Mining, Ranking, and Using Acronym Patterns

Xiaonan Ji[1⋆], Gu Xu[2], James Bailey[1], and Hang Li[2]

[1] NICTA Victoria Laboratory, Department of CSSE, University of Melbourne, Australia
{xji,jbailey}@csse.unimelb.edu.au
[2] Microsoft Research Asia, 4F, Sigma Center, No. 49, Zhichun Road, Haidian District, Beijing, 100080, China
{guxu,hangli}@microsoft.com

Abstract. Techniques for being able to automatically identify acronym patterns are very important for enhancing a multitude of applications that rely upon search. This task is challenging, due to the many ways that acronyms and their expansions can be embedded in text. Methods for ranking and exploiting acronym patterns are another related, yet mostly untouched area. In this paper we present a new and extensible approach to discover acronym patterns. Furthermore, we present a new approach that can also be used for both ranking the patterns, as well as utilizing them within search queries. In our pattern discovery system, we are able to achieve a clear separation between higher and lower level functionalities. This enables great flexibility and allows users to easily configure and tune the system for different target domains. We evaluate our system and show how it is able to offer new capabilities, compared to existing work in the area.

1 Introduction

An acronym is a word formed from the parts of a full name and is used to stand for that name. For example, CPU can be used to stand for "Central Processing Unit". Recognizing acronyms and their full names is useful in many document processing applications. Alternatively, acronyms are often used by users as terms within search queries. By being able to replace acronyms with their most appropriate expansions, a search engine can potentially deliver better search results.

Most of the previous work in this area concentrates on identifying acronym patterns. The drawbacks are either a lack of extensibility (such as [10, 6, 9, 8]) or heavy reliance on the availability of a remarked training corpus (such as [12, 2]). In contrast, our new acronym mining system clearly separates higher-level mapping strategies from lower-level mapping rules and is available to achieve a high degree of flexibility. Users can easily add new rules or turn on/off existing ones.

⋆ Part of the work has been done at Microsoft Research Asia.

A new and interesting related problem is the ability to rank and deploy acronym patterns for online search. The purpose of ranking method is to properly appraise different patterns, by taking into account the popularity of acronym patterns in conjunction with standard confidence measures in their correctness. The ability to rank acronym patterns can be particularly attractive for online search applications, where the ambiguity of acronyms can cause problems for keyword-matching based IR systems. If the IR system can recognize acronyms and their most popular expansions, the retrieved results can be ranked more appropriately.

Related work. Some algorithms discover acronym patterns by finding a best alignment from acronym letters to letters of words in the full name. Some of them use predefined rules [10, 14, 11, 9], while others use machine learning methods [2, 12]. An interesting idea that aims to identify syntax-based relationships for word phrases is studied in [1, 15]. These methods can be used to discover various types of patterns such as acronyms and expansions or books and authors. Acrophile is an online acronym dictionary based on work in [6]. It uses a very simple method of acronym ranking, which is just based on counting acronym frequencies. We are not aware of any research which has addressed the problems of how to rank acronym patterns and how to use them for query extension.

Our contributions. In this paper, we introduce a new acronym pattern mining framework called `AcroMiner`. `AcroMiner` uses an architecture built on mapping rules, but allows users to flexibly configure predefined rules or add new ones. Experiments show that `AcroMiner` is able to handle large data sets and can achieve promising results. Furthermore, we also address the novel problems of how to rank acronym patterns and extend acronym queries using their expansions.

2 Preliminary Definitions

The full name of an acronym is called its *expansion*. The way of coming up with the acronym from the expansion is called the *mapping* from the former to the latter. This mapping can be broken into lower-level mappings from individual letters of the acronym to positions in the expansion. In this paper, letters of acronyms and the corresponding positions of the expansions are underlined, in order to show how acronyms are mapped to expansions. An acronym pattern is the tuple of an acronym word (A) and its expansion (E), denoted as <A, E>.

An acronym may stand for multiple expansions. For example, WTO stands for "World Trade Organization" as well as "World Tourism Organization" and sometimes even "World Toilet Organization". This property is called *the ambiguity of acronyms*. On the other hand, an expansion is usually abbreviated by only one acronym. We call this property *the unambiguity of expansions*.

This paper discusses the following problems. *The mining problem:* identifying acronym patterns from documents. *The ranking problem:* recognizing the acronym patterns and then ordering them by a popularity measure. *The acronym query extension problem:* using ranked acronym pattern set to improve the capability of an IR system to handle queries containing acronyms.

Given a set of documents $D = \{s_1, s_2, ..., s_n\}$, the *occurrence frequency* (*tf*) of an object (e.g. a word or an acronym pattern) p is the total number of occurrences of p in documents of D , that is, $tf(p) = \sum_{i=1}^{n} |\{o \in s_i | o = p\}|$. The *document frequency* (*df*) of p is the number of documents containing p, that is, $df(p) = |\{s_i | o \in s_i \text{ and } o = p\}|$.

Letters contained in documents[3] are categorized as follows. *Invisible letters*: tab (ASCII 09), line feed (ASCII 10), carriage return (ASCII 13) and space (ASCII 32). *Punctuation letters*: exclamation mark (`!`), question mark (`?`), brackets (`(`, `)`, `[`, `]`, `<`, `>`, {, }), hyphen (`-`), underscore (_), colon (`:`), semicolon (`;`), comma (`,`), period (`.`), slash (`/`), apostrophe (’), quotation marks(`""`, `‘’`). *Alphabetical letters*: [`A-Z`] and [`a-z`]. *Numerical letters*: [`0-9`]. *Symbolic letters*: `@`, `~`, `#`, `$`, `%`, `^`, \, `&`, `*`, `+`, `=`, |.

Letters that are not allowed to be contained in acronyms and expansions are marked as delimiters. Delimiters indicate the boundaries of potential expansions.

3 The Framework of `AcroMiner`

`AcroMiner` consists of four components: document preprocessing, identification of acronyms, identification of acronym patterns and postprocessing. The four components operate sequentially: the outputs of earlier components are input into the later ones.

Document Preprocessing. The input documents are reformatted in the first component. The preprocessing removes meta data, marks some punctuation and symbolic letters as delimiters and concatenates sentences from a document into a single long sequence.

Identifying Acronym Words. The second component identifies acronyms from input sequences. A regular expression $\mathcal{R}$ is used to identify acronyms. During the scan of word sequences, each word w is checked to see whether it satisfies $\mathcal{R}$ or not. The regular expression currently used in `AcroMiner` is:

$$\mathcal{R}{=}\{num\}?((U\{sep\}?)^{+}([\{num\}, L, \{sym\}])?)?(U\{sep\}?)^{+}([\{num\}, S])?$$

In $\mathcal{R}$, $\{num\}$ stands for numerical letters; U means alphabetical letters written in uppercase ([`A-Z`]); $\{sep\}$ stands for period (`.`); L stands for alphabetical letters written in lowercase ([`a-z`]); $\{sym\}$ stands for ‘`&`’, ‘`/`’ or ‘`-`’; S stands for lowercase letter ‘`s`’; (...) is used to group subexpressions; [...] is used to wrap alternative subexpressions; ‘`?`’ means that the preceding subexpression may appear at most once; ‘`+`’ means that the preceding subexpression appears at least once. $\mathcal{R}$ covers the format of the majority of acronyms, such as “U.S.A.”, “SVMs”, “3D”, “P2P”, “DoD”, “AT&T” and “TCP/IP”.

It could be the case that there are acronyms written in different ways to that specified by $\mathcal{R}$. For example, the acronym of “Tool Command Language” is written as Tcl rather than TCL[4]. In order to discover these acronyms, two

[3] In this paper we only consider documents written in English and encoded by ASCII.

[4] According to Tcl Wikipedia: `http://en.wikipedia.org/wiki/Tcl`.

writing formats are additionally considered. The first writing format is: a single word embedded in a pair of parentheses, following closely after a sequence of words. It is denoted as $\ldots$ *a sequence of words* $(w)\ldots$. The other writing format is: a sequence of words embedded in a pair of parentheses, preceded closely by a word. It is denoted as $\ldots w$ (*a sequence of words*)$\ldots$. According to [15], acronyms are usually written at the position w and the expansions are usually contained in the sequence of words next to it.

Identifying Acronym Patterns. Once an acronym is found, the next task is to identify the expansion it stands for. We make the assumption that *an acronym's expansion lies close to the acronym*. In other words, we only try to identify expansions in regions appearing W words before or after the acronym occurrence. We call these expansion regions *context windows* or CWs.

This problem can be divided into two levels. (a) The lower-level subproblem deals with how to map individual letters in the acronym to individual positions in the CW. Several letter-to-word mapping rules are used as constraints to define such mappings. Mapping rules are weighted with different mapping scores according to a confidence measure of correctness. (b) The higher-level subproblem deals with the strategy of mapping multiple letters to multiple positions. A backtracking algorithm is used to discover all letter-to-word mappings. The quality of the mapping from acronym to an expansion is measured by the sum of mapping scores of the applied rules. We next discuss these subproblems in detail.

(a) Lower-level letter-to-word mapping rules. The letter-to-word mapping rules are categorized into three types: *fixed*, *shiftable* and *neglectable*. These types are considered in the higher-level mapping strategy and their meanings will be explained later. We now list the mapping rules.

1. A letter can be mapped to the 1st position of a non-neglectable word[5], if it is the same as the letter in that position. This is a *fixed* mapping rule. For example, in <CMU, "Carnegie Mellon University">, each letter in the acronym is mapped to the 1st position of each word.
2. A letter can be mapped to the 2nd or 3rd position of a non-neglectable word if it is the same as the letter in that position and its preceding letter is mapped to the preceding position. For example, in <DASFAA, "Database Systems for Advanced Applications">, the first '`A`' in the acronym is mapped to the 2nd position of the first word in the expansion. This is a *shiftable* mapping rule.
3. A letter can be mapped to the leading position of a neglectable word if it is the same as the initial letter of this word. For example, in <WOW, "World of Warcraft">, the letter '`O`' in the acronym is mapped to the 1st position of the neglectable word "of". This is a *shiftable* mapping rule.
4. A letter can be mapped to a hyphen-connected word if it is the same as the first letter of that word. A hyphen-connected word is a non-leading word in a hyphenated phrase. For example, in the hyphenated phrase "Peer-to-Peer",

[5] Neglectable words currently used are: &, after, an, and, are, as, at, de, en, for, from, in, is, la, of, on, or, the, to, up, with. New neglectable words can be added easily.

"to" and the second "peer" are hyphen-connected words. By applying this rule to <XWC, "X-Windows Commander">, the letter 'W' in the acronym is mapped to the 1st position of "Windows". This is a *shiftable* mapping rule.

5. For words containing English prefixes[6], it is common to map two letters from the acronym to the prefix and suffix separately. A letter can be mapped to the first letter of the suffix by applying rule 1 given that its preceding letter is mapped to the first letter of the prefix. For example, in <MSDN, "Microsoft Developer Network">, 'S' is mapped to the first letter of "-soft" given its preceding letter 'M' is mapped to the first letter of "micro-". This is a *shiftable* rule.
6. Letters of acronyms can be mapped to sub-words of some compound words[7]. A letter can be mapped to the first letter of the latter sub-word according to rule 1 given that its preceding letter is mapped to the initial letter of the former word. For example, in <DBA, "Database Administrator">, 'B' can be mapped to the sub-word of "database" given that its preceding letter 'D' is mapped to the first letter of the former sub-word. The compound word list can be extended by users. This is a *shiftable* rule.
7. A letter from the acronym can be mapped to the 4th, 5th or 6th position of a word if that word contains at least 8, 9 or 10 letters and its preceding letter is mapped to the first letter of this word by applying rule 1. It is *shiftable*.
8. 'X' can be mapped to any word initialized with prefix "ex". This is a *fixed* rule.
9. The ending letter 'S' (or 's') of the acronym can be neglected if no proper mapping can be found. The reason is that the ending 'S' (or 's') can represent the plural format of the acronym thus have no position to be mapped to. For example, in <SVMS, "Support Vector Machine(s)">, the ending 'S' is mapped to nothing. This is a *neglectable* mapping rule.
10. Numerical letters [0-9] can be mapped to their English names, e.g. in <3D, "Three Dimension">, '3' is mapped to "three". This is a *fixed* mapping rule.
11. Some special letters can be mapped to English words. In the current implementation, '&' is mapped to "and", '2' is mapped to "to" and '4' is mapped to "for". For example, in <AT&T, "American Telephone and Telegraph">, '&' is mapped to "and". It is a *fixed* mapping rule.
12. If the acronym contains numerical letters, its preceding letter (if there is any) or its following letter is repeated that many times to create a new acronym. This newly-created acronym is also used to find possible expansions by using other rules. For example, "W3C" can be changed to "WWWC" in order to discover <W3C, "World Wide Web Consortium">.

(b) Higher-level mapping strategy. The higher-level mapping strategy tells how to map multiple letters to multiple positions. When more than one map-

[6] The English prefix currently used are: anti-, auto-, bi-, bio-, cent-, centi-, chem-, circum-, contra-, counter-, deci-, dis-, euro-, ex-, extra-, fore-, inter-, kilo-, mega-, micro-, mini-, multi-, out-, over-, post-, pre-, pro-, quad-, semi-, sub-, super-, tele-, trans-, tri-, manu-, ultra-. New prefixes can be added easily.

[7] Compound words currently used are: data-base, play-station, on-line, world-wide, north-west. New compound words can be added easily.

ping way is found from the acronym to a substring in the CW, the higher-level mapping strategy picks one expansion with the highest mapping score.

We use a backtracking algorithm to explore all possible ways of mapping. During the backtracking, the types of mapping rules decide which mappings are not able to be changed and which ones are able to be shifted after they are created.

If a letter is mapped to some position by a fixed mapping rule, it cannot be changed to map to other positions. If a fixed mapping cannot be applied, backtracking is required until a previous mapping established by a shiftable mapping rule is found. If a letter is mapped to some position by a shiftable mapping rule, this mapping can be removed and the letter can be mapped to some other position to the right. The neglectable mapping rules allow a letter to be neglected.

Each mapping rule is assigned a mapping score. These mapping scores are used to measure the quality (the confidence of correctness) of an acronym-to-expansion mapping. The general principles of the score setting are: longer expansions are preferred to shorter ones and regular mappings are preferred to less regular ones. Writing formats can also give hints about correct mappings, such as letters written in uppercase. Mapping scores can be set approximately and still achieve good performance, as long as they reflect these principles.

Algorithm 1 AcroMiner(A, CW)

Require: A: the acronym word. CW: the context window.
Ensure: $\mathbb{P}$: the set of mined acronym patterns whose acronym is A.
1: $j = 1$;
2: $\mathbb{P}' = \varnothing$;
3: **while** $j < |CW|$ **do**
4: FindAE(A, CW, 1, j); /* $\mathbb{P}'$ is used to store discovered acronym patterns whose expansions start from the j-th position in CW. */
5: $<A, E> = \arg\max_{ms(<A,E_i>)}\{<A, E_i> \in \mathbb{P}'\}$; /* $ms(<A, E>)$ is the mapping score from the acronym A to the expansion E. */
6: add $<A, E>$ to $\mathbb{P}$;
7: $j =$ 1st position of the word following the last word of E;
8: $\mathbb{P}' = \varnothing$;
9: **end while**

Given the acronym word and a CW, the mapping strategy works according to Algorithm 1. `AcroMiner` was designed to explicitly separate lower-level mapping rules (the mapping constraints) from higher-level mapping strategy. Line 1 of Algorithm 2 deals with choosing the applicable lower-level mapping rule for the specific letter and the rest of the code deals with the higher-level mapping strategy. If this line is treated as a black box, the higher-level mapping strategy is clearly separated from lower-level mapping rules.

Acronym Pattern Postprocessing. A postprocessing step is necessary for better acronym pattern ranking, as well as better acronym query extension. This component merges acronym patterns containing duplicate expansions that have the same meaning, but are written in different expressions. Once a duplication

Algorithm 2 FindAE(A, CW, i, j)

Require: i: the starting position of A. j: the starting position of CW. $\mathbb{P}'$: store patterns whose expansions start from position j in CW. '#': the delimiter.
Ensure: Map letters A[i...$|A|$] to CW[j...k] according to certain k. If new pattern is found, add it to $\mathbb{P}$.

```
1: k = arg min_{j≤k≤|CW|}{k|∃r related to the position of k}; /* r is lower-level mapping rule */
2: if k == ∅ then
3:     return
4: end if
5: if '#'∈ CW[j...k] then
6:     return /* Expansions containing delimiters are illegal. */
7: end if
8: if r can be applied to the mapping from A[i] to CW[k] then
9:     if i==|A| then
10:        add the newly discovered acronym pattern to ℙ';
11:    else
12:        FindAE(A, CW, i+1, k+1);
13:        if i==1 then
14:            return /* The mapping from A[1] to other positions than j is handled in AcroMiner()
               */
15:        else if r is shiftable then
16:            FindAE(A, CW, i, k+1); /* Try to find all the possible mappings */
17:        end if
18:    end if
19: else
20:    if r is shiftable then
21:        FindAE(A, CW, i, k+1); /* On failure, shift the letter to the next applicable position */
22:    else
23:        return /* If r is fixed, the failure is not shiftable, backtrack to earlier shiftable mappings
           */
24:    end if
25: end if
```

is detected between two acronym patterns, the one having smaller document frequency is removed and its document frequency is added to the one retained.

4 Ranking Acronym Patterns

Acronym patterns are not all equally useful. Some patterns are not popular and used by few people. `AcroMiner` may mistakenly discover false patterns, where the acronyms do not stand for mapped expansions. It is not desirable to treat the less popular or false acronym patterns in an equal fashion to the more popular or correct ones. A method for ranking, based on scores for the acronym patterns, is described next.

The qualities of acronym patterns can be quantitatively measured by *ranking scores*. The ranking score is controlled by three factors: (I) *pattern popularity*, (II) *gap between the acronym and its expansion* for every occurrence of the pattern and (III) *mapping score from the acronym to the expansion* for every occurrence of the pattern.

Acronym pattern popularity is measured by document frequency. The gap between the acronym and the expansion is measured by the number of words in-between the acronym and the expansion. The larger the gap is, the weaker the relevance is between the acronym and the expansion and thus there is less confidence to say the mapping is correct. The mapping score of an acronym

pattern (more accurately, from the acronym to the expansion) is the sum of the scores of the mapping rules that were applied to map letters of the acronym to positions of the expansion.

The *rating score* (rs) of an occurrence of an acronym pattern is measured by combining the gap and the mapping score and is given by:

$$rs(o) = \frac{\sum_i^{|A|} ms(A[i])}{f \times |A| \times g}. \quad (1)$$

There, o is an occurrence of the acronym pattern <A, E>. $|A|$ is the length of the acronym. $ms(A[i])$ is the score of the mapping rule applied to map the i-th letter of A to certain position of E. $\sum_i^{|A|} ms(A[i])$ is the mapping score from A to E. f is the maximum score among all the mapping rules. $f \times |A|$ can be thought as the "highest achievable" mapping score obtainable for A.

The *ranking score* is calculated by multiplying the average rating score among all occurrences of the acronym pattern with the popularity (df) of the pattern:

$$rank(p) = \frac{\sum_{o \in s_i, o=p} rs(o)}{tf(p)} \times df(p). \quad (2)$$

As we can see, an acronym pattern is ranked by considering the "fitness" of mapping the acronym word to the expansion and the frequency of seeing this pattern in the data set. Patterns with higher ranking scores should be placed at higher positions in the result list.

5 Acronym Query Extension Using Acronym Patterns

We now discuss the following questions about using acronym patterns (more precisely, the expansions) for query extension. If a user query is submitted that contains a word not found in a dictionary, should the system consider it as an acronym? Should every acronym be extended by its expansions for the retrieval task? Should the system consider all of the acronym's possible expansions, or only a subset of them?

Addressing these questions requires us to estimate the probability that a query term T is an acronym corresponding to an expansion E, i.e. $P(T \text{ is } A, E)$. Now $P(T \text{ is } A, E) = P(E|T \text{ is } A) \times P(T \text{ is } A)$, where on the right hand side, the former term is the probability that T stands for E, if T is definitely an acronym. The latter term is the probability that T is used as acronym. $P(E|T \text{ is } A)$ can be computed by:

$$P(E|T \text{ is } A) = \frac{rank(<A, E>)}{\sum_{E_i, <A, E_i> \in \mathbb{P}} rank(<A, E_i>)}. \quad (3)$$

Equation (3) says that the probability for acronym A to stand for E, is the ratio of the ranking score of acronym pattern <A, E>, to the sum of the ranking scores of all patterns having acronyms as A. $P(T \text{ is } A)$ can be computed by:

$$P(T \text{ is } A) = \frac{df(T \text{ is } A)}{df(T)}, \quad (4)$$

Table 1. Statistical information about misidentified patterns from V.E.R.A. data set.

Acronym length	# ground truth	# misidentified	percentage(%)
2	700	207	30
3	5537	153	3
4	3255	11	0.3
5 and above	1709	0	0

where $df(T \text{ is } A)$ is the number of documents containing acronym pattern <A, E> and $df(T)$ is the number of documents containing the word T. If T is used without any associated expansion (i.e. no expansion is discovered) in many documents, it is likely that T is not an acronym and is instead just a normal English word.

Finally, $P(T \text{ is } A, E)$ is derived by combining equations (3) and (4):

$$P(T \text{ is } A, E) = \frac{rank(<A, E>)}{\sum_{E_i, <A, E_i> \in \mathbb{P}} rank(<A, E_i>)} \times \frac{df(T \text{ is } A)}{df(T)}. \quad (5)$$

For a query term T, all expansions whose probabilities (as calculated by Equation (5)) are larger than a predefined threshold, will be selected and submitted along with T for the IR system to process with the query.

6 Experiments

Our experiments have been designed to evaluate the accuracy of `AcroMiner`, the efficiency of the ranking method and the usefulness of using acronym patterns to extend acronym queries.

Experiments on `AcroMiner`. The V.E.R.A[8] acronym dictionary was used to evaluate the performance of `AcroMiner`. This data set contains 11201 computer and IT related acronyms and their intended expansions. `AcroMiner` discovered 9134 acronym patterns, of which 8763 ones were correct. The recall is 78.2% and the precision is 95.9%. Statistics for the misidentified 371 patterns are given in Table 1. The table shows that it is more difficult to identify acronyms with shorter lengths. The fewer letters are in the acronym, the easier it is to mismatch them.

The performance of the online acronym extraction system Acrophile[9] can be compared with `AcroMiner` on the same data set. It discovered 8570 acronym patterns, among which 8058 ones were correct. The recall is 72% and the precision is 94%. This is not a precise comparison, because there is no means to adjust parameters, such as the window size W, for Acrophile. A large number of mistakes made by Acrophile were because it rigidly tried to map a letter to the 4th, 5th or 6th position in a word.

[8] Virtual Entity of Relevant Acronyms: `http://cgi.snafu.de/ohei/user-cgi-bin/veramain-e.cgi`.

[9] Acrophile demo: `http://ciir.cs.umass.edu/irdemo/acronym/getacros.html`.

`AcroMiner` was run on two real-world data sets in order to generate comprehensive acronym dictionaries for querying and other applications. The Web database is a collection of around 20 million crawled web pages. The Wiki data set[10] contains around 4 million Wikipedia articles written in XML. Only the plain text content was used for pattern mining and lower-level mapping rules 2 and 7 were switched off. It took roughly 12 hours to discover 563440 acronym patterns from the Web database and 2.5 hours to discover 118028 acronym patterns from the Wiki database. This time includes both the mining and ranking processes. On average, almost three new acronym patterns were discovered from every 100 Wikipedia articles or web pages. For the Web database, each acronym was mapped to 4.2 expansions on average. Interestingly, the acronym "ACE" was mapped to 493 expansions (not all of which may be valid).

It is difficult to evaluate the performance on these databases since no ground truth is available, but precision based on random sampling can be computed in the following way: 200 acronym patterns were randomly selected at a time and checked in Google for their correctness. This process was repeated for four times and the average precision measured in this way was 81%. If only the acronym patterns that ranked among the top five were selected, the precision was 91% and for only the top one, the precision was 98%.

Experiments on Ranking Acronym Patterns. Table 2 lists the top ranked acronym patterns for (I) `AcroMiner` using the Web database, (II) `AcroMiner` using the Wiki database and (III) The Acrophile [6] system. The Acrophile online acronym dictionary was mined using military and government documents. Many of its top ranked patterns are less well-known. For the ranking result for `AcroMiner` using the Web database, we compared with the ranks from AcronymFinder[11]. This web site is used for human assisted collection and ranking of acronym patterns using 5 levels, based on the popularity of use. The number next to each expansion indicates the ranking level given by that web site. As shown, most of the acronym patterns were ranked similarly by both `AcroMiner` and AcronymFinder, but in `AcroMiner` the ranking was done automatically, to a smaller level of granularity.

Experiments on Acronym Query Extension. 161 acronym queries were picked out from a query set[12] used as the benchmark data to evaluate algorithms of query extension.

The normalized discounted cumulative gain scoring measure (NDCG), for each of the first-page search results from a Web search engine was used to compare the qualities of the query results before and after being extended by acronym patterns. We use a special operation[13] to embed acronyms and expansions. The operation tells the search engine that phrases embedded should

[10] Hyperlink to download Wikipedia database: `http://en.wikipedia.org/wiki/Wikipedia:Database_download`.

[11] AcronymFinder online dictionary: `http://www.acronymfinder.com`.

[12] This data set is product-related and its information is hidden due to privacy issues.

[13] The operation is product-related and is hidden due to privacy issues. We denote it as `OP`.

Table 2. Ranking results of some acronym patterns returned from `AcroMiner` and Acrophile.

`AcroMiner` using Web database			
CS	AI	CSU	ACA
1.Counter Strike(1)	1.Artificial Intelligence(1)	1.Channel Service Unit(1)	1.American Correctional Association(1)
2.Computer Science(1)	2.Amnesty International(1)	2.California State University(1)	2.American Camp Association(1)
3.Customer Service(1)	3.American Idol(1)	3.Colorado State University(1)	3.Australian Communications Authority(1)
4.Creative Suite(1)	4.Adobe Illustrator(1)	4.Charles Sturt University(1)	4.American Chiropractic Association(1)
5.Community Server(-)	5.All Inclusive(1)	5.Christian Social Union(1)	5.American Counseling Association(1)
`AcroMiner` using Wiki database			
CS	AI	CSU	ACA
1.Counter Strike	1.Artificial Intelligence	1.Colorado State University	1.American Camp Association
2.Club Sport	2.Amnesty International	2.Christian Social Union	2.American Chiropractic Association
3.Computer Science	3.Artificial Insemination	3.California State University	3.American Counseling Association
4.Credit Suisse	4.Appenzell Innerrhoden	4.Cleveland State University	4.Amputee Coalition of America
5.Chief of Staff	5.Air India	5.Charles Sturt University	5.Agile Combat Aircraft
Acrophile			
CS	AI	CSU	ACA
1.Combat Support	1.Artifical Intelligence	1.California State University	1.Associate Contractor Agreements
2.Containment Spray	2.Amnesty International	2.Colorado State University	2.American Counseling Association
3.Congenital Syphilis	3.Active Ingredient	3.Computer Software Unit	3.Airspace Control Authority
4.Computer Science	4.Assignment Instruction	4.Computer Software Units	4.Airspace Coordination Area
5.Core Spray	5.Action Items	5.Conservation System Units	5.Administrative Cost Allowance

be treated equally and interchangeably but documents containing more of the phrases are not necessarily ranked higher than the ones containing fewer of the phrases. The results of using and not using this operation are compared.

45 acronym queries, after being extended to include their expansions, had improvements of their NDCG scores. 86 extended queries had no NDCG score change, while another 30 extensions led to decreases of NDCG scores. While these results indicate that acronym extension is indeed promising, it is hard to explain the 86 unchanged cases. The search engine we used considers many factors, which give alternative clues about how to properly rank results, regardless of any use of acronym patterns. It is therefore perhaps not surprising if the operation does little to affect the overall results. Also, the data sets we used for mining by `AcroMiner` to create its acronym dictionaries are still small compared to the entire Web. So this data limitation may cause `AcroMiner` to miss some popular acronym patterns. Acronym patterns are likely to provide more obvious improvements for other types of IR systems, where keyword-matching plays a more important role in overall ranking function.

7 Conclusion

In this paper we have studied the problem of mining acronym patterns from unstructured documents. We have developed `AcroMiner`, a highly open and flexible mining system that can handle large-scale data sets with high accuracy. We also presented a study on how to rank acronym patterns and use them for acronym query extension, a new problem in the area.

Acknowledgement. This work was partially supported by Microsoft Research Asia and National ICT Australia. We thanks Jiafeng Guo and Zhichao Zhou for their helpful suggestion.

References

1. Sergey Brin: Extracting Patterns and Relations from the World Wide Web. WebDB (1998) 172–183
2. Jeffrey T. Chang, Hinrich Schütze, Russ B. Altman: Creating an Online Dictionary of Abbreviations from MEDLINE. Journal of the American Medical Informatics Association **9** (2003) 612–620
3. David Hawking, Nick Craswell, Peter Bailey, Kathleen Griffihs: Measuring Search Engine Quality. Information Retrieval **4(1)** (2001) 33–59
4. James Pustejovsky, Jose Castano, Maciej Kotecki, Michael Morrell: Automatic Extraction of Acronym-Meaning Pairs from Medline Databases. Medinfo **10** (2001) 371–375
5. Kalervo Järvelin, Jaana Kekäläinen: IR evaluation methods for retrieving highly relevant documents. SIGIR (2000) 41–48
6. Leah S. Larkey, Paul Ogilvie, M. Andrew Price, Brenden Tamilio: Acrophile: An Automated Acronym Extractor and Server. ACM DL (2000) 205–214
7. Vladimir I. Levenshtein: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Doklady Akademii Nauk SSSR **163(4)** (1965) 845–848
8. Mendell Rimer, Michael O'Connell: BioABACUS: A Database of Abbreviations and Acronyms in Biotechnology and Computer Science. Bioinformatics **14(10)** (1998) 888–889
9. Ariel S. Schwartz, Marti A. Hearst: A Simple Algorithm for Identifying Abbreviation Definitions in Biomedical Texts. Pacific Symposium on Biocomputing (2003)
10. Kazen Taghva, Jeff Gilbreth: Recognizing Acronyms and Their Definitions. IJDAR **1(4)** (1999) 191–198
11. Jonathan D. Wren, Harold R. Garner: Heuristics for Identification of Acronym-Definition Patterns Within Text: Toward an Automated Construction of Comprehensive Acronym-Definition Dictionaries. Methods of Information in Medicine **41(5)** (2002) 426–434
12. Jun Xu, Yalou Huang: Using SVM to Extract Acronyms from Text. Soft Comput. **11(4)** (2007) 369–373
13. Ricardo A. Baeza-Yates, Berthier A. Ribeiro-Neto: Modern Information Retrieval. ACM Press/Addison-Wesley (1999) 0-201-39829-X
14. Stuart Yeates: Automatic Extraction of Acronyms from Text. New Zealand Computer Science Research Students' Conference (1999) 117–124
15. Jeonghee Yi, Neel Sundaresan: Mining the Web for Acronyms Using the Duality of Patterns and Relations. Workshop on Web Information and Data Management (1999) 48–52