Linguistic Knowledge Generator

Satoshi SEKINE *
Tokyo Information and Communications Research Laboratory
Matsushita Electric Industrial Co.,Ltd.

Sofia ANANIADOU **Jeremy J.CARROLL** **Jun'ichi TSUJII**
Centre for Computational Linguistics
University of Manchester Institute of Science and Technology
PO Box 88, Manchester M60 1QD, United Kingdom

1 Introduction

The difficulties in current NLP applications are seldom due to the lack of appropriate frameworks for encoding our linguistic or extra-linguistic knowledge, but rather to the fact that we do not know in advance what actual *instances* of knowledge should be, even though we know in advance what *types* of knowledge are required.

It normally takes a long time and requires painful trial and error processes to adapt knowledge, for example, in existing MT systems in order to translate documents of a new text-type and of a new subject domain. Semantic classification schemes for words, for example, usually reflect ontologies of subject domains so that we cannot expect a single classification scheme to be effective across different domains. To treat different *sublanguages* requires different word classification schemes. We have to construct appropriate schemes for given sublanguages from scratch [1].

It has also been reported that not only knowledge concerned with extra-linguistic domains but also syntactic knowledge, such as subcategorization frames of verbs (which is usually conceived as a part of general language knowledge), often varies from one sublanguage to another [2].

Though re-usability of linguistic knowledge is currently and intensively prescribed [3], our contention is that the *adaptation* of existing knowledge requires processes beyond mere *re-use*. That is,

1. There are some types of knowledge which we have to discover from scratch, and which should be integrated with already existing knowledge.
2. It is often the case that knowledge, which is normally conceived as valid regardless of subject domains, text types etc., should be revised significantly.

In practical projects, the ways of achieving such adaptation and discovery of knowledge rely heavily on human introspection. In the adaptation of existing MT systems, linguists add and revise the knowledge by inspecting a large set of system translation results, and then try to translate another set of sentences from given domains, and so on. The very fact that this trial and error process is time consuming and not always satisfactory indicates that human introspection alone cannot effectively reveal regularities or closure properties of sublanguages.

There have been some proposals to aid this procedure by using programs in combination with huge corpora [4] [5] [6] [7]. But the acquisition programs in these reports require huge amounts of sample texts in given domains which often makes these methods unrealistic in actual application environments. Furthermore, the input corpora to such learning programs are often required to be properly tagged or annotated, which demands enormous manual effort, making them far less useful.

In order to overcome the difficulties of these methods, we propose a Linguistic Knowledge Generator (LKG) which working on the principle of "Gradual Approximation" involving both human introspection and discovery programs.

In the following section, we will explain the Gradual Approximation approach. Then a scenario which embodies the idea and finally we describe an experiment which illustrates its use.

2 Gradual Approximation

Some of the traditional learning programs which are to discover linguistic regularities in a certain dimension requires some amount of training corpus to be represented or structured in a certain way. For example, a program which learns disambiguation rules for parts-of-speech may require training data to be represented as a sequence of words with their correct parts-of-speech. It may count frequencies of trigram of parts-of-speech in corpora to learn rules for disambiguation. On the other hand, a program to discover the semantic classes of nouns may require input data (sentences) to be accompanied by their correct syn-

*SEKINE is currently a visitor at U.M.I.S.T. sekine@ccl.umist.ac.uk

tactic structures, and so on.

This is also the case for statistical programs. Meaningful statistics can be obtained only when they are applied to appropriate units of data. Frequencies of characters or trigrams of characters, for example, are unlikely to be useful for capturing the structures of the semantic domains of a given sublanguage. In short, discovery processes can be effectively assisted or carried out, if corpora are appropriately represented for the purpose. However, to represent or tag corpora appropriately requires other sorts of linguistic or extra-linguistic knowledge or even the very knowledge which is to be discovered by the program.

For example, though corpora annotated with syntactic structures are useful for discovering semantic classes, to assign correct syntactic structures to corpora requires semantic knowledge in order to prevent proliferation of possible syntactic structures.

One possible way of avoiding this chicken-and-egg situation is to use roughly approximated, imperfect knowledge of semantic domains in order to hypothesize correct syntactic structures of sentences in corpora. Because such approximated semantic knowledge will contain errors or lack necessary information, syntactic structures assigned to sentences in corpora may contain errors or imperfections.

However, if a program or human expert could produce more accurate, less imperfect knowledge of semantic domains from descriptions of corpora (assigned syntactic structures), we could use it to produce more accurate, less erroneous syntactic descriptions of corpora, and repeat the same process again to gain further *improvement* both in knowledge of semantic domains and in syntactic descriptions of corpora. Thus, we may be able to converge gradually to both correct syntactic descriptions of corpora, and semantic classifications of words.

In order to support such convergence processes, LKG has to maintain the following two types of data.

1. knowledge sets of various dimensions (morphology, syntax, semantics, pragmatics/ontology of extra-linguistic domains etc.), which are hypothesized by humans or by discovery programs, and all of which are imperfect in the sense that they contain erroneous generalizations, lack specific information, etc.

2. descriptions of corpora at diverse levels, which are based on the hypothesised knowledge in 1. Because of the hypothetical nature of the knowledge in 1, descriptions based on it inevitably contain errors or lack precision.

Based on these two types of data, both of which contain imperfect, the whole process of discovering regularities in sublanguage will be performed as a relaxation process or a gradual repetitive approximation process. That is,

1. human specialists or discovery programs make hypotheses based on imperfect descriptions of corpora

2. hypotheses thus proposed result in more accurate, less imperfect knowledge

3. the more accurate, less imperfect knowledge in 2., results in a more accurate description of the corpora

The same process will be repeated from 1., but this time, based on the more accurate descriptions of corpora than the previous cycle. It will yield further, more accurate hypothesized knowledge and descriptions of corpora and so on.

3 Algorithm

In this section, we describe a scenario to illustrate how our idea of the "Gradual Approximation" works to obtain knowledge from actual corpora. The goal of the scenario is to discover semantic classes of nouns which are effective for determining (disambiguating) internal structures of compound nouns, which consist of sequences of nouns. Note that, because there is no clear distinction in Japanese between noun phrases and compound nouns consisting of sequences of nouns, we refer to them collectively as compound nouns. The scenario is comprised of three programs, ie. Japanese tagging program, Automatic Learning Program of Semantic Collocations and clustering program.

There is a phase of human intervention which accelerates the calculation, but in this scenario, we try to minimize it. In the following, we first give an overview of the scenario, then explain each program briefly, and finally report on an experiment that fits this scenario. Note that, though we use this simple scenario as an illustrative example, the same learning program can be used in another more complex scenario whose aim is, for example, to discover semantic collocation between verbs and noun/prepositional phrases.

3.1 Scenario

This scenario takes a corpus without any significant annotation as the input data, and generates, as the result, plausibility values of collocational relations between two words and word clusters, based on the calculated semantic distances between words.

The diagram illustrating this scenario is shown in Figure 1. The first program to be applied is the "Japanese tagging program" which divides a sentence into words and generates lists of possible parts-of-speech for each word.

Sequences of words with parts-of-speeches are then used to extract candidates for compound nouns (or noun phrases consisting of noun sequences), which are the input for the next program, the "Automatic Learning Program for Semantic Collocations" (ALPSC). This program constitutes the main part of

the scenario and produces the above-mentioned output.

The output of the program contain *errors*. *Errors* here mean that the plausibility values assigned to collocations may lead to wrong determinations of compound noun structures. Such *errors* are contained in the results, because of the errors in the tagged data, the insufficient quality of the corpus and inevitable imperfections in the learning system.

From the word distance results, word clusters are computed by the next program, the "Clustering Program". Because of the errors in the word distance data, the computed clusters may be counter-intuitive. We expect human intervention at this stage to formulate more intuitively reasonable clusters of nouns.

After revision of the clusters by human specialists, the scenario enters a second trial. That is, the ALPSC re-computes plausibility values of collocations and word distances based on the revised clusters, the "Clustering Program" generates the next generation of clusters, and humans intervene to formulate more reasonable clusters, and so on, and so forth. It is expected that word clusters after the (i+1)-th trial becomes more intuitively understandable than that of the i-th trial and that the repetition eventually converges towards *ideal* clusters of nouns and plausibility values, in the sense that they are consistent both with human introspection and the actual corpus.

It should be noted that, while the overall process works as gradual approximation, the key program in the scenario, the ALPSC also works in the mode of gradual approximation as explained in Section 3.2.2.

3.2 Programs and Human interventions

We will explain each program briefly. However the ALPSC is crucial and unique, so it will be explained in greater detail.

3.2.1 *Program:* Japanese tagging program

This program takes Japanese sentences as an input, finds word boundaries and puts all possible parts-of-speech for each word under adjacency constraints. From the tagged sentences, sequences of nouns are extracted for input to the next program.

3.2.2 *Program:* Automatic Learning Program of Semantic Collocations (ALPSC)

This is the key program which computes plausibility values and word distances. In this scenario, the ALPSC treats only sequences of nouns, but it can generally applied for any structure of syntactic relationships. It is an unique program with the following points [8]:

1. it does not need a training corpus, which is one of the bottle necks of some other learning programs
2. it learns by using a combination of linguistic knowledge and statistical analysis
3. it uses a parser which produces all possible analyses
4. it works as a relaxation process

While it is included as a part of a larger repetitive loop, this program itself contains a repetitive loop.

Overview

Before formally describing the algorithm, the following simple example illustrates its working.

A parser produces all possible syntactic descriptions among words in the form of syntactic dependency structures. The description is represented by a set of tuples, for example, `[head word, syntactic relation, argument]`. The only syntactic relation in a tuple is MOD for this scenario, but it can be either a grammatical relation like MOD, SUBJ, OBJ, etc. or a surface preposition like BY, WITH, etc. When two or more tuples share the same argument and the same syntactic-relation, but have different head-words, there is an ambiguity.

For example, the description of a compound noun: `"File transfer operation"` contains three tuples:

```
[transfer, MOD, file]
[operation, MOD, file]
[operation, MOD, transfer]
```

The first two tuples are redundant, because one word can only be an argument in one of the tuples. As repeatedly claimed in the literature of natural language understanding, in order to resolve this ambiguity, a system may have to be able to infer extra-linguistic knowledge. A practical problem here is that there is no systematic way of accumulating such extra-linguistic knowledge for given subject fields.

That is, unless a system has a full range of contextual understanding abilities it cannot reject either of the possible interpretations as 'impossible'. The best a system can do, without full understanding abilities, is to select more plausible ones or reject less plausible ones. This implies that we have to introduce a measure by which we can judge plausibility of 'interpretations'.

The algorithm we propose computes such measures from given data. It gives a plausibility value to each possible tuple, based on the sample corpus. For example, when the tuples `(transfer, MOD, file)` and `(operation, MOD, file)` are assigned 0.5 and 0.82 as their plausibility, this would show the latter tuple to be more plausible than the former.

The algorithm is based on the assumption that the ontological characteristics of the objects and actions denoted by words (or linguistic expressions in general), and the nature of the ontological relations

among them, are exhibited, though implicitly in sample texts. For example, nouns denoting objects which belong to the same ontological classes tend to appear in similar linguistic contexts.

Note that we talk about extra-linguistic 'ontology' for the sake of explaining the basic idea behind the actual algorithm. However, as you will see, we do not represent such things as ontological entities in the actual algorithm. The algorithm simply counts frequencies of co-occurrences among words, and word similarity algorithms interpret such co-occurrences as contexts.

The algorithm in this program computes the plausibility values of hypothesis-tuples like `(operation, MOD, file)`, etc., basically by counting frequencies of instance-tuples `[operation, MOD, file]`, etc. generated from the input data.

Terminology and notation

instance-tuple $[h, r, a]$: a token of a dependency relation; part of the analysis of a sentence in a corpus.

hypothesis-tuple (h, r, a) : a dependency relation; an abstraction or type over identical instance-tuples.

cycle : repeat time of the relaxation cycle.

$C_{T,i}$: Credit of instance-tuple T with identification number i. $[0, 1]$

V_T^g : Plausibility value of a hypothesis-tuple T in cycle g. $[0, 1]$

$D^g(w_a, w_b)$: distance between words, w_a and w_b in cycle g. $[0, 1]$

Algorithm

The following explanation of the algorithm assumes that the inputs are sentences.

1. For a sentence we use a simple grammar to find all tuples possibly used. Each instance-tuple is then given credit in proportion to the number of competing tuples.

$$C_T = \frac{1}{number\ of\ competing\ tuples} \quad (1)$$

This credit shows which rules are suitable for this sentence. On the first iteration the split of the credit between ambiguous analyses is uniform as shown above, but on subsequent iterations plausibility values of the hypothesis-tuples V_T^{g-1} before the iteration are used to give preference to credit for some analyses over others. The formula for this will be given later.

2. Hypothesis-tuples have a plausibility value which indicates their reliability by a number between 0 and 1. If an instance-tuple occurs frequently in the corpus or if it occurs where there are no alternative tuples, the plausibility value for the corresponding hypothesis must be large. After analysing all the sentences of the corpus, we get a set of sentences with weighted instance-tuples. Each instance-tuple invokes a hypothesis-tuple. For each hypothesis-tuple, we define the plausibility value by the following formula. This formula is designed so that the value does not exceed 1.

$$V_T^g = 1 - \prod_i (1 - C_{T,i}) \quad (2)$$

3. At this stage, the word-distances can be used to modify the plausibility values of the hypothesis-tuples. The word-distances are either defined externally using human intuition or calculated in the previous cycle with a formula given later. Distance between words induces a distance between hypothesis-tuples. To speed up the calculation and to get better results, we use similar hypothesis effects. The plausibility value of a hypothesis-tuple is modified based on the word distance and plausibility value of a similar hypothesis. For each hypothesis-tuple, the plausibility-value is increased only as a consequence of the similar hypothesis-tuple which has the greatest effect. The new plausibility value with similar hypothesis-tuple effect is calculated by the following formula.

$$V_T^g = V_T^g + (1 - V_T^g) * V_{T'}^g * (1 - D^g(w_a, w_b))^2 \quad (3)$$

Here, the hypothesis-tuple T'' is the hypothesis-tuple which has the greatest effect on the hypothesis-tuple T (original one). Hypothesis-tuple T and T' have all the same elements except one. The distance between T and T'' is the distance between the different elements, w_a and w_b. Ordinarily the difference is in the head or argument element, but when the relation is a preposition, it is possible to consider distance from another preposition.

4. Distances between words are calculated on the basis of similarity between hypothesis-tuples about them. The formula is as follows:

$$D^g(w_a, w_b) = \frac{\sum_T (V_T^g - V_{T'}^g)^\beta}{n} \quad (4)$$

T and T' are hypothesis-tuples whose arguments are w_a and w_b, respectively and whose heads and relations are the same. β is a constant parameter.

5. This procedure will be repeated from the beginning, but modifying the credits of instance-tuples between ambiguous analyses using the plausibility values of hypothesis-tuples. This will hopefully be more accurate than the previous cycle. On the first iteration, we used just a constant figure for the credits of instance-tuples. But this time we can use the plausibility value of the hypothesis-tuple which was deduced in the previous iteration. Hence with each iteration we expect more reliable figures. To calculate the new credit of instance-tuple T, we use:

$$C_T = \frac{V_T^{g\,\alpha}}{\sum_T \left(V_T^{g\,\alpha}\right)} \qquad (5)$$

Here, V_T^g in the numerator, is the plausibility value of a hypothesis-tuple which is the same tuple as the instance-tuple T. V_T^g in the denominator are the plausibility values of competing hypothesis-tuples in the sentence and the plausibility value of the same hypothesis-tuple itself. α is a constant parameter.

6. Iterate step 1 to 5 several times, until the information is saturated.

3.2.3 *Program:* Clustering program

Word clusters are produced based on the word distance data which are computed in the previous program. A non-overlapping clusters algorithm with the maximum method was used. The level of the clusters was adjusted experimentally to get suitable sizes for human intervention.

3.2.4 *Human intervention:* Select clusters

The clusters may inherit errors contained in the word distance data. The errors can be classified into the following two types.

1. A *Correct* cluster overlaps with two or more generated clusters.

2. A generated cluster overlaps with two or more *correct* clusters.

Note that *'correct'* here means that it is correct in terms of human intuition. To ease the laborious job of correcting these errors by hand, we ignore the first type of error, which is much harder to remove than the second one. It is not difficult to remove the second type of error, because the number of words in a single cluster ranges from two to about thirty, and this number is manageable for humans. We try to extract purely *'correct'* clusters or a subset of a correct cluster, from a generated cluster.

It is our contention that, though clusters contain errors, and are mixtures of clusters based on human intuition and clusters computed by process, we will gradually converge on correct clusters by repeating this approximation.

At this stage, some correct clusters in the produced clusters are extracted. This information will be an input of the next trial of ALPSC.

4 Experiment

We conducted an experiment using compound nouns from a computer manual according to the scenario. The result for other relations, for example prepositional attachment, would be not so different from this result.

The corpus consisted of 8304 sentences. As the result of Japanese tagging program, 1881 candidates, 616 kinds of compound nouns were extracted.

Then ALPSC took these compound nouns as an input. Tuple relations were supposed between all words of all compound nouns with the syntactic relation 'MODIFY'. A tuple has to have a preceding argument and a following head. For example, from a compound noun with 4 words, 5 ambiguous tuples and 1 firm tuple can be extracted, because each element can be the argument in only one tuple. An initial credit of 1/3 was set for each instance-tuple whose arguments are the first word of the compound noun. Similarly, a credit 1/2 was set for each instance-tuple in which the second word is an argument.

No word distance information was introduced in the first trial. Then the learning process was started.

We have shown the results of first trial in Table 1 and examples in Figure 2.

The results were classified as correct or incorrect etc.. 'Correct' means that a hypothesis-tuple which has the highest plausibility value is the correct tuple within ambiguous tuples. 'Incorrect' means it isn't. 'Indefinite' means that plausibility values of some hypothesis-tuples have the same value. 'Uncertain' means that it is impossible to declare which hypothesis tuple is the best without context.

Words	correct	incorrect	indefinite	uncertain
3	72	33	4	14
4	41	7	5	1
5	4	0	0	2
total	117	40	9	16
(%)	(70.5)	(24.1)	(5.4)	(-)

Table 1: Results of experiment after first ALPSC

The clustering program produced 44 clusters based on the word distance data. a sample of the clusters is shown in Figure 3. The average number of words in a cluster was 3.43. Each produced cluster contained one to twenty five words. This is good number to treat manually. The human intervention to extract correct clusters resulted in 26 clusters being selected from 44 produced clusters. The average number of

words in a cluster is 2.96. It took a linguist who is familiar with computer 15 minutes. A sample of the selected clusters is shown in Figure 4.

These clusters were used for the second trial of ALPSC. The results of second trial are shown in Table 2.

Words	correct	incorrect	indefinite	uncertain
3	76	30	3	14
4	43	5	5	1
5	4	0	0	2
total	123	35	8	16
(%)	(74.1)	(21.1)	(4.8)	(-)

Table 2: Results of experiment after second trial

5 Discussion

The scenario described above embodies a part of our ideas. Several other experiments have already been conducted, based on other scenarios such as a scenario for finding clusters of nouns by which we can resolve ambiguities caused by prepositional attachment in English. Though this works in a similar fashion as the one we discussed, it has to treat more serious structural ambiguities and more diverse syntactic structures.

Though we have not compared them in detail, it can be expected that the organization of semantic clusters of nouns that emerge in these two scenarios will be different from each other. One reflects collocational relations among nouns, while the other reflects those between nouns and verbs. By merging these two scenarios into one larger scenario, we may be able to obtain more accurate or intuitively reasonable noun clusters. We are planning to accumulate a number of such scenarios and larger scenarios. We hope we can report it soon.

As for the result of the particular experiment in the previous section, one eighth of the incorrect results have progressed after one trial of the gradual approximation. This is significant progress in the processing. For humans it would be a tremendously laborious job as they would be required to examine all the results. What humans did in the experiment is simply divide the produced clusters.

Although the clusters are produced by a non-overlapping clustering algorithm in this experiment, we are developing an overlapping clustering program. Hopefully it will produce clusters which involve the concept of word sense ambiguity. It will mean that a word can belong to several clusters at a time. The method to produce overlapping clusters is one of our current research topics.

Examining the results, we can say that the cluster effect is not enough to explain word relations of compound nouns. There might be some structural and syntactic restrictions. This feature of compound nouns made it hard to get a higher percentage of correct answers in our experiment. Extra-processing to address these problems can be introduced into our system.

Because the process concerns huge amount of linguistic data which also has ambiguity, it is inevitable to be experimental. A sort of repetitive progress is needed to make the system smarter. We will need to perform a lot of experiments in order to determine the type of the human intervention required, as there seems to be no means of determining this theoretically.

This system is aiming not to simulate human linguists who conventionally have derived linguistic knowledge by computer, but to discover a new paradigm where automatic knowledge acquisition programs and human effort are effectively combined to generate linguistic knowledge.

6 Acknowledgements

We would like to thank our colleagues at the CCL and Matsushita, in particular, Mr.J.Phillips, Mr.K.Kageura, Mr.P.Olivier and Mr.Y.Kanno, whose comments have been very useful.

References

[1] Ralph Grishman: Discovery Procedures for Sublanguage Selectional Patterns: Initial Experiments *Comp. Linguistics Vol.12 No.3* (1986)

[2] Sofia Ananiadou: Sublanguage studies as the Basis for Computer Support for Multilingual Communication *Proceedings of Termplan '90, Kuala Lumpur* (1990)

[3] A Zampolli: Reusable Linguistic Resources (Invited paper) *5th Conference of the E.A.C.L.* (1991)

[4] Kenneth Ward Church: A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text *2nd Conference on A.N.L.P* (1988)

[5] Donald Hindle and Mats Rooth: Structural Ambiguity and Lexical Relations *29th Conference of the A.C.L.* (1991)

[6] Smaja and McKeown: Automatically Extracting and Representing Collocations for language generation. *28th Conference of the A.C.L.* (1991)

[7] Uri Zernik and Paul Jacobs: Tagging for Learning: Collecting Thematic Relations from Corpus *13th COLING-90* (1990)

[8] S.Sekine, J.J.Carroll, S. Ananiadou, J.Tsujii: Automatic Learning for Semantic Collocation *3rd Conference on A.N.L.P.* (1992)

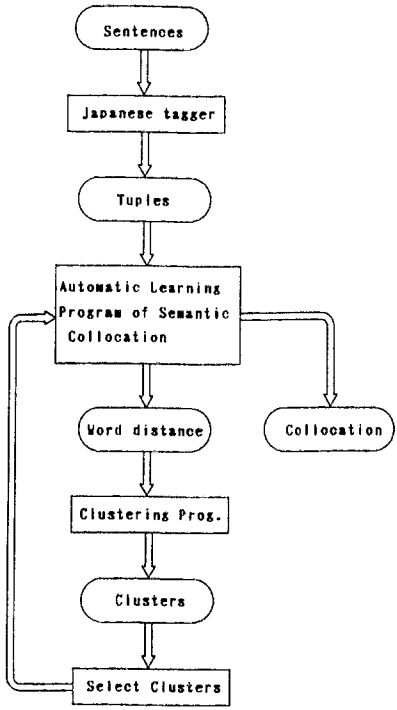

Figure 1. Diagram of the scenario

[グループ:group] [実行:execute]
[許可:permission] [文字:charcter]
(グループ 実行) 0.000
(グループ 許可) 0.002
(グループ 文字) 0.997 X
(実行 許可) 1.000 O
(実行 文字) 0.000

[エラー:error] [処理:management] [ルーチン:routin]
(エラー 処理) 0.505 O
(処理 ルーチン) 0.494

[ループ:loop] [終了:finish] [テスト:test]
(ループ 終了) 0.025
(終了 テスト) 0.976 X

Figure 2. Sample of the Results

{ファイル(file), 高速(high speed), 文脈(discource), ループ(loop), グループ(group), テーブル(table), 二重(duplication), 逆(reverse), 剰余(surplus), スケジュール(schedule), 優先(pliority), 用紙(paper), 浮動(float), 正規(regular), 文字(character), 構文(sentence-structure), 字句(words-and-phrase), 仮想, supposition), 仮字(KANA), 漢字(chinese-charcter), ディレクトリ(directory), ポップアップ(pop-up), レター(letter), バック(back-), 空白(white-)}
{ワイルド(wild), 計算(calculation)}
{エラー(error), 有限(infinite), 待ち(wait-)}
{C, 解析(analysis), 対象(object)}
{ハード(hard), 主(main-), 補助(assistance), 終了(finish)}
{アクセス(access), 使用(usage)}
{実行(execute), 参照(refer)}
{中(middle-)}
{接続(connection)}
{検索(retrieval)}
{規則(regulation)}
{前(forward-), コピー(copy)}
{複数(multiple), デフォルト(default)}
{フィールド(field)}
{復帰(return), 方向(direction), 幅(width), 全(full-), 半(half-)}
{表示(display), 変更(change)}
{上(upper-), 処理(management), 削除(delete)}
{表現(expression), フォント(font)}
{内部(internal-), 文法(grammar)}
{指定(specification), 入力(input), 出力(output)}
{テープ(tape)}
{修正(modification), 作成(creation), 最終(final-), モード(mode)}
{項(item)}
{右(right), 左(left)}

Figure 3. Sample of Produced Clusters in first trial

{文字(character), 字句(words-and-phrases), 仮字(KANA), 漢字(chinese-character), レター(letter)}
{文脈(discourse), 構文(sentence-structure)}
{ファイル(file), ディレクトリ(directory)}
{主(main-), 補助(assistance)}
{実行(execute), 参照(refer)}
{方向(direction), 幅(width)}
{全(full-), 半(half)}
{表示(display), 変更(change)}
{処理(management), 削除(delete)}
{表現(expression), フォント(font)}
{入力(input), 出力(output)}
{修正(modification), 作成(creation)}
{右(right), 左(left)}

Figure 4. Sample of Selected clusters