



Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019

Topic Identification Through Ontology-based Concept Generalization

Raman Adaikkalavan, Laali Elkhalfa,
Y. Alp Aslandogan
Contact author: alp@cse.uta.edu

Technical Report CSE-2003-26

1.	Introduction.....	3
2.	Rationale	3
2.1.	Topic Identification in Web pages containing text-based information	4
2.2.	Topic Identification in Corporate email messages	4
3.	Various Approaches for Topic Identification	4
4.	Architecture and Implementation for Topic Identification.....	5
4.1.	Input Module.....	6
4.2.	Feature Extraction Module	6
5.	Text Feature Extraction using Natural Language Processing (<i>NLP</i>)	7
5.1.	Implementation of Natural Language Processing	7
5.2.	HTML Parsing	8
5.3.	Elimination of stop words and special characters	9
5.4.	Parts of Speech Tagging	9
5.5.	Converting Plural to Singular form	10
5.6.	Web Structure Oriented Weighting (WSOW)	10
5.7.	Assigning weights by position	11
5.8.	Concept Mapping/Generalization	11
5.9.	Using WordNet Lexical Database.....	14
6.	Text Feature Extraction Using IBM Intelligent Miner.....	14
6.1.	IBM Intelligent Miner Feature Extraction tool.....	15
6.2.	Implementation of IBM Intelligent Miner Feature Extraction tool.....	15
7.	Experiments and Performance Study	17
7.1.	Test Cases for Topic Categorization using NLP:.....	17
7.2.	For the test case 3, the snap shot of the output is shown below.....	18
8.	Limitations	20
9.	Conclusion	21
10.	References.....	21

Abstract

We present a method for topic identification for HTML and similar structured documents. Our approach relies on part-of-speech tagging, HTML parsing, frequency and position-based term weighting, and concept expansion. Concept expansion is achieved through Wordnet, a general purpose lexical ontology. We apply the method for categorizing web pages and compare the performance with human perception. These preliminary experimental results show that the approach is promising and can be adapted to many categorization tasks.

1. Introduction

Topic identification concerns a problem of predicting terms in text, which indicates its subject or themes [1]. Topic identification plays an important role, but not limited to the following areas

1. Topic Identification in Corporate email messages
2. Topic Identification in Web pages containing text-based information
3. Topic Identification in Multimedia Broadcast Data
4. Topic Identification in Natural Language Dialogues Using Neural Networks
5. Topic Identification in a Text Corpus
6. Topic Identification in Dynamical Text by extracting minimum complexity time components
7. Topic Identification using Ontology Hierarchy
8. Topic Identification of News Speech based on Keyword Spotting
9. Topic Identification in Discourse

This proposal will provide information that is needed to show the rationale behind choosing this problem, its significance, and different approaches to handle this problem, experimental setup and performance study.

2. Rationale

In this section, we take the first two aforementioned roles to explain why topic identification is important.

2.1. Topic Identification in Web pages containing text-based information

Project "How Much Information", done at UC Berkeley in 2000 [2] specifies that the "surface" Web consists of approximately 2.5 billion documents, up from 1 billion pages at the beginning of the year, with a rate of growth of 7.3 million pages per day. With this rapid growth of information in the web, there is an important need for topic identification, since all these documents are not useful unless manually read. Identifying the topics and categorizing it (i.e., information retrieval) is an efficient way in order to get useful information from these documents. Category or categories a document belongs to can be determined according to predefined categories provided by the user or by using traditional learning techniques.

2.2. Topic Identification in Corporate email messages

Topic identification is an important problem for corporations that use computers to classify email. On daily basis, most of the corporations receive messages through email from customers concerning orders, complaints, inquires, shipping dates, latest releases, different products. Those corporations would want to automate the classification of such emails and forward them to their respective departments. Since the emails are usually sent in an unstructured form, it will be likely that the only way to determine the topic of an email is to manually review it, which is not an efficient way.

3. Various Approaches for Topic Identification

There are various approaches to handle these problems and we have specified 6 of them below.

- Topic Identification by word counting. Predicting important terms involves numerical weighting of terms in documents. Terms with top weights are judged important and representative of document [1].
- Topic Identification by position (i.e. Title/Meta Tags) [6].
- Topic Identification by using the inverse document frequency (IDF) for finding the important nouns and their connectivity with other nouns and verbs [3].

- Topic Identification by using the “traditional” data mining techniques [4].
- Trace the hyperlinks in the web page and form a graph, showing the relationship between the linked documents. Then identify the theme of each document and then come up with an overall theme for the web page or the web structure [5].
- Topic Identification by cue phrases [7].

In this project, we are using a combination of the first two approaches.

4. Architecture and Implementation for Topic Identification

This section describes the architecture and implementation of our project, Topic Identification/Concept Generalization. In this project, we are using Natural Language Processing for Topic Identification/ Concept Generalization and Intelligent Miner feature extraction tool for extracting the keywords from HTML pages.

Architecture and Implementation for Topic Identification

Figure 1 shows the architecture for the Topic/Keyword Identifier. The first module is the Input module and the second one is the feature extraction module which is explained in the following subsections. Topic Identification/ Concept Generalization using Natural Language Processing include HTML Parser, Parts of Speech (POS) tagging, Web Structure Oriented Weighting (WSOW), and WordNet Database tool. The WordNet lexical database tool is used to roll-up the concept hierarchy and to generate a generalized concept for our topic identified. List of keywords/phrases is extracted from the webpage using IBM Intelligent Miner feature extraction tool along with WSOW. The approach is carried out in steps such as Feature Extraction, WSOW, Topic Identification and Concept Mapping/Generalization, and Keywords/phrases extraction. For each of these steps, a module developed and is linked to the other modules. Below we provide a brief description of each module in the architecture.

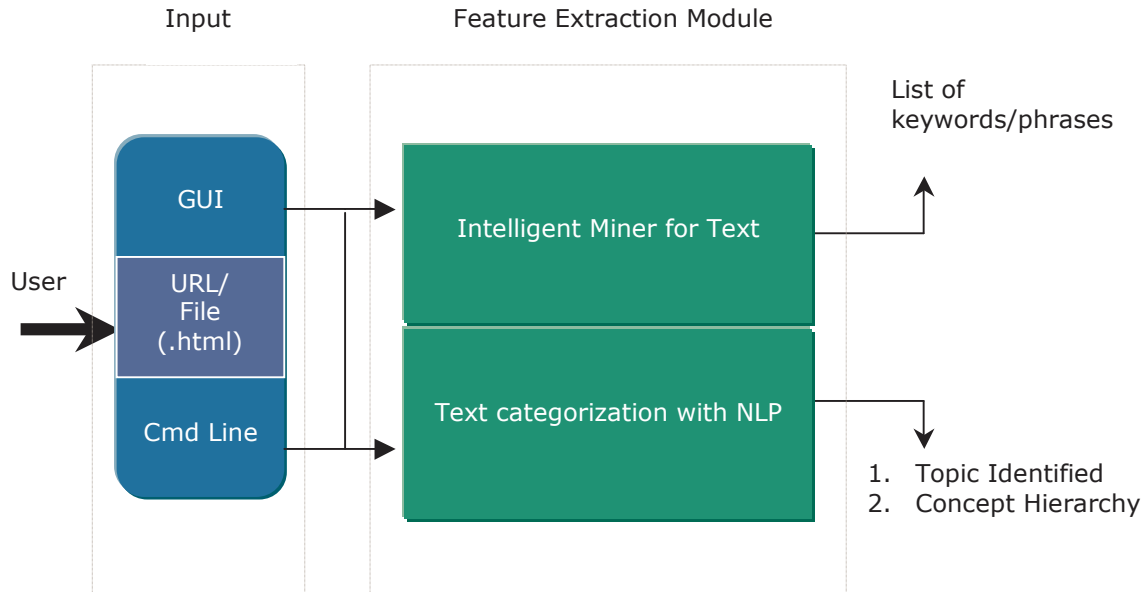


Figure 1. Architecture for Topic Identification/ Concept Generalization.

4.1. Input Module

With the input module (see in **Figure 1** GUI/Command Line) user can either type a URL or HTML filename of an existing file on the GUI text field provided by our application, or feed it directly to the program as a command line argument. Our application is loosely coupled with the GUI in terms that the application can run without the use of GUI from the command prompt providing the necessary parameters. The GUI provides the flexibility of assigning various parameters to the intelligent miner text extraction tool as well as the concept hierarchy display module (explained in details in following sections).

4.2. Feature Extraction Module

This module parses the html page and extracts words, computes their term frequency and ranks the top “n” terms based on descending frequency count. This module in turn has two modules, one for feature extraction using NLP and the other for IBM Intelligent miner. We will explain in detail about these in the following sections. Output of this module will be the Topic/ Keywords identified.

5. Text Feature Extraction using Natural Language Processing (NLP)

This module has 5 modules and they are Parser, Stop word eliminator, WSOW, Parts of Speech (POS) tagging and Topic Identification as shown in **Figure 1**. A web page is parsed, and tokenized. Each tokenized text undergoes additional processing elimination of stop words and special characters, parts of speech tagging and conversion of plural to singular form. During the preprocessing of the extracted words, word/ term frequency is taken into account. Each term/ word is assigned a value that represents the occurrence of the term in the input. The value is computed based on WSOW. After computing the value, the top ranked keywords with high values are selected to generate a general concept using WordNet lexical database tool. These modules are explained in detail in the following sections.

5.1. Implementation of Natural Language Processing

This module has been implemented in different phases using different Perl modules. Prior to these phases, the application checks to see if the input is URL or a HTML file. If it is a URL, using the *Perl LWP*, a set of Perl modules that provides an application-programming interface (API) to the World Wide Web, the html page is fetched if it exists. The API sends a request that contains the URL name to the remote file server and gets the response. As a first step, we instantiate a *user agent*, an object that represents the application on the network and provides an interface that accepts requests for fetching web pages and returns responses to the application. *LWP* has a class name *HTTP::Request* that communicates with the user agent. The main attributes of the request object:

- The **method** that tells what kind of a request this is. The method we used is *GET* (get the web page).
- The **uri**, a string comprising the name of the URL to be accessed.

The request object sends the request to the user agent and gets a response back. The response contains the content of the document, which is then printed out to a file “*source.txt*” and is passed to the parser. However, if the input is an existing file, the module directly feeds the file to the parser.

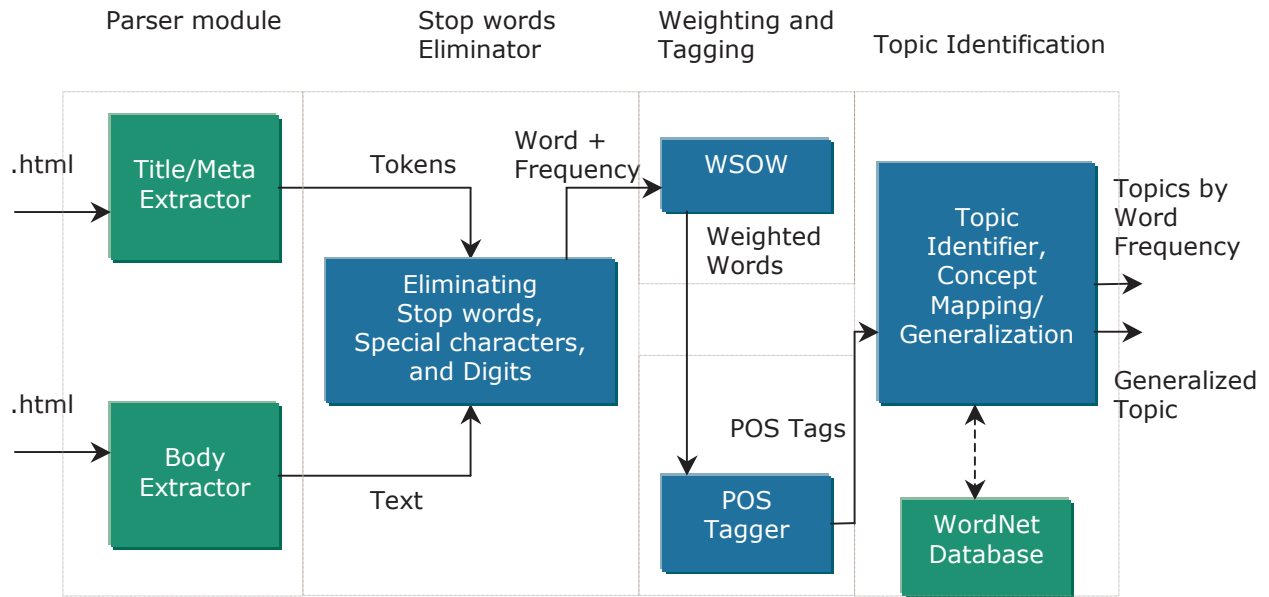


Figure 1. Architecture for Topic Identification using NLP.

5.2. HTML Parsing

In this first phase, the input is processed by the *HTML::Tokeparser*, an interface that parses an html document. We use this module to extract words that appear in the Title or Meta. An *HTML::Tokeparser* object is instantiated passing the filename as an argument. The module invokes *get_token* method to read tokens from html document. If the token has a tag Meta or Title, the text comprised in that html element is extracted.

To extract keywords from the body of the html document, we use *HTML::TreeBuilder*, an html parser that parses an html file and produces a syntax tree, to extract keywords appeared in the body. The following steps detail how we use the *HTML::TreeBuilder*:

- The module creates a new *HTML::TreeBuilder* object, \$tree.
- Then use one of the methods of *HTML::Parser*, *parse_file(\$filename)*, to parse the HTML document into the tree \$tree.
- Using *look_down* method, we traverse the tree for body tag and retrieve the textual information contained in the body.

5.3. Elimination of stop words and special characters

In this phase, we eliminate stop words, digits and special characters from both the retrieved text present in the body of html document and array of keywords present in the Meta or Title. The information retrieval system developed at Cornell called SMART uses a stop word list of 571 words (<ftp://ftp.cs.cornell.edu/pub/smart/>). We use this list of stop words that are available in the file “stop.wrd”.

5.4. Parts of Speech Tagging

In order to have WordNet tool identify the keywords selected to generate a general concept for the topic, we need to specify the tag for each keyword. Hence in our application we utilize Eric Brill’s tagger (provided with the application). The tagger specifies whether the keyword is a verb, adjective, adverb or a noun. Since Eric Brill’s tagger has different tags than the ones used in WordNet, we map the tags using the following table:

Brill’s Tags	Parts of speech	Wordnet Tags
NN	Noun, singular	“n” noun
NNS	Noun, plural	
NNP	Proper noun, singular	
NNPS	Proper noun, plural	
RB	Adverb	“V” adverb
RBR	Adverb, comparative	
RBS	Adverb, superlative	
VB	Verb, base form	“v” verb
VBD	Verb, past tense	
VBG	Verb, gerund or present participle	
VBN	Verb, past participle	
VBP	Verb, singular	
VBZ	Verb, 3 rd person singular	
JJ	Adjective	
JJR	Adjective, comparative	
JJS	Adjective, superlative	

5.5. Converting Plural to Singular form

During the tagging process, we keep record of the plural nouns. Some of the plural nouns are not recognized by WordNet, therefore we convert them using *WordNet::Querydata*. *WordNet::Querydata* provides a direct interface to the WordNet database files. To use *Querydata*, we specify the path of the WordNet database by passing it as an argument invoking “new” method. Then we invoke “validForms” method to retrieve the singular form of the plural keyword example it returns baby for babies. Basically, in this phase we do stemming.

During our implementation, we started using *Lingua::Stem* a perl module and other approaches for the *Porter's stemming algorithm* to stem words to get their singular form. After experimenting with it, we found that when the words are stemmed for e.g. *printable* to *printable* WordNet database did not have the corresponding word, thus we used the approach explained before.

5.6. Web Structure Oriented Weighting (WSOW)

This module (see **Figure 1** for WSOW) accepts inputs from the Parts of Speech Tagger module. From the set of features extracted by the modules described before, this module chooses the terms that are representative of the document, not only considering the number of occurrences of terms in the document, but also the HTML tag element the terms are present in. For achieving good performance, we assign extra weight to terms that belong to the elements that are more suitable for representing web pages such as Titles, and Meta tag elements. For example, the words been extracted from the title element receives an extra weight (more details about the weighing approach is provided in the next section). This phase is known as feature selection. We select the top “n” ranked (n=5 based on experiment) terms that have a greater weight than the other terms. And those terms are later passed to the WordNet database tool to generate a general concept for the topic description.

5.7. Assigning weights by position

Certain positions in the input html page tend to carry important information about the topic. In this module we recognize words or terms that appear in those important positions and to assign greater importance by adding weights to those terms. Weights are assigned by the following function [8]:

$$SWT_w(t_i, d) = \sum_{e_k} (w(e_k) \cdot TF(t_i, e_k, d))$$

e_k : HTML element,

$w(e_k)$: Weight assigned to the element e_k

d = 1 (number of web pages)

$TF(t_i, e_k, d)$ = number of times the term t_i occurs in the web page.

Weight function $w(e)$ will be defined as

$$w(e) = \begin{cases} \alpha & \text{if } e = \text{Meta or Title tags} \\ 1 & \text{elsewhere} \end{cases}$$

In this function, greater weight is given to the Meta and Title tags. Term frequency (TF) of term t_i , number of times the term t_i appeared in html element e_k within the input page, will be multiplied by html element e_k weight. Then the results of the product are summed up and considered to be the weight of the term in the document.

5.8. Concept Mapping/Generalization

The basic idea of the topic identification is to provide a general topic together with list of keywords that best describes the input web page in a form of a concept hierarchy. In this phase as shown in **Figure 1**, top five extracted keywords are mapped into one concept. Concept mapping is one of the challenging parts of this project. Topic mapping requires prior knowledge about the general concepts. To handle this problem, we exploit WordNet lexical database tool to generalize words or terms and then consider their frequent

occurrence in the web page. We utilize the word or term frequency and assign it to the word's associated concept as a concept frequency. We stop generalizing when we reach the appropriate concept considering the distance between the concept and the actual keywords. We specify a certain threshold as to how far we can roll up the hierarchy. The concept, which is more frequent and has shortest distance, is selected. We had explained the whole process of concept generalization below using an example.

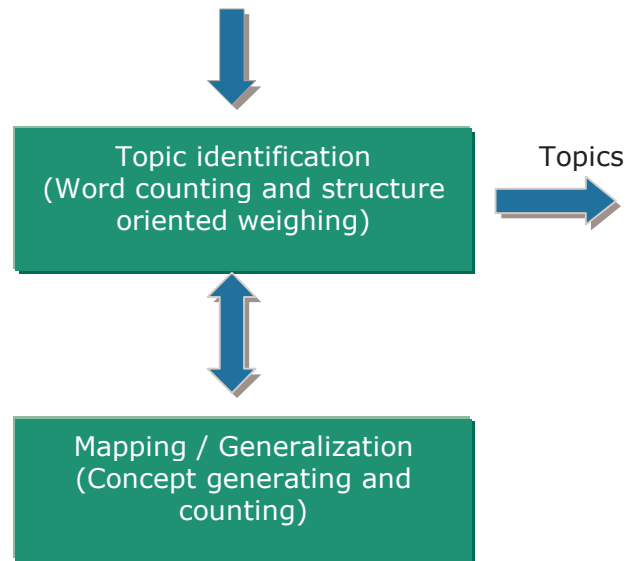


Figure 2. Topic Identification

Lets assume the top five ranked words that are need to be generalized are

1. Earthquake, POS
2. Magnitude, POS
3. Rock, POS
4. Damage, POS
5. Energy, POS

Now these words along with the POS are passed to WordNet database to get the hypernyms in all the senses. We then intersect these hypernyms to get the concept hierarchy that can represent this page. Since the intersection that we get is more than one, we will use some heuristics. We will take the intersection that is most frequent, since this

5.9. Using WordNet Lexical Database

In this project, our approach is applied as a combination of different techniques. It is used for finding concept hierarchy using WordNet and other lexical resources such as morphological module.

Wordnet is a large and free lexical database tool [9]. Its database is divided by parts of speech into nouns, verbs, adjectives and adverbs. The nouns are organized as a lexical hierarchy of nodes. Each node is a word meaning or a synset. Synset is set of English words that refer to the same meaning. In this project, WordNet will be exploited to extract the semantic relations between words and be able to generalize them. It is the relation of subordination, which in this context is called *hyponymy*. For example, the noun *robin* is a hyponym (subordinate) of the noun *bird*, or, conversely, *bird* is a hypernym (superordinate) of *robin*. It is this semantic relation that organizes nouns into a lexical hierarchy.

In Wordnet, hypernymy is a relation between lexicalized concepts, a relation that is represented by a pointer between the appropriate synsets. Synsets are the sets of synonyms that form the basic building blocks. Thus, a lexical hierarchy can be reconstructed by following the path of hypernymically related synsets: {*robin, redbreast*} @→ {*bird*} @→ {*animal, animate_being*} @→ {*organism, life_form, living_thing*}, for example, where the brackets indicate a synset and @→ is the transitive, asymmetric, semantic relation that can be read 'IS-A' or 'IS-A-KIND-OF'. (By convention, @→ is said to point upward.) This design creates a sequence of levels, or a hierarchy, going from many specific terms at the lower levels to a few generic terms at the top.

6. Text Feature Extraction Using IBM Intelligent Miner

In this module (see **Figure 2** for Intelligent Miner for Text), we exploit feature extraction tool in the IBM intelligent miner for text with various input parameters and options. HTML page is fed to the miner as the input. The miner produces all the features that have been extracted from the html page according to the parameters given. This project allow

users to configure the extraction tool by having the option of choosing what type of features they need to be extracted, such as words, multiword terms, names, and/or abbreviations.

6.1. IBM Intelligent Miner Feature Extraction tool

IBM Intelligent Miner for text feature extraction tool provides the means to extract features from a web page. It accepts textual documents in ASCII or HTML format. It allows extraction of features of various types such as single or multiword terms, names and abbreviation.

6.2. Implementation of IBM Intelligent Miner Feature Extraction tool

This module has been implemented in Perl. The module checks to see if the input is URL or a file. If it is URL, using the *Perl LWP*, a set of Perl modules that provides an application programming interface (API) to the world wide web, the module fetches the html page if it exists. The API sends a request that contains the URL name to the remote file server and gets the response. As a first step, we instantiate a *user agent*, an object that represents the application on the network and provides an interface that accepts requests for fetching web pages and returns responses to the application. *LWP* has a class name *HTTP::Request* that communicates with the user agent. The main attributes of the request object:

- The **method** that tells what kind of a request this is. The method we used is *GET* (get the web page).
- The **uri**, a string comprising the name of the URL to be accessed.

The request object sends the request to the user agent and gets a response back. The response contains the content of the document, which is then printed out to a file “*source.txt*”. However, if the input is an existing file, the module directly feeds the file to the miner. In both cases, the miner produces a list of extracted features together with the number of occurrences for each feature.

Before sending the content to the miner, our module parses the input using Perl *HTML::Tokeparser*, an interface that parses an html document. We use this module to extract words that appear in the Title or Meta, a functionality that is not supported by Intelligent Miner (extracting words in specific html elements). An *HTML::Tokeparser* object is instantiated passing the filename as an argument. The module invokes *get_token* method to read tokens from html document. If the token has a tag Meta or Title, the text comprised in that html element is extracted into an array.

After the input has been sent to the miner, the miner stores the features extracted in a file, “*features.htm*”. For each feature from the miner we add the weights computed using

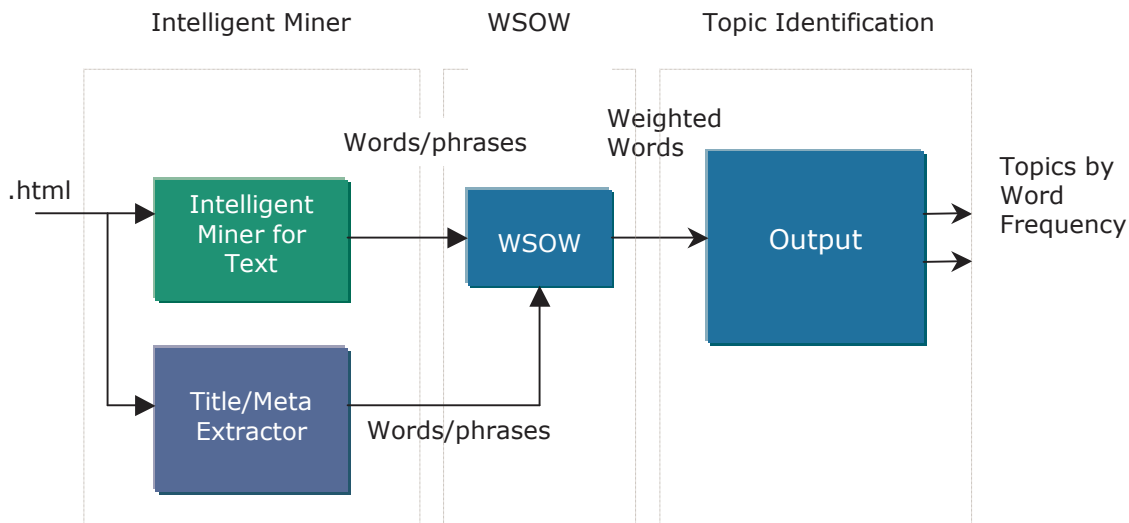


Figure 2. Architecture for Intelligent Miner module.

WSOB. Then the features are stored in a hash with features as keys and frequency as the values. The hash is sorted in descending frequency and the top five words or phrases are selected to represent the keywords of the web page.

7. Experiments and Performance Study

Experiments were performed on HTML web pages. We conducted experiments by selecting web pages from different categories ranging from Vegetables, Tornado, Earthquake, English writing tips, Science, Home stuffs, Travel, Money related web pages. We have listed all the tested web pages in the below table with the explanation of what is that particular web page is about along with the topics identified.

We also studied the quality of topics produced when WordNet is used. We have some limitations that were explained below.

7.1. Test Cases for Topic Categorization using NLP:

No.	HTML Web Pages	CS*	Pages about	Topics Identified
1.	http://www.urbanext.uiuc.edu/veggies/	Yes	Vegetables	Vegetable
2.	http://www.spc.noaa.gov/faq/tornado/	Yes	Tornado	Atmospheric_phenomenon
3.	http://www.agso.gov.au/urban/factsheets/20010919_15.jsp	Yes	Earthquakes	Natural_phenomenon
4.	http://ranger.uta.edu/~alp	Yes	Home Page	Alp
5.	http://lorien.ncl.ac.uk/ming/Dept/Tips/writing/writeindex.htm	Yes	Writing Tips	Communication
6.	http://www.oreilly.com/catalog/oraclescrp/chapter/ch01.html script -- (a particular orthography or writing system)	Yes	Script	Communication
7.	http://www.trolleycars.com/archaeology.htm	Yes	Trolley cars	Instrumentality
8.	http://www.lord.ca/publications/articles/museum_planning_urban_engagement.html	Yes	Museum Planning	Museum
9.	http://science.howstuffworks.com/virus-human.htm/printable	Yes	Human Virus	Virus_thing
10.	http://science.howstuffworks.com/cruise-missile.htm/printable	Yes	Cruise Missile	Instrumentality
11.	http://science.howstuffworks.com/telescope.htm/printable	Yes	Telescope	Device
12.	http://science.howstuffworks.com/caffeine.htm/printable	Yes	Caffeine	Substance
13.	http://science.howstuffworks.com/mosquito.htm/printable	Yes	Mosquito	Mosquito (wsow)
14.	http://science.howstuffworks.com/brain.htm/printable	Yes	Human Brain	Body part
15.	http://science.howstuffworks.com/laughter.htm/printable	Yes	Laughter	Communication
16.	http://science.howstuffworks.com/iron.htm/printable	Yes	Iron	Substance
17.	http://science.howstuffworks.com/helicopter.htm/printable	Yes	Helicopter	Instrumentality

18.	http://science.howstuffworks.com/zipper.htm/printable	Yes	Zipper	Device
19.	http://home.howstuffworks.com/fax-machine.htm/printable	Yes	Fax-Machine	Instrumentality
20.	http://home.howstuffworks.com/burglar-alarm.htm/printable	Yes	Burglar-Alarm	Object
21.	http://home.howstuffworks.com/doorbell.htm/printable	Yes	Door-Bell	Switch
22.	http://auto.howstuffworks.com/odometer.htm/printable	Yes	Odometer	Device
23.	http://auto.howstuffworks.com/radar-detector.htm/printable	Yes	Radar-Detector	Instrumentality
24.	http://money.howstuffworks.com/patent.htm/printable	Yes	Patent	Written_communication
25.	http://money.howstuffworks.com/bank.htm/printable	Yes	Bank	Organization
26.	http://money.howstuffworks.com/lottery.htm/printable	Yes	Lottery	Event
27.	http://money.howstuffworks.com/mortgage.htm/printable	Yes	Mortgage	Possession
28.	http://travel.howstuffworks.com/rock-climbing.htm/printable	Yes	Rock-Climbing	Climbing(wsow)
29.	http://travel.howstuffworks.com/bicycle.htm/printable	Yes	Bicycle	Cycle
30.	http://home.howstuffworks.com/cholesterol.htm/printable	Yes	Cholesterol	Cholesterol (wsow)

CS* → Condition Satisfied (Yes / No)

(WSOW) → represents the concepts got by web structure oriented weighting alone

7.2. For the test case 3, the snap shot of the output is shown below

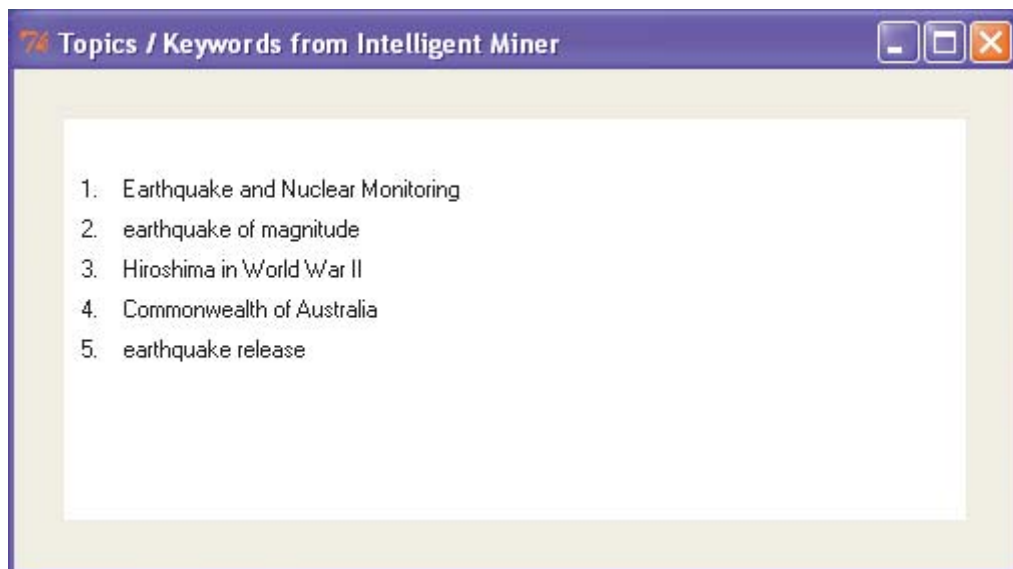


Figure 3. Keyword/Phrases using IBM Intelligent Miner.

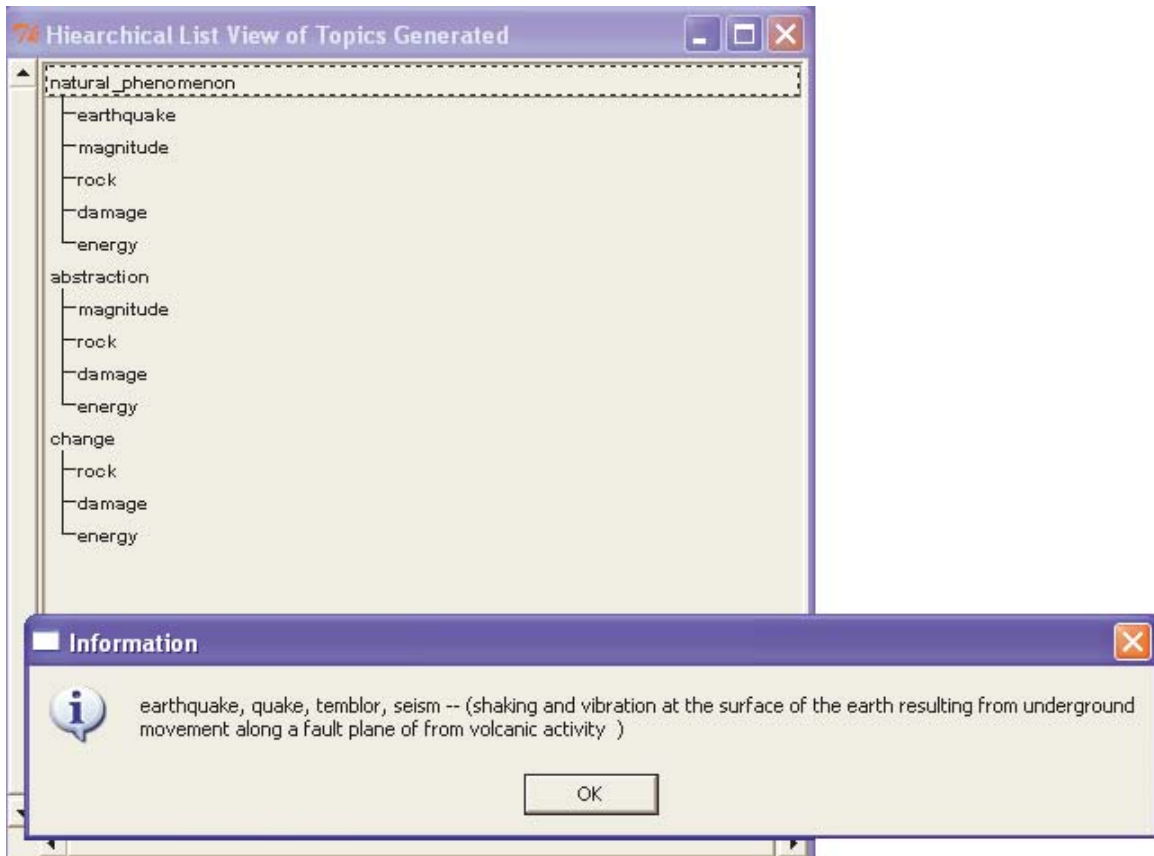


Figure 4. Concept Hierarchy using NLP

- Generating concepts was explicitly dependable on WordNet lexical database tool. WordNet couldn't recognize names such as a name of a person. To handle this problem, we provide the option of selecting the word as a possible topic if it has not been recognized by WordNet.

9. Conclusion

We have presented a method for topic identification that can be used for categorizing web pages and similar structured documents. Our approach relies on part-of-speech tagging, HTML parsing, structural term weighting, and concept expansion through Wordnet. Our experimental results show that the approach is promising and can be adapted to many categorization tasks.

10. References

- [1] Exploiting Text Structure for Topic Identification <http://acl.ldc.upenn.edu/W/W96/W96-0109.pdf>
- [2] <http://www.sims.berkeley.edu/research/projects/how-much-info/internet.html>
- [3] Topic Identification in Discourse <http://acl.ldc.upenn.edu/E/E95/E95-1037.pdf>
- [4] TopCat: Data Mining for Topic Identification in a Text Corpus (2002) <http://citeseer.nj.nec.com/499751.html>
- [5] Web Document Classification based on Hyperlinks and Document semantics <http://citeseer.nj.nec.com/kuo00web.html>
- [6] Identifying Topics by Position <http://acl.ldc.upenn.edu/A/A97/A97-1042.pdf>
- [7] Sentence extraction as a classification task <http://citeseer.nj.nec.com/teufel97sentence.html>
- [8] Future Selection for Web Page Classification <http://citeseer.nj.nec.com/554644.html>
- [9] WordNet: an electronic lexical database (Book)
- [10] Lin, Chin-Yew. *Knowledge-based Automatic Topic Identification*. In proceeding of the 33rd Annual Meeting of the Association for Computational Linguistics '95. (1995)