# High-recall extraction of acronym-definition pairs with relevance feedback

Anna Yarygina
Saint-Petersburg University
anya_safonova@mail.ru

Natalia Vassilieva
HP Labs
nvassilieva@hp.com

## ABSTRACT

This paper addresses the problem of extracting acronyms and their definitions from large documents in a setting, when high recall is required and user feedback is available. We propose a three step approach to deal with the problem. First, acronym candidates are extracted using a weak regular expression. This step results in a list of acronyms with high recall but low precision rates. Second, definitions are constructed for every acronym candidate from its surrounding text. And last, a classifier is used to select genuine acronym-definition pairs. At the last step we use relevance feedback mechanism to tune the classifier model for every particular document. This allows achieving reasonable precision without losing recall. As opposed to existing approaches, either created to be generic and domain independent or tuned to one particular domain, our method is adaptive to an input document. We evaluate the proposed approach using three datasets from different domains. The experiments prove the validity of the presented ideas.

## 1. INTRODUCTION

Automatic extraction of acronyms along with their definitions from a text is important for information extraction and information retrieval tasks due to the wide usage of acronyms in textual data. An acronym is usually introduced along with its definition when it is first mentioned in a document. Many approaches have been proposed for acronym extraction in the literature [2–12], all of them following a similar scheme: (1) candidate acronyms recognition, (2) definition extraction, (3) filtering of acronym-definition pairs (an optional step). The first and second steps usually exploit a set of hand-crafted rules or patterns, designed to match common formats of acronyms definitions. At the third step either scoring rules with predefined thresholds are applied, or a classifier is used. The classifier is trained on a labeled dataset with a set of features which are commonly derived from the similar rules. The result of applying an acronym extraction algorithm to a set of documents is a dictionary of acronyms defined in these documents. The rules and patterns used in the extraction process are commonly designed to achieve a higher precision of the resulting dictionary rather than a higher recall. A missed acronym-definition pair is considered less important than an invalid entry in a dictionary. This paper addresses the problem of extracting acronym-definition pairs with high recall.


*BEWEB 2012*, March 30, 2012, Berlin, Germany.


We have been faced the problem of extracting acronyms from a specific set of documents: financial and government reports. The extracted acronym-definition pairs are used to enable correct aggregation of financial data from these documents. Acronym-definition extraction is one of the important components of the system aimed at capturing, in depth, intelligence about customers, competitors and any aspect of the environment needed to support executives and managers in making strategic decisions for an organization. The system allows to automate the extraction of relevant business and information technology data from available unstructured data sources and to record the knowledge procured in a formal, structured data repository. This task requires high recall of results of every component including a high recall of a resulting acronym-definition dictionary. A missed acronym-definition pair might lead to a missed opportunity for a business. Precision of the resulting dictionary is not that important. After the dictionary is constructed, it is reviewed by an expert prior to the data aggregation step. During the data aggregation step, the dictionary from a particular processed document is integrated into the common data repository and used for extracting facts from tabular data.

The documents under our consideration are typically large (tens of hundreds of pages) and contain many domain-specific and document-specific acronyms, which are defined and used only within the same document or within a small set of documents produced by the same company or government department. These acronyms cannot be found in general dictionaries and are often introduced in different and sometimes uncommon formats. The rules and patterns of common acronym extraction algorithms are not wide enough to match all definition formats. But although a format might differ for different documents, it is usually consistent within the same document. Thus, relevance feedback might be used to adapt an extraction algorithm to a particular large document.

In this paper we propose an algorithm for extracting acronym-definition pairs from large documents in a setting,

when high recall is required and user feedback is available. We exploit the hypothesis that rules and patterns for acronym creation and definition are the same within the document. Following a common scheme, we propose a three step approach: (1) acronym candidates are extracted using a weak regular expression, which matches a wider set of acronyms compared to other approaches; (2) definitions are constructed for every acronym candidate from the surrounding text using a wide set of rules; (3) a classifier is used to select genuine acronym-definition pairs. At the last step we use relevance feedback mechanism to adapt the classifier model for every particular document. According to our knowledge, this is the first attempt to develop a recall-oriented algorithm for extracting acronym-definition pairs and to use relevance feedback to make an algorithm adaptive to a particular input document.

The rest of the paper is organized as follows. Section 2 contains an overview of related work. The approach description in section 3 is followed by the experimental settings and the analysis of the experimental results in section 4.

# 2. RELATED WORK

## 2.1 Acronym Extraction Approaches

At the acronym extraction phase, manually written rules or regular expressions are usually applied. Rule-based approaches are described in [2, 7, 9–11]. Regular expressions are applied at the acronym extraction phase in [4–6, 8]. All these techniques exploit a specific set of features: the length of acronyms, uppercase letters and special symbols in acronyms, brackets surrounding acronyms in the text. Some of the proposed regular expressions are naive and select acronyms with high accuracy but miss a lot of less known patterns [11], the others are too wide and select too many false positive acronyms [2].

The acronym extraction step is concluded by filtering in [2, 9, 10]. All extracted acronyms are matched with a dictionary of common words and well-known acronyms. This step improves precision of the selected set of acronyms. However, we can miss some acronyms, for example, "FAST (Five hundred meter Aperture Spherical Telescope)" and "EU (Entropia Universe)".

The majority of the proposed heuristic-based techniques select acronyms with a small number of false positives, but with low recall. In our setting, acronym extraction step is required to find acronym candidates with high recall. However, we achieve high recall results at the expense of decreasing precision. We apply candidates refinement to solve this problem.

## 2.2 Definition Extraction Approaches

At the definition extraction step the majority of known algorithms construct an appropriate definition near the acronym position in the text [2–12]. In most cases authors use different heuristics to decide where to search for the definition and how to find it. The length and position of windows surrounding the acronym are analyzed and heuristic-based patterns to match acronym-definition pair are developed.

For example, in [9] left and right windows of $2n$ words near an acronym are constructed, where $n$ is the number of letters in the acronym. The sequence of symbols in the acronym is compared to the sequence of the first letters of the words inside the window to select a valid definition. Classes of words such as stopwords, normal words, and hyphenated words are taken into account during matching a definition with an acronym.

Several letters in definition words are sequentially compared with acronym letters in [11]. This approach enables the extraction of more complex acronym definitions compared to the previous one.

Less restrictive and more language-independent rules for definitions extraction are also applied [2, 7, 10]. The definition construction phase is based on techniques for terms extraction from a text in [6].

Some of these approaches construct no more than one definition and are very precise but produce a lot of false negatives [5, 6, 8, 9]. In this case further acronym-definition pairs dictionary refinement is not needed. Other techniques construct acronym-definition dictionary with high recall, but in this case refinement is needed because of high rate of false positive entries in the resulting dictionary [2–4, 7, 10–12]. We will follow the second approach to keep recall of acronym-definition dictionary at this step.

## 2.3 Dictionary Filtering Approaches

The authors of [7] use acronym-definition filtering rules for dictionary refinement. This technique is not flexible and takes a lot of manual work to adopt it to a special set of documents.

Machine learning techniques are proposed to refine the dictionary of acronym-definitions in [2, 10, 11]. However, sets of features for classification and classification algorithms vary in these papers. A limited set of features is selected in [11]: the length of the acronym, the length of the definition, the number of words in the definition, the number of stop words in the definition. In [2] the refinement of candidates is implemented as classification based on 12 features, such as the number of characters in the definition and the acronym, or whether the acronym or its definition is between parentheses. In [10] the authors use the features of the acronym and the definition and contextual features in their dictionary refinement model. Independently of the feature sets, all dictionary refinement techniques are based on a fixed training set which does not change in time. In this case the processing of a significantly different document may produce an unacceptable result.

Unlike other algorithms, the unsupervised ranking algorithm enables the refinement of the acronym-definition dictionary in [3]. A method for acronyms disambiguation based on graph of links between pages is also described there. The authors of [4] similar to their colleagues discuss techniques for ranking acronyms and their definitions according to the context of their usage.

An approach based on comparison of the numbers of bytes needed to code the acronym and its definition is proposed for the candidates filtering in [12].

## 2.4 Baseline and Similar Techniques

One of the most known and widely used techniques for acronyms extraction is described in [8]. During the first step acronyms are extracted as words in brackets. During the second step a definition is consequently constructed in the left window. The size of the window depends on the length of the acronym. The advantage of this approach is that the resulting dictionary is quite precise and no marked up data are needed for training at the filtering step. In our experiments we use the implementation of this approach in JULIE Lab (JACRO - JULIE Lab Acronym Annotator) as a baseline.

Our approach is similar to the one proposed in [10]. The authors clearly distinguish three steps for extraction of acronym-definition pairs: definition of "all likely acronyms", construction of expansion candidates, and selection of genuine expansions for acronyms. Similar to our solution they apply heuristic-based rules at the first and at the second steps and exploit a machine learning approach to select genuine acronym-definition pairs. We have implemented the approach proposed in [10] but unfortunately have not received results comparable with those described in the paper. We believe we were not able to exactly reproduce the proposed solution because of very vague description of the feature set used for the classification in the paper.

In this work we inherit ideas from [10] and, in addition, introduce an iterative dictionary refinement technique using the relevance feedback. The proposed approach enables our application to adapt to specific documents. To our knowledge, there are no acronym-definition pairs extraction techniques which apply incremental changing of a dictionary refinement model based on relevance feedback from the user.

# 3. ACRONYM-DEFINITION PAIRS EXTRACTION TECHNIQUE

We divide our technique for extraction of acronyms and their definitions from plain text into three steps: acronym extraction, definition extraction, and candidates refinement. Let us discuss each step of the proposed technique in turns.

## 3.1 Acronym Extraction

The Acronyms Extraction module extracts words and groups of words which are likely to be acronyms. Our final goal is to construct a dictionary of acronyms and their definitions with reasonably high recall. Therefore, the Acronym Extraction module should extract as many acronym candidates as possible.

We started from the baseline regular expression presented in JACRO (see, equation 1). Acronym candidates are determined by adjacency to parentheses. Acronyms are considered valid candidates only if they consist of one word which contains at least one uppercase letter.

$$\begin{array}{l}[\ \backslash(\ \backslash[\ ] \\ \quad [\ -\backslash w\ ]*? \\ \quad\quad (\ [A-Z]-?\ \backslash w \mid \backslash w\ -?\ [A-Z]\ ) \\ \quad [\ -\backslash w\ ]*? \\ [\ \backslash)\ \backslash]\ ]\end{array} \tag{1}$$

We have built a regular expression (see, equation 2) to extend the baseline regular expression. The Acronym Extraction module selects acronyms as all words or groups of words which match this regular expression. The regular expression matches substrings constituted of two groups of uppercase letters or digits, which may be joined together by other symbols. It is important to mention that the regular expression matches different symbols framing the acronym in the text.

$$\begin{array}{l}[=\ \backslash(\ ,.-\ ]+ \\ \quad [\ 0-9a-zA-Z-\ ]* \\ \quad\quad \Big(([A-Z]+\ [\ ]\ [A-Z]+) \mid ([A-Z]+) \mid ([0-9])\Big) \\ \quad [']/0-9a-z\ \backslash)\ \backslash(\ A-Z\&.-]* \\ \quad\quad \Big(([A-Z]+\ [\ ]\ [A-Z]+) \mid ([A-Z]+) \mid ([0-9])\Big) \\ \quad [']/0-9a-z\ \backslash)\ \backslash(\ A-Z\&.-]* \\ [-,=.\ \backslash)\ \backslash(\ ]+\end{array} \tag{2}$$

The regular expression proposed in equation 2 matches many false acronym candidates but enables the extraction of acronyms with higher recall. For example, the regular expression in equation 1 does not match the acronyms in several cases: "NCES -National Center for Education Statistics"; "PS Act (Packers and Stockyards Act of 1921)"; "(R/EOC) Regional/ Emergency Operations Center" and others. In contrast, the regular expression in equation 2 matches a large portion of words, which are not acronyms (e.g., "OTHER PURCHASES", "In Service", "Sub-Activity"). Matching acronyms with dictionaries of common words was proposed to solve this problem [9, 10]. However, we do not apply this technique to keep recall of our approach.

## 3.2 Definition Extraction

The Definition Extraction module extracts definitions for every previously identified acronym.

Our implementation (Rule-based) constructs definitions in the left and right windows of the words surrounding the acronym. The proposed algorithm consequentially adds a new word into a definition. A new candidate definition is checked against the following set of heuristics.

- The definition consists of no more than 8 words.
- The first word (in case of the left window) or the last one (in case of the right window) in the definition is not a stop word.
- At least one letter in the acronym is the first letter in some word in the definition.
- Almost all symbols in the acronym consequently appear in the definition, i.e. there are no more than 2 missed letters.
- The length of the definition is less than 100 symbols.
- The definition contains at least one lowercase letter.
- The acronym is not a substring of its definition.

- A newly constructed word in the definition contains a letter from the acronym.
- The acronym identified by the Acronym Extraction module is compared to the acronym which is automatically generated from the definition as a sequence of the first letters of the definition words. The Levenshtein distance [1] between the extracted and generated acronyms is less than 2.

Definitions are considered as valid candidates only if they satisfy all these rules.

This technique is less restrictive than the previous one and finds more definitions. For example, this algorithm can extract pairs "Autonomous Aerial Refueling (AAR)" and "STOVL - Short Take-off and Landing". In contrast with the baseline approach (JACRO), the Rule-based technique may construct several definitions for one acronym. This approach enables correct extraction of definitions in several special cases such as "Office **o**f **P**ersonnel Manage**m**ent (OPM)", which are not covered by the baseline. In this example, the baseline technique matches bold letters and stops the process of the definition construction.

## 3.3 Candidates Refinement

Our extensions in the Acronym Extraction and the Definition Extraction modules improve recall of the resulting dictionary without taking into consideration its precision. Therefore, we developed a technique to filter the resulting dictionary without significant loss of recall. We selected methods proposed in [2, 10] as a concept. Classification based on a specific set of features is used in the Candidates Refinement module.

### 3.3.1 Classification Problem for Candidates Refinement

We exploit a machine learning approach to address the problem. C4.5 classifier was used in our implementation; however preliminary experiments with other classification algorithms have shown comparable results.

A set of acronym-definition pairs is labeled in texts in advance to use it for training a classification model and to test performance.

We have a dictionary $D = \{x_i, y_i\}$. A classification model is learned in order to minimize error in prediction $y$ given $x$. Here $x_i$ represents an acronym-definition pair and $y_i \in \{true, false\}$ indicates the validity of this pair in the dictionary.

### 3.3.2 Classification Features

Our classifier is based on a heuristic-based feature set. We use 100 features of acronym-definition pairs in total. They can be divided into several groups:

- The number of symbols in the acronym and its definition;
- The number of uppercase symbols in the acronym and its definition;
- The number of lowercase symbols in the acronym and its definition;
- The number of words in the definition;
- The number of different special symbols between the acronym and its definition in the text;
- The statistics on parts of speech in the acronym and its definition (based on part of speech tagger in OpenNLP[1]).

### 3.3.3 Training set

We need to learn a classification model from a training set when a feature set is chosen.

The behavior of a classifier highly depends on the balance of the training set. If the training set consists mostly of negative examples then the classifier tends to classify all objects (acronym-definition pairs) as invalid. The balance of the training set, i.e. the proportion of positive and negative examples, enables tuning the classification model and therefore increases the effectiveness of the Candidates Refinement module.

Another parameter of the Candidates Refinement module is the number of instances in a training set. The size of the training set influences the performance of the classifier and the time required for its training. Creation of large handcrafted training sets is an expensive process. Therefore, we investigate the following questions: how the size of a training set affects the performance of the classification algorithm and whether duplicated examples (the use of examples several times) in the training set are useful.

### 3.3.4 Refinement Model Based on Relevance Feedback

At the Candidates Refinement phase we use relevance feedback provided by the user in order to improve the quality of the resulting dictionary of acronyms and their definitions. In our setting, the user reads the document and marks up correctly and incorrectly extracted acronym-definition pairs. Therefore, we are able to collect new instances for the training set and retrain the classification model.

In this work we address the problem of extracting acronyms from large documents. Therefore, it is useful to retrain a classification model based on a locally collected training set and process the unread part of a document with the new classifier in order to provide to the user the results with higher quality.

Figure 1 presents the schema of document processing based on the user relevance feedback. The Candidates Refinement module stores a Global Classification model, which is used to classify candidate acronym-definition pairs during the first processing of a document. Dictionaries for the initial training of the Global Classification model are selected in advance and are usually based on known benchmarks.

After that, the user observes the document and marks up a portion of it.

[1]http://incubator.apache.org/opennlp/index.html

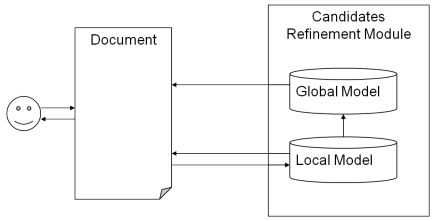



**Figure 1: Iterative refinement model for the Candidates Refinement phase**

The data collected from the user feedback is used for training a Local Classification model. The Local Classification model is constructed in order to take into account specificity of the document. The training set for the Local Classification model is collected as a combination of instances marked up by the user and those from the global training set.

After learning the new Local Classification model, it is applied to re-classify the dictionary and the user observes new results in the unseen part of the document. During the processing of the document the number of instances accumulated from the user increases, and the Local Classification model is periodically retrained and applied again. One may see that the Local Classification model becomes more specific over time.

When the whole document is processed, the Candidates Refinement module uses the relevance feedback provided by the user to retrain the Global Classification model. In our experiments we have a fixed-size training set for the Global Classification model. Therefore, at this step newly constructed local examples replace a portion of old examples in the training set. The newly constructed Global Classification model is used for processing subsequent documents.

It is important to mention that the behavior of the Candidates Refinement module highly depends on the ratio of examples from the current document and from the global training set. This parameter of the Candidates Refinement module can also be tuned in order to improve its effectiveness.

# 4. EXPERIMENTS

In this section, we analyze the effectiveness of the proposed pipeline for extracting acronyms and their definitions from plain text. Experimental analysis demonstrates how tuning of parameters and combination of algorithms affect performance of the whole approach.

## 4.1 Data Sets and Metrics

We used three data sets in our experiments to evaluate the performance of the proposed approach (see table 1):

1. *Medstract* data set is publicly available at http://medstract.org together with the corresponding list of acronym-definition pairs. This data set consists of 401 paper abstracts from the medical domain.
2. *Military* data set is a report from 2010 by the Office of Budget Department of the Navy. The report contains a list of acronyms, which we have used to mark up this document automatically. After that, the automatically constructed set of acronyms and their definitions was manually purified.
3. *Disa* data set is 2010 Budget Estimates report of the Defense Information Systems Agency. The acronym-definition pairs were collected manually.

**Table 1: Data sets for experiments**

| | medstract | military | disa |
|---|---|---|---|
| # of pairs | 368 | 207 | 350 |
| # of words | 81762 | 44936 | 60392 |
| # of sentences | 3596 | 1316 | 1543 |

**Table 2: Performance of the Acronyms Extraction phase**

| | medstract | | military | | disa | |
|---|---|---|---|---|---|---|
| Regexp | P | R | P | R | P | R |
| JACRO | 0.79 | 0.93 | 0.61 | 0.88 | 0.77 | 0.93 |
| Extended | 0.12 | 0.95 | 0.10 | 0.99 | 0.12 | 0.97 |

We have constructed training sets for the Candidates Refinement phase from these data sets. To extend the training sets with negative examples we executed the Acronym Extraction and the Definition Extraction steps and then marked up the resulting dictionaries with already known labels.

We used C4.5 classifier to refine the acronym-definition pairs dictionary extracted by the Rule-based algorithm. This classifier builds decision trees from a set of training data, using the concept of information entropy.

In our experiments, we measure and analyze the effectiveness of the proposed technique based on precision (P) and recall (R) of the resulting dictionary.

$$P = \frac{\#\ of\ correct\ objects\ found}{total\ \#\ of\ objects\ found}$$

$$R = \frac{\#\ of\ correct\ objects\ found}{total\ \#\ of\ objects\ in\ document}$$

## 4.2 Experiments Analysis

Table 2 presents precision and recall of acronym dictionaries constructed by regular expressions 1, 2 described in subsection 3.1.

The results presented in table 2 show that the proposed extension of the baseline regular expression increases the recall of the dictionaries, but dramatically reduces their precision. In all subsequent experiments, we use the standard implementation of each module in JACRO [8] as a baseline.

The results obtained by the Definition Extraction module are presented in table 3 in comparison with the baseline. It is important to mention that the metrics show the quality of the acronym-definition dictionaries instead of measuring the

**Table 3: Performance of the Definitions Construction phase**

| | medstract | | military | | disa | |
|---|---|---|---|---|---|---|
| Algorithm | P | R | P | R | P | R |
| JACRO | 0.87 | 0.91 | 0.71 | 0.85 | 0.64 | 0.87 |
| Rule-based | 0.21 | 0.91 | 0.30 | 0.87 | 0.23 | 0.94 |

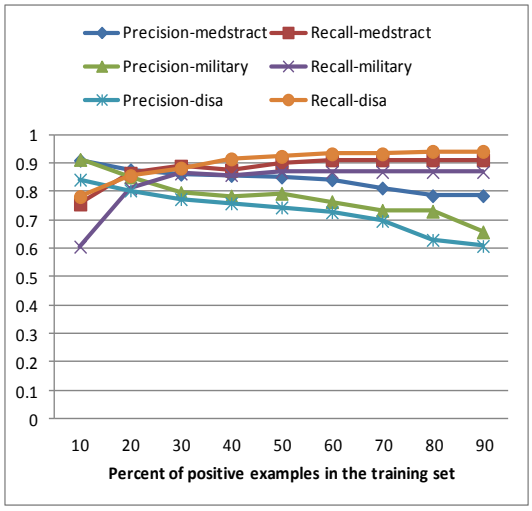
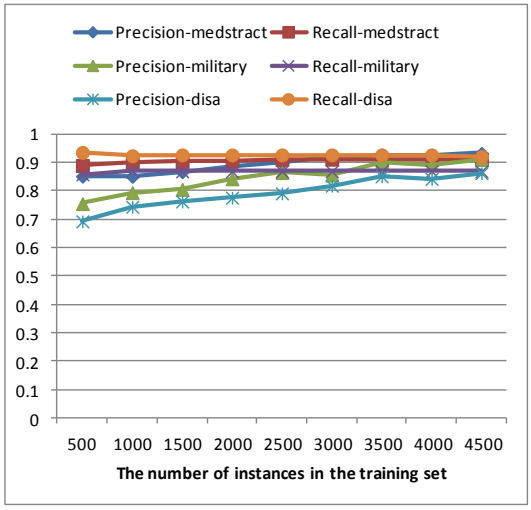



**Figure 2: Combination of positive and negative examples in the global training set**

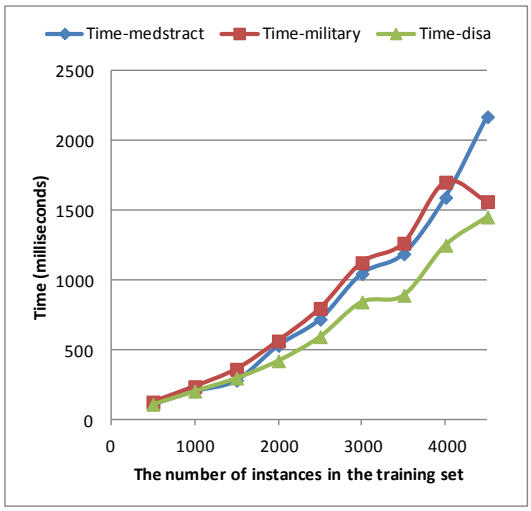
effectiveness of the Definition Extraction module independently of the previous step. One may see that the proposed combination of the extended regular expression for acronyms recognition together with the developed Rule-based implementation of the Definition Extraction module in most cases outperforms the baseline technique in recall but significantly yields in precision of the resulting dictionary.

We have constructed several Global Classification models for the Candidates Refinement step and have analyzed their effectiveness depending on the balance of the training set. In our experiments we apply acronym-definition dictionary refinement techniques to the results achieved by the Rule-based algorithm.

Figure 2 shows how the performance of the Candidates Refinement step depends on the rate of positive and negative examples in the global training set. This parameter enables us to balance the recall and precision of the classification algorithm and therefore to optimize the result obtained by the whole pipeline. According to our problem statement recall is more important than precision. Therefore, we construct a training set with a larger portion of positive examples compared to the original proportion of positives and negatives in a given data set. For this experiment we used global training sets of 1000 instances. According to the obtained results the Candidates Refinement step almost does not decrease the recall if the training set consists of equal portions of positive and negative examples.




(a)




(b)

**Figure 3: Variation of the global training set size**

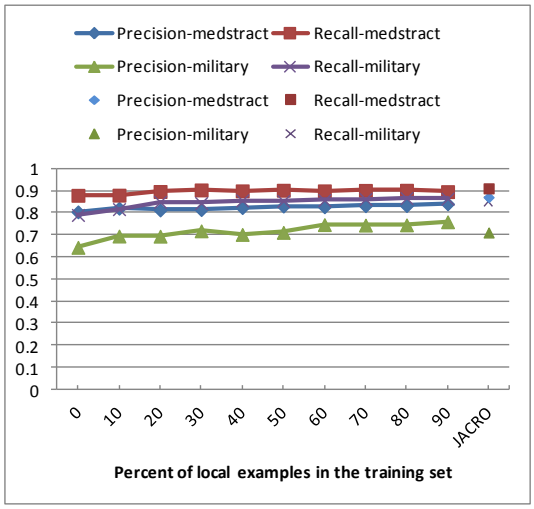



**Figure 4: Variation of the local training set size**

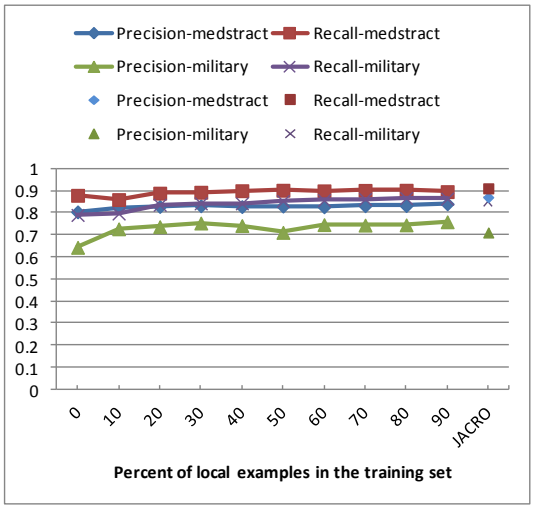



**Figure 5: Variation of the local training set size with redundant local examples**

Figure 3(a) shows how variation of the global training set size affects the effectiveness of the Candidates Refinement module. For this experiment we applied global training sets with equal portions of positive and negative examples and various numbers of instances. One may see that a large and even redundant training set improves the precision of the result.

However, increasing a training set size causes a dramatic increase of time needed to retrain a classification model (see, figure 3(b)). This figure shows absolute time required to train a classification model based on a training set of a fixed size.

The improvement, which can be obtained by increasing the portion of local examples in the training set for the Local Classification model, is presented in figure 4. For this experiment we used training sets of 1000 instances and varied the portion of local examples. We selected *disa* data set as a global one and *medstract* and *military* as local ones. The first point on each line shows zero-iteration, when no feedback is received from the user. At this point the Global Model (learned based on *disa* training set) is used to classify the entries of the acronym-definition dictionaries obtained at the Definition Extraction step on *medstract* and *military* data sets. The Global and Local Models are learned based on balanced training sets. We force the proportion of positive and negative examples in both training sets to be 50/50. The amount of manual work required from the user is not that high. For example, the user marks up about 30 acronym definition-pairs to receive 100 local examples. The rightmost points in figure 4 marked as JACRO show the precision and recall values for the baseline on the corresponding data sets.

Figure 4 shows that both precision and recall of the resulting acronym-definition dictionary slightly go up with increase of the portion of local examples in the training set.

Figure 5 presents the results of the experiment on the artificial duplication of local examples in the training set during the first iterations with the user. This experiment is similar to the previous one. During the iterations, we construct a training set by replacing 500 global examples with 500 duplicate local examples, in case the number of received local examples is less than 500. When the number of local examples exceeds 500, we use all of them and replace the corresponding number of global examples in the training set. Comparison of figures 4 and 5 shows that this duplication trick enables us to decrease the time needed for the Candidate Refinement step to adapt to the specific document.

The summary of the results achieved with the previously described experimental settings is presented in table 4. The first row shows the precision and recall of the baseline (JACRO) approach. The second row demonstrates the precision and recall of our pipeline at zero-iteration when no feedback from the user is available. The next row contains the precision and recall of our pipeline when the proportion of local and global examples is 50/50. The last row reflects the last iteration with 90% of the local examples in the training set. Table 5 summaries the results of a similar experiment when *military* data set is used as a global one.

One may see, that the proposed acronym-definition extraction pipeline provides better results on *military* and *disa* data sets and lose precision and recall on *medstract* data set, as compared to JACRO approach. The baseline was originally designed for texts in medical domain. Therefore, it shows better results on *medstract* data set, but provides poorer quality of results for *military* and *disa* data sets. And what concerns our approach, it is domain-independent and more adaptive to a particular data set.

## 5. CONCLUSION

In this paper, we have proposed and analyzed a tunable and adaptive pipeline for extracting acronyms with their definitions. Experimental analysis have shown that the proposed approach extracts acronyms from text with high recall and is able to use relevance feedback from the user in order

**Table 4: Performance of the Candidates Refinement phase (*disa* global training set)**

| | medstract | | military | |
|---|---|---|---|---|
| Algorithm | P | R | P | R |
| JACRO | 0.87 | 0.91 | 0.71 | 0.85 |
| Global - 100%, Local - 0% | 0.80 | 0.88 | 0.64 | 0.79 |
| Global - 50%, Local - 50% | 0.83 | 0.90 | 0.71 | 0.85 |
| Global - 10%, Local - 90% | 0.84 | 0.90 | 0.76 | 0.86 |

**Table 5: Performance of the Candidates Refinement phase (*military* global training set)**

| | medstract | | disa | |
|---|---|---|---|---|
| Algorithm | P | R | P | R |
| JACRO | 0.87 | 0.91 | 0.64 | 0.87 |
| Global - 100%, Local - 0% | 0.83 | 0.68 | 0.67 | 0.74 |
| Global - 50%, Local - 50% | 0.84 | 0.90 | 0.71 | 0.92 |
| Global - 10%, Local - 90% | 0.84 | 0.90 | 0.76 | 0.91 |

to improve precision. However, the behavior of the proposed pipeline highly depends on its parameters and might be tuned for a specific task. Provided experiments have shown that the proposed Candidates Refinement technique is adaptive. However, further research in this direction is needed to analyze the approach and its applicability in case when training and application data sets differ significantly.

# 6. REFERENCES


[1] *Algorithms and Theory of Computation Handbook*. CRC Press LLC, 1999. , "Levenshtein distance", in Dictionary of Algorithms and Data Structures [online], Paul E. Black, ed., U.S. National Institute of Standards and Technology. 14 August 2008. (accessed TODAY) Available from: http://www.nist.gov/dads/HTML/Levenshtein.html.

[2] D. Dannélls. Automatic acronym recognition. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations*, EACL '06, pages 167–170, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[3] S. Feng, Y. Xiong, C. Yao, L. Zheng, and W. Liu. Acronym extraction and disambiguation in large-scale organizational web pages. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1693–1696, New York, NY, USA, 2009. ACM.

[4] X. Ji, G. Xu, J. Bailey, and H. Li. Mining, ranking, and using acronym patterns. In *Proceedings of the 10th Asia-Pacific web conference on Progress in WWW research and development*, APWeb'08, pages 371–382, Berlin, Heidelberg, 2008. Springer-Verlag.

[5] A. Kempe. Acronym-meaning extraction from corpora using multi-tape weighted finite-state machines. *CoRR*, abs/cs/0612033, 2006.

[6] N. Okazaki and S. Ananiadou. A term recognition approach to acronym recognition. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 643–650, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[7] D. Sánchez and D. Isern. Seeking acronym definitions: a web-based approach. In *Proceeding of the 2009 conference on Artificial Intelligence Research and Development: Proceedings of the 12th International Conference of the Catalan Association for Artificial Intelligence*, pages 339–348, Amsterdam, The Netherlands, The Netherlands, 2009. IOS Press.

[8] A. S. Schwartz and M. A. Hearst. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Pacific Symposium on Biocomputing*, pages 451–462, 2003.

[9] K. Taghva and J. Gilbreth. Recognizing acronyms and their definitions. *ISRI (Information Science Research Institute) UNLV*, 1:191–198, 1999.

[10] J. Xu and Y. Huang. Using svm to extract acronyms from text. *Soft Comput.*, 11:369–373, November 2006.

[11] S. Yeates. Automatic extraction of acronyms from text. In *New Zealand Computer Science Research Students' Conference*, pages 117–124, 1999.

[12] S. Yeates, D. Bainbridge, and I. H. Witten. Using compression to identify acronyms in text. In *Proceedings of the Conference on Data Compression*, DCC '00, pages 582–, Washington, DC, USA, 2000. IEEE Computer Society.