International Journal For Technological Research In Engineering
Volume 1, Issue 9, May-2014 *ISSN (Online): 2347 - 4718*

Comparison of Algorithms Used to make Automatically Identification of Acronyms-Definitions Pairs Systems in English language

Manpreet Kaur Aulakh[1], Jagroop Kaur[2] (Asst. Prof)
[1]Computer Science Department, Punjabi University, Patiala, Malout, India.
[2]Department of Computer Science, UCOE, Punjabi University, Patiala, India.

Abstract: This paper addresses the task of finding acronym-definition pairs in text. One such challenge derives from the common and uncontrolled use of acronyms in the literature. Each additional acronym increases the effective size of the vocabulary for a field. Therefore, to create an automatically generated and maintained lexicon of acronyms, various algorithms have been developed to match acronyms in text with their expansions. In this paper we will give review of AFT (Acronym Finding Program), TLA (Three Letter Acronym), Using simple algorithms with adding constraints, Rule-based method approaches used to extract acronyms and their expansions from text. In these Approaches performance measured using recall and precision.

I. INTRODUCTION

Acronym is word formed from the initial letter or letters of each of the successive parts or major parts of a compound term. Acronyms are a subset of abbreviations and are generally formed with capital letters from the original word or phrase, however many acronyms are realized in different surface forms i.e. use of Arabic-numbers, mixed alpha-numeric forms, low-case acronyms etc.

Properties of acronyms and expansions:
Acronyms are often defined by preceding (or following) their first use with a textual explanation. Acronyms are used in place of their expansions, where either reader is familiar with the concepts and entities they stand for or their meaning is clear from the context. Acronyms have following special properties:

- Generally, acronyms do not appear in standard dictionaries. To make their meaning clear, authors may give their expansions at their first use.
- Acronyms may be nested.
- Acronyms are not necessary unique.

Acronyms are generally three to ten characters in length, although shorter or longer acronyms do exist. Acronyms' characters are alphabetic, numeric, and special characters such as '-', '/', '.' or '&' etc. White spaces rarely appear. Key differences between acronyms and other abbreviations include the lack of symbols such as apostrophe (') and full stops (.) in acronyms, more standard construction and the use of capital letters. Can't and etc. are abbreviations but not acronyms, in the first case both because of the inclusion of other than initial letters and because of the inverted comma (') and the in second case because of the use of a (.), both lack of capital letters. Acronym lists are available from a number of sources, but these are static—they list acronyms current in some domain at the time of compilation or officially in use in a domain or organisation. While these may be of use in specific organisational or domain they are unlikely to be useful for an arbitrary piece of text at some point in the future. Abbreviations such as acronyms are used in places where either reader are familiar with the concepts and entities they stand for or their meanings are clear from the context of the discussion. Unlike other abbreviations, acronyms are usually introduced in a standard format when used for the first time in a text. Acronyms are not necessarily unique Acronym identification is the task of processing text to extract pairs consisting of a word (the acronym) and an expansion (the definition), where the word is the short form of (or stands for) the expansion. In this work, we do not discriminate between acronyms (short forms of multiword expressions) and abbreviations (contractions of single words). We use the term *acronym* to include both cases. we tackle the core task only. That is, given an input text, our algorithm will attempt to extract all explicit acronym-definition pairs. Our goal is to create a dictionary of acronym-definition pairs specific to a single text. 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. 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. Two concepts, distinguishing Global and Local abbreviations .Global abbreviations are not defined within the document, similar to common abbreviation. Local abbreviations appear in the document alongside the long form, similar to dynamic abbreviations.

II. OBJECTIVES

Our Objective is to create automatic dictionary of acronym-definition pairs in given text. That is, given an input text, our

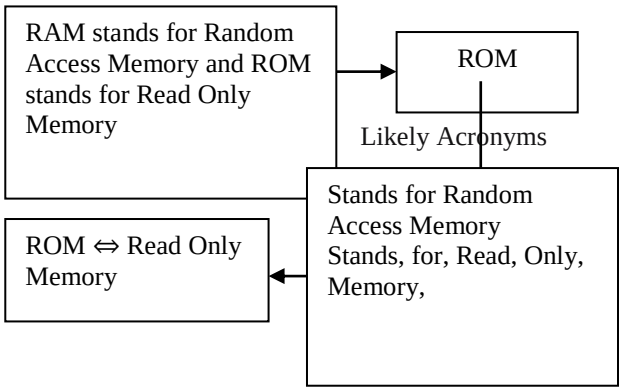
algorithm will attempt to extract all explicit acronym-definition pairs**.** People will find it helpful if we develop a system that can automatically recognize acronyms and their expansions from Text. This is because many organizations have a large number of online documents which contain many acronyms. One specific issue is the high rate at which new abbreviations are introduced in texts. Existing databases, ontologies, and dictionaries must be continually updated with new abbreviations and their definitions. In an attempt to help resolve the problem, introduced to automatically extract abbreviations and their definitions.

III. METHOD

We decompose the Abbreviation-finding problem into three steps:-

A. Identifying likely acronyms

The aim of this step is to identify all possible acronyms from original textWe scan the text to find the Acronym candidates .There are many approaches like All uppercase[5] ,parentheses matching[6] , Non parentheses matching [3].In first Approach ,The input is pre-processed to disregard lines of text that are all uppercase .In parentheses matching Abbreviation approach ,most of the acronym-definition pairs come inside parentheses and can correspond to two different patterns: (i) definition (acronym) (ii) acronym (definition). The algorithm extracts acronym-definition candidates which correspond to one of these two patterns .In Non parentheses matching; the algorithm seeks for acronym candidates that follow the constraints and are not enclosed in parentheses .There are some rules defined to identify likely Acronym.



genuine acronym/expansion candidates

Fig. 1. Architecture of Acronym Identification System

B. Generating expansion candidates

In this step, we are to generating all expansion candidates for acronyms identified .We notice that expansions always occur in surrounding text where acronyms appear in. Based on this, before generating expansion candidates, sentence broken and tokenization should be conducted on text. Sentences are split and tokens in sentences are segmented by white spaces. Please note that punctuations such as ‘,’, ‘.’, ‘)’, ‘(’ and ‘!’etc. are considered as single tokens. A given acronym splits the sentence into two parts: the substring that precedes the acronym (left context) and the substring that follows the acronym (right context). All of the substrings of left and right context are considered as candidates. Two parameters are used for identify an expansion candidate: length and offset. Maxlength is calculated as:-

Maxlength = min (length (acronym) +5, length (acronym) × 2) We can also use lexical Analyzer in which Firstly remove remove all non-alphabetic characters and break text into chunks based on occurrence of (,) and. Characters. After determine candidate acronym , it is compared with preceeding chunk and following chunk ,looking for matching definition .We can also uses rules to generating expansion candidates from surrounding text.

C. Selecting genuine expansions

In the last step we select the genuine expansions for acronyms from candidate set. we select the genuine expansions Using longest comman substring(LCS) algorithm[5],Using heuristic checker[6],Applying simple algorithm[1],applying some rules. The longest common subsequence (LCS) of any two strings X and Y is a common subsequence with the maximum length among all common subsequence.In heuristic checkers, once candidate acronyms have been found they are passed through a number of heuristics, any one of which may fail the acronym. The heuristics are loosely based on the definition of acronyms. The main idea of simple algorithm is starting from the end of the short form and the long form, move right to left, trying finding the shortest long form that matches the short form by applying constraints. In rule based we define some rules to select genuine expansions.

IV. ALGORITMS

A. Pattern Matching Algorithm

The program consists of three phases: initialization, input filtering and the application of the acronym algorithm.

(a) Initialization Phase:

The input for the algorithm is composed of several lists of words, with the text of the document as the final input stream. These inputs are:

- A list of stop words - commonplace words that are often insignificant parts of an acronym (e.g., “the", “and", “of").
- A list of reject words |- words that are frequent in the document, or in general, but are known not to be acronyms (e.g., “TABLE", “FIGURE", Roman Numerals.
- A database of acronyms and their accompanying definitions.
- The text of the document or collection to be searched.

(b) Input filtering Phase:

The input is pre-processed to disregard lines of text that are all uppercase (e.g., titles and headings) [5]. Upon identifying an acronym candidate, the reject word list is consulted before subsequent processing. If the candidate does not appear in the reject list, then an appropriate text window surrounding

the acronym is searched for its definition. The text window is divided into two sub windows, the pre-window and the post-window. Each sub window's length in words is set to twice the number of characters in the acronym.

(c) Applying the algorithm Phase:
The algorithm identifies a common subsequence of the letters of the acronym and the leader array to find a probable denition. The longest common subsequence(LCS*)* [5] of any two strings X and Y is a common sub-sequence with the maximum length among all common Subsequence For Example: if *X = acbceac and Y = cebaca, then cba is a com*mon subsequence of X and Y of length 3.There are well known and efficient algorithms that find only one LCS. To fully explain *AFP,* we present an algorithm to generate all possible LCS's. Now, for two strings X[1….m] and Y [1….n], let c[i; j] be the length of an LCS of the sequences X[1….i] and Y [1….j].c[i; j] can be obtained from the following recursive formula.

$$c[i;j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1;j-1]+1 & \text{if } i;j > 0 \text{ and } Xi = Yj \\ max(c[i;j-1];c[i-1;j]) & \text{if } i;j > 0 \text{ and } Xi \mathrel{!}= Yj \end{cases}$$

Two matrix made:matrix c: The algorithm computes the length of an LCS for strings X and Y and stores this value in c[m; n].matrix b :The LCS construction method utilizes the matrix b to show the path from which an LCS can be constructed.A " " entry in b[i; j] asserts that X[i] = Y [j], and c[i−1; j−1]+1 is the selected value.

B. Heuristic Algorithm
TLA (Three-Letter Acronyms) [6] was developed to provide enhanced browsing facilities in a digital library. As with AFP, candidate acronyms and their definitions are selected from a stream of words. All non-alphabetic characters are converted to spaces and any multiple spaces replaced with a single space. Candidate acronyms are determined by matching the initial letter of each word in the context of a potential acronym against the appropriate letter in the acronym. If the first letter does not match, the word is skipped. Otherwise, the next letter of the same word is tested against the next letter of the acronym, and if it matches the algorithm continues to move along the word. A maximum of six letters are used from each word, and a potential acronym must be entirely upper-case. In order to determine which candidate acronyms should be output, a machine learning scheme is used. Four attributes [6] are calculated for each candidate:

- the length of the acronym in characters (generally between 2 and 6);
- the length of the acronym's definition in characters (generally between 10 and 40);
- the length of the acronym's definition in words (generally between 2 and 6);
- The number of stop words in the acronym's definition.

These features clearly include redundancy the fourth is the difference between the third and the first. The machine learning approach is to generate a model using training data in which acronyms have already been marked by hand. The model determines what attributes, and what combinations of attributes, are the important ones for making the decision.
Simple Algorithm with Adding Constraints
This consists of two phases: Identifying Short Form and Long Form Candidates, Algorithm for Identifying Correct Long Forms

(a)Identifying Short Form and Long Form Candidates:
For identifying likely Acronym two cases are considered [1]:
(i) Long form '('short form ')'
(ii) Short form '('long form ')'
For Identifying candidates for long form, we select surrounding window in which no more than min (|A| + 5, |A| * 2) words, where |A| is the number of characters in the short form.

(b) Algorithm for Identifying Correct Long Forms:
The main idea is: starting from the end of the short form and the long form, move right to left, trying find the shortest long form that matches the short form. Every character in the short form must match a character in the long form, and the matched characters in the long form must be in the same order as the characters in the short form .One exception: the match of the character at the beginning of the short form must match a character in the initial position of the word in the long form.
Example:-
To illustrate the algorithm [1], consider the following pair *<HSF, Heat shock transcription factor>*. The algorithm starts by setting *sIndex* to point to the end of the short form (*HSF*), and *lIndex* to point to the end of the long form (*factor*). It then decrements *lIndex* until a match is found (*factor*). *sIndex* is decremented by one (*HSF). lIndex* is decremented until a match is found (*transcription*). *sIndex* is decremented again (*HSF*). Since *sIndex* now points to the beginning of the short form, the next match should be found at a beginning of a word in the long form. Therefore, *lIndex* is decremented until a valid match is found (*Heat*). Note that another match was skipped (*shock*) because it was not in the beginning of a word. Also note that although the algorithm did not match the second character correctly (*transcription* instead of *shock*) it still found the right long form.

C. *Rule Based Algorithm*
This consists of two phases two Phases: Finding Acronym-Definition Candidates, Matching Acronyms with Definitions

(a)Finding Acronym-Definition Candidates:
A valid Acronym Candidate is found if the string satisfies the conditions [3] (i) and (ii) and either (iii) or (iv): (i) The string contains at least two characters. (ii) The string is not in the list of rejected words. (iii) The string contains at least one

capital letter. (iv) The strings' first or last character is lower case letter or numeric. The algorithm searches for a definition candidate that satisfies the following conditions (i) At least one letter of the words in the string matches the letter in the acronym. (ii) The string doesn't contain a colon, semi-colon, question mark or exclamation mark. (iii) The maximum length of the string is min (|A|+5,|A|*2), where |A| is the acronym length. (iv)The string doesn't contain only upper case letters.

(b)Matching Acronyms with Definition:

The next step is to choose the correct substring of the definition candidate for the acronym candidate. This is done by reducing the definition candidate string as follows: the algorithm searches for identical characters between the acronym and the definition starting from the end of both strings and succeeds in finding a correct substring for the acronym candidateIf it satisfies the following conditions[3]: (i) at least one character in the acronym string matches with a character in the substring of the definition (ii) the first character in the acronym string matches the first character of the leftmost word in the definition substring, ignoring upper/lower case letters.

V. SIGNIFICANCE OF WORK

- Useful tool for reader.
- Used to build new tools based on gathered acronym data in digital libraries.
- An automatic method to define abbreviations would help researchers by providing a self-updating abbreviation dictionary and also facilitate computer analysis of text.
- Used for hypertext browsing system.
- Used to enhance text or information retrieval.
- Help existing tools work more smoothly.
- Annotation and decoration of text presented to user in digital libraries.
- Acronym detection improves the quality of spell checker.
- Used in Post Processing System (PPS).

VI. PERFORMANCE AND COMPARISON

In different approaches performance measured using standard measures Recall and Precision. These different systems tested on collection of documents. The training and test sets, while mutually exclusive, involved only a fraction of the documents in the collection. To select these sets, the full collection was automatically analysed and sequenced according to the approximate ratio of acronyms to document length.

APPROACH	RECALL	PRECISION
Pattern Matching Algorithm	93%	98%
Heuristic Algorithm	91%	68%
simple algorithm with adding constraints	82%	96%
Rule-based method	72.5%	93%

There are five main differences between Systems made by using technique Using Pattern Matching Algorithm and Using Heuristic Algorithm-

- First algorithm only considers the first letter of each word when searching for acronyms. Second algorithm considers the first three letters in each word. This enables system to match acronyms such as AmVets.
- First algorithm uses probabilistic techniques to determine matches. A probability is computed that a given definition matches a candidate acronym and if this probability is higher than a certain threshold, then it is accepted. Second algorithm uses a set of heuristics each of which can reject any candidate acronym in a Boolean fashion.
- First algorithm candidate acronyms are all upper case; upper case sentences are ignored. Second algorithm is independent of case (but heuristics have case information available to them).
- First algorithm parses words with embedded punctuation as single words, whereas Second algorithm parses them as separate words. This allows matching of U.S. Geographic Service (USGS), but may prevent matching of other acronyms.
- First accepts some errors. This enables matches for acronyms which second miss. For example DBMS (Database Management System), which Second misses because the "B" is the middle of the "Database".

The first three differences appear to indicate that system made using Heuristic Algorithm is more general than system made Using Pattern Matching Algorithm.The fifth difference indicates the system made using Pattern Matching Algorithm is more general than system made using Heuristic Algorithm. System made using Pattern Matching Algorithm reports precision and recalls rates as high as 98%, this is far higher that system made using Heuristic Algorithm has so far achieved.

VII. CONCLUSION AND FUTURE WORK

System made using Pattern Matching Algorithm reports 98% precision and 93% recall rates, this is far higher than other approaches have so far achieved. Because most of algorithms find only one LCS (Longest Common Subsequence) but System made using Pattern Matching Algorithm generates all possible LCS's. For future work some adjustments like special acronym characters or acronym length could be provided as options to System made using Pattern Matching

Algorithm so the program could be tailored to a document's or collection's content .Future work can be to generate system that will find Acronym-Definitions pairs from text that is in other languages like piunjabi, hindi etc.

VIII. ACKNOWLEDGMENT

First and foremost, I would like to thank God almighty for life itself. All that I have is due to His grace and I give all glory to him. Even though I have let him down many times in life, he has always carried me through. With deep sense of gratitude I express my sincere thanks to my esteemed and worthy supervisor Er. Jagroop Kaur, Assistant professor, Department of Computer Engineering, Punjabi University, Patiala for her valuable guidance in carrying out this work under her effective supervision, encouragement, enlightment and cooperation. I shall be failing in my duties, if I do not express my deep sense of gratitude towards Dr. Lakh winder Kaur, Professor and Head of, Department of Computer Engineering, Punjabi University, Patiala for providing necessary facilities in the department to carry out this work.

REFERENCES

[1] Ariel S. Schwartz and Marti A. Hearst. 2003. "*A simple algorithm for identifying abbreviation definitions in biomedical texts"*. Proc. of the Pacific Symposium on Biocomputing. University of California, Berkeley.
[2] Dana Dannells,"*Automatic Acronym Recognition*"
[3] David Nadeau and Peter Turney. 2005." *A Supervised Learning Approach to Acronym Identification"*. Information Technology National Research Council, Ottawa, Ontario, Canada
[4] J.T. Chang, H Schütze, and R.B. Altman, "Creating an Online Dictionary of Abbreviations from MEDLINE" *JAMIA*, to appear.
[5] Kazen Taghva and Jeff Gilbreth. 1999. Technical Report. "*Recognizing Acronyms and their Definitions"*. University of Nevada, Las Vegas.
[6] Stuart Yeates. 1999." *Automatic extraction of acronyms from text"*. Proc. of the Third New Zealand Computer Science Research Students' Conference. University of Waikato, New Zealand

