A Discriminative Alignment Model for Abbreviation Recognition

Naoaki Okazaki†
okazaki@is.s.u-tokyo.ac.jp

Sophia Ananiadou‡
sophia.ananiadou@manchester.ac.uk

Jun’ichi Tsujii†‡
tsujii@is.s.u-tokyo.ac.jp

†Graduate School of Information Science and Technology
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo 113-8656, Japan

‡School of Computer Science,
University of Manchester
National Centre for Text Mining (NaCTeM)
Manchester Interdisciplinary Biocentre
131 Princess Street, Manchester M1 7DN, UK

Abstract

This paper presents a discriminative alignment model for extracting abbreviations and their full forms appearing in actual text. The task of abbreviation recognition is formalized as a sequential alignment problem, which finds the optimal alignment (origins of abbreviation letters) between two strings (abbreviation and full form). We design a large amount of fine-grained features that directly express the events where letters produce or do not produce abbreviations. We obtain the optimal combination of features on an aligned abbreviation corpus by using the maximum entropy framework. The experimental results show the usefulness of the alignment model and corpus for improving abbreviation recognition.

1 Introduction

Abbreviations present two major challenges in natural language processing: term variation and ambiguity. Abbreviations substitute for expanded terms (e.g., dynamic programming) through the use of shortened term-forms (e.g., DP). At the same time, the abbreviation DP appearing alone in text is ambiguous, in that it may refer to different concepts, e.g., data processing, dirichlet process, differential probability. Associating abbreviations and their full forms is useful for various applications including named entity recognition, information retrieval, and question answering.

The task of abbreviation recognition, in which abbreviations and their expanded forms appearing in actual text are extracted, addresses the term variation problem caused by the increase in the number of abbreviations (Chang and Schütze, 2006). Furthermore, abbreviation recognition is also crucial for disambiguating abbreviations (Pakhomov, 2002; Gaudan et al., 2005; Yu et al., 2006), providing sense inventories (lists of abbreviation definitions), training corpora (context information of full forms), and local definitions of abbreviations. Hence, abbreviation recognition plays a key role in abbreviation management.

Numerous researchers have proposed a variety of heuristics for recognizing abbreviation definitions, e.g., the use of initials, capitalizations, syllable boundaries, stop words, lengths of abbreviations, and co-occurrence statistics (Park and Byrd, 2001; Wren and Garner, 2002; Liu and Friedman, 2003; Okazaki and Ananiadou, 2006; Zhou et al., 2006; Jain et al., 2007). Schwartz and Hearst (2003) implemented a simple algorithm that finds the shortest expression containing all alphanumeric letters of an abbreviation. Adar (2004) presented four scoring rules to choose the most likely expanded form in multiple candidates. Ao and Takagi (2005) designed more detailed conditions for accepting or discarding candidates of abbreviation definitions.

However, these studies have limitations in discovering an optimal combination of heuristic rules from manual observations of a corpus. For example, when expressions transcription factor 1 and thyroid transcription factor 1 are full-form candidates for the abbreviation TTF-1, an algorithm should choose the latter expression over the shorter...
expression (former). Previous studies hardly handle abbreviation definitions where full forms (e.g., water activity) shuffle their abbreviation letters (e.g., AW). It is also difficult to reject ‘negative’ definitions in a text; for example, an algorithm should not extract an abbreviation definition from the text, “the replicon encodes a large RepA protein (RepA),” since the text, “the replicon encodes a large RepA protein (RepA),” should not extract an abbreviation definition from the sentence as a sequence of letters (abbreviation recognition as a sequential alignment problem) since the number of abbreviation letters but only pairs of abbreviations and full forms. It was thus difficult to design effective features for abbreviation recognition and to reuse the knowledge obtained from the training processes.

In this paper, we formalize the task of abbreviation recognition as a sequential alignment problem, which finds the optimal alignment (origins of abbreviation letters) between two strings (abbreviation and full form). We design a large amount of features that directly express the events where letters produce or do not produce abbreviations. Preparing an aligned abbreviation corpus, we obtain the optimal combination of the features by using the maximum entropy framework (Berger et al., 1996). We report the remarkable improvements and conclude this paper.

2 Proposed method
2.1 Abbreviation alignment model
We express a sentence $x$ as a sequence of letters $(x_1, ..., x_L)$, and an abbreviation candidate $y$ in the sentence as a sequence of letters $(y_1, ..., y_M)$. We define a letter mapping $a = (i, j)$ to indicate that the abbreviation letter $y_j$ is produced by the letter in the full form $x_i$. A null mapping $a = (i, 0)$ indicates that the letter in the sentence $x_i$ is unused to form the abbreviation. Similarly, a null mapping $a = (0, j)$ indicates that the abbreviation letter $y_j$ does not originate from any letter in $x$. We define $a_{(x)}$ and $a_{(y)}$ in order to represent the first and second elements of the letter mapping $a$. In other words, $a_{(x)}$ and $a_{(y)}$ are equal to $i$ and $j$ respectively, when $a = (i, j)$. Finally, an abbreviation alignment $a$ is defined as a sequence of letter mappings, $a = (a_1, ..., a_T)$, where $T$ represents the number of mappings in the alignment.

Let us consider the following example sentence:

We investigate the effect of thyroid transcription factor 1 (TTF-1).

This sentence contains an abbreviation candidate TTF-1 in parentheses\(^2\). Figure 1 illustrates the correct alignment $a$ (bottom line) and its two-dimensional representation for the example sentence\(^3\); the abbreviation letters ‘t,’ ‘t,’ ‘f,’ ‘-,’ and ‘1’ originate from $x_{30}$, $x_{38}$, $x_{52}$, nowhere (null mapping), and $x_{70}$ respectively.

We directly model the conditional probability of the alignment $a$, given $x$ and $y$, using the maximum entropy framework (Berger et al., 1996),

$$P(a|x, y) = \frac{\exp \{ \Lambda \cdot F(a, x, y) \}}{\sum_{a \in C(x, y)} \exp \{ \Lambda \cdot F(a, x, y) \}}.$$  \hspace{1cm} (1)

In Formula 1, $F = \{f_1, ..., f_K\}$ is a global feature vector whose elements present $K$ feature functions, $\Lambda = \{\lambda_1, ..., \lambda_K\}$ denotes a weight vector for the feature functions, and $C(x, y)$ yields a set of possible alignments for the given $x$ and $y$. We obtain the following decision rule to choose the most probable alignment $\hat{a}$ for given $x$ and $y$,

$$\hat{a} = \arg\max_{a \in C(x, y)} P(a|x, y).$$  \hspace{1cm} (2)

Note that a set of possible alignments $C(x, y)$ always includes a negative alignment whose elements are filled with null-mappings (refer to Section 2.3 for further detail). This allows the formula to withdraw the abbreviation candidate $y$ when any expression in $x$ is unlikely to be a definition.

\(^2\)Refer to Section 3.1 for text makers for abbreviations.
\(^3\)We ignore non-alphabetical letters in abbreviations.
2.2 Features

The main advantage of the discriminative alignment model is its ability to incorporate a wide range of non-independent features. Inspired by feature engineering for Conditional Random Fields (CRFs) (Lafferty et al., 2001), we design two kinds of features: unigram (state) features defined on each letter mapping, and bigram (transition) features defined on each pair of adjacent letter mappings. Given a triplet, \( a, x, \) and \( y, \) a global feature function \( f_k(a, x, y) \in F \) sums up the boolean values (0 or 1) of the corresponding local feature \( g_k(a, x, y, t) \) at \( t \in \{1, \ldots, T\} \),

\[
f_k(a, x, y) = \sum_{t=1}^{T} g_k(a, x, y, t). \tag{3}
\]

In other words, \( f_k(a, x, y) \) counts the number of times the local feature is fired in the alignment \( a \).

A unigram feature corresponds to the observation at \( x_i \) and \( y_j \) associated by a mapping \( a_t = (i, j) \). A unigram feature encodes the condition where the letter in the full form \( x_i \) is chosen or unchosen for producing the abbreviation letter \( y_j \). For example, we may infer from the letter mapping at \( a_8 = (30, 1) \) in Figure 1, that \( x_{30} \) is mapped to \( y_1 \) because: \( x_{30} \) is at the head of the word, \( y_1 \) is a capital letter, and both \( x_{30} \) and \( y_1 \) are at the head of the word and abbreviation.

Bigram features, combining two observations at \( a_s \) and \( a_t \) (\( 1 \leq s < t \leq T \)), are useful in capturing the common characteristics shared by an abbreviation definition. For instance, we may presume in Figure 1 that the head letters in the full form might be selectively used for producing the abbreviation, based on the observations at \( a_8 = (30, 1) \) and \( a_9 = (38, 2) \). In order to focus on the conditions for consecutive non-null mappings, we choose the previous position \( s \) for the given \( t \).

\[
s = \begin{cases} 
    t-1 & (a_{t(y)} = 0 \lor \forall u : a_{u(y)} = 0) \\
    \max_{1 \leq u < t} \{ u \mid a_{u(y)} \neq 0 \} & \text{(otherwise)}
\end{cases}
\tag{4}
\]

Formula 4 prefers the non-null mapping that is the most adjacent to \( t \) over the previous mapping \((t-1)\). In Figure 1, transitions \( a_9 \rightarrow a_{11} \) and \( a_{11} \rightarrow a_{13} \) exist for this reason.

In this study, we express unigram and bigram features with atomic functions (Table 1) that encode observation events of \( x_{a_t(x)}, y_{a_t(y)}, a_t, x_{a_{u}(x)} = x_{a_{t}(x)}, \) and \( y_{a_{u}(y)} = y_{a_{t}(y)} \). Atomic functions \( x_{\text{ctype}}, y_{\text{ctype}}, x_{\text{position}}, \) and \( y_{\text{position}} \) present common heuristics used by previous studies. The function \( x_{\text{pos}} \) examines the existence of stop words (e.g., the, of, in) to prevent them from producing abbreviation letters. We also include \( x_{\text{pos}} \) (part-of-speech of the word) since a number of full forms are noun phrases.

Functions \( x_{\text{cliff}}, x_{\text{cliff wd}}, \) and \( y_{\text{cliff}} \) are designed specifically for bigram features, receiving two positions \( s \) and \( t \) in their arguments. The function \( x_{\text{cliff}} \) mainly deals with abbreviation definitions that include consecutive letters of their full forms, e.g., \textit{amplifier} (AMP). The function
Function | Return value
--- | ---
x\_ctype\_t(a, x, t) & $x_{at}(x) + \delta$ is \{U (uppercase), L (lowercase), D (digit), W (whitespace), S (symbol)\} letter
x\_position\_t(a, x, t) & $x_{at}(x)$ is at the \{H (head), T (tail), S (syllable head), I (inner), W (whitespace)\} of the word
x\_char\_t(a, x, t) & The lower-cased letter of $x_{at}(x)$
x\_word\_t(a, x, t) & The lower-cased word (offset position $\delta$) containing the letter $x_{at}(x)$
x\_pos\_t(a, x, t) & The part-of-speech code of the word (offset position $\delta$) containing the letter $x_{at}(x)$
y\_ctype\_t(a, y, t) & $y_{at}(y)$ is \{N (NIL) U (uppercase), L (lowercase), D (digit), S (symbol)\} letter
y\_position\_t(a, y, t) & $y_{at}(y)$ is at the \{N (NIL) H (head), T (tail), I (inner)\} of the word
y\_char\_t(a, y, t) & The lower-cased letter of $y_{at}(y)$

Table 1: Atomic functions to encode observation events in $x$ and $y$

| Combination | Rules |
| --- | --- |
| unigram(t) | $xy\_unigram(t) \oplus \{a\_state\(t)\}$ |
| xy\_unigram(t) | $x\_unigram(t) \oplus \{a\_state\(t)\}$ |
| xy\_unigram(t) | $x\_state\_t(t) \ominus x\_state\_t(t) \ominus x\_state\_t(t)$ |
| y\_unigram(t) | $y\_ctype\_t(t), y\_position\_t(t), y\_ctype\_t(t), y\_position\_t(t)$ |
| x\_state\_t(t) | $x\_ctype\_t(t), x\_position\_t(t), x\_char\_t(t), x\_word\_t(t), x\_pos\_t(t), x\_ctype\_t(t), x\_position\_t(t), x\_pos\_t(t), x\_ctype\_t(t), x\_ctype\_t(t), x\_pos\_t(t)$ |
| bigram(s, t) | $xy\_bigram(s, t) \ominus \{a\_state\(t)\}$ |
| xy\_bigram(s, t) | $x\_state\_t(t) \ominus x\_state\_t(t) \ominus x\_state\_t(t) \ominus x\_state\_t(t)$ |
| trans(s, t) | $x\_diff(s, t), x\_diff\_wd\_t(s, t), x\_diff\_t(s, t)$ |

Table 2: Generation rules for unigram and bigram features.

$x\_diff\_wd$ measures the distance of two words. The function $y\_diff$ models the ordering of abbreviation letters; this function always returns non-negative values if the abbreviation contains letters in the same order as in its full form.

We express unigram and bigram features with the atomic functions. For example, Formula 5 defines a unigram feature for the event where the capital letter in a full-form word $x_{at}(x)$ produces the identical abbreviation letter $y_{at}(y)$.

$$
g_k(a, x, y, t) = \begin{cases} 
1 & x\_ctype\_t(a, x, t) = U \\
& \land y\_ctype\_t(a, y, t) = U \\
& \land a\_state\(a, y, t) = MATCH \\
0 & \text{(otherwise)}
\end{cases} \tag{5}
$$

For notation simplicity, we rewrite this boolean function as (arguments $a, x, y$ are omitted),

$$
1\{x\_ctype\_t(t), y\_ctype\_t(t), a\_state\(t) = U, U, MATCH\} \tag{6}
$$

In this formula, $1\{v = \tilde{v}\}$ is an indicator function that equals 1 when $v = \tilde{v}$ and 0 otherwise. The term $v$ presents a generation rule for a feature, i.e., a combination rule of atomic functions.

Table 2 displays the complete list of generation rules for unigram and bigram features$^4$, unigram(t) and bigram(s, t). For each generation rule in unigram(t) and bigram(s, t), we define boolean functions that test the possible values yielded by the corresponding atomic function(s).

### 2.3 Alignment candidates

Formula 1 requires a sum over the possible alignments, which amounts to $2^{LM}$ for a sentence (L letters) with an abbreviation (M letters). It is unrealistic to compute the partition factor of the formula directly; therefore, the factor has been computed by dynamic programing (McCallum et al., 2005; Blunsom and Cohn, 2006; Shimbo and Hara, 2007) or approximated by the $n$-best list of highly probable alignments (Och and Ney, 2002; Liu et al., 2005). Fortunately, we can prune alignment rules that are unlikely to present full forms, by introducing the natural assumptions for abbreviation definitions:

$^4$In Table 2, a set of curly brackets $\{\}$ denotes a list (array) rather than a mathematical set. Operators $\oplus$ and $\ominus$ present concatenation and Cartesian product of lists. For instance, when $A = \{a, b\}$ and $B = \{c, d\}$, $A \oplus B = \{a, b, c, d\}$ and $A \ominus B = \{ac, ad, bc, bd\}$.
Figure 2: A part of the possible alignments for the abbreviation TTF-1 in the example sentence.

1. A full form may appear \( \min(m + 5, 2m) \) words before its abbreviation in the same sentence, where \( m \) is the number of alphanumeric letters in the abbreviation (Park and Byrd, 2001).

2. Every alphanumeric letter in an abbreviation must be associated with the identical (case-insensitive) letter in its full form.

3. An abbreviation letter must not originate from multiple letters in its full form; a full-form letter must not produce multiple letters.

4. Words in a full form may be shuffled at most \( d \) times, so that all alphanumeric letters in the corresponding abbreviation appear in the rearranged full form in the same order. We define a shuffle operation as removing a series of word(s) from a full form, and inserting the removed word(s) to another position.

5. A full form does not necessarily exist in the text span defined by assumption 1.

Due to the space limitation, we do not describe the algorithm for obtaining possible alignments that are compatible with these assumptions. Alternatively, Figure 2 illustrates a part of possible alignments \( C(\mathbf{x}, \mathbf{y}) \) for the example sentence. The alignment #2 represents the correct definition for the abbreviation TTF-1. We always include the negative alignment (e.g., #0) where no abbreviation letters are associated with any letters in \( \mathbf{x} \).

The alignments #4–#8 interpret the generation process of the abbreviation by shuffling the words in \( \mathbf{x} \). For example, the alignment #6 moves the word ‘of’ to the position between ‘factor’ and ‘1’. Shuffled alignments cover abbreviation definitions such as receptor of estrogen (ER) and water activity (AW). We call the parameter \( d \), distortion parameter, which controls the acceptable level of reordering (distortion) for the abbreviation letters.

2.4 Parameter estimation

Parameter estimation for the abbreviation alignment model is essentially the same as for general maximum entropy models. Given a training set that consists of \( N \) instances, \( \{(\mathbf{a}^{(1)}, \mathbf{x}^{(1)}, \mathbf{y}^{(1)}), ..., (\mathbf{a}^{(N)}, \mathbf{x}^{(N)}, \mathbf{y}^{(N)})\} \), we maximize the log-likelihood of the conditional probability distribution by using the maximum a posterior (MAP) estimation. In order to avoid overfitting, we regularize the log-likelihood with either the \( L_1 \) or \( L_2 \) norm of the weight vector \( \mathbf{\Lambda} \),

\[
L_1 = \sum_{n=1}^{N} \log P(\mathbf{a}^{(n)}|\mathbf{x}^{(n)}, \mathbf{y}^{(n)}) - \frac{||\mathbf{\Lambda}||_1}{\sigma_1},
\]

\[
L_2 = \sum_{n=1}^{N} \log P(\mathbf{a}^{(n)}|\mathbf{x}^{(n)}, \mathbf{y}^{(n)}) - \frac{||\mathbf{\Lambda}||_2^2}{2\sigma_2^2}.
\]

In these formulas, \( \sigma_1 \) and \( \sigma_2 \) are regularization parameters for the \( L_1 \) and \( L_2 \) norms. Formulas 7 and 8 are maximized by the Orthant-Wise Limited-memory Quasi-Newton (OW-LQN) method (Andrew and Gao, 2007) and the Limited-memory BFGS (L-BFGS) method (Nocedal, 1980)\(^5\).

3 Experiments

3.1 Aligned abbreviation corpus

The Medstract Gold Standard Corpus (Pustejovsky et al., 2002) was widely used for evaluating abbreviation recognition methods (Schwartz and Hearst, 2003; Adar, 2004). However, we cannot use this corpus for training the abbreviation alignment model, since it lacks annotations on the origins of abbreviation letters. In addition, the size of the corpus is insufficient for a supervised machine-learning method.

Therefore, we built our training corpus with 1,000 scientific abstracts that were randomly chosen from the MEDLINE database. Although the alignment model is independent of linguistic patterns for abbreviation definitions, in the corpus we found only three abbreviation definitions that were described without parentheses. Hence, we employed parenthetical expressions, full-form ('abbreviation'), to locate possible abbreviation definitions (Wren and Garner, 2002). In order to exclude parentheses inserting clauses into passages,\(^5\)

\(^5\)We used Classias for parameter estimation: http://www.chokkan.org/software/classias/
we consider the inner expression of parentheses as an abbreviation candidate, only if the expression consists of two words at most, the length of the expression is between two to ten characters, the expression contains at least an alphabetic letter, and the first character is alphanumeric.

We asked a human annotator to assign reference abbreviation alignments for 1,420 parenthetical expressions (instances) in the corpus. If a parenthetical expression did not introduce an abbreviation, e.g., “... received treatment at 24 months (RRMS),” the corresponding instance would have a negative alignment (as #0 in Figure 2). Eventually, our aligned corpus consisted of 864 (60.8%) abbreviation definitions (with positive alignments) and 556 (39.2%) other usages of parentheses (with negative alignments). Note that the log-likelihood in Formula 7 or 8 increases only if the probabilistic model predicts the reference alignments, regardless of whether they are positive or negative.

3.2 Baseline systems

We prepared five state-of-the-art systems of abbreviation recognition as baselines: Schwartz and Hearst’s method (SH) (Schwartz and Hearst, 2003), SaRAD (Adar, 2004), ALICE (Ao and Takagi, 2005), Chang and Schütze’s method (CS) (Chang and Schütze, 2006), and Nadeau and Turney’s method (NT) (Nadeau and Turney, 2005). We utilized the implementations available on the Web for SH, CS, and ALICE, and we reproduced SaRAD and NT, based on their papers.

Our implementation of NT consists of a classifier that discriminates between positive (true) and negative (false) full forms, using all of the feature functions presented in the original paper. Although the original paper presented heuristics for generating full-form candidates, we replaced the candidate generator with the function \( C(x, y) \), so that the classifier and our alignment model can receive the same set of full-form candidates. The classifier of the NT system was modeled by the LIBSVM implementation with Radial Basis Func-

---

Table 3: Performance on our corpus.

| System                  | P     | R     | F1    |
|-------------------------|-------|-------|-------|
| Schwartz & Hearst (SH)  | .978  | .940  | .959  |
| SaRAD                   | .891  | .919  | .905  |
| ALICE                   | .961  | .920  | .940  |
| Chang & Schütze (CS)    | .942  | .900  | .921  |
| Nadeau & Turney (NT)    | .954  | .871  | .910  |
| Proposed \((d = 0; L_1)\) | .973  | .969  | .971  |
| Proposed \((d = 0; L_2)\) | .964  | .968  | .966  |
| Proposed \((d = 1; L_1)\) | .960  | .981  | .971  |
| Proposed \((d = 1; L_2)\) | .957  | .976  | .967  |

---

6 Abbreviation Definition Recognition Software: http://biotext.berkeley.edu/software.html
7 Biomedical Abbreviation Server: http://abbreviation.stanford.edu/
8 We applied a score cutoff of 0.14.
9 Abbreviation Lifter using Corpus-based Extraction: http://uudb3.hgc.jp/ALICE/ALICE_jindex.html
10 LIBSVM – A Library for Support Vector Machines: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

---

11 We tuned kernel parameters \( C = 128 \) and \( \gamma = 2.0 \) by using the grid-search tool in the LIBSVM distribution.

12 We determined the regularization parameters as \( \sigma_1 = 3 \) and \( \sigma_2 = 3 \) after testing \{0.1, 0.2, 0.3, 0.5, 1, 2, 3, 5, 10\} for the regularization parameters. The difference between the highest and lowest F1 scores was 1.8%.

13 On Intel Dual-Core Xeon 5160/3GHz CPU, excluding time for feature generation and data input/output.
Table 4: Performance on Medstract corpus.

| System                      | P    | R    | F1  |
|-----------------------------|------|------|-----|
| Schwartz & Hearst (SH)      | .942 | .891 | .916 |
| SaRAD                       | .909 | .859 | .884 |
| ALICE                       | .960 | .945 | .953 |
| Chang & Schütze (CS)        | .858 | .852 | .855 |
| Nadeau & Turney (NT)        | .889 | .875 | .882 |
| Proposed (d = 1; L₁)        | .976 | .945 | .960 |

Table 5: Effect of atomic functions (d = 0; L₂).

| # Feature                  | λ    |
|---------------------------|------|
| 1: x.position = H₁, ctype = U₁, position = H₁/M | 1.7370 |
| 2: y.position = dY, position = dY, diff = 1/M/M | 1.3470 |
| 3: x.ctype = 1 = x.ctype = 1/S | 0.96342 |
| 4: x.ctype = L₁, ctype = L₁, diff = 0/M/M | 0.94009 |
| 5: x.position = x.char = L/S | 0.91645 |
| 6: x.position = H₁, pos = H₁, ctype = H₁/M | 0.86786 |
| 7: x.ctype = 1 = x.ctype = L/M | 0.86474 |
| 8: x.char = 0 = x.ctype = L₁, diff = 0/M-S | 0.71262 |
| 9: x.char = 0 = x.ctype = L/M | 0.69764 |
| 10: x.position = H₁, ctype = U₁, diff = 1/M-M | 0.66418 |

Table 6: Top ten features with high weights.

weights assigned by the MAP estimation with L₁ regularization. A unigram and bigram features have prefixes “U:” and “B:” respectively; a feature expresses conditions at s (bigram features only), conditions at t, and mapping status (match or skip) separated by ‘|’ symbols. For example, the #1 feature associates a letter at the head of a full-form word with the uppercase letter at the head of its abbreviation. The #4 feature is difficult to obtain from manual observations, i.e., the bigram feature suggests the production of two abbreviation letters from two lowercase letters in the same word.

4 Conclusion

We have presented a novel approach for recognizing abbreviation definitions. The task of abbreviation recognition was successfully formalized as a sequential alignment problem. We developed an aligned abbreviation corpus, and obtained fine-grained features that express the events wherein a full forum produces an abbreviation letter. The experimental results showed remarkable improvements and usefulness of the alignment approach for abbreviation recognition. We expect the usefulness of the discriminative model for building a comprehensible abbreviation dictionary.

Future work would be to model cases in which a full form yields non-identical letters (e.g., ‘one’ → ‘1’ and ‘deficient’ → ‘-‘), and to demonstrate this approach with more generic linguistic patterns (e.g., ‘aka’, ‘abbreviated as’, etc.). We also plan to explore a method for training a model with an unaligned abbreviation corpus, estimating the alignments simultaneously from the corpus.

Acknowledgments

This work was partially supported by Grant-in-Aid for Scientific Research on Priority Areas (MEXT, Japan), and Solution-Oriented Research for Science and Technology (JST, Japan).
References

Adar, Eytan. 2004. SaRAD: A simple and robust abbreviation dictionary. *Bioinformatics*, 20(4):527–533.

Andrew, Galen and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pages 33–40.

Ao, Hiroko and Toshihisa Takagi. 2005. ALICE: An algorithm to extract abbreviations from MEDLINE. *Journal of the American Medical Informatics Association*, 12(5):576–586.

Berger, Adam L., Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Blunsom, Phil and Trevor Cohn. 2006. Discriminative word alignment with conditional random fields. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (Coling-ACL 2006)*, pages 65–72.

Chang, Jeffrey T. and Hinrich Schütze. 2006. Abbreviations in biomedical text. In Ananiadou, Sophia and John McNaiht, editors, *Text Mining for Biology and Biomedicine*, pages 99–119. Artech House, Inc.

Gaudan, Sylvain, Harald Kirsch, and Dietrich Rebholz-Schuhmann. 2005. Resolving abbreviations to their senses in Medline. *Bioinformatics*, 21(18):3658–3664.

Jain, Alpa, Silviu Cucerzan, and Saliha Azzam. 2007. Acronym-expansion recognition and ranking on the web. In *Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI 2007)*, pages 209–214.

Lafferty, John, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, pages 282–289.

Liu, Hongfang and Carol Friedman. 2003. Mining terminological knowledge in large biomedical corpora. In *the 8th Pacific Symposium on Biocomputing (PSB 2003)*, pages 415–426.

Liu, Yang, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL 2005)*, pages 459–466.

McCallum, Andrew, Kedar Bellare, and Fernando Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005)*, pages 388–395.

Nadeau, David and Peter D. Turney. 2005. A supervised learning approach to acronym identification. In *the 8th Canadian Conference on Artificial Intelligence (AI'2005)* (LNAI 3501), page 10 pages.

Nocedal, Jorge. 1980. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782.

Och, Franz Josef and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL 2002)*, pages 295–302.

Okazaki, Naoaki and Sophia Ananiadou. 2006. Building an abbreviation dictionary using a term recognition approach. *Bioinformatics*, 22(24):3089–3095.

Pakhomov, Serguei. 2002. Semi-supervised maximum entropy based approach to acronym and abbreviation normalization in medical texts. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL 2002)*, pages 160–167.

Park, Youngja and Roy J. Byrd. 2001. Hybrid text mining for finding abbreviations and their definitions. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, pages 126–133.

Pustejovsky, James, José Castaño, Roser Saurí, Anna Rumphinsky, Jason Zhang, and Wei Luo. 2002. Medstract: creating large-scale information servers for biomedical libraries. In *Proceedings of the ACL-02 workshop on Natural language processing in the biomedical domain*, pages 85–92.

Schwartz, Ariel S. and Marti A. Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. In *the 8th Pacific Symposium on Biocomputing (PSB 2003)*, pages 451–462.

Shimbo, Masashi and Kazuo Hara. 2007. A discriminative learning model for coordinate conjunctions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 610–619.

Wren, Jonathan D. and Harold R. Garner. 2002. Heuristics for identification of acronym-definition patterns within text: towards an automated construction of comprehensive acronym-definition dictionaries. *Methods of Information in Medicine*, 41(5):426–434.

Yu, Hong, Won Kim, Vasileios Hatzivassiloglou, and John Wilbur. 2006. A large scale, corpus-based approach for automatically disambiguating biomedical abbreviations. *ACM Transactions on Information Systems (TOIS)*, 24(3):380–404.

Zhou, Wei, Vetle I. Torvik, and Neil R. Smallheiser. 2006. ADAM: another database of abbreviations in MEDLINE. *Bioinformatics*, 22(22):2813–2818.