -> communism -> commun
(m>1) ATE -> activate -> activ
(m>1) ITI -> angulariti -> angular
(m>1) OUS -> homologous -> homolog
(m>1) IVE -> effective -> effect
(m>1) IZE -> bowdlerize -> bowdler

The suffixes are now removed. All that remains is a little tidying up.

Step 5a:
(m>1) E -> and probate -> probat rate -> rate
(m=1 and not *o) E -> and cease -> ceas

Step 5b:
(m > 1 and *d and *L) -> single letter
control -> control
roll -> roll

The algorithm is careful not to remove a suffix when the stem is too short, the length of the stem being given by its measure, m. There is no linguistic basis for this approach. It was merely observed that m could be used quite effectively to help decide whether or not it was wise to take off a suffix.

LIST 1: For example in the following list

List A	List B
Relate	Derivate
Probate	Activate
Conflate	Demonstrate
Pirate	Necessitate
Prelate	Renovate

-ATE is removed from the list B words, but not from the list A words. This means that the pairs DERIVATE/DERIVE, ACTIVATE/ACTIVE, DEMONSTRATE/DEMONS-TRABLE, NECESSITATE/NECESSITOUS, will conflate together. The fact that no attempt is made to identify prefixes can make the results look rather inconsistent. Thus PRELATE does not lose the -ATE, but ARCHPRELATE becomes ARCHPREL. In practice this does not matter too much, because the presence of the prefix decreases the probability of an erroneous conflation.

Complex suffixes are removed bit by bit in the different steps.
Thus
GENERALIZATIONS is stripped to GENERALIZATION (Step 1), then to GENERALIZE (Step 2), then to GENERAL (Step 3) and then to GENER (Step 4).
OSCILLATORS is stripped to OSCILLATOR (Step 1), then to OSCILLATE (Step 2), then to OSCILL (Step 4) then to OSCIL (Step 5). In a vocabulary of 10,000 words, the reduction in size of the stem was distributed among the steps as follows:
Suffix stripping of a vocabulary of 10,000 words

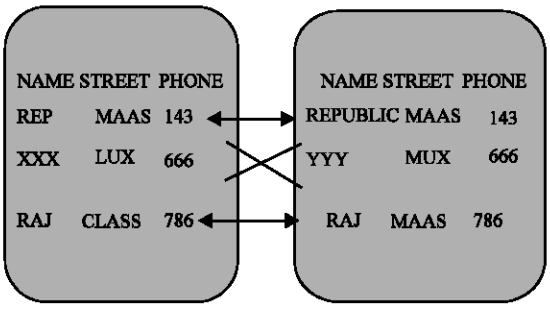


Fig. 3: Mapping system

Number of words reduced in step 1:	3597	
"	2:	766
"	3:	327
"	4:	2424
"	5:	1373
Number of words not reduced:		3650

The resulting vocabulary of stems contained 6370 distinct entries. Thus the suffix stripping process reduced the size of the vocabulary by about one third.

Mapping system or record linkage system: The task of object identification occurs when integrating information from different web pages[8]. The same data objects can exist in inconsistent text formats across pages, making it difficult to identify matching objects using exact text match. Examples of the object identification problem are shown in below Figure. In the first example the restaurant referred to as republic on the one website may appear "rep's Delicatessen" on the other.

Different representations of persons at different places: I have developed a record linkage system which compares the objects shard attributes in order identify matching object. Certain attributes are more important for deciding if mapping should exist between two objects.

Our approach is to select a primary source for an entity's name and then provide a mapping from that source to each of the other sources where a different naming scheme is used. One way to do this is to create a *mapping table*, which specifies for each entry in one data source what the equivalent entity is called in another data source. Alternatively, if the mapping is computable, it can be represented by a *mapping function*, which is a program that converts one form into another form.

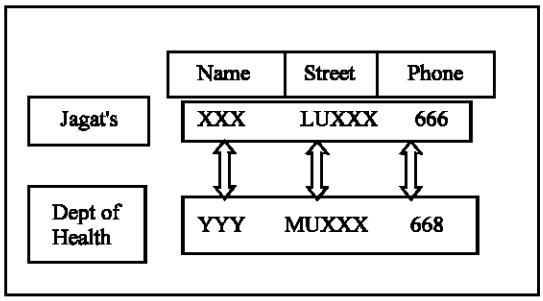


Fig. 4: Mapping Function

Our method attempts to pair each entity in one source with a corresponding entity (or entities, in some cases) in another source. The basic idea is to use information retrieval techniques to provide an initial mapping and then to apply machine learning techniques to improve the mapping. The initial mapping matches entities from two sources based on their textual similarity. In the subsequent learning phase, the system learns two types of rules to help improve/verify the initial mapping. Transformation rules identify textual transformations[9] like acronyms, abbreviations and phrase orderings that are common in the domain. For instance, the system can learn that Rep is a commonly used abbrievation for `Republic, or that one source commonly employs acronyms, or that one source represents person names as "LastName, Firstname, while the other uses FirstName LastName. The system also learns Mapping rules which are used when we can compare entities along multiple attributes.

We are prototyping an active learning method for learning transformation rules and mapping rules. In our approach, a human is asked to verify some of the pairs in the initial mapping, i.e., to indicate whether the pairs are correctly or incorrectly matched. Then the system attempts to learn new rules and selects additional pairs for the human to verify. The system selects the pairs that will be most valuable for confirming/disconfirming the hypotheses explored by the learning algorithm. The goal is to obtain a mapping for which the system is highly confident, while minimizing the time the human must spend.

Parsing for information extraction: Relationships between Entities

Input:

Boeing is located in Seattle

Output:

{Relationship = Company-Location
Company = Boeing
Location = Seattle}

Input:

Boeing is located in Seattle. Alan Mulally is the CEO.

Output:

{Relationship=Company-Location
Company = Boeing Location = Seattle}
{Employer =Boeing Co.Relationship = Employer-Employee Employee = Alan Mulally}

Integration phase:

Integrating vector data and imagery: We focused on the problem on integration of structure data sources, such as databases or XML data. With the more complex geospatial data types, such as imagery, maps and vector data accurately integrating vector data with images. WeI developed efficient techniques to the even more challenging problem conflating maps with imagery. There is a wide variety of data available on the Internet that provides satellite imagery and maps of various regions. In addition, a wide variety of maps are available from various government agencies, such as property survey maps and maps of oil and natural gas fields. Satellite imagery and aerial photography have been utilized to enhance real estate listings, various military targeting applications and other applications. By integrating these data sets, one can support a rich set of knowledge discovery queries that could not have been answered given any of these data sets in isolation. For example, when you are looking for a park in a neighborhood, the imagery may provide you better view of the park, while the map is essential to see the surrounding streets and how to get to the park. However, accurately integrating maps and imagery from different data sources remains a challenging task. This is because data obtained from various data sources may have different projections and different accuracy levels.

RESULT

Inputs to the project: HTML, XML pages which are stored in the
Apache Tomcat Server4.1.2

Screen shots for the input:

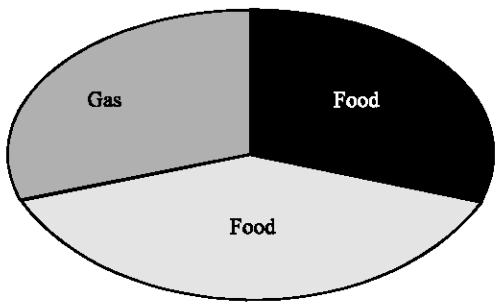


Fig. 5: Input

Table .1: Input

Months	Food	Gas	Motel
Jan	12	11	13
Feb	21	14	17
Mar	11	14	12
Apr	20	21	19
May	11	16	18
Jun	14	15	12

Fig. 6: Opening screen

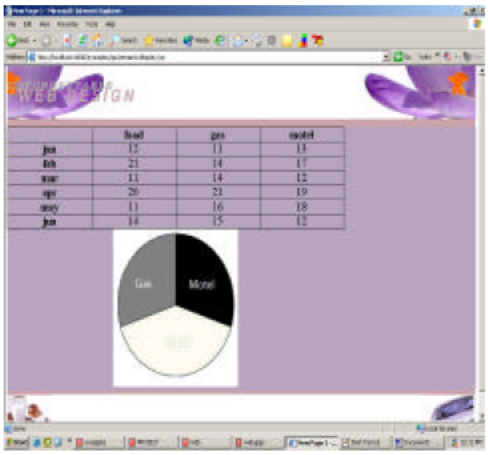


Fig.7: Output

We are launching our new Products Shop Boss for retail sector very shortly. By introducing new marketing strategy we produce and supply our products in a very short period. The monthly report of this year is, in Jan the supplied unit of Food is 12, Gas is 11 and Motel is 13. (Table 1 and Figs)

Output: Semi-semantically integrated heterogeneous data as per user specification (Fig. 6 and 7)

CONCLUSION AND FUTURE ENHANCEMENT

To conclude, in Coding basically there are 5 stages,

- Retrieve phase or retrieving web pages
- Extract phase or source wrapping phase
- Analyzing the wrapped data
- Record linkage system or mapping system
- Integration phase or transformation phase

The features that are given in the definition of the problem are successfully implemented in this study. This study aims To integrate semi-semantically Heterogeneous data (text, pie charts, bar charts) from different web pages and performing sophisticated detail analysis to provide the relevant data so that a user can retrieve the data easily as per his query.

This study combines the features of some the already available Ones (porter stemmer algorithm, dictionary of words) but develops a new application with advanced features, which is still not available in world.

- Designing semantic search engines at large scale
- Providing Semantic web services
- Implementing semantic web-applications
- Solving the semantic web challenges

REFERENCES

1. Michalowski, M., J.L. Ambite, S. T. Rattapoom Tuchinda, Craig and A. Knoblock, 2004. Retrieving and semantically integrating heterogeneous data from the web. University of Southern California and Fetch Technologies, Steve Muiton, Fetch Technologies. IEEE, intelligent systems.
2. Hendler, *et al.*, 2002. Integrating applications on the semantic web. J. the Institute of Electrical Engineers of Japan, 122: 676-680.
3. Hendler, agents and the semantic web, IEEE intelligents systems J, pp:
4. Saalfel, A., 1993. conflation: Automated map compilationtech, report, computer vision lab,centre for automatic research,Univ.of Mary land 1993.
5. Chenetal, C.A., 2003. automatically annotating and integrating spaptial data sets, proc.8th Intl. Symposium, Spatial and Temporaldatabases, lncs 2750, springer -verleg, pp: 469-488.
6. Knoblocketal, C.A., 2003. accurately and reliably extracting data from the Web: A Machine Learning Approachintelligent Explanation of the Web, Springer-verlag, pp:275-287.
7. Barish, G. and C.A. Knoblocketal, 2002. An expressive and efficient language for information gathering on the web.proc.6th Intl Con. AI Planing and Schedulin(AIPS 2002) Workshop:

8. Levy, A.Y., 2000. Logic-based Techniques in Data Integration; logic Based Artificial intelligence. J. Muiker, ed Kluwer Publishers, pp: 575-595.

9. Berners T. and W. Lee, 1998. RDF model is different from the XML model.Informal note, 1998.http://www.w3.org/design issues/RDF-XML.html