A Supervised Learning Approach to Acronym Identification

David Nadeau and Peter D. Turney

Institute for Information Technology
National Research Council Canada
Ottawa, Ontario, Canada
{david.nadeau, peter.turney}@nrc-cnrc.gc.ca

Abstract. This paper addresses the task of finding acronym-definition pairs in text. Most of the previous work on the topic is about systems that involve manually generated rules or regular expressions. In this paper, we present a supervised learning approach to the acronym identification task. Our approach reduces the search space of the supervised learning system by putting some weak constraints on the kinds of acronym-definition pairs that can be identified. We obtain results comparable to hand-crafted systems that use stronger constraints. We describe our method for reducing the search space, the features used by our supervised learning system, and our experiments with various learning schemes.

1 Introduction

Acronym identification is the task of processing text to extract pairs consisting of a word (the acronym) and an expansion (the definition), where the word is the short form of (or stands for) the expansion. For instance, in the sentence, "The two nucleic acids, deoxyribonucleic acid (DNA) and ribonucleic acid (RNA), are the informational molecules of all living organisms," there are two acronyms, "DNA" and "RNA", along with their respective definitions, "deoxyribonucleic acid" and "ribonucleic acid". In this work, we do not discriminate between acronyms (short forms of multiword expressions) and abbreviations (contractions of single words). We use the term *acronym* to include both cases.

The acronym identification task can be extended in many ways. It is possible to try to resolve acronyms even when there are no explicit definitions in the text. For instance, the familiar acronym "HIV" will often appear without being defined. Another extension to the task is to try to disambiguate polysemous acronyms (e.g., "CMU" means "Carnegie Mellon University" but also "Central Michigan University"). The task requires identifying the intended sense of the acronym even when its definition is absent. Ambiguous acronyms are particularly problematic for information retrieval.

In this paper, we tackle the core task only. That is, given an input text, our algorithm will attempt to extract all explicit acronym-definition pairs. Our goal is to

create a dictionary of acronym-definition pairs specific to a single text. An algorithm that addresses the core task can be used, for example, to enhance a list of author keyphrases by resolving acronyms. More importantly, such an algorithm is a key component in systems that handle the various extended tasks, such as co-reference resolution for named-entity recognition or automatic query expansion for information retrieval. The literature on automatic acronym identification presents many attempts to solve the core task, and our contribution is to present a supervised learning approach with weak constraints on the forms of acronyms and definitions that can be identified. Our results are comparable to what is achieved (on the same testing data) by human-engineered rule systems with stronger constraints.

The next section presents a detailed summary of related work. Section 3 presents our supervised learning approach to acronym identification and Section 4 discusses the training and testing corpus we used. At least three other papers use the same corpus for evaluating their systems (Pustejovsky *et al*., 2001; Chang *et al.*, 2002; Schwartz and Hearst, 2003). The remaining sections discuss our experimental results and conclude the paper.

2 Related Work

In this section, we present previous work on the acronym identification task. We focus on the constraints that these systems use to extract valid acronym–definition pairs.

One of the earliest acronym identification systems (Taghva and Gilbreth, 1999) is AFP (Acronym Finding Program). The AFP system first identifies candidate acronyms, which the authors define as uppercase words of three to ten letters. It then tries to find a definition for each acronym by scanning a 2*n*-word window, where *n* is the number of letters in the acronym. The algorithm tries to match acronym letters against initial letters in the definition words. Some types of words receive special treatment: stopwords can be skipped, hyphenated words can provide letters from each of their constituent words and, finally, acronyms themselves can be part of a definition. Given these special cases, the longest common sequence (LCS) between acronym letters and initial letters in definitions is computed.

Yeates (1999) proposes the automatic extraction of acronyms-definitions pairs in a program called TLA (Three Letter Acronyms). Although the name suggests that acronyms must have three letters, the system can find *n*-letter acronyms as well. The algorithm divides text into chunks using commas, periods, and parentheses as delimiters. It then checks whether adjacent chunks have acronym letters matching one or more of the initial three letters of the definition words. Further heuristics are then applied to each candidate, ensuring that the acronym is uppercase, is shorter than the definition, contains the initial letters of most of the definition words, and has a certain ratio of words to stopwords.

Larkey *et al.* (2000) developed Acrophile. They compared various strategies and found their Canonical/Contextual method to be the most accurate. First they force

candidate acronyms to be in upper-case, allowing only embedded lower case letters (internal or final), periods (possibly followed by spaces), hyphens (or diagonal slashes) and digits (at most one, non-final digit). They allow a maximum of nine alphanumeric characters in acronyms. They search for expansions in a window of 20 words, adjacent to the given acronym. Stopwords can contribute to an inner letter, but only once for the entire acronym. Furthermore, an expansion is only valid if it fits a given pattern, such as being surrounded by parentheses or preceded by a cue phrase (e.g.,"also known as").

Recently the fields of Genetics and Medicine have become especially interested in acronym resolution (Pustejovsky *et al*., 2001, Yu *et al.* 2002). Pustejovsky *et al*., present an approach with weak constraints, designed to capture the wide range of acronyms that are abundant in medical literature. For example, "PMA" stands for "phorbol ester 12-myristade-13-acetate" and "E2" stands for "estradiol-17 beta". Pustejovsky *et al.*'s acronym resolution technique searches for definitions of acronyms within noun phrases. Acronym-definition candidate pairs must match a given set of regular expressions, designed to be very general, and the final decision about whether a pair is valid relies on counting the number of acronym characters and definition words that match.

Another strategy, also developed for the medical field, is from Schwartz and Hearst (2003).[1] Their approach is similar to Pustejovsky *et al*.'s (2001) strategy and the emphasis is again on complicated acronym-definition patterns for cases in which only a few letters match (e.g., "Gen-5 Related N-acetyltransferase" [GNAT]). They first identify candidate acronym-definition pairs by looking for patterns, particularly "*acronym (definition)*" and "*definition (acronym)*". They require the number of words in the definition to be at most $\min(|A|+5, |A|\times 2)$, where $|A|$ is the number of letters in the acronym.[2] They then count the number of overlapping letters in the acronym and its definition and compare the count to a given threshold. The first letter of the acronym must match with the first letter of a definition word. They also handle various cases where an acronym is entirely contained in a single definition word.

Byrd and Park (2001) combine mechanisms such as text-markers and linguistic cues with pattern-based recognition. The same combination was used by Larkey (2000). This removes some constraints on the acronyms that can be identified. The reason for these mechanisms is to cope with the growing popularity of acronyms that diverge from the tradition of using only the first letter of each word of the definition. They use cue expressions (e.g., "or", "short", "acronym", "stand") to reinforce the confidence in acronym-definition pairs. They also allow acronyms to include a digit at the beginning or the end; thus, "5GL (Fifth Generation Language)" would be a valid candidate.

Adar (2002) presents a technique that requires only four scoring rules for acronym-definition pair evaluation: (1) add one to the score if an acronym letter begins a

---

[1] The Java source code for their system is available at http://biotext.berkeley.edu/software.html.

[2] This formula is borrowed from Byrd and Park (2001).

definition word, (2) subtract one for each extra word that does not match acronym letters, (3) add one if the definition is next to a parenthesis and (4) the number of definition words should be less than or equal to the number of acronym letters; therefore, subtract one for each extra word.

Chang *et al.* (2002) present a supervised learning approach to acronym identification. In order to circumscribe the learning, they impose a strongly restrictive condition on candidate acronym-definition pairs, by searching only for "*definition (acronym)*" patterns. Interestingly, this pattern accounts for the majority of positive cases in their evaluation corpus. Chang *et al.*'s learning algorithm uses eight features describing the mapping between acronym letters and definition letters (e.g., percentage of letters aligned at the beginning of a word, number of definition words that are not aligned to the acronym, etc.). The learning algorithm they used is logistic regression.

Zahariev (2004) presents a complete review of the acronym identification literature in his thesis. He also extends the task to multi-lingual acronym identification and he offers an in-depth analysis of acronym phenomena. However, the proposed system uses the same strongly constraining patterns as Larkey *et al.* (2000).

Table 1 summarizes related work on acronym identification. In this table and in the forthcoming sections, "participation" means that a letter of the acronym is found in a word of the definition. Generally, either the constraints on the acronym are strong (e.g., "all acronym letters must be capitals" or "the number of letters must exceed some minimum") or the definition pattern is fixed (e.g., "the definition must be in parentheses"). Such strong constraints ensure reasonable precision but, in general (for heterogeneous text from unrestricted domains), they necessarily limit recall. In our work, we try to use only weak constraints on both the acronym and the definition.

Table 1. Summary of constraints on acronyms and definitions

Author (Year)	Strongest constraints on acronym candidate	Strongest constraints on definition candidate
Taghva and Gilbreth (1999)	▪ uppercase word of 3 to 10 characters	▪ must be adjacent ▪ only first letters of definition words can participate
Yeates (1999)	▪ uppercase word	▪ must be adjacent ▪ first three letters of definition words can participate
Larkey *et al.* (2000)	▪ need some uppercase letters ▪ maximal size of 9 characters	▪ pattern "acronym (definition)" or "definition (acronym)" ▪ cue (e.g., "also known as")
Pustejovsky *et al.* (2001)	▪ a word between parentheses or adjacent to parentheses	▪ pattern "acronym (definition)" or "definition (acronym)"
Schwartz and Hearst (2003)	▪ a word between parentheses or adjacent to parentheses	▪ pattern "acronym (definition)" or "definition (acronym)"

Byrd and Park (2001)	▪ at least 1 capital ▪ from 2 to 10 characters	▪ parentheses pattern or linguistic cue (also known as, short for, etc.)
Adar (2002)	▪ one word between parentheses	▪ adjacent on the left of parenthesis
Chang *et al.* (2002)	▪ one word between parentheses	▪ adjacent on the left of parenthesis
Zahariev (2004)	▪ a word between parentheses or adjacent to parentheses	▪ pattern "acronym (definition)" or "definition (acronym)"

3 Supervised Learning Approach

The acronym identification task can be framed in terms of supervised learning. The concept we want to learn is a pair $\langle A, D \rangle$ made of an acronym A (a single token) and a definition D (a sequence of one or more consecutives tokens). Given a sequence T of n tokens, $T = \langle t_1, ..., t_n \rangle$, from which we wish to extract a pair $\langle A, D \rangle$, there are n possible choices for $A = t_i$. Each possible acronym ($A = t_i$) can be defined (D) by any combination of one or more consecutive tokens taken from the left context $\{t_1, ..., t_{i-1}\}$ or from the right context $\{t_{i+1}, ..., t_n\}$. The number of possible pairs is $O(n^3)$ (n choices for $A = t_i$ multiplied by n choices for the first token in D multiplied by n choices for the last token in D). Therefore, before applying supervised learning, we reduce the space of possible $\langle A, D \rangle$ pairs with some heuristics.

Section 3.1 describes our heuristics for reducing the search space for candidate acronyms and Section 3.2 discusses the constraints for candidate definitions. Together, these sections explain how we reduce the space of $\langle A, D \rangle$ pairs that must be considered by the supervised learning algorithm. After the space has been reduced, the remaining candidate pairs must be represented as feature vectors, in order to apply standard supervised learning algorithms (Witten and Frank, 2000). Section 3.3 outlines our set of seventeen features.

The constraints that follow (Sections 3.1 and 3.2) are relatively weak, compared to most past work on acronym identification, but they still exclude some possible acronym-definition pairs from consideration by the supervised learning algorithm. The resulting decrease in recall is discussed in Section 5.

3.1 Space-reduction Heuristics for Candidate Acronyms

The acronym space (the set of choices for $A = t_i$) is reduced using syntactic constraints on the tokens, $T = \langle t_1, ..., t_n \rangle$, expressed by the conjunction of the following statements:

1. $A = t_i$, where $1 \leq i \leq n$.
2. $\text{Size}(t_i) \geq 2$, where $\text{Size}(t_i)$ is the number of characters in the token t_i (including numbers and internal punctuation).
3. $\text{NumLetter}(t_i) \geq 1$, where $\text{NumLetter}(t_i)$ is the number of alphabetic letters in the token t_i (excluding numbers and punctuation).
4. $(\text{Cap}(t_i) \wedge \text{UnknownPOS}(t_i)) \vee \text{Cue}(t_i)$, where $\text{Cap}(t_i)$ means that the token starts with a capital letter, $\text{UnknownPOS}(t_i)$ means that the part-of-speech of the token is neither conjunction, determiner, particle, preposition, pronoun nor verb, and $\text{Cue}(t_i)$ means that the token contains a digit, punctuation, or a capital letter.

The rationale behind $\text{Size}(t_i) \geq 2$ is that, in most cases, isolated letters such as "H" will not be acronyms (although "H" can stand for "Hydrogen"). Statement (4) says that the token t_i should have some capitalization or special characters, but in the former case, the token should not have a known part-of-speech. The calculation of $\text{UnknownPOS}(t_i)$ requires applying a part-of-speech tagger to the text. We used QTAG (Tufis and Mason, 1998) as our part-of-speech tagger.

The above heuristic constraints are less restrictive than previous approaches (compare with Table 1).

3.2 Space-reduction Heuristics for Candidate Definitions

Once a candidate acronym $A = t_i$ is found in the text, we search for its definition D on both sides of t_i. First, we require that both acronym and definition must appear in the same sentence. This considerably reduces the search space for $\langle A, D \rangle$ by reducing the size n of T, although the space is still $O(n^3)$. We then need stronger criteria to define a reasonable set of candidate definitions. We impose the following additional constraints:

1. The first word of a definition must use the first letter of the acronym (Pustejovsky *et al.*, 2001).
2. A definition can skip one letter of the acronym, unless the acronym is only two letters long.
3. The definition can skip any number of digits and punctuation characters inside the acronym.

4. The maximum length for a definition is $\min(acronymlen+5, acronymlen\times 2)$ (Byrd and Park, 2001). (Definition length is measured by number of words and acronym length is measured by number of characters.)
5. A definition cannot contain a bracket, colon, semi-colon, question mark, nor exclamation mark. (We found counter-examples for other punctuation. For instance, the acronym "MAM" expands to "meprin, A5, mu", where the comma is used.)

Typically, these constraints will dramatically reduce the number of candidate definitions (increasing precision) while including the vast majority of true positive cases (preserving recall).

To illustrate the remaining search space, consider the following sentence:

```
Microbial control of mosquitoes with special emphasis
on bacterial control (Citation).
```

The word "Citation" is not an acronym, but it fits our constraints, since it is a capitalized noun. Even with the above constraints, there are 92 candidate definitions in this example. Note that, according to the second rule above, the definition can skip one letter (except the leading 'C') of the acronym. Here is one of the candidate definitions (acronym letters are marked with square brackets):

```
[c]ontrol of mosqu[i]toes wi[t]h speci[a]l emphas[i]s
[o]n bacterial co[n]trol
```

3.3 Acronym-Definition Features for Supervised Learning

The above heuristics reduce the search space significantly, so that the number of ways to extract a pair $\langle A, D\rangle$ from a token sequence $T=\langle t_1,...,t_n\rangle$ is now much less than $O(n^3)$. The next step is to apply supervised learning, to select the best $\langle A, D\rangle$ pairs from the remaining candidates. Standard supervised learning algorithms assume input in the form of feature vectors. We defined seventeen features to describe a candidate acronym-definition instance. The hand-crafted rules that are described in previous work inspired the design of many of the following features. Our features mainly describe the mapping of acronym letters to definition letters and syntactic properties of the definition.

1. the number of participating letters matching the first letter of a definition word;
2. (1) normalized by the acronym length;
3. the number of participating definition letters that are capitalized;
4. (3) normalized by the acronym length;
5. the length (in words) of the definition;
6. the distance (in words) between the acronym and the definition;

7. the number of definition words that do not participate;
8. (7) normalized by the definition length;
9. the mean size of words in the definition that do not participate;
10. whether the first definition word is a preposition, a conjunction or a determiner (inspired by Byrd and Park, 2001);
11. whether the last definition word is a preposition, a conjunction or a determiner (inspired by Byrd and Park, 2001);
12. number of prepositions, conjunctions and determiners in the definition;
13. maximum number of letters that participate in a single definition word;
14. number of acronym letters that do not participate;
15. number of acronym digits and punctuations that do not participate;
16. whether the acronym or the definition is between parentheses;
17. the number of verbs in the definition.

If the heuristics in Sections 3.1 and 3.2 propose a candidate acronym-definition pair $\langle A_1, D_1 \rangle$ then there are three possibilities:

1. In the manual annotation of the corpus, there is an officially correct acronym-definition pair $\langle A_2, D_2 \rangle$ such that $A_1 = A_2$ and $D_1 = D_2$. In this case, $\langle A_1, D_1 \rangle$ is labeled as positive for both training and testing the algorithm.
2. In the manual annotation of the corpus, there is an officially correct acronym-definition pair $\langle A_2, D_2 \rangle$ such that $A_1 = A_2$ but $D_1 \neq D_2$. In this case, $\langle A_1, D_1 \rangle$ is ignored during training but it is labeled as negative during testing (see Section 6.3 for details).
3. In the manual annotation of the corpus, there is no officially correct acronym-definition pair $\langle A_2, D_2 \rangle$ such that $A_1 = A_2$. In this case, $\langle A_1, D_1 \rangle$ is labeled as negative for both training and testing.

4 Evaluation Corpus

We use the Medstract Gold Standard Evaluation Corpus (Pustejovsky *et al.*, 2001) to train and test our algorithm.[3] The corpus is made of Medline abstracts in which each acronym-definition pair is annotated. The training set is composed of 126 pairs and the testing set is composed of 168 pairs. The main interest of this corpus is that it was annotated by a biologist using an informal definition of a valid pair. Therefore the

[3] http://medstract.org/gold-standards.html

corpus reflects human interpretation of acronym-definition pairs and acronym identification is challenging for an automated process.

Past results with this corpus are reported in Table 2. All of the results are based on modified versions of the Medstract Gold Standard Evaluation Corpus, and (unfortunately) they all use different modifications. Here are some remarks on each of the modifications:

1. Chang *et al.* (2002) do not describe their modifications.
2. Pustejovsky *et al*. (2001) note that they removed eleven elements that they judged were not acronyms.
3. Schwartz and Hearst (2003) mention that they made modifications, but do not describe what modifications they made.
4. We attempted to replicate the results of Schwartz and Hearst (2003), while making only minimal modifications to the original corpus. Our modifications were aimed at creating a valid XML file and a consistent set of tags. We had to remove embedded acronyms and remove or correct obvious errors.

Since Schwartz and Hearst's (2001) system is available online[4], we were able to repeat their experiment on our modified version of the corpus. This is the version of the corpus that we use in the following experiments, in Section 5.

Table 2. Performance reported by teams using their own version of the Medstract corpus

Team	Precision	Recall	F1	Corpus Modification
Chang *et al.*, 2002	80%	83%	81.5%	See (1)
Pustejovsky *et al.*, 2001	98%	72%	83.0%	See (2)
Schwartz and Hearst, 2003	96%	82%	88.4%	See (3)
Schwartz and Hearst (our replication)	89%	88%	88.4%	See (4)

5 Experimental Results

We use the Weka Machine Learning Toolkit to test various supervised learning algorithms (Witten and Frank, 2000). The results are reported in Table 3. We found that the performance varies greatly depending on the chosen algorithm. A good classifier was PART rules (rules obtained from a partially pruned decision tree) with somewhat low recall but high precision. The Support Vector Machine (Weka's SMO) reaches F1 = 88.3%, a performance that rivals hand-craft systems. The Bayesian net also performs well. The OneR classifier (one rule) is shown as a baseline. Table 3

[4] http://biotext.berkeley.edu/software.html

includes our replication of Schwartz and Hearst (2003) for comparison. Note that all results in this table are based on the same corpus.

Table 3. Performance of various classifiers on the Medstract corpus

Learning Algorithm	Precision	Recall	F1
OneR[5]	69.0%	33.1%	44.7%
Bayesian Net	89.6%	81.7%	85.5%
PART rules	95.3%	79.6%	86.7%
SVM (SMO kernel degree = 2)	92.5%	84.4%	88.3%
Schwartz and Hearst (our replication)	88.7%	88.1%	88.4%

We claim that our system has weaker hand-coded constraints than competing approaches. In support of this claim, it is worth mentioning that 1,134 candidate acronym-definition pairs satisfied the constraints in Sections 3.1 and 3.2, but only 141 candidates (12%) were classified as positive by the supervised learning algorithms. Therefore the hand-coded part of our system allowed more candidates than, for example, Schwartz and Hearst's system allows. In comparison, their system considered 220 patterns that involve parentheses and 148 (67%) are accepted by the rule-based system. In our system, the reduction from 1,134 candidates to 141 candidates is done by the supervised learning component, rather than by hand-coded constraints. The advantage of this approach is that the supervised learning component can easily be retrained for a new corpus. The hand-coded constraints are designed to be weak enough that they should not require modification for a new corpus.

6 Discussion

In this section, we discuss the interpretation of our experimental results.

6.1 The Parenthesis Feature

In our examination of previous work (Section 2), we criticized many authors for making use of overly constraining patterns. One of the problems is the use of parentheses. Many authors only accept acronym-definition pairs when one of the expressions is between parentheses. To avoid this kind of limitation, we did not impose this constraint in our model. However, the only way we were able to perform as well as hand-built systems was to use the feature "*whether the acronym or the definition is between parentheses*" (feature 16 in Section 3.3). The learner uses this feature, since it works well on the Medstract corpus. Our relatively weak constraints (Sections 3.1 and 3.2) allow 889 candidate acronym-definition pairs for which the

[5] The rule for OneR says that the pair is valid if 70.8% of acronym letters match the first letter of a definition word.

parenthesis feature is false (neither the candidate acronym nor the candidate definition is between parentheses). In the Medstract corpus, all of these 889 candidates are negative instances (none are true acronym-definition pairs). Thus this feature dramatically increases precision with no loss of recall. It is a very informative feature, but we do not wish to hard-code it into our constraints, since we believe it may not generalize well to other corpora. With a new corpus, our system can learn to use the feature if it is helpful or ignore it if it does not apply. This robustness is an advantage of using weak constraints combined with supervised learning.

6.2 The Best Features

When evaluating the contribution of the individual features (using the Chi Square Test), we found that three features significantly outperform others. Those features are, in order of predictive power, (1) the distance between the definition and the acronym (feature 6), (2) the number of acronym letters that match the first letters of definition words (feature 1), and (3) the parentheses feature (feature 16).

6.3 Effects of the Space-reduction Heuristics

In Section 3, we presented heuristics for reducing the space of possible acronym-definition candidates. A particular case can be misleading for the supervised learning algorithm.

Consider a case in which our heuristics identify <PKA, protein kinase A> but the corpus annotation is <PKA, cAMP-dependent protein kinase A>. It is tempting to say that <PKA, protein kinase A> must count as a negative example for the supervised learner, but this could confuse the learner, since the match between PKA and protein kinase A is actually very credible and reasonable. Instead of counting <PKA, protein kinase A> as a negative example, we found that it is better to ignore this case during training. It would be incorrect to count this case as a positive example, but it would be misleading to count it as a negative example, so it is best to ignore it. During testing, however, such instances are added to the false negatives (thus reducing recall), because this is an error and the system must be penalized for it. (See Section 3.3.)

7 Conclusion

In this paper, we described a supervised learning approach to the acronym identification task. The approach consists in using weak hand-coded constraints to reduce the search space, and then using supervised learning to impose stronger constraints. The advantage of this approach is that the system can easily be retrained for a new corpus, when the previously learned constraints no longer apply. The hand-coded constraints reduce the set of candidate acronym-definition pairs that must be

classified by the supervised learning system, yet they are weak enough that they should be portable to a new corpus with little or no change.

In our experiments, we tested various learning algorithms and found that a Support Vector Machine is comparable in performance to rigorously designed hand-crafted systems presented in the literature. We reproduced experiments by Schwartz and Hearst (2003) and showed that our test framework was comparable to their work.

Our future work will consist in applying the supervised learning approach to different corpora, especially corpora in which acronyms or definitions are not always indicated by parentheses.

References

Adar, E. (2002) S-RAD A Simple and Robust Abbreviation Dictionary, *HP Laboratories Technical Report*, September.

Chang, J.T., Schütze, H. and Altman R.B., (2002), Creating an Online Dictionary of Abbreviations from MEDLINE, *Journal of American Medical Informatics Association (JAMIA)*, 9(6), p.612-620.

Larkey, L., Ogilvie, P., Price, A. and Tamilio, B. (2000) Acrophile: An Automated Acronym Extractor and Server, *In Proceedings of the ACM Digital Libraries conference*, pp. 205-214.

Park, Y., and Byrd, R.J., (2001), Hybrid Text Mining for Finding Abbreviations and Their Definitions, *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, Pittsburgh, PA.

Pustejovsky, J., Castao, J., Cochran, B., Kotecki, M., Morrell, M. and Rumshisky, A. (2001) "Extraction and Disambiguation of Acronym-Meaning Pairs in Medline", *unpublished manuscript.*

Schwartz, A. and Hearst, M. (2003), A simple algorithm for identifying abbreviation definitions in biomedical texts, *In Proceedings of the Pacific Symposium on Biocomputing* (PSB).

Taghva, K. and Gilbreth, J. (1999), Recognizing acronyms and their definitions, *International journal on Document Analysis and Recognition*, pages 191-198.

Tufis, D. and Mason, O. (1998). Tagging Romanian Texts: a Case Study for QTAG, a Language Independent Probabilistic Tagger, *Proceedings of the First International Conference on Language Resources and Evaluation (LREC)*, Spain, p.589-596.

Yeates, S. (1999), Automatic extraction of acronyms from text. *In Third New Zealand Computer Science Research Students' Conference*, pages 117-124.

Yu H, Hripcsak G, Friedman C. (2002) Mapping abbreviations to full forms in biomedical articles, *Journal of the American Medical Informatics Association (*9) 262-272.

Witten I, H, and Frank, E. (2000) *Data Mining: Practical machine learning tools with Java implementations*, Morgan Kaufmann, San Francisco.

Zahariev, M. (2004). *A (Acronyms)*, Ph.D. thesis, School of Computing Science, Simon Fraser University.