DRTC Workshop on
Digital Libraries: Theory and Practice
March 2003
DRTC, Bangalore

Paper: E

Search Algorithms

Brijesh Shanker Singh
Documentation Research and Training Centre
Indian Statistical Institute
Bangalore-560 059
e-mail: *brijesh@drtc.isibang.ac.in*

Abstract

This paper is a brief discussion about the some of most popular search algorithms, used in searching, browsing and retrieving of digital information. It also gives the brief outlook about the various search algorithms used by digital library and similar species of the software.

1. INTRODUCTION

Recently, as the Internet and the World Wide Web are becoming more and more popular, the topic of digital libraries has gained focus in library and information world. In today's information explosion environment it is very difficult to retrieve relevant document. As digital libraries become larger, current techniques of information retrieval cannot be used efficiently both for browsing and querying. Browsing and Querying for exact document in digital libraries is not easy as in conventional databases because digital documents are free-formatted, voluminous and multiple media types in nature. In the absence of efficient retrieval system it would be very difficult to retrieve relevant document speedily. Search algorithms play a very vital role in effective and interactive browsing and querying. The present paper, attempts to explain some of the most popular and useful search algorithms, which are used in searching of digital documents.

2. WHAT IS AN ALGORITHM?

An algorithm is a procedure to solve a problem in an orderly, finite, step-by-step logical and straightforward manner. An algorithm must be finite (terminating after a finite number of steps) and be effective (accomplish the solution of the problem in a series of simple steps).
Here some of the important search algorithms are discussed.

- Soundex algorithm
- Metaphone algorithm
- Phonex algorithm
- Word Stemming Algorithms

3. SOUNDEX ALGORITHM

Margaret O' Dell and Robert C. Russell patented the original soundex algorithm in 1918 (1). The method is based on the six phonetic classifications of human speech sounds, which in turn are based on how you put yours lips and tongue to make the sounds.

1. **Bilabial:** Sound produce with both lips.
2. **Labiodentals:** Utter with the participation of the lip and the teeth (the labiodentals sounds \f\ and \v\).
3. **Dental:** Articulated with the tip or blade of the tongue against or near the upper front teeth.
4. **Alveolar:** Articulated with the tip of the tongue touching or near the teeth ride.
5. **Velar:** Formed with the back of the tongue touching or near the soft palate (The velar \k\ of \kill\ cool).
6. **Glottal:** the interruption of the breath stream during speech by closure of the glottis.

Terms that are often misspelled can be a problem for database designers. Names, for example, are variable length, can have strange spellings, and they are not unique. Words can be misspelled or have multiple spellings, especially across different cultures or national sources. To solve this problem, we need phonetic algorithms, which can find similar sounding terms and names. Such families of algorithms exist and are called Soundex, after the first patented version.

The **Soundex** algorithm is used for search of words on the basis of their phonetic quality. So the results will contain words that might have different spellings and yet sound similar when spoken. This algorithm is useful to search for keywords when the exact spelling is unknown. A Soundex algorithm takes a word, such as a person's name, as input and produces a character string, which identifies a set of words that are (roughly) phonetically alike. It is very handy for searching large databases when the user has incomplete data. The algorithm is fairly straightforward to code and requires no backtracking or multiple passes over the input word. Following is the outline of soundex algorithm.

1. Capitalize all letters in the word and drop all punctuation marks. Pad the word with rightmost blanks as needed during each procedure step.
2. Retain the first letter of the word.
3. Change all occurrences of the following letters to '0' (zero): 'A', E', 'I', 'O', 'U', 'H', 'W', 'Y'. (Except when they come as an initial letter)
4. Change letters from the following sets into the digit given:
   a. 1 = 'B', 'F', 'P', 'V'
   b. 2 = 'C', 'G', 'J', 'K', 'Q', 'S', 'X', 'Z'
   c. 3 = 'D','T'
   d. 4 = 'L'
   e. 5 = 'M','N'
   f. 6 = 'R'
5. Remove all pairs of digits, which occur beside each other from the string that resulted after step (4).
6. Remove all zeros from the string that results from step 5.0 (placed there in step 3)
7. Pad the string that resulted from step (5) with trailing zeros and return only the first four positions, which will be of the form <uppercase letter> <digit> <digit> <digit>.

Example
Steps 0 1 2 3 4 5 6 7

William's → WILLIAMS → W → W0LL00MS → W0440052 → W04052 → W452
→ **W452** would be the soundex code for word WILLIAM'S.

3.1 Benefits

The computational benefits of Soundex-type algorithms are easy to see: by exchanging a name for a code, all variant spellings of the name and words can be expected to share that same code, allowing a relatively efficient search of database. Now it is basic feature of spell-checker, information retrieval system and Internet search engines and it is used specially for quiry refining.

3.2 Limitations

Following are the some drawbacks of the Soundex (2).

3.2.1. Dependence on Initial Letter

The Soundex algorithm uses the first letter of a name as a key component of the code it generates to represent the name. Names that do not begin with the same letter will never match each other. A data entry operator hearing the name "Korbin" might type in the much more common "Corbin." Although "Korbin" may be in the database, it will not be returned by a standard Soundex search on "Corbin."

3.2.2. Differing Transcription Systems

Languages written in non-Roman scripts may use multiple systems for converting names from native to Roman characters. The same Russian surname may occur as "Ivanov" or "Ivanoff" or even as "Iwanow." The Soundex codes for the members of these spelling variants do not always match each other. in other words, soundex is largely language specific.

3.2.3. Silent Consonants

Soundex cannot capture the phonetic similarity between names with silent consonants and alternate or simplified spellings of those names in which these consonants are omitted. An

uncommon name like "Coghburn" may be spelled as "Coburn," or "Deighton" may be recorded as "Dayton." Soundex assigns different codes to these pairs, guaranteeing that they can never match each other.

3.2.4. Initial

Initials are often substituted for full names. Records for a "Michael Kissinger" and "M. Kissinger," for example, may well belong to the same person. However, a standard Soundex query on "Michael Kissinger" will not retrieve "M. Kissinger," and vice versa.

3.2.5. Unranked, Unordered Returns

Because its goal is to group names by assigning a common code, Soundex does not have the capability to measure the degree of similarity between a pair of names found within a group. This means that returns from a Soundex-based query cannot be ranked and presented best first—names are typically returned as they are found in the database.

4. METAPHONE SEARCH ALGORITHM

The original Metaphone algorithm was described in the December 1990 issue of Computer Language magazine and was designed as a replacement for the Soundex algorithm. It returns a rough approximation of how an English word sounds and uses many common rules of English pronunciation that Soundex does not cover (3). The algorithm reduces the word to a 1 to 4-character code using relatively simple phonetic rules for typical spoken English. Metaphone ignores the vowels that occur after the leading letter, only vowels that occur at the beginning of a word are retained. It then reduces the remaining alphabet to sixteen constant sounds (B, X, S, K, J, T, F, H, L, M, N, P, R, O, W and Y). The “sh” sound is represented by an X and a zero is used as a representation of the “th” sound, as it is similar to the Greek theta (Ø). If a letter is duplicated

Table 1 and Table 2 shows all the rules applied and the exceptions respectively

Letter	Code	Comments
B	B	Unless at the end of a word after “m” as in
C	X	X (sh) if “-cia-” or “-ch-”
	S	S is “-ci-”, “-ce-”, or “-cy-”
		Silent if “-sci-”, “-sce-”, or “-scy-”
	K	Otherwise, including “-sch-”
D	J	If in “-dge-”, “-dgy-”, “-dgi”
	T	Otherwise
F	F	
G		Silent if in “-gh-” and not at end or before a vowel
		In “-gn” or “-gned”
		In “-dge-”, etc., as in above rule
	J	If before “i”, or “e”, or “y”, if not double “gg”
	K	Otherwise
H		Silent if after vowel and no vowel follows
	H	Otherwise
J	J	
K		Silent if after “c”
	K	Otherwise

L	L	
M	M	
N	N	
P	F	If before "h"
	P	Otherwise
Q	K	
R	R	
S	X	(sh) if before "h" or in "-sio-" or "-sia-"
	S	Otherwise
T	X	(sh) if "-tia-" or "-tio-"
	Ø	(th) if before "h"
		Silent if in "tch-"
	T	Otherwise
V	F	
W		Silent if not followed by a vowel
	W	If followed by a vowel
X	KS	
Y		Silent if not followed by a vowel
	Y	If followed by a vowel
Z	S	

Table 1: Metaphone Coding Rules

Initial "kn-", "gn-", "pm-", "ae-", "wr-"	Drop the first letter
Initial "x"	Change it to s
Initial "wh-"	Change to w

Table 2: Metaphone Coding Exceptions

5. PHONEX

The Phonex algorithm is the result of work by A. J. Lait and B. Randell, which describes a comparative analysis of a number of name matching algorithms, including Soundex and metaphone. Based on the analysis of their strengths and weaknesses, it proposes a new and improved algorithm. They decided to call it "Phonex", which is derived from the two methods on which it is based (Soundex and Metaphone) (3).

The algorithm converts each word into an equivalent four-character code using the following steps:

1. Pre-process the name according to the following rules:
2. Remove all trailing 'S' characters at the end of the name.
3. Convert leading letter-pairs as follows:

KN -> N	WR -> R
PH -> F	

Convert leading single letters as follows:

H -> Remove	
E, I, O, U, Y -> A	K, Q -> C
P -> B	J -> G
V -> F	Z -> S

Code the pre-processed name according to the following rules:

4. Retain the first letter of the name, and drop all occurrences of A, E, H, I, O, U, W, Y in other positions.

5. Assign the following numbers to the remaining letters after the first:

B, F, P, V -> 1	
C, G, J, K, Q, S, X, Z -> 2	
D, T -> 3	If not followed by C.
L -> 4	If not followed by vowel or end of name.
M, N -> 5	Ignore next letter if either D or G.
R -> 6	If not followed by vowel or end of name.

6. Ignore the current letter if it has the same code digit as the last character of the code.

7. Convert to the form 'letter, digit, digit, digit' by adding trailing zeros (if there are less than three digits), or by dropping rightmost digits if there are more than three.

6. WORD STEMMING ALGORITHMS

6.1. What is Word Stemming?

Word Stemming is common form of language processing in most Information Retrieval (IR) systems. Word stemming is an important feature supported by present day indexing and search systems. In natural language processing, stemming is the process of merging or lumping together non-identical words, which refer to the same principal concept. This can relate both to words, which are entirely different in form (e.g., "group" and "collection"), and to words which share some common root (e.g., "group", "grouping", "subgroups".) (8). In the first case the words can only be mapped by referring to a dictionary or thesaurus but in second case idea is to improve recall by automatic handling of word endings by reducing the words to their word roots, at the time of indexing and searching. Stemming is usually done by removing any attached suffixes, and prefixes from index terms before the assignment of the term. Since the stem of a term represents a broader concept than the original term, the stemming process eventually increases the number of retrieved documents (8).

6.2. Algorithms for Word Stemming

A stemming algorithm is an algorithm that converts a word to a related form. One of the simplest such transformations is conversion of plurals to singulars, another would be the derivation of a verb from the gerund form (the "-ing" word). An algorithm that attempts to convert a word to its linguistically correct root ("compute" in this case) is sometimes called a **lemmatiser.** A number of stemming or conflation algorithms have been developed for IR in order to reduce morphological variants to their root form. A stemming algorithm would normally be used for document matching and classification by using it to convert all likely forms of a word in the input document to the form in a reference document. This requires that the reference document also be processed through the stemming algorithm (7).

- Paice/Husk Stemming Algorithm (1990)
- Porter Stemming Algorithm (1980)
- Krovetz Stemming Algorithm (1993)
- Lovins Stemming Algorithm (1968)
- Salton Stemming Algorithm (1968).
- Dawson Stemming Algorithm (1974).

In these algorithms, the Porter Stemming Algorithm is widely used and effective algorithm and it is much more popular comparison to others. Examples of products using stemming algorithms would be search engines such as Lycos and Google, and also thesaurus and other products using NLP for the purpose of IR. Stemmers and lemmatizes also have applications more widely within the field of Computational Linguistics.

7. PORTER STEMMING ALGORITHM

The Porter Stemmer is a conflation Stemmer developed by Martin Porter at the University of Cambridge in 1980. Porter stemming algorithm (or 'Porter stemmer') is a process for removing the commoner morphological and in-flexional endings from words in English. It is the most effective and widely used. Porter's Algorithm works based on number of vowel characters, which are followed be a consonant character in the stem (Measure), must be greater than one for the rule to be applied (8).

The rules in the Porter algorithm are separated into five distinct steps. They are applied to the words in the text starting from step 1 and moving on to step 5.

Step 1. Deal with plurals and past participles. The subsequent steps are much more straightforward.
Ex. plastered->plaster, motoring-> motor

Step 2. Deals with pattern matching on some common suffixes.
Ex. happy -> happi, relational -> relate, callousness -> callous

Step 3. Deals with special word endings.
Ex. triplicate-> triplic, hopeful-> hope

Step 4. Checks the stripped word against more suffixes in case the word is compounded.
Ex. revival -> reviv, allowance-> allow, inference-> infer etc.,

Step 5. Checks if the stripped word ends in a vowel and fixes it appropriately
Ex. probate -> probat, cease -> ceas, controll -> control

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. There is no linguistic basis for this approach (9).

Example.

In a typical IR environment, one has a collection of documents, each described by the words in the document title and possibly by words in the document abstract. Ignoring the issue of precisely where the words originate, we can say that a vector of words, or terms represents a document. 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 (10)

8. SOUNDEX FOR AFFIX REMOVAL

8.1 SOUNDEX for Prefix Removal

Name with prefixes: if a surname has a prefix, such as Van, De, Di, La or Le code both with and without the prefix because the surname might be listed under their code. Note, however that Mc and Mac are not considered prefixes. For Example Van-Deusen might be coded two ways, first with prefix, V532 (V, N=5, D=3, S=2) Or without prefix D250 (D, S=2, N=5, 0 added) (11).

8.2 SOUNDEX for Suffix Removal

Soundex is also useful for suffix problem because it in code only four initial letters of word (except vowels) so it doesn't matter what are the ending le tters (11).

9. CONCLUSION

In conclusion I would like to mention some search algorithms used by Digital library and similar software's like Greenstone digital library software, Harvest, Htdig, Cdsware, Ganesh .etc

9.1. Ht://Dig: WWW Search Engine Software

Ht://Dig was developed at San Diego State University as a way to search the various web servers on the campus network The ht://Dig system is a complete world wide web indexing and searching system for a domain or intranet. It is meant to cover the search needs for a single company, campus, or even a particular sub section of web site. Htdig supports the text base searching so it uses the word matching algorithms. Currently the following algorithms are supported (in any combination): **Soundex, Metaphone , Common word endings (stemming), Synonyms, Accent stripping, Sub string and prefix, Exact-match algorithms** (4).

9.2. Search Algorithm in Greenstone Digital Library

Greenstone digital library provides the text base searching. For this purpose GSDL use **Kea (Keyword Extracting Algorithm) Search algorithm.** Kea is an algorithm for extracting keyphrases from the text of a document (We say keyphrases because not all keywords are words). Kea automatically extracts keyphrases from the full text of documents. The set of all candidate phrases in a document are identified using rudimentary lexical processing, features are computed for each candidate, and machine learning is used to generate a classifier that determines which candidates should be assigned as keyphrases and it create links to similar documents and to suggest appropriate query phrases to users (5).

9.3. Harvest

Harvest uses the three-index/search engines for the purpose of indexing and searching. By default, harvest uses GIMPSE (**GL**obal **IMP**licit **SE**arch) index / search engine and rest of two **WAIS** (Wide Area Information Server) and **SWISH** (Simple Web Indexing System for Humans), as supportive index/search engine. They uses exact matching, similar matching, string search, etc. as technique for supporting to searching (6).

The application of search algorithms in digital library environment makes the searching easy and meaningful. And improve the recall and precision in information retrieval.

10. REFERENCES

1. *Understanding Classic SoundEx Algorithms*. from http://www.creativyst.com/Doc/Articles/SoundEx1/SoundEx1.htm/
2. Franki, P., & Leonard, S. (2001). *Is soundex good enough for you?* from http://www.onomastix.com/about_ono/whitepapers/wp_soundex.htm/
3. Lang, R. A. (2001). *The Compact Disc Search Engine*, from http://www.rlang.co.uk/projects/soundz/Soundz.html/
4. *WWW Search Engine Software*. from http://www.Htdig.org/
5. *Greenstone digital library software*. from http://www.greenstone.org/
6. Hardy, D. R., & Lee, K.-J. (2002). *Harvest User's Manual*, from http://www.harvest.sourceforge.net/harvest/doc/html/manual.html/
7. *What is Stemming?* from http://www.comp.lancs.ac.uk/computing/research/stemming/general/
8. Lakshmi, K. V. (2002).*Developing a word-stemming program using Porter's Algorithm*, from http://144.16.72.189/opendl/cdrom/test-collections/gsdl/project-reports/Lak-minor/lakproj.ppt.
9. *The Porter Stemming Algorithm*. from http://www.tartarus.org/~martin/PorterStemmer/voc.txt
10. *An algorithm for suffix stripping.* from http://www.tartarus.org/~martin/PorterStemmer/def.txt
11. *The Soundex Indexing System.* from http://www.archives.gov/research_room/genealogy/census/soundex.html
12. *Merriam-Webster Unabridged Expands*. from http://www.m-w.com/home.htm