Hybrid text mining for finding abbreviations and their definitions

Youngja Park, Roy J. Byrd
IBM Thomas J. Watson Research Center
P.O. Box 704, Yorktown Heights
New York 10598, USA
{pyoungja, roybyrd}@us.ibm.com

Abstract

We present a hybrid text mining method for finding abbreviations and their definitions in free format texts. To deal with the problem, this method employs pattern-based abbreviation rules in addition to text markers and cue words. The pattern-based rules describe how abbreviations are formed from definitions. Rules can be generated automatically and/or manually and can be augmented when the system processes new documents. The proposed method has the advantages of high accuracy, high flexibility, wide coverage, and fast recognition.

Introduction

Many organizations have a large number of on-line documents -- such as manuals, technical reports, transcriptions of customer service calls or telephone conferences, and electronic mail -- which contain information of great potential value. In order to utilize the knowledge these data contain, we need to be able to create common glossaries of domain-specific names and terms. While we were working on automatic glossary extraction, we noticed that technical documents contain a lot of abbreviated terms, which carry important knowledge about the domains. We concluded that the correct recognition of abbreviations and their definitions is very important for understanding the documents and for extracting information from them [1, 6, 9, 11].

An abbreviation is usually formed by a simple method: taking zero or more letters from each word of its definition. However, the tendency to make unique, interesting abbreviations is growing. So, it is easy to find new kinds of abbreviations which cannot be processed by hard-coded heuristics-based algorithms [1, 6, 7, 13, 14], since they are formed in ways not anticipated when the algorithms were devised.

We propose a hybrid text mining approach to deal with these problems. We use three kinds of knowledge: pattern-based abbreviation rules, text markers, and linguistic cue words. An abbreviation rule consists of an abbreviation pattern, a definition pattern and a formation rule. The formation rule describes how an abbreviation is formed from a definition. There may exist multiple formation rules for a given pair of abbreviation and definition patterns. Abbreviation rules are described in Section 3.

Text markers are special symbols frequently used to imply the abbreviation-definition relationship in texts. They include characters such as '(…)', '[…]', and '='. Cue words are particular words occurring in the local contexts of abbreviations and the definitions, which strongly imply the abbreviation relationship. They include words such as "or", "short", "acronym" and "stand". Text markers and cue words are discussed in section 2.4.

This system has 5 components -- abbreviation recognizer, definition finder, rule applier, abbreviation matcher and best match selector -- as shown in Figure 1. The abbreviation recognizer seeks candidate abbreviations in a text and generates their patterns (Section 1). When an abbreviation candidate is found, the system determines the contexts within which to look for a definition. When it finds a candidate definition, it generates a pattern for it also (Section 2).

Having generated the abbreviation pattern and the definition pattern, the system first searches the rulebase for a rule which would

generate the abbreviation from the definition. The rules for the given candidates are applied in the order of rule priorities (Section 4.1). If the rulebase is empty or if no existing rule matches the candidate abbreviation with the candidate definition, the system runs the abbreviation matcher and generates a new abbreviation rule. The abbreviation matcher consists of 5 layered matching algorithms (Section 4.2). If the matcher succeeds, new rules may be added to the rulebase, allowing it to grow as the system processes new documents.

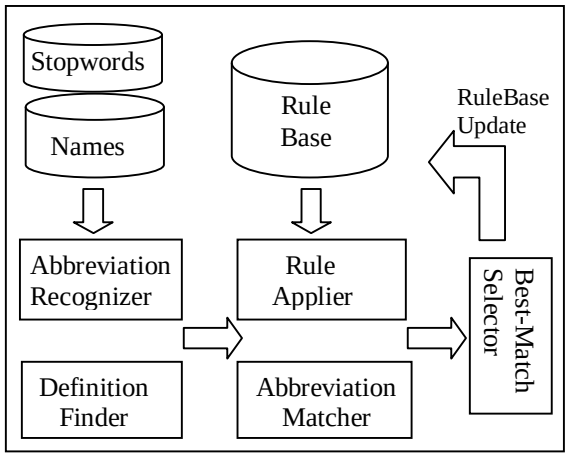

Fig. 1. System Overview

1. Abbreviation Recognition

1.1 Candidate Abbreviations

An abbreviation is defined as a shortened form of a written word or phrase used in place of the full form [2]. Acronyms are a special case of abbreviations which are devoted to multi-word full forms. In this work, we consider a string of alphabetic, numeric and special characters as a candidate abbreviation if it satisfies the following three conditions:

(1) Its first character is alphabetic or numeric
(2) Its length is between 2 and 10 characters
(3) It contains at least one capital letter

and if the string meets the following restrictions:

(1) It is not a known (dictionary) word containing an initial capital letter and appearing as the first word in a sentence.
(2) It is not a member of a predefined set of person and location names.
(3) It is not a member of user-defined list of stopwords.

The first restriction keeps many common words from being treated as possible abbreviations. Many proper names have the same characteristics above and may be recognized as abbreviations. To reduce generating false candidates and thus improve system performace, we use a list of proper names created by the Talent system[3, 10]. We also provide users with a way to create a user defined list of stopwords.

Based on these conditions, AI (Artificial Intelligence), Baracuda (Boldly Advanced and Refined Aircraft Concept Under Development for AGATE), SgRP (seating reference point), 2MASS (Two-Micron All Sky Survey), ACIS (Advanced CCD Imaging Spectrometer), W3C (World Wide Web Consortium), T/C/F (Trim/Chassis/Final) are recognized as candidate abbreviations.

1. 2 Abbreviation Patterns

Once a candidate abbreviation is identified an abbreviation pattern is generated from it. An abbreviation pattern is a string of ‘c’ and ‘n’ characters. An alphabetic character is replaced with a ‘c’ and a sequence of numeric characters (including ‘.’ and ‘,’) is replaced with an ‘n’ regardless of its length. Non-alphanumeric characters such as hyphen, slash, and ampersand are not reflected in abbreviation patterns. Some examples of candidate abbreviations and their patterns are in Table 1.

Table 1 Abbreviation Patterns

Abbreviations	Patterns
2MASS	ncccc
NEXT	cccc
R&D	cc
SN1987A	ccnc
T/C/F	ccc
V3.5	cn

2. Definition Finding

2.1 Search Space

This system searches for a possible definition of a candidate abbreviation in its left and right contexts. The size of the search space is a function of the length of the abbreviation and the

maximally allowed distance (offset) between a definition and its abbreviation.

We have analyzed about 4,500 abbreviations and their definitions in computer science texts. The maximum number of skipped words (words in a definition that are not reflected in the abbreviation) was 4 in our sample data. Based on this analysis, we decided that, for relatively short abbreviations (from two to four characters), the length of a definition in words should not be greater than twice the abbreviation length. For long abbreviations (five or more characters), the definition should not be longer than the abbreviation length plus 5. Thus, the maximum length of a definition D of an abbreviation A is calculated as follows.

$$\text{max. } |D| = \min \{|A| + 5,\ |A| * 2\}$$

The maximum offset means the longest distance of a definition from an abbreviation. If a definition is in the left context, the distance is the number of words from the last word of the definition to the abbreviation. If a definition is in the right context, the distance is the number of words from the abbreviation to the first word of the definition. We set the maximum offset to 10 in this experiment. Therefore, the size of each search space is {max. |D| + 10} words to the left and right of the candidate abbreviation as shown in Fig 2.

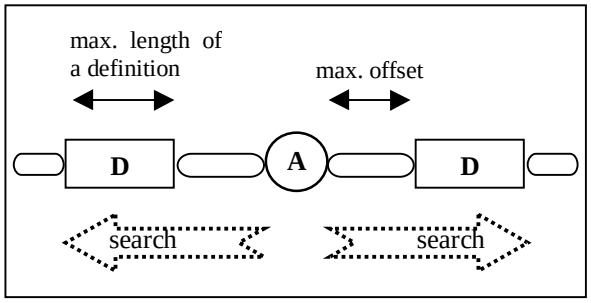

Fig. 2 Search Space for Definitions

2.2 Definition Search

The system searches for candidate definitions within the search space. A sequence of words in the contexts is considered as a candidate definition if it satisfies the following conditions.

(1) The first character of the first word of a definition is matched with the first character of the abbreviation (including 'replacement match (Section 3)').
(2) All words in a definition are in the same sentence.
(3) The first word and the last word of a definition are not prepositions, be-verbs, modal verbs, conjunctions or pronouns.
(4) Some symbols such as (,), [,], {, }, =, !, ? may not be inside of a definition.

2.3 Definition Patterns

Next, we preprocess the candidate definition as follows to generate a pattern for the candidate.

(1) replace special symbols with spaces.
Input/Output => Input Output
(2) separate strings of numerical characters.
Windows98 => Windows 98
(3) separate prefixes[1] from the headword
reusable => re usable

A definition pattern consists of the characters 'w' (word), 's' (stopword), 'p' (prefix), 'h' (headword) and 'n' (number). Some examples of definitions and their patterns are in Table 2.

Table 2 Definition Patterns

Definitions	Patterns
Product Database	phw
Supernova 1987A	phnw
Trim/Chassis/Final	www
Two-Micron All Sky Survey	wwwww
U.S. Department of Agriculture	wwsw

In the above examples, the definition pattern for 'product database' is 'phw', which is not morphologically correct. This happens because 'pro' is included in our prefix list and 'duct' is found in the dictionary and we don't do any semantic-level processing.

2.4 Syntactic Cues

We extract some orthographic and syntactic structure information as well as possible definitions from the contexts. If there exist text markers and/or cue words in the contexts of a candidate abbreviation and its candidate definition, the pair is highly likely to be valid and to be useful for augmenting the abbreviation

[1] We currently have 60 prefixes such as anti, bi, electro, inter, pre, sub, trans, un.

rulebase. The structures we take into account include:

(1) **(**abbr**)** or **[**abbr**]**
(2) **(**definition**)** or **[**definition**]**
(3) abbr = definition
(4) definition = abbr
(5) abbr***, or*** definition
(6) definition***, or*** abbr
(7) abbr … ***stands/short/acronym*** … definition
(8) definition, abbr ***for short***

3. Abbreviation Rules

3.1 Abbreviation Rule Format

An abbreviation rule describes how an abbreviation is formed from its definition. An abbreviation rule, R, consists of an abbreviation pattern (A_Pattern), a definition pattern (D_Pattern) and a formation rule (F_Rule).

R = <A_Pattern, D_Pattern, F_Rule>

A formation rule defines how each character in an abbreviation is formed from a definition. An element in a formation rule has a pair of values, a word number and a formation method. A word number is the sequential location of a word within the preprocessed definition. A formation method represents how a character (or characters) in the word takes part in the abbreviation.

We have defined five kinds of formation methods: 'F', 'I', 'L', 'E' and 'R'. 'F' means that the first character of a word occurs in the abbreviation. Similarly, 'I' refers to an interior character and 'L' indicates the last character of a word. 'E' means 'exact match' and 'R' means 'replacement match'. Exact match ('E') is used only for matching numbers. If both a candidate abbreviation and a candidate definition contain a string of numeric characters, the two numbers must be exactly same. For example, "V3" is not an abbreviation of "Version 3.5" but may be an abbreviation of "Version 3". Replacement match ('R') is for recognizing multi-characters abbreviated with one character. In our current list of replacement matches, 'x' in an abbreviation may stand for "hex", "ex", "trans", or "cross"; 'f' may stand for "ph"; '1' may stand for "first", "one", or "1st"; and so on. Some examples of abbreviation rules are listed in Table 3.

In one example, the abbreviation rule for the abbreviation "NEXT" and its definition "Near-End CrossTalk" is <cccc, www, (1, F) (2, F) (3, R) (3, I)>. The definition is encoded in "www" because "Near-End" is divided in two words "Near" and "End. This rule means that the first character in the abbreviation 'N' is formed from the first letter of the first word '(1, F)'; the second character 'E' is from the first letter of the second word '(2, F)'; the third character 'X' is formed by a replacement match within the third word ("Cross" is replaced into 'X'); and 'T' comes from an interior character of the third word.

Table 3 Abbreviation Rule Expamples

2-MASS Two-Micron All Sky Survey <ncccc, wwwww , (1,R) (2,F) (3,F) (4,F) (5,F)>
CFCs chlorofluorocarbons <cccc, w, (1,F) (1,I) (1, I) (1, L) >
CONTOUR Comet Nuclear Tour <ccccccc, www, (1,F)(1,I)(2,F)(3,F)(3,I)(3,I)(3,L) >
NEXT Near-End CrossTalk <cccc, www, (1,F) (2, F) (3, R) (3, I)>
SN1987A Supernova 1987A <ccnc, phnw, (1, F) (2, F) (3, E) (4, F)>
TAXI Transparent Asynchrous Transceiver Interface <cccc, wwww, (1,F) (2,F) (3,R) (4,F)>
X2B Hexadecimal to Binary <cnc, phsw, (1,R) (3, R) (4, F)>
W3C World Wide Web Consortium <cnc, wwww, (1,F) (2,F) (3,F) (4,F)>

3.2 The Initial RuleBase

We constructed an initial rulebase from our analysis of 4,500 abbreviations in the field of computer science, which were collected from the Web. We ran the Abbreviation Matcher routine to generate patterns and formation rules for the abbreviations and their definitions and selected frequent rules for the initial rulebase. The initial rulebase currently contains 45 abbreviation rules, some of which are shown in Table 4.

4. Matching Abbreviations and Definitions

4.1 Rule Application

When the system has found a candidate abbreviation with a candidate definition, it generates the A_pattern and the D_pattern,

Table 4 Initial RuleBase

Pattern		Formation Rule
A_Pat	D_Pat	
cc	wph	(1,F)(2,F)
cc	wsw	(1,F)(3,F)
cc	ww	(1,F)(2,F)
ccc	phww	(1,F)(3,F)(4,F)
ccc	wphw	(1,F)(2,F)(4,F)
ccc	wsw	(1,F)(2,F)(3,F)
ccc	wsww	(1,F)(3,F)(4,F)
ccc	wwph	(1,F)(2,F)(3,F)
ccc	wwsw	(1,F)(2,F)(4,F)
ccc	www	(1,F)(2,F)(3,F)
ccc	www	(1,R)(2,F)(3,R)
cccc	phwsww	(1,F)(3,F)(5,F)(6,F)
cccc	wphww	(1,F)(2,F)(4,F)(5,F)
cccc	wsww	(1,F)(2,F)(3,F)(4,F)
ccn	wwn	(1,F)(2,F)(3,E)
cnc	wsw	(1,F)(2,R)(3,F)
ncc	www	(1,R)(2,F)(3,F)

respectively, and then looks up the pattern pair in the rulebase. If the pair exists, the system applies the associated formation rules in priority order. If any rule can generate the given abbreviation from the definition, the pair is regarded as valid.

Suppose, for example, that the abbreviation "5GL" and the definition "fifth generation language" are found in a text. The system preprocesses them and generates their patterns. In this case, the A_Pattern is "ncc" and the D_Pattern is "www". A formation rule <(1, R) (2, F) (3, F)> is associated with this pattern pair in the rulebase. Thus, the system applies the rule to determine the validity of the abbreviation/definition pair. The first word ('fifth') can be replaced into '5' [(1, R)]; the first character of the second word is 'G' [(2, F)]; and the first character of the third word is 'L' [(3, F)]. Hence the pair is valid and 'fifth generation language' is considered to be a definition of '5GL'.

4.2 The Abbreviation Matcher

If the rulebase does not have rules for the pattern pair or if no rule successfully generates the abbreviation from the definition, and if the pair occurs in one of the cue environments described in section 2.4, the system activates the Abbreviation Matcher routine. This routine is also used for creating the initial rulebase.

The Abbreviation Matcher contains five layered matching algorithms. We categorized abbreviations into five different types, one for each layer, based on the relationship between the abbreviation length and the length of the corresponding definition. Abbreviations of type 1 are the most frequent in our 4,500 item sample. Type 2 is the next most frequent, and so on.

(1) Layer 1 : |A| = |D|
EDS, Electronic Data System
MPH, miles per hour
2MASS, Two Micron All Sky Servey

(2) Layer 2 : |A| = |D| - |S|
F&M, Facilities and Materials
ASCA, Advanced Satellite for Cosmology and Astrophysics

(2) Layer 3 : |A| < |D|
4GT, 4 Gigabyte memory Tuning
FAME, Full Sky Astrometric Mapping Explorer

(3) Layer 4 : |A| > |D|
DB, DataBase
CONTOUR, Comet Nuclear Tour

(4) Layer 5 : special numerals
W3C, World Wide Web Consortium
D2T2, Dye Diffusion Thermal Transfer

|A| represents the length of an abbreviation pattern. |D| is the length of a definition pattern. |S| indicates the number of stopwords in a definition.

This system processes an <A, D> pair by applying the algorithms in layer order, beginning at layer 1. If the pair is matched at any layer, matching stops and the successful formation rule is returned. If a match is not found at any layer, the candidate pair is discarded.

4.3 Best Match Selection

The system may generate multiple definition candidates in many cases, but we assume that there exists only one definition in a local context. In order to select the best candidate, we employ several weighting features.

(1) syntactic cues
A definition has a higher weight than other candidates if it has syntactic cues.

(2) rule priority
A definition has a higher weight if it was matched by a higher priority rule.

(3) distance
The definition closest to the abbreviation is favored over other candidate definitions.

(4) capitalization
A definition with initial capital letters is preferred.
(5) number of words
A definition is preferred based on the following sequence of length comparisons: |A| = |D|, |A| < |D| and |A| > |D|.
(6) number of stopwords
A definition having fewer stopwords is preferred.

If multiple candidate definitions are found in a document for an abbreviation, these six features are tested on the definitions in the order given, until one definition remains. If ambiguity still remains at the end of the test, the first definition is selected.

Users can specify whether they want to update the rulebase with the results of processing a new document. If an existing rule successfully matches an abbreviation/definition pair, then that rule's frequency is updated in the rulebase, thereby increasing the rule's priority. Users may also specify a rule threshold; new rules which occur with a frequency exceeding the threshold will be added to the rulebase.

5. Experiments and Results

We have conducted experiments with three documents: a book about automotive engineering (D1), a technical book from a pharmaceutical company (D2), and NASA press releases for 1999 (D3). The data used in the experiments and experimental results are shown in Table 5. Performance is evaluated using recall and precision.

Table 5 Test Data and Experimetal Results

		D1	D2	D3
Size (# of words)		20,379	97,000	83,539
No. of Abbreviations		33	63	81
Found	Correct	31	60	76
	Incorrect	1	0	2
	Total	32	60	78
Missed		1	3	5

For D1, the system found 32 abbreviations and their definitions but among them 1 abbreviation is incorrect. Thus, it shows 93.9% recall and 96.9% precision. For D2, it found 60 pairs and missed 3 pairs showing 95.2% recall and 100% precision. For D3, it found 78 pairs with 2 incorrect results and missed 5 pairs. The recall rate is 93.8 % and precision is 97.4 %.

The reasons for missing some abbreviations are (a) the definitions fell outside of the search space (b) misinterpretation by the part-of-speech tagger (c) matches beyond system's current capability. Some examples of missed abbreviations are:

(1) DEHP di-2-ethylhexylphthalate
(2) ALT alanine aminotransferase
(3) ASI Italian Space Agency
(4) MIDEX medium-class Explorer
(5) CAMEX-3 Third Convection and Moisture Experiment

For (1), we would need to add the domain-specific prefixes "ethyl and "hexyl" to the prefix list. In general, adaptation of our method to new technical domains will probably involve the addition of domain-specific prefixes to the prefix list. (2) failed because there was no first letter match for "aminotransferase". The abbreviation in (3) is an acronym of the Italian translation of the definition. In (4), there is no credible source for the "I" in the abbreviation. In (5), the numeric replacement in the abbreviation is permuted. These and other phenomena such as compound word processing will be the subject of further investigation.

6. Related Work

AFP (Acronym Finding Program) is an early attempt to automatically find acronyms and their definitions in free text [13]. In this work, however, an acronym candidate is simply an upper-case word from 3 to 10 characters in length. AFP looks for candidate expansions in two sub-windows – the pre-window and the post-window - of the acronym by applying an LCS (longest common subsequence) algorithm. Each subwindow's length in words is set to twice the number of characters in the acronym and it looks for matching letters occurring at word beginnings or after hyphens.

However, AFP does not support 2-letter acronyms that are very common in texts (e.g., AI, DB, and IP) and it does not allow interior-

letter matches that are not uncommon in abbreviations.

TLA (Three Letter Acronym) [14] removes all non-alphabetic characters and breaks the text into chunks based on the occurrences of '(', ')' and '.' characters. It looks for candidate acronyms in each chunk and attempts to find matching definitions in the preceding and following chunks. Candidate acronyms and candidate definitions are compared by matching up to the first three letters of each word in the chunks. The potential matches are passed through a number of ad-hoc heuristics below, each of which can reject any candidate acronyms.

- Acronyms are shorter than their definitions
- Acronyms contain initial characters of most of the words in their definitions
- Acronyms are given in upper case
- Shorter acronyms tend to have longer words in their definition
- Longer acronyms tend to have more stop words

As part of a larger study of the topology of relations across the World-Wide Web, Sundaresan and Yi [12] explore specific relations involving acronyms and their definitions. Similar to other work on mining the Web for relations (e.g., Kleinberg [5] for hyperlinks and Larson [8] for bibliometrics), their work uses duality-based methods to build networks of interrelated syntactic cues, acronym-definition pairs, and formation rules. It develops iterative techniques for finding new acronym-definition pairs, given a set of syntactic cues, and for finding new syntactic cues, given a set of known pairs. It can also learn new formation rules.

While the overall system frameworks are quite different, our hybrid text mining method and the duality-based method both use similar underlying machinery: syntactic cues, abbreviation-definition pairs, and formation rules. Differences include the hybrid method's use of a more abstract representation for formation rules, the central use of abbreviation patterns and definition patterns as the organizing principle for the rule base, and the use of cue words among the syntactic cues.

The developers of the Acrophile system at UMass Amherst [7] evaluated four different acronym extraction algorithms against manually-analyzed test documents and against hand-crafted acronym dictionaries. Their "canonical-contextual" algorithm, which shares elements with our hybrid method, was the most successful one. In particular, Acrophile uses a fixed 40-word search space for their "contextual" definition search and has a set of syntactic cues similar to ours for defining the "canonical" environments in which abbreviation-definition pairs may be found.

Beyond special handling for numeric characters in acronyms, however, there is no provision for replacement matches; for explicit lists of prefixes, known words, and proper names; or for adaptively learning new acronym patterns. Acrophile's system environment and experimental results are quite interesting; by directed search of the World-Wide Web, the system was able to exceed the coverage of the largest publicly available hand-crafted on-line acronym dictionary.

Conclusions and Future Work

We have introduced a new hybrid approach for finding abbreviations and their definitions in unstructured texts. The problem of abbreviation processing has attracted relatively little attention in NLP field. However, technical documents use a lot of abbreviations to represent domain-specific knowledge. Thus, the ability to find correct abbreviations and their definitions is very important to being able to utilize the information contained in those documents. It is also very useful for many NLP applications such as information retrieval [1] and glossary extraction [4, 9, 11].

The proposed method has the following advantages:

(1) It is simple and fast.
A small number of formation rules can describe many abbreviations. By keeping these rules in the rulebase, this system can process most abbreviations by simple pattern matches. Furthermore, the abbreviation matcher consists of 5 simple match routines and each routine is dedicated

to a certain type of abbreviations. Thus, it is conceptually simple and fast.

(2) It shows high recall and precision rates.

(3) It provides for flexible user customization. For example, users can specify rule thresholds for updating the rulebase.

(4) It is trainable. The rulebase may be automatically refined as the system processes new documents.

(5) It is adaptable to new styles and editorial conventions. It can process new types of abbreviations by inserting appropriate rules in the rulebase without modifying the system. Rules are symbolic, so users can easily add, modify, or delete the rules by hand.

(6) It can be adapted to new technical domains. The dictionary, set of replacement matches, stopword list, and prefix list, can be tailored for new domains.

In addition to the lacunae mentioned in Section 5, we are aware that there are classes of abbreviations which our current method does not handle adequately. These are typically written with all lower-case characters and are almost never introduced with text markers or cue words. Examples are :

- cu – customer
- hw – hardware
- mgr – manager
- pgm – program
- sw – software

Mechanisms for processing these abbreviations, which tend to occur in informal text such as email, chat rooms, or customer service call records, are the subject of ongoing research in our project.

References

[1] Byrd, Roy, Yael Ravin, and John Prager. Lexical Assistance at the Information-Retrieval User Interface. *Research Report, RC19484, IBM T.J. Watson Research Center*, 1994.

[2] G. & C. Merriam Co. Webster's New Collegiate Dictionary. *1981*

[3] IBM T. J. Watson Research. The Talent (Text Analysis and Language Engineering) project. http://www.research.ibm.com/talent/. 2001.

[4] Justeson, John and Slava Katz. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering, 1(1):9-17*, 1995.

[5] Kleinberg, Jon. Authoritative sources in a hyperlinked environment, In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, May 1997.

[6] Kugimiya, Shuzo, Yoji Fukumochi, Ichiko Sata, Tokyuki Hirai, and Hitoshi Suzuki. Machine Translation apparatus having a process function for proper nouns with acronyms. *US Patent 5161105, 1992*

[7] Larkey, Leah, Paul Ogilvie, Andrew Price and Brenden Tamilio. Acrophile:An Automated Acronym Extractor and Server, In *Proceedings of the ACM Digital Libraries conference*, pp. 205-214, 2000.

[8] Larson, R. Bibliometrics of the World-Wide Web: An exploratory analysis of the intellectual structure of cyberspace, *Technical Report, School of Information Management and Systems, University of California, Berkeley*, 1966. http://sherlock.sims.berkeley.edu/docs/asis96/asis96.html.

[9] Maynard, Diana and Sophia Anaiadou. Term Extraction using a Similarity-based Approach. In *Recent Advances in Computational Terminology*, John Benjamins, 1999.

[10] Ravin, Yael, Nina Wacholder and Misook Choi. Disambiguation of proper names in text. *17th Annual ACM-SIGIR Conference, 1997.*

[11] Roark, Brian and Eugene Charniak. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pp 1110-1116, 1998.

[12] Sundaresan, Neel and Jeonghee Yi. Mining the Web for Relations, In *The Ninth International World Wide Web Conference*, 2000. http://www9.org/w9cdrom/363/363.html.

[13] Taghva, Kazem and Jeff Gilbreth. Recognizing Acronyms and their Definitions, *Technical Report 95-03, Information Science Research Institute, University of Nevada, Las Vegas*, June 1995.

[14] Yeates, Stuart. Automatic Extraction of Acronyms from text. In *Proceedings of the Third New Zealand Computer Science Research Students' Conference*. Pp 117-124, 1999.