A MACHINE LEARNING APPROACH TO RECOGNIZING ACRONYMS AND THEIR EXPANSIONS*

JUN XU, YA-LOU HUANG

College of Software, Nan Kai University, No. 94 Weijin Road, Tianjin, China
E-MAIL: nkxj@yahoo.com.cn, yellow@nankai.edu.cn

Abstract:

The paper addresses the problem of automatically recognizing acronyms and their corresponding expansions in free format text. To deal with the problem, we propose a machine learning approach. First, all likely acronyms are identified from text. Second, candidate expansions against likely acronyms are generated from their surrounding text. Last, we employ Support Vector Machines (SVM) to select the genuine expansions for acronyms. Experimental results show that our approach outperforms baseline method of using patterns. Experimental results also show that the trained SVM model is generic and performs well on different domains.

Keywords:

Acronym extraction; expansion; text mining; machine learning

1. Introduction

People will find it helpful, if we develop a system that can automatically recognize acronyms (e.g., *ROM*) and their expansions (e.g., *Read Only Memory*) in text. This is because many organizations have a large number of on-line documents which contain many acronyms. In many cases, however, acronyms occur frequently enough to make it difficult for outsiders to comprehend text. We conclude that the correct mapping of acronyms to their expansions is very important for understanding the documents. Detecting acronyms and their expansions is also useful in an information retrieval context. Explicit recognition of acronym/expansion pairs could make existing search engines work more effectively by performing in-line substitution.

Manually collected acronym lists have been complied and many are available in Internet [16], [17]. However, they are confined in specific domains/organizations. Moreover, with the rapid growth of acronyms, maintaining the lists is a big problem.

Finding all the acronyms, along with their expansions automatically in documents is a problem of text mining that has often been tackled using ad hoc heuristics. Previous efforts fall into three categories: natural language based approaches, rule based approaches, and multiple alignment based approaches. these approaches depend heavily on manually written rules and patterns.

In this paper, we propose a novel machine learning approach to recognizing acronyms and their expansions in text. First, all 'likely acronyms' are identified from text by heuristic rules. Second, candidate expansions against each likely acronym are generated from surrounding text. Last, we employ Support Vector Machine (SVM) model [11] to select the genuine expansions for acronyms.

Compared with conventional pattern-based approach, our machine learning approach has several advantages. First, machine learning frees human labors from the boring process of writing and tuning patterns/rules. Second, it is easier to conduct domain adaptation. With appropriate features, model trained in one domain will still performs well in another. Third, machine learning can utilize more evidences than patterns. Only *strong* evidences can be included into patterns for the difficulties of creating patterns. Machine learning model, however, can consider both *strong* and *weak* evidences collectively.

Experimental results indicate that our approach to recognizing acronyms and their expansions performs well on real world. We show that, the built SVM model can always choose the correct expansions for acronyms from a set of candidates. Experimental results also show that the SVM model trained on one domain performs well on another. This indicates that our trained model is generic and our approach can be adapted to different domains easily.

The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 gives a description of acronym recognition and Section 4 discusses our approach in detail. Section 5 reports experimental results and Section 6 summarize our work in the paper.

* Supported by the Ministry of Education under the grant 02038, and by the Asia Research Center at Nan Kai University under the grant AS0405

2. Related Work

In this section, we introduce previously work on acronym recognition, including manually maintained acronym lists and automatically recognizing acronyms.

2.1 Manually Built Acronym Lists

Many acronym lists have been compiled and published and many lists are available on the Web (e.g., Acronym Finder [16] and Canonical Abbreviation/Acronym List [17]). All these lists appear to be extracted manually rather than automatically.

The primary problem with many large lists on the Web is that they are hard to maintain, with the number of acronyms increasing fast. Some sites allow people to submit expansions and check them carefully. However, it is still time consuming to check people's submissions.

2.2 Automatic Acronym Recognition

Research on the topic of finding acronyms and their expansions in documents roughly falls into three categories: natural language based approaches, rule based approaches, and multiple alignment based approaches.

Natural language based approaches attempt to utilize results of natural language such as part-of-speech tagging to find expansions around acronyms. AcroMed [8] is such a system finding acronym-expansion pairs from medical documents. However, this approach is hard in practical for the complex pattern matching algorithm and its dependence on results of part-of-speech tagging.

Acrophile[6] is an example of rule based system. Authors of Acrophile observed web documents carefully and wrote a large number of patterns and rules for identifying acronyms and their expansions from text. Unfortunately, due to the complexity of relationship between acronyms and expansions in web documents, precision of Acrophile is low. For other work on rule-based system please refer to [7], [15].

As an example of multiple alignment based approaches, AFP (Acronym Finding Program) is a method of extracting algorithm in an OCR environment [10]. AFP is based on an inexact pattern matching algorithm applied to text surrounding the possible acronym. Other work fall into this category can be found at [2], [3], [4].

For other work on extracting acronyms please refer to [1], [9], [12], [13], [14].

In this paper, we present a novel machine learning approach which learns recognizing strategy from labeled examples.

3. Recognizing Acronyms and Expansions from Text

In Webster Dictionary, 'acronym' is defined as:

> a word formed from the first (or first few) letters of a series of words, as *radar*, from *ra*dio *d*etecting *a*nd *r*anging.

Acronyms are a relatively new linguistic feature in English language, first appearing in the 1940s and 1950s [12]. They are often defined by preceding (or following) their first use with a textual explanation. Acronyms are used in place of their expansions, where either readers are familiar with the concepts and entities they stand for or their meaning is clear from the context. Acronyms have following special properties:

- Generally, acronyms don't appear in standard dictionaries. To make their meaning clear, authors of documents may give their expansions at their first use, usually in form of *ROM (Read Only Memory)* or *Read* Only memory (ROM).
- Acronyms may be nested. For example ROM (Read Only Memory) is part of Programmable ROM (PROM).
- Acronyms are not necessary unique. For example, MIT may stand for both Massachusetts Institute of Technology and Management Information Tree. The later expansion comes from CISCO glossary list.
- Acronyms are generally three to ten characters in length, although shorter or longer acronyms do exist (e.g., AI for Artificial Intelligence). Acronyms' characters are alphabetic, numeric, and special characters such as '-', '/', or '&' etc. White spaces rarely appear in acronyms.

Heuristics rules/patterns are immediate and obvious approach to extracting acronyms. For example, AFP [10] is based on an inexact pattern matching algorithm applied to text surrounding the possible acronym. However, creating and tuning patterns is boring and time consuming. It is hard to conduct domain adaptation for pattern-based system. Also, writing patterns manually confines the usage of information. Only *strong* evidences can be reflected in patterns.

Machine learning approach can overcome these difficulties in a natural way.

Machine learning model is built on labeled examples. Labeling data is much easier than writing patterns/rules.

As to domain adaptation, experimental results in Section 5.3 show that the SVM model built in one domain performs well in another. With appropriate features, we can get a generic model.

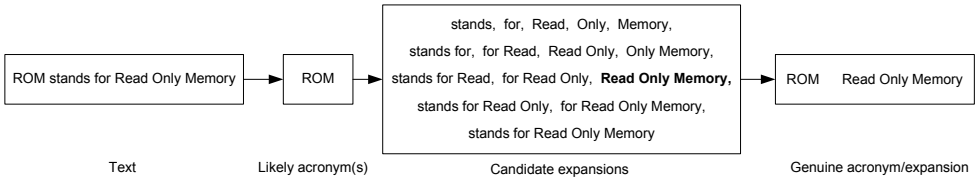

Figure 1. Illustration of recognizing acronyms and their expansions

Moreover, machine learning model can utilize variant kinds of evidences easily. In machine learning, acronyms, expansions and context they co-occur in are represented by features. Some features present *strong* evidences and also be implemented in pattern-based approaches. For example, the feature defined as: if first characters of expansion words constitute the acronym, set the feature value 1, otherwise 0. Some features present *weak* evidences that cannot be included into pattern-based approaches easily. For example, the feature defined as: if the acronym appears in its precede text (text left to the acronym), set the feature value 1, otherwise 0. Since acronyms are often defined at their *first* appearance in text, the feature indicates that perhaps the candidate is not a genuine expansion. However, this is not always the truth. Following real world sentence is a counterexample. "*The SVG DOM conforms to key aspects of Document Object Model (DOM) Level 1 Specification and Document Object Model (DOM) Level 2 Specification.*"

4. Our Approach

Our approach consists of three steps: identify likely acronyms, generate candidate expansions, and select genuine expansions. Figure 1 illustrates the outputs of each step based on a simple example text "*ROM stands for Read Only Memory*".

4.1 Identify Likely Acronyms

The aim of this step is to identify all possible acronyms from original text. Just as the first step of example illustrated in Figure 1, we are to identify acronym *ROM* from text *ROM stands for Read Only Memory*.

Based on properties of acronyms listed in Section 3, in this paper, we consider a string of alphabetic, numeric, and special characters as likely acronyms if it satisfies following conditions:

1. Its length is between 2 and 10 characters and at most consists of one white space.
2. Its first character is alphabetic or numeric. At least one letter is capital. (The first letters of the initial words of sentences are not counted in.)
3. It is not a known word in dictionary. It is not person name, location name or word in a predefined list of stop words.

The first condition restricts the length of acronyms. The second and third keeps many person names, location names, and common words from being treated as likely acronyms. Based on these restrictions, we can get a list of likely acronyms from text. Obviously, these three restrictions are much loose. Many likely acronyms have no expansions at all. These false identified acronyms will not impact final result because they will be removed automatically in following steps.

4.2 Generate Candidate Expansions

The aim of this step is to search all candidate expansions for acronyms identified in above step. As the second step of the example shown in Figure 1, we are to generate a set of candidate expansions for the acronym *ROM*.

We observed that expansions always occur in surrounding text where acronyms appear in and always in the same sentence. Based on this observation, we search candidate expansions for an identified acronym from its surrounding text. One of these candidates may be the genuine candidate against the acronym, if the expansion exists.

Before generating candidate expansions, 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.

We search all candidate expansions for a given acronym within a sentence. Two parts of the sentence are searched respectively – the substring precedes the acronym (left context) and substring follows the acronym (right context). Let us consider searching candidate expansions from left context first.

We get all the candidate expansions by procedure shown in Figure 2.

In the procedure, `offset` is the number of tokens between acronym and candidate. `length` is the number of tokens in candidate. `tokens_in_leftcontext` indicates

```
For offset from 0 to tokens_in_leftcontext
   For length from 1 to tokens_in_leftcontext-offset
      Add GetLeftExpansion(leftcontext,offset,length) to Candidates Set
   End
End
```

Figure 2. Procedure for searching all candidate expansions

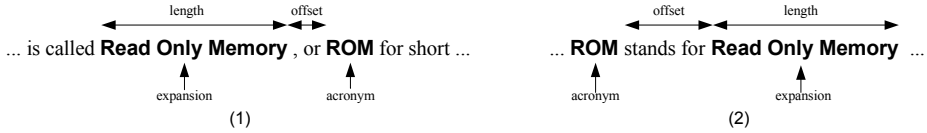

Figure 3. Generating candidate expansions

the number of tokens in `leftcontext`. Function `GetLeftExpansion` returns the candidate expansion which:

1. Distance from candidate expansion to right boundary is `offset` tokens.
2. The candidate contains `length` tokens.

The returned candidate expansion is added to the candidate set.

As the shown by (1) of Figure 3, we get the genuine expansion '*Read Only Memory*' for acronym *ROM* when `offset=2` and `length=3`. Note ',' is considered as a token.

Generating candidate expansions from right context is similar to the case of from left, except that `offset` is counted from left boundary. In (2) of Figure 3, we get the genuine expansion at `offset=2` and `length=3`.

Theoretically, all substrings within the sentence can be enumerated, just as the procedure shown above. However, this is not necessary. We observed that the genuine expansions always appear near to acronyms. Lengths of genuine expansions (by tokens) always approximately equals to or a little longer than their acronyms (by characters).

Thus we define two parameters for dramatically reducing the number of candidates generated. The first parameter, *maxoffset*, controls the maximum distance between acronyms and their expansions. In this paper, we set *maxoffset* to 10 tokens; it is sufficient large on our corpus. The second parameter, *maxlength*, controls the maximum length of expansions. We assume that, for long acronyms, the length of expansions (by tokens) should not longer than the acronym length (by characters) plus 5. Also, for short acronyms, it should not longer than twice of the acronym length. Thus we get following formula for calculating *maxlength*:

$$maxlength = \min(length(acronym) + 5, \; length(acronym) \times 2), \quad (1)$$

where *length*(*acronym*) is the number of characters in acronym.

4.3 Select Genuine Expansions

The aim of this step is to select the genuine expansions for acronyms from candidate set. As the last step of the example shown in Figure 1, we are to select candidate *Read Only Memory* as the genuine expansion for *ROM*.

In this step, acronym, candidate expansion, and the context they appear in are represented by a set of features. We employ a SVM model to determine whether the candidate is the genuine expansion for the acronym or not, based on the features. Figure 4 illustrated the flowchart of selecting expansions.

Section 4.3.1 and 4.3.2 discuss the SVM model and features utilized, respectively.

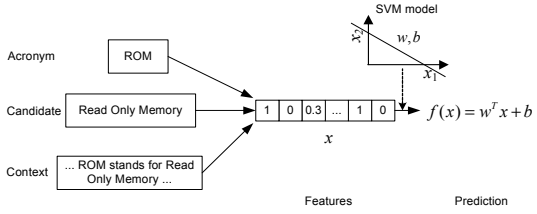

Figure 4. Selecting genuine expansions from candidates

4.3.1 SVM Model

We take a statistical machine learning classifier to address the problem. We labeled acronym/expansion candidates in advance, and use them for training. Let us describe the problem more formally. Given a training data set $D = \{x_i, y_i\}_1^n$, we construct a model that can minimize error in prediction of y given x (generalization error). Here $x_i \in X$ and $y_i \in \{+1, -1\}$ represent a definition candidate and a label representing whether it is a genuine expansion for the acronym, respectively. When applied to a new instance x, the model predicts the corresponding y and outputs the score of prediction. We employ SVM as the classifier model to assign prediction scores. Given an instance x, SVM assign as score to it based on

$$f(x) = w^T x + b \quad (2)$$

where w denotes a vector of weights and b denotes a intercept. The sign of $f(x)$ is used. If it is positive, x is classified into the positive category, otherwise into the negative category. The construction of SVM needs labeled training data. Details of the learning algorithm can be found in [11]. In a few words, the learning algorithm creates the 'hyper plane' in (2), such that the hyper plane separates the positive and negative instances in the training data with the largest 'margin'.

In some cases, an acronym may be mapped to more than one candidate within one sentence. (SVM model assigns more than one positive score to different candidates within a sentence for one acronym.) We choose the one has highest prediction score as the final genuine expansion. In some other cases, no candidate is identified as expansion for an acronym. (SVM model assigned all generated candidates negative scores.) The acronyms are automatically discarded.

4.3.2 Features

In SVM model, we utilize both binary and real valued features as described below. These features are created to characterize the likely acronyms, candidate expansions, and context they appear in.

- *Features characterizing acronym and expansion*

  Length of acronym, lower case, numeric, special characters, and white spaces in acronym are important indicators to determine whether it is a good acronym.

  Length of expansion, words with initial capital letters in expansion, first/last expansion words and preposition/conjunction in expansion are good indicators to tell whether it is a good expansion. Symbols such as '(', ')', '[', ']', '{', '}', '=', '!', and '?' are indicators that the expansion are not good.

  Relationship between acronym and expansion are considered seriously as features. E.g., if the acronym letters matched with the first letter of words in expansion, it is likely that the candidate is the genuine expansion against the acronym.

- *Features characterizing context*

  We also rely upon context features, though some of them are not so strong evidences. For example, most acronyms are surrounded by parenthesis and appear immediately after expansions. If acronyms already appear in their preceding text (left context), the probability that the candidate is a genuine expansion is low.

5. Experiments

We have conducted experiment to verify the effectiveness of our proposed approach. Particularly, we have investigated whether the problem of recognizing expansion and their expansion can be solved by machine learning approach proposed. Also, we experimentally proved that the trained model is generic.

5.1 Baseline and Measures for Evaluation

As baseline method, we used a pattern-based approach, which is similar to AFP[10].

We made use of *precision* and *recall* for measuring performance. Following equations (3) and (4) give the details.

$$precision = \frac{\text{\# of correct acronym expansions found}}{\text{total \# of acronym expansions found}} \quad (3)$$

$$recall = \frac{\text{\# of correct acronym expansions found}}{\text{total \# of acronym expansions in documents}} \quad (4)$$

5.2 Recognizing Acronyms and Expansions from UCI data set

We used the UCI data set '*NSF Research Awards Abstracts 1990-2003*' [5] for this experiment. The dataset consists of 129,000 abstracts describing NSF awards for basic research. Acronyms and expansions often appear in these abstracts.

We randomly selected 1000 documents. All occurrences of acronyms and their expansions are labeled manually. Our final data set contains 639 times of acronym/expansion co-occurrences. We used the labeled set for training and evaluation.

The dataset is split into two parts randomly: one for training the SVM model (train set) and another for testing performance (test set). Results reported in Table 1 are performance on test set.

From Table 1, we can see that our approach outperforms baseline on both precision and recall. The results indicate that our approach of using machine learning for recognizing acronyms and their expansions are effective.

Table 1. Performance on UCI data set

	Precision	Recall
Baseline	0.8157	0.8201
Our approach	**0.9086**	**0.8413**

It is not surprising that our approach performs better than baseline. In Section 3, we have shown the advantages

of our approach, especially that machine learning model can utilize both *strong* and *weak* evidences collectively. In patterns, however, many *weak* but useful information (e.g., acronym appear in preceding text) is ignored.

5.3 Recognizing Acronyms and Expansions from W3C

In this experiment, we tested whether a generic model (SVM model) can be constructed for recognizing acronyms.

As training data, we used the train set used in Section 5.2. To create the test data set, we randomly selected 100 documents from TREC W3C corpus, WWW scope. All the co-occurrences of acronyms-expansion pairs are labeled and the final data set consists of 151 times of acronym/expansion co-occurrences.

From results reported in Table 2, we see that our approach still outperforms the baseline method, though the SVM model is trained in another domain. In Section 4.3.2, we have described the features used in SVM. The features are domain independent. That is why we can create a domain independent generic model.

Table 2. Performance on W3C data set

	Precision	Recall
Baseline	0.8493	0.8212
Our approach	**0.8936**	**0.8344**

6 Conclusions

In this paper, we proposed to address the issue of recognizing acronyms and their expansions from free format text using SVM model. We have proposed a novel approach based on machine learning. Specifically, we first identify likely acronyms and then generate candidate expansions from text surrounding acronyms. Last, we use a SVM model to choose the genuine expansions for acronyms from candidates. We have shown the advantages of the machine learning approach, especially that it can utilize variant evidences easily and effectively.

Experimental results indicate that our proposed method performs better than the baseline method of using patterns. The results also show that our proposed method works well in different domains and can be adapted to other domains easily.

References

[1] Adar, E.: SaRAD: a Simple and Robust Abbreviation Dictionary. HP Laboratories Technical Report, 2004.

[2] Bowden, P. R.: Automatic Glossary Construction for Technical Papers, Nottingham Trent University, Department Working Paper, December 1999.

[3] Bowden, P. R., Halstead, P., and Rose, T. G. Dictionaryless English Plural Noun Singularisation Using a Corpus-Based List of Irregular Forms, in: Ljung M., ed. In Proc. of 17th Int. Conf. on English Language Research on Computerized Corpora, Rodopi, Amersterdam, Netherlands. pp. 130-137.

[4] Chang., J. T., Schutze H., and Altman R. B.: Creating an Online Dictionary of Abbreviations from MEDLINE, J. Am. Med. Inform. Assoc., 9.

[5] Hettich, S. and Bay, S. D.: The UCI KDD Archive. Irvine, CA: University of California, Department of Information and Computer Science.

[6] Larkey, L. S., Ogilvie, P., Price, M. A., and Tamilio, B.: Acrophile: An automated acronym extractor and server. In the Proc. of 5th ACM Conf. on Digital Libraries. San Antonio, TX: ACM Press, 2000.

[7] Park, Y., and Byrd, R. J.: Hybrid text mining for finding abbreviations and their definitions. In Proc. of EMNLP, 2001.

[8] Pustejovsky J., Castano J., Cochran B., Kotecki M., and Morrell M.: Automatic Extraction of Acronym-meaning pairs from MEDLINE databases. In Proc. of Medinfo, 2001.

[9] Schwartz, A. S., and Hearst, M. A.: A simple algorithm for identifying abbreviation definitions in biomedical text. In Proc. of the Pacific Symposium on Bio-computing, 2003.

[10] Taghva, K. and Gilbreth, J.: Recognizing Acronyms and their Definitions. Information Science Research Institute, University of Nevada, Technical Report TR 95-03.

[11] Vapnik, V.N.: The Nature of Statistical Learning Theory. by VN Vapnik. Berlin: Springer-Verlag, 1995

[12] Yeates, S.: Automatic Extraction of Acronyms from Text. In Proc. of the Fourth New Zealand Computer Science Research Students' Conference, 1999

[13] Yeates, S., Bainbridge, D., and Witten, I.H.: Using compression to identify acronyms in text. In Proc. of Data Compression Conf., IEEE Press, New York, NY, p. 582.

[14] Yoshida, M., Fukuda, K., and Takagi, T.: PNAD-CSS: a workbench for constructing a protein name abbreviation dictionary, Bioinformatics, 16, pp. 169-175.

[15] Yu, H., Hripcsak, G., and Friedman, C.: Mapping abbreviations to full forms in biomedical articles. J Am Med Inform Assoc, 2002.

[16] Acronym Finder: http://www.acronymfinder.com/

[17] The Canonical Abbreviation/Acronym List: http://www.astro.umd.edu/~marshall/abbrev.html