Inducing Differences among Documents using Aleph with Construction of Background Knowledge

Chieko Nakabasami

Toyo University, 1-1-1 Izumino Itakura Oura Gunma 374-0193, Japan
chiekon@itakura.toyo.ac.jp

Abstract. This paper presents a case study in which an Inductive Logic Programming (ILP) technique is applied to natural language processing. Aleph, an ILP system, is used to induce differences among documents. A Case-Based Reasoning (CBR) system is proposed for the purpose of compiling the background knowledge inputted into Aleph. In the CBR system, lexical and syntactic information concerning words in close proximity to the target word(s) in training sentences is provided in order to infer new cases effectively. The compiled background knowledge is used to determine the semantic differences in documents that are written in natural language. Tentative experiments with this technique have produced encouraging results.

1 Introduction

This paper describes a method to induce differences among natural language documents using Aleph[1], a successor to Progol[2]. Aleph is an acronym for "A Learning Engine for Proposing Hypotheses" and an Inductive Logic Programming (ILP) system. Aleph's main purpose was to understand the concept of inverse entailment, which appeared in [2]. Aleph requires three files to construct theories. They are "filestem.b, filestem.f, and filestem.n," which stand for background knowledge file, positive example file, and negative example file, respectively. Aleph receives these three files in order to construct a theory that is consistent with positive examples and inconsistent with negative ones.

The background knowledge inputted into Aleph is generated by means of Case-Based Reasoning (CBR) with user interaction if necessary. Sophisticated background knowledge should be provided to efficiently process natural language documents since they are rich in representation. The acquisition of background knowledge is one of the problems to be overcome. On the other hand, the acquisition and compilation of background knowledge have been reported to cause a bottleneck in knowledge engineering because they are very time-consuming tasks. The automatic generation of background knowledge provides a solution to the problem.

In this study, CBR is applied to obtain such knowledge. Though background knowledge is constructed with a great deal of user interaction during the initial phase of CBR, the frequency of interaction decreases as the information is processed by CBR. The system processes a sentence in turn with human interaction, and novel representations acquired are stored into cases so that they may be applied to future processes. The acquisition technique locates words that are in close proximity to target words, and it functions by matching syntactic and lexical patterns. Users validate the selected cases by judging them correct or incorrect. The CBR system enables users to add new appropriate cases and delete inappropriate ones.

The knowledge obtained includes discriminating features that have been extracted from the target documents and incorporated into the system. The knowledge that people want to acquire is not general knowledge concerning similar documents; instead, it is specific and distinctive knowledge contained only in the target document. In this paper, scientific manuscripts dealing with material design serve as the domain. The focus is on material design because collaborative engineering studies on the application of artificial intelligence have been conducted by metallurgy experts[3], and plenty of on-line manuscripts in this discipline are available. In my former research, a semantic representation of background knowledge in technical papers was described based on Pustejovsky's Generative Lexicon[4]. In this case, an object-oriented modeling approach was taken in order to infer the differences among papers that used similar experimental methods. However, an automatic mechanism could not be implemented in this study due to the difficulty entailed in implementing the automatic modeling of the background knowledge presented in the papers. The approach taken in this paper would be expected in order to save considerable time in the preparation of background information data for manual compilation.

Researchers in this field want to retrieve the differences in the details of the experiments written in the target paper and compare them to similar papers because small differences in experiments affect the properties of materials. The details cannot be understood by examining the abstracts alone. However, since it is time-consuming for researchers to read whole papers, some effective methods should be developed. Aleph, which contains the assembled background knowledge, detects and reports structural differences as the manuscripts are compared. Aleph produces a representative body of knowledge contained in the target manuscripts that is missing in the other manuscripts.

The remainder of this paper is organized as follows. Section 2 defines knowledge representation of the background knowledge. Section 3 explains the CBR system for generating the background knowledge. Section 4 illustrates the mechanism by which Aleph induces the intended results. Section 5 shows the tentative results produced by Aleph. Section 6 mentions related works for applying ILP to NLP. Section 7 is the conclusion.

2 Forms of Background Knowledge

2.1 Predicate Definitions

The background knowledge applied to Aleph has the form of Horn clauses. The predicate names and their arguments are defined in advance so that Aleph may efficiently construct understandable results. The predicates depend on the experimental papers on the subject of material design. They are as follows:

– result(*Paper*): The result of *Paper*
– has_focus(*Paper, Focus*): The *Paper* has the *Focus*.
– focus (*Focus, Content*): The content of the *Focus* is *Content*.
– action (*Paper, Action, Target*): The *Paper* describes the *Action* that is conducted for the *Target*.
– change (*Paper, Result, Target*): The *Paper* describes the *Result* that is achieved for the *Target*.
– purpose (*Paper, Purpose, Target*): The *Paper* describes the *Purpose* of processing the *Target*.
– show (*Paper, Result, Target*): The *Paper* describes the *Result* of processing the *Target*.
– detail (*Paper, Detail, Target*): The *Paper* describes the *Detail* of the *Target*.
– condition (*Paper, Proposition, Target*): The *Paper* describes under what condition the *Target* is processed. The *Proposition* represents a proposition used in the text.

2.2 Declaration Part of Background Knowledge for Aleph

The background knowledge for Aleph is composed of two parts[1]. One is the declaration part for each predicate, and the other is the representation of knowledge that expresses the content of the target document. The representation uses the word(s) from the documents as arguments in each predicate. The declaration part for each predicate is as follows:

Declaration Part

```
:-modeh(1,result(+paper)).
:-modeb(*,has_focus(+paper,-focus)).
:-modeb(*,focus(+focus,-content)).
:-modeb(*,action(+focus,-action,-target)).
:-modeb(*,detail(+focus,-detail,-target)).
:-modeb(*,show(+focus,-action,-target)).
:-modeb(*,condition(+focus,-condition,-target)).
:-modeb(*,change(+focus,-action,-target)).
:-modeb(*,purpose(+focus,-action,-target)).
:-determination(result/1,has_focus/2).
:-determination(result/1,focus/2).
:-determination(result/1,action/3).
:-determination(result/1,detail/3).
:-determination(result/1,show/3).
:-determination(result/1,condition/3).
:-determination(result/1,change/3).
:-determination(result/1,purpose/3).
```

In the Declaration Part, *modeh* declares the predicate and its argument type(s) that should be used as the head of the theory constructed by Aleph, and *modeb* as the body. The first argument of modeh and modeb can be either a number specifying the number of successful calls to the predicate; or * specifying that the predicate has bounded non-determinacy. For instance, *modeh(1,result(+paper))* declares the predicate *result/1*, which has an argument of type *paper*. The symbols "+" and "-" in the declarations represent the input and output variables, respectively.

In addition, *determination* declares the predicates that can be used to construct a hypothesis. For instance, *determination(result/1,has_focus/2)* declares that a hypothesis should be constructed with *result/1* as the head of a clause and *has_focus/2* as the body. A precise explanation of the declarations of background knowledge for Aleph is given in the Aleph manual[1].

2.3 Knowledge Representation of Sentences in Documents

The background knowledge for Aleph has been extracted from the sentences in the target documents according to the predicate declarations presented in section 2.2. This information is developed by using the CBR process. For example, the following representation (3) is constructed from sentence (1) via (2). The words in the sentences are tagged with a part-of-speech[5] such as (2).

```
(1) Feedstock material is melted in a high vacuum chamber
    by electron beam guns into water.
(2) [feedstock/nn,material/nn],is/vbz,melted/vbn,
    in/in,[a/dt,high/jj,vacuum/nn,chamber/nn],
    by/in,[electron/nn,beam/nn,guns/nns],
    into/in,[water/nn].
(3) action([melt],[feedstock, material]).
    condition([in],[high,vacuum,chamber]).
    condition([by],[electron,beam,guns]).
    condition([into],[water]).
```

In (2), the determiners are removed, and "nn," "vbz," "vbn," "dt," "jj," "in," and "nns" represent singular noun, singular present verb, past participle, determiner, adjective, proposition, and plural noun, respectively. The representations in (3) correspond to the predicates in the declaration part of the background knowledge in section 2.2. The transformation presented above is explained in section 3.

3 CBR Process for Construction of Background Knowledge

Case-Based Reasoning (CBR) is applied to obtain the background knowledge concerning the contents of the documents. In the CBR system, each sentence stored is represented with a set of *pred/3* predicate and composed of three arguments. These three have a list form of the predicate introduced in section 2.1, which deals with lexical and syntactic information. For instance, sentence (1) shown in section 2.3 is represented by a form such as (4), if it is stored as one of the training cases.

```
(4) pred([action,[melt],[feedstock,material]],[],[is/vbz,melted/vbn]).
    pred([condition,[in],[high,vacuum,chamber]],[in/in],[]).
    pred([condition,[by],[electron,beam,guns]],[by/in],[]).
    pred([condition,[into],[water]],[into/in],[]).
```

The first argument of *pred/3* corresponds to one of the predicates introduced in section 2.1. As a second argument of *pred/3*, the lexical and syntactic information that appears in a previous position in a sentence is based on the third element of the first argument of *pred/3* (e.g., [feedstock, material]) and as the third argument in the postposition. Hereafter, the term used for the third element of the first argument of *pred/3* is "pivot words." Empty lists are assigned when there is not a pair of keywords.

The CBR system locates the words nearest the target by matching syntactic and lexical patterns. The criterion for finding neighboring words is whether a word (or phrase) in novel sentences includes the word(s) in the pivot words and whether the part-of-speech of the word(s) in front of and behind the pivot words matches the part-of-speech in the original cases. For example, sentence (5) is translated into (7) because the word "material" in (5) is included among the pivot words of the first *pred/3* in (4). In addition to word matching and from the fact that the part-of-speech of the words behind "material" matches those of (4) (i.e., "vbz," "vbn"), it can be assumed that one of the neighboring cases of sentence (1) is (5). Then (7) is constructed from (5) via (6).

```
(5) The material is sampled for that experiment.
(6) [the/dt,material/nn],is/vbz,sampled/vbn,for/in,
    [that/dt,experiment/nn].
(7) action([sample],[material]).
```

The CBR system enables users to add new appropriate cases and delete inappropriate ones. The schema of the CBR system is sketched in Fig. 1.

In the CBR system, about 120 cases are stored manually as initial cases. These cases are randomly extracted from among 630 manuscripts on the subject of the material design of superalloys. The system processes a sentence as humans interact, and the acquired novel representations are stored into cases so that they may be applied to future processes. The lexical and syntactic information around the pivot words is assigned automatically when the new case is stored. Background knowledge is constructed with a great deal of user interaction during the initial phase of CBR. The frequency of interaction decreases as the information is processed by CBR because the system can easily determine the neighbors that are similar to others. Some interaction that has been processed by the CBR system is illustrated in Appendix.

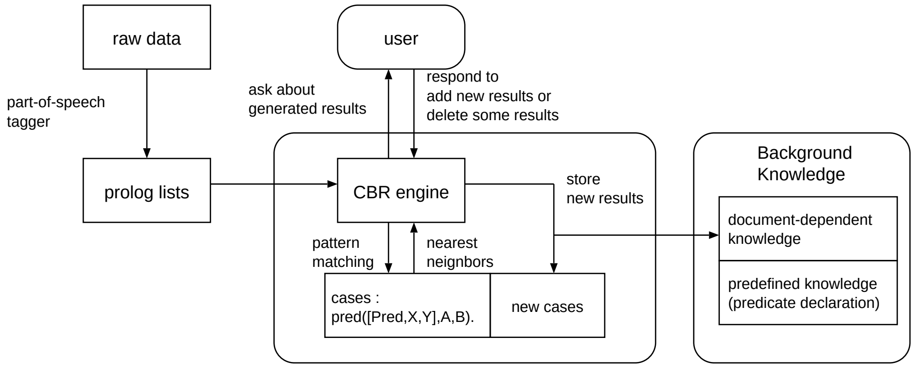

Fig. 1. Schema of the system

4 Mechanism for Inducing Differences among Documents

The following data are prepared for Aleph:

- Positive example: A target paper (or papers) from which to extract information;
- Negative example: Other papers that have some content in common and some that differs from the target paper; and
- Background knowledge: Descriptions in terms of content for each example.

The three kinds of data are inputted into Aleph, and descriptions of the content of the target paper that is absent in the other related papers are developed. At first, researchers pick up the paper that they want to process and also choose other similar papers (e.g., the title words might include the same material name or type of mechanical testing). Each paper is given a unique identification (ID). Next, these papers are inputted into the CBR system one by one, and the system outputs the background knowledge used in Aleph. After that process, Aleph induces the description that differentiates the target papers from others. For example, the following files are prepared for Aleph. These file stems are identified as "alloy" because of the material design domain.

- alloy.f: Positive examples are written.

  ```
  result(p01).  result(p02).
  ```
- alloy.n: Negative examples are written.

  ```
  result(p03).
  ```
- alloy.b: The knowledge representation of the papers in both positive and negative examples, which has been constructed by the CBR system, is written.

In the example above, Aleph attempts to induce the differences in content present in paper "p01" and "p02" but absent in paper "p03." The body of result/1 is hypothesized by means of the ILP technique. The hypothesis represents the differences between the positive and negative example papers.

5 Results

Tentative results are obtained using sample papers. Experts chose the papers carefully to ensure that the experimental methodology was similar. Three sets, each consisting of three papers, were chosen. The sections on the experimental methods in the three papers are selected as sample documents. These documents are divided into positive and negative examples. Two documents are considered positive and one negative. The positive and negative documents are as follows:

Positive

```
Experimental (paper id: p01)
Five new alloy compositions were selected to analyze the potential
mechanical property improvements obtained when multiphase
dispersions of fine gamma, carbide, and oxide particles are
combined in typical, fully alloyed nickel-based superalloy
compositions.

Experimental alloys and procedures (paper id: p02)
Alloy compositions, listed in Table 1, were selected to investigate
a wide range of refractory alloying additions and to permit
statistical analysis of results.
```

Negative

```
Experimental methods (paper id: p03)
GTD-111 specimens were machined into cylindrical shapes
with four different diameters of 10, 15, 25, and 35 mm
and a length of 30 mm.
```

Using these samples, Aleph induced the following descriptions. Aleph is somewhat customized so that the descriptions will be more understandable for users.

Output

```
result(A) :- has_focus(A,B),purpose(B,C,D).

The difference compared to other papers:

paper p02 describes in terms of [experimental,alloy]
The purpose of the [experimental,alloy] is to [investigate]
the [refractory,alloying,addition]

The difference compared to other papers:

paper p02 describes in terms of [experimental,alloy]
The purpose of the [experimental,alloy] is to [permit]
```

```
the [statistical,analysis]

The difference compared to other papers:

paper p01 describes in terms of [experimental]
The purpose of the [experimental] is to [analyze]
the [potential,mechanical,property,improvement]
```

Positive

```
Experiment (paper id: p04)
The crystallographic mode of propagation indicates
enhanced cracking resistance of the grain boundaries.
While this mode of propagation was observed in all
specimens tested at 1400 F, the amount of crystallographic
cracking varied. This transgranular fracture mode
at 1400 F appears to be a characteristic of the hafnium effect.

Experimental procedures (paper id: p05)
With the addition of hafnium to D.S. MAR-M200
in 1969, transverse grain boundary strength and
ductility were significantly improved.
Work on single crystals was discontinued at this time
because they offered no significant improvements
in properties over D.S. and they were more expensive.
```

Negative

```
Experimental methods (paper id: p06)
SC blades were used to inhibit grain nucleation on mold walls.
No significant investment mold-superalloy reactions
were observed as a result of the higher superheats
required for SC castings.
```

Output

```
result(A) :- has_focus(A,B),show(B,C,D).

The difference compared to other papers:

paper p05 describes in terms of [single,crystal]
The observation of the [single,crystal] is to [improve]
the [mar,m200]
```

The difference compared to other papers:

paper p05 describes in terms of [single,crystal]
The observation of the [single,crystal] is to [improve]
the [ductility]

The difference compared to other papers:

paper p05 describes in terms of [single,crystal]
The observation of the [single,crystal] is to [offer]
the [no,significant,improvement]

The difference compared to other papers :

paper p04 describes in terms of [hafnium,addition]
The observation of the [hafnium,addition] is to [indicate]
the [enhanced,cracking,resistance]

The difference compared to other papers :

paper p04 describes in terms of [hafnium,addition]

The observation of the [hafnium,addition] is to [observe]
the [propagation]

Positive

Fatigue at high Delta-K at 700 C (paper id: p07)
Particles were observed on fracture surfaces and
there was evidence that intergranular crack growth
had involved fine-scale cavitation.
Striations with spacings close to the macroscopic
growth rate were present on transgranular areas.

Experimental procedure (paper id: p08)
Fatigue-crack-growth data were obtained by testing
compact-tension (CT) specimens cut from the forging
blanks such that specimens were equidistant from
the centre of the forging and the direction of
crack growth was radial.

Negative

```
Materials and experimental approach (paper id: p09)
The crack length was monitored during the precrack period
using the plastic replica technique. Several of the cracks
were naturally initiated at elevated temperature under the
actual strain controlled test conditions to better simulate
the actual crack growth process.
```

Output

```
result(A) :- has_focus(A,B),detail(B,C,D).

The difference compared to other papers :

paper p07 describes in terms of [fatigue]
The detail of the [fatigue] is [high,delta-k] of the [700,deg,c]

The difference compared to other papers :

paper p07 describes in terms of [growth]
The detail of the [growth] is [striation] of the [spacing]
```

6 Related Work

There are some studies although the search was not exhaustive, applying Progol to Natural Language Processing[6][7][8]. In these cited studies, Progol is used to develop rules for making words less ambiguous by tagging them according to the part of speech they represent. As in the other papers, this study uses context information found in close proximity to the target word, which we call "pivot words." In addition, ILP methods are applied to NLP applications such as conceptual clustering[9], transfer-rule learning[10], learning morphology[11][12][13], and learning phonetic rules[14]. A note by [15] mentions the European works on ILP. It has been proposed that ILP could be applied to semantic parsing as well as to the analysis of the syntactic aspects of language[16]. In [16], syntactic and semantic parsers are learned so that a natural language database query can be mapped directly into an executable Prolog query that will answer the question. Promising advantages are being claimed from the synthesis of ILP and NLP because of the knowledge that can be gained[17].

On the other hand, as for applying CBR methods to acquire knowledge representation, [18] proposes a system, which is used to extract structural knowledge in terms of a specific domain. While the purpose of [18] is to assign a correct conceptual role to each word by making it less syntactically and semantically ambiguous, a goal of this study is to extract discriminating features from target documents by acquiring background knowledge from them for an ILP system such as Aleph.

7 Conclusion

In this paper, a case study for inducing differences among natural language documents was reported. Aleph was used to develop explanations for the semantic differences that exist in a variety of documents. The background knowledge for Aleph is constructed using a CBR system. In the CBR system, lexical and syntactic information is examined to develop information that can help interpret novel sentences. The experiments that make use of cumulative background knowledge in this paper show promising results. How semantic representations are produced and the structure of background knowledge are subjects for future research. In addition, a sound synthesis of ILP and NLP should be carefully considered, and NLP applications should be developed on the basis of such a synthesis.

Acknowledgements

I thank Dr. Ken'ichi Hoshimoto, former researcher at the National Research Institute for Metals (NRIM), for his valuable advice and assistance with the development and evaluation of the system.

References

1. Srinivasan, A. and Camacho, R.: The Aleph Manual. http://www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/(1993)
2. Muggleton, S.: Inverse entailment and Progol. New Generation Computing Journal,13. (1992) 245–286
3. Nakabasami, C., Hoshimoto, K.: Extracting Knowledge from Technical Papers in Metallurgy on the basis of a Generative Lexicon. Proc. of the Conference Pacific Association for Computational Linguistics. (1999) 137–142
4. Pustejovsky, J.: The Generative Lexicon. MIT Press. (1995)
5. Brill, E.: A simple rule-based part of speech tagger. Proc. of the 3rd Conference on Applied Natural Language Processing (ACL'92). (1992)
6. Cussens, J.: Part-of-Speech Tagging Using Progol. Proc. of 7th International Conference on Inductive Logic Programming (ILP'97). (1997) 93–108
7. Cussens, J.: Using Inductive Logic Programming for Natural Language Processing. Workshop Notes on Empirical Learning of Natural Language Tasks (ECML'97). (1997) 25–34
8. Eineborg, M., Lindberg, N.: Induction of Constraint Grammar-Rules Using Progol. Proc. of 8th International Conference on Inductive Logic Programming (ILP'98). (1998) 116–124
9. Faure, D., Nédellec, C.: Aquisition of Semantic Knowledge using Machine Learning Methods: The System "ASIUM". Technical Report ICS-TR-88-16, LRI University. (1998)

10. Böstrom, H.: Induction of Recursive Transfer Rules. Proc. of Learning Language in Logic (LLL) Workshop. (1999)
11. Džeroski, S., Erjavec, T.: Induction of Slovene Nominal Paradigms. Proc. of 7th International Conference on Inductive Logic Programming (ILP'97). (1997) 141–148
12. Manandhar, S., Džeroski, S., Erjavec, T.: Learning Multilingual Morphology with CLOG. Proc. of 8th International Conference on Inductive Logic Programming (ILP'98). (1998) 135–144
13. Kazakov, D., Manandhar, S.: A Hybrid Approarch to Word Segmentation. Proc. of 8th International Conference on Inductive Logic Programming (ILP'98). (1998)
14. Alexin, Z., Csirik, J., Jelasity, M., Tóth, L.: Learning Phonetic Rules in a Speech Recognition System. Proc. of 7th International Conference on Inductive Logic Programming (ILP'97). (1997) 37–44
15. Cussens, J.: Notes on Inductive Logic Programming Methods in Natural Language Processing (European Work). Manuscript. (1998).
16. Zelle, J. M., Mooney, R. J.: Learning Semantic Grammars with Constructive Inductive Logic Programming. Proc. of the 11th National Conference on Artificial Intelligence. (1993) 817–822.
17. Mooney, R. J.: Inductive Logic Programming for Natural Language Processing. Proc. of 6th International Conference on Inductive Logic Programming (ILP'96). (1996) 3–22
18. Cardie, C.: A Case-Based Approarch to Knowledge Aquisition for Domain-Specific Sentence Analysis. Proc. of the 11th National Conference on Artificial Intelligence. (1993) 798–803

Appendix: Some interaction processes in the CBR system for constructing background knowledge.

Words as the arguments in predicates are transformed into basic forms (e.g., present forms for verbs; singular for nouns) when they are stored as new cases in the CBR system.

Sample Sentence

```
Prior to releasing the alloy for producing engine applications, an
extensive experimental engine test program was conducted under very
severe operating conditions.
```

Interaction Output *(See the next page)*

***** now processing
*** result candidate(s) ***
detail([producing,engine,application], [alloy]).
action([conducted], [alloy]).
purpose([conducted], [alloy]).
action([conducted], [extensive,experimental,engine,test,program]).
focus([extensive,experimental,engine,test,program]).
condition([under], [severe,operating,condition]).

*** present sentence ***
[prior,to,releasing,[the,alloy],for,[producing,engine,application],
[an,extensive,experimental,engine,test,program],was,conducted,under,
very,[severe,operating,condition]].

***** Any other clauses ? (y or n):
|: y.
***** Tell me the number of the appropriate predicate:
** 1: focus ** 2: action ** 3: show
** 4: purpose ** 5: change ** 6: condition
** 7: detail
|: 6.

***** Tell me an appropriate action or a focus word:
|: [prior,to].

***** Tell me an appropriate target (in list form):
|: [releasing].

*** added term ***
condition([prior,to], [releasing]).

*** current term(s) ***
condition([prior,to], [releasing]).
detail([producing,engine,application], [alloy]).
action([conducted], [alloy]).
purpose([conducted], [alloy]).
action([conducted], [extensive,experimental,engine,test,program]).
focus([extensive,experimental,engine,test,program]).
condition([under], [severe,operating,condition]).

*** present sentence ***
[prior,to,releasing,[the,alloy],for,[producing,engine,application],
[an,extensive,experimental,engine,test,program],was,conducted,under,
very,[severe,operating,condition]].

***** Any other clauses ? (y or n):
|:y.
***** Tell me the number of the appropriate predicate:
** 1: focus ** 2: action ** 3: show
** 4: purpose ** 5: change ** 6: condition
** 7: detail
|: 2.

***** Tell me an appropriate action or a focus word:
|: [alloy].

*** added term ***
action([release], [alloy]).

*** current term(s) ***
action([release], [alloy]).
condition([prior,to], [releasing]).
detail([producing,engine,application], [alloy]).
action([conducted], [alloy]).
purpose([conducted], [alloy]).
action([conducted], [extensive,experimental,engine,test,program]).
focus([extensive,experimental,engine,test,program]).
condition([under], [severe,operating,condition]).

*** present sentence ***
[prior,to,releasing,[the,alloy],for,[producing,engine,application],
[an,extensive,experimental,engine,test,program],was,conducted,under,
very,[severe,operating,condition]].

***** Any other clauses ? (y or n):
|: n.
*** result candidate(s) ***
** 1: condition([prior,to], [releasing]).
** 2: action([release], [alloy]).
** 3: detail([producing,engine,application], [alloy]).
** 4: action([conducted], [alloy]).
** 5: purpose([conducted], [alloy]).
** 6: action([conducted], [extensive,experimental,engine,test,program]).
** 7: focus([extensive,experimental,engine,test,program]).
** 8: condition([under], [severe,operating,condition]).

*** present sentence ***
[prior,to,releasing,[the,alloy],for,[producing,engine,application],
[an,extensive,experimental,engine,test,program],was,conducted,under,
very,[severe,operating,condition]].

***** Are there any clauses to be deleted ? (y or n):
|: y.
***** Tell me the position of the element to be deleted:
|: 4.

***** Is the element to be deleted similar to the following ? (y or n):
action([conducted], [alloy]).
|: y.
*** current term(s) ***
** 1: condition([prior,to], [releasing]).
** 2: action([release], [alloy]).
** 3: detail([producing,engine,application], [alloy]).
** 4: purpose([conducted], [alloy]).
** 5: action([conducted], [extensive,experimental,engine,test,program]).
** 6: focus([extensive,experimental,engine,test,program]).
** 7: condition([under], [severe,operating,condition]).

*** present sentence ***
[prior,to,releasing,[the,alloy],for,[producing,engine,application],
[an,extensive,experimental,engine,test,program],was,conducted,under,
very,[severe,operating,condition]].

***** Are there any clauses to be deleted ? (y or n):
|: n.
*** generated term(s) ***
condition([prior,to], [releasing]).
action([release], [alloy]).
detail([producing,engine,application], [alloy]).
purpose([conducted], [alloy]).
action([conducted], [extensive,experimental,engine,test,program]).
focus([extensive,experimental,engine,test,program]).
condition([under], [severe,operating,condition]).

Yes