NUL System at NTCIR RITE-VAL tasks

Ai Ishii
Nihon Unisys, Ltd.
ai.ishi@unisys.co.jp

Hiroshi Miyashita
Nihon Unisys, Ltd.
hiroshi.miyashita@unisys.co.jp

Mio Kobayashi
Nihon Unisys, Ltd.
mio.kobayashi@unisys.co.jp

Chikara Hoshino
Nihon Uniisys, Ltd.
chikara.hoshino@unisys.co.jp

ABSTRACT

This paper describes the submitted strategy and the methods of NUL team on NTCIR-11 RITE-VAL[1] fact validation (FV) and system validation (SV) tasks. We started to follow the shallow approach by Tian et al[3]. Then, we improved the named entity recognition accuracy and transformed some variables by the cross validation score of training sets. Especially, in the FV tasks, we used Apache Solr as the base search system. We compared several units of chunk to the texts index and the weighting of the ranking score for search results. After several modification, we achieved the highest cross validation score to the RITE-10 Exam bc and Exam Search tasks. Our final submitted system achieved Macro-f1 score 61.93 in FV and 69.59 in SV respectively.

Team Name

NUL

Subtasks

Fact Validation and System Validation (Japanese)

Keywords

named entity recognition, search results ranking, machine learning, logical reasoning

1. INTRODUCTION

Recognizing Textual Entailment (RTE) is the task of determining whether "hypothesis sentence" can be inferred from "text sentence". This task is one of the recent challenge of Natural Language Processing (NLP), and one of the essential function to solve Question Answering (QA), Information Retrieval (IR), Information Extraction (IE), Summarizing (SUM), and so on. We examined some papers associated with RTE and investigated the entailment examples in RITE-2. The consequence of that, we found one of the most important key factor to solve RTE task is to map words between sentences ("hypothesis sentence" and "text sentence"). Thus, in this task we focused on words mapping between sentences, made features (numerical value vector) to determine entailment from that mapping and solved the task by using the technique of machine learning. Besides, we used existing tools (Cabocha[1], KNP[2] , NormalizeNumexp[3], Wikipedia Hyponymy extraction tool[4], Solr[5]) and data (Wikipedia, Wikipedia redirect[6], WordnetJP[7], NihongoGoiTaikei[8]) to map words between sentences.

2. SHALLOW APPROACH

In this section, we describe our shallow approach for the SV task.

2.1 System Architecture

Figure 1 shows our textual entailment system architecture. Our system is mainly consisted of three subsystems (feature extraction, learning and judgment). In the feature extraction, it takes a Japanese sentence pair (T/H) as input, then make a feature vector for each sentence pair. In the learning, it takes feature vectors and answer labels (Y/N) as input, then adjust hypothesis formula's parameters by Logistic Regression. In the judgment, it takes feature vectors as input, then outputs inferred answer labels. Moreover, we eliminate unwanted characters and normalize characters in raw inputs (T/H pairs) to make subsequent process easy.

2.2 Features

We make following features from each T/H sentence pair.

2.2.1 *Number Expression Correspondence*

First, we use normalizedNumexp to extract number expressions from a sentence. Then we convert them into number or date range. Number Expression Correspondence feature is defined by following formula.

$$f_1 = \begin{cases} 1, \ When\ F_1 = 1 \\ 0.1, \ Otherwise \end{cases}$$

F_1 is defined by following formula.

$$F_1 = \frac{\#(Corr\ NumE_{h,t})}{\#(NumE_h)}$$

Where $\#(Corr\ NumE_{h,t})$ is number of H's Number expression which are corresponding to T's Number expressions. $\#(NumE_h)$ is number of H's Number expression. When

[1] https://code.google.com/p/cabocha/
[2] http://nlp.ist.i.kyoto-u.ac.jp/index.php?KNP
[3] http://www.cl.ecei.tohoku.ac.jp/~katsuma/software/normalizeNumexp/
[4] http://alaginrc.nict.go.jp/hyponymy/
[5] http://lucene.apache.org/solr/
[6] https://code.google.com/p/wikipedia-redirect/
[7] http://nlpwww.nict.go.jp/wn-ja/
[8] http://www.iwanami.co.jp/hotnews/GoiTaikei/

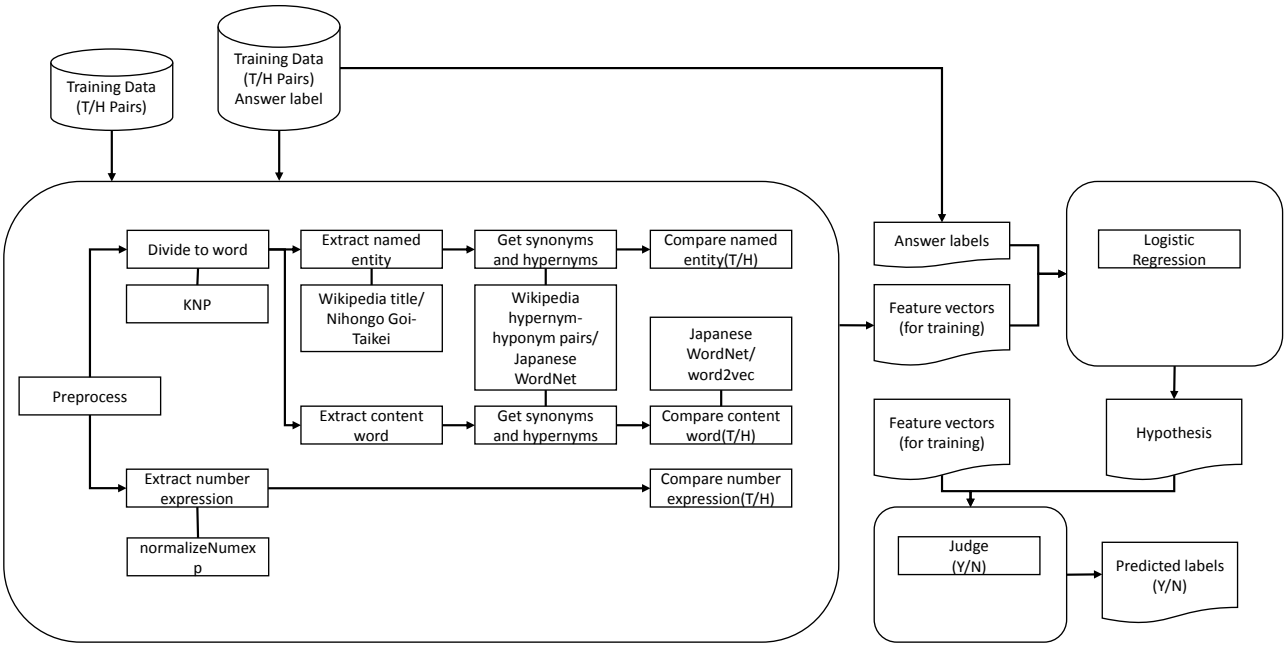


Figure 1: System architecture

a H's Number expression satisfies following condition, we judge the Number expression to be correspond.

$$H's\ Number\ Expression \supset T's\ Number\ Expression$$

If Number expression doesn't exist in H, we assign 1 to this feature.

2.2.2 Named Entity Correspondence

First, we use KNP to divide a sentence into word chunks (Basic Clause). We concatenate adjacent word chunks in order, then search longest match string from Wikipedia title and Nihongo goi taikei. If word chunks matches, we identify them as Named Entity (NE). When concatenating word chunks, we eliminated unwanted suffix. Next, we retrieve synonyms and hypernyms from Wikipedia Redirect and Wikipedia hypernym dictionary. We made Wikipedia hypernyms dictionary by Wikipedia Redirect tool beforehand. Named Entity Correspondence feature is defined by following formula.

$$f_2 = \begin{cases} 1, & When\ F_2 = 1 \\ 0.1, & Otherwise \end{cases}$$

where F_2 is defined by following formula.

$$F_2 = \frac{\#(Corr\ NE_{h,t})}{\#(NE_h)}$$

In this formula, $\#(Corr\ NE_{h,t})$ is number of H's Named Entity which are corresponding to either T's Named Entity. $\#(NE_h)$ is number of H's Named Entity. If H's Named Entity satisfied following condition, we judge it to be correspond.

$$(NE_h \cup (synonyms\ of\ NE_h) \cup (hypernyms\ of\ NE_h)) \cap (NE_t \cup (synonyms\ of\ NE_t)) \neq \phi$$

Where NE_h is H's Named Entity and NE_t is T's Named Entity. Moreover, to solve orthographic variation, we calculate Levenshtein Distance between strings, then if that distance is under a threshold value, we treated these strings are corresponding. If Named Entity doesn't exist in H, we assign 1 to this feature.

2.2.3 Content Word Correspondence Rate

We identify word chunks as a Content Word. Then eliminate stop words like "する", "ある", "こと", "もの" ... etc. Then we retrieve synonyms and hypernyms from Wikipedia Redirect, Wikipedia hypernym dictionary, Nihongo goi taikei and Japanese WordNet. Content Word Correspondence Rate feature is defined by following formula.

$$f_3 = \frac{\#(Corr\ CW_{h,t})}{\#(CW_h)}$$

In this formula, $\#(Corr\ CW_{h,t})$ is number of H's Content Word which are corresponding to T's either Content Words. $\#(CW_h)$ is number of H's Content Words. If a H's Content Word satisfied following conditions, we judge the Content Word to be correspond.

$$(CW_h \cup (synonyms\ of\ CW_h) \cup (hypernyms\ of\ CW_h) \cap (CW_t \cup (synonyms\ of\ CW_t))) \neq \phi$$

Where CW_t is T's Content Words. Moreover, to solve orthographic variation, we calculate Levenshtein Distance be-

tween strings, then if that distance is under a threshold value, we treated these strings are corresponding.

2.2.4 Content Word First Character Correspondence Rate

Content Word First Correspondence Rate feature is defined by following formula.

$$f_4 = \frac{\#(FC\ Corr\ CW_{h,t}) - \#(Corr\ CW_{h,t})}{\#(CW_h)}$$

In this formula, $\#(FC\ Corr\ CW_{h,t})$ is number of H's first character which is corresponding to either first character of T's Content Word. We compare first character of Content Word and its synonyms in both sentences (T/H).

2.2.5 Word2vec Cosign Distance

Word2vec Cosign Distance feature is defined by following formula.

$$f_5 = \frac{\sum_{CW_h \in NMCW_{h,t}} (\max_{CW_t}(\cos(v(CW_h), v(CW_t))))}{\#(CW_h)}$$

Where $NMCW_{h,t}$ is a set of H's Content Word which is not corresponding to T. $v(w)$ is a numerical vector mapped to word w. We use word2vec to map word and numerical vector. $cos(v_1, v_2)$ is a cosign similarity between vector v_1 and v_2. If a H's Content Word is not registered in word2vec, we assign 0 to $v(CW_h)$. Word2vec was trained by Wikipedia content in advance.

2.2.6 Exclusive Word

First, we retrieve adjective's attribute's categories by searching Japanese WordNet. When searching, we use SQL (Appendix A) to retrieve them. Second, we compare attribute's categories of Content Word in both sentences (T/H). Then, if a H' s Content Word satisfies following condition, we judge sentence H has an exclusive word against sentence T.

$$(\bigcup_{CW_h \in NMCW_h} AC(CW_h)) \cup (\bigcup_{CW_t \in CW_{all}} AC(CW_t)) \neq \phi$$

Where CW_{all} is a set of T's Content Word and AC(CW) is an attribute category set related to Content Word CW. Exclusive Word feature is defined by following formula.

$$f_6 = \begin{cases} 1, & When\ H\ has\ exclusive\ word \\ 0.1, & Otherwise \end{cases}$$

2.2.7 Non Match Content Word Rate

Non Match Content Word Rate feature is defined by following formula.

$$f_7 = \frac{\#(NMCW_{h,t})}{\#(CW_h)}$$

Where $\#(NMCW_{h,t})$ is number of H's Content Word which doesn't correspond any T's Content Words.

2.2.8 Learning

We use following Hypothesis for Logistic Regression.

$$h_\theta(f) = \frac{1}{1 + e^{-z}}$$

where

$$z = \sum_{i=1}^{n} \theta_i f_i \quad (f_i : feature\ value.\ f_0 \equiv 1.)$$

In our system, if the Hypothesis returns 0.5 or over, we predict that sentence T entails sentence H. We use following cost function to adjust parameters $(\theta_0, \ldots, \theta_7)$.

$$J_\theta = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(f^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(f^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

Where m is number of training data and $y^{(i)}$ is answer label of training data.

$$y^{(i)} = \begin{cases} 1, & When\ answer\ label\ of\ i-th\ training\ data\ is\ Y \\ 0, & Otherwise \end{cases}$$

$f^{(i)}$ is the feature vector of i-th training data. λ is regularization parameter. We assign 1 to the regularization parameter. We used "RITE2_JA_dev_bc.xml" and "RITE2_JA_dev _exambc.xml" for training. These two files were distributed for NTCIR-11 RITE-VAL dry run.

2.3 Search

2.3.1 Features of FV

This section describes features of FV. These features is almost the same in respect of the system architecture shown in section 2. The point that this approach differs from Shallow Approach is that there are few features and use Cabocha instead of KNP as the base processing systems. We defined features referred to shallow approach by Tian et al[3]. Features are defined by:

1. Correspondence of named entity (include number expression). $f_1 = 1$ if every named entity in H as a synonym in T. Otherwise $f_1 = 0.1$.
2. Content Word Correspondence Rate $f_2 = \frac{\log(D_H+1)}{\log(L_H+1)}$, where L_H is the length of the content words list of H and D_H is the number of words in H that have found their synonyms in T.
3. Length of H. $f_3 = \log(L_h + 1)$.

Three ideas appeared in the feature design are worth mentioning:

1. We can be pretty sure that all named entity being included and the correspondence rate of content word were important for the positivity of the examples, and set up f_1 and f_2.
2. When H is long, it is difficult for f_2 become high, so we applied the number of words of H to f_3.
3. About f_2 and f_3, because of the length of a sentence is well applied to the log-normal distribution, so we take logarithm.

2.3.2 Search Strategy

We used Apache Solr as the base search system in the FV tasks. We created the search index of Wikipedia and textbook data, respectively, and merged each search results by Distributed Search of Solr. Moreover, we chose the highest-scoring entry as T among search results. We evaluated the following two measures, in order to decide the search strategy.

1. Unit of Search Index
   We assumed that the entry is appropriate as T which includes many search keywords and these keywords are near each other. However, Solr doesn't provide proximity measure which ignore word order. Therefore, for substitution of proximity search we chose the unit of index appropriately. It is suggested that making a search index by a paragraph unit is effective than a section and a subsection unit by Kano[5]. Therefore, we divide textbook into paragraphs by document structure, and Wikipedia into paragraphs by a newline.

2. Search Query
   We assumed that it is important that almost all named entity of H are included in T. Therefore we construct the search query which weights named entity 5 times.

3. MACHINE LEARNING

In the shallow approach, we apply the classification model with the extracted features as described above. We investigate the non-linearity of the classification functions by Bayesian Neural Networks [2].

$$p(y|x,w) = g(x,w)^y(1-g(x,w))^{1-y}$$
$$g(x,w) = \frac{1}{1+\exp(-f(x,w))}$$
$$f(x,w) = \sum_{j=1}^{J} a_j \tanh(\sum_{k=1}^{K} b_{jk}x_k + c_j) + d$$

For the criterion of the validity of the several network architectures, we use WAIC and WBIC [4]. WAIC is the estimate of the generalization error which is given by

$$p(w|x^n, y^n, \beta) \equiv \frac{\prod_{i=1}^n p(y_i|x_i,w)^\beta \varphi(w)}{\int \prod_{i=1}^n p(y_i|x_i,w)^\beta \varphi(w) dw}$$
$$WAIC = T_n + \frac{V_n}{n}$$
$$T_n = -\frac{1}{n}\sum_{i=1}^{n} \log E[p(y_i|x_i,w)]_{p(w|y^n,x^n,1)}$$
$$V_n = \sum_{i=1}^{n} V[\log p(y_i|x_i,w)]_{p(w|y^n,x^n,1)},$$

where, $E[f]_p$ and $V[f]_p$ are expectation and variance of the function f over the distribution p. Additionally, WBIC is the estimate of the free energy which is given by

$$WBIC = -E[\sum_{i=1}^{n} \log p(y_i|x_i,w)]_{p(w|y^n,x^n,\frac{1}{\log n})}.$$

We evaluate these values to the training data which merge dev-bc, testlabel-bc, dev-exambc and testlabel-exambc data by Markov chain Monte Carlo. The results are given by table1. These results show that WBIC prefers the linear model but WAIC prefers the 5-hidden units non-linear model. In this submission (NUL-04), we choose WAIC's non-linear model because of the known fact that WBIC is more conservative than WAIC. The validation results indicate that the non-linear model outperforms at accuracy but under perform at Macro-F1. This results suggest that we should carefully design the target of the objective function in such a subtle situation.

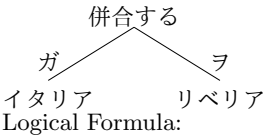
model	WAIC	WBIC	Accuracy	Macro F1
linear	0.4904	1079.7	77.67	68.87
5-hidden	0.4858	1111.4	77.81	68.73

Table 1: Model Comparison

4. DEEP APPROACH

In the deep approach, we refer to Inference-based Approach by Tian et al[2]. First, we convert dependency trees by KNP(syntax analysis system) to logical formula with some simple rule, and evaluate it. For this conversion, we apply Neo-Davidsonian Event Semantics[6]. We solve this logical formula with Prolog program. Each of the nodes is converted to facts, and hypernym/hyponym relations are converted to rules. We convert text T to facts and H to goal. In Figure 2, on the dependency tree, an edge between "イタリア"(Italy) and "併合する"(annex) is labeled case "ガ" by KNP as tag "格解析結果". If there is one edge labeled "ガ", this parent/child relation will be described as predicate "SBJ" stands for subject; If there is other label, as "ヲ" or "ニ", this relation will be described as predicate "OBJ" stands for object; And if there is not labeled by KNP tag "格解析結果", this relation will be described as predicate "ATT" stands for attribute.

Text:
イタリアは，リベリアを併合した。(Italy annexed Liberia.)
Syntax analysis:



Logical Formula:
イタリア (x) ∧ リベリア (y) ∧ 併合する (z) ∧ SBJ(z,x) ∧ OBJ(z,y)

Figure 2: Conversion of dependency trees

For this conversion, main operations are following four:

(1) Remove function words
From parsing result by KNP, we remove function words like auxiliary verb.

(2) Unite basic clauses
With Wikipedia, WordnetJP and parsing result by KNP, we unite basic clauses divided excessively. (We unite basic clauses with tag "文節内"or "Wikipedia エントリ". Moreover, we do longest suffix matching for other basic clauses.)

(3) Integration of synonyms
We get the ID of each words in the dependency trees from Wikipedia redirect and WordnetJP. If there are two words with the same meaning, these words have same ID. The words with the same ID would be described as the same things. In addition, we describe hypernym/hyponym relations. About this relations, we consider if clause A is hypernym of clause B, B is one of A. Therefore, we convert the relations to rules like "hypernym(X) :- hyponym(X)". However, if both "A is hypernym of B" and "B is hypernym of A"

are present, the rules would be cause an infinite loop. Therefore we delete the one of them.

(4) Change predicates or structure of tree
As mentioned earlier, we describe parent/child relation with some predicates. However, there are relationship as passive or parallel relation. When we choice the predicates from KNP tag "格解析結果", these relations may be unable to be described. Therefore, we change some predicates. If a parent node labeled by KNP tag "態:受動" and an edge to child node labeled case "ヲ", this relation will be described as predicate "SBJ" instead of "OBJ", and labeled "ガ" it will be described as predicate "OBJ". When there is parallel relation, if necessary, we add new edges on dependency tree.

However, this approach has a problem that text H include many words not in text T. In such a case, our answer will be always False. On formal SV run, as expected, Deep Approach answered to False on most of problems. Therefore we rejected these answers.

For reference, result of UnitTest provided as training data is shown at table2, table3. For the above reasons, "disagree" category has been solved well. In "modifier" or "list" category, we achieved good results. "phrase" expect "disagree:phrase" was not a good score. From the results, deep approach will be possible to get a good result at specific categories.

Category	Frequency	Accuracy
disagree:lex	5	100.00
disagree:modality	2	100.00
disagree:spatial	1	100.00
disagree:phrase	25	96.00
modifier	30	66.67
list	11	63.64
relative_clause	10	40.00
synonymy:lex	10	40.00
scrambling	16	37.50
clause	29	34.48
apposition	3	33.33
case_alternation	9	33.33
entailment:phrase	28	7.14
synonymy:phrase	45	4.44
coreference	12	0.00
entailment:lex	1	0.00
hypernymy:lex	6	0.00
hypernymy:phrase	3	0.00
implicit_relation	10	0.00
inference	4	0.00
meronymy:lex	1	0.00
nominalization	2	0.00
quantity	1	0.00
spatial	4	0.00
temporal	2	0.00
transparent_head	2	0.00
Total	272	33.46

Table 2: Result of UnitTest(Training data)

5. EXPERIMENTAL RESULTS

Y-F1	N-F1	Y-Prec.	Y-Rec.	N-Prec.	N-Rec.
39.46	26.12	98.33	24.69	15.09	96.97

Table 3: Macro F1, Precision, Recall of UnitTest(Training data)

Our highest submission score are shown at table4 and table5.

id	accuracy	Marco F1	Y-F1	N-F1
NUL-JA-FV-03	63.23	61.93	54.89	68.97

Table 4: Best result of FV task

id	accuracy	Marco F1	Y-F1	N-F1
NUL-JA-SV-04	77.81	69.59	53.78	85.40

Table 5: Best result of SV task

6. DISCUSSION

6.1 Relevant Features

For the examination of the relevance of proposed features, we performed replication study. As a result of that, we found effective features are following (see section 2.2).

- f_1 : "Number Expression Correspondence"
- f_2 : "Named Entity Correspondence"
- f_3 : "Content Word Correspondence Rate"

Therefore, we examined mean difference of Micro-F1 score between Case0 and Case1-Case4. Each case is defined as follow.

- Case 0 : Use all $(1, f_1, f_2, \ldots, f_7)$ features.
- Case 1 : Use f_1, f_2 features
- Case 2 : Use f_1, f_3 features
- Case 3 : Use f_2, f_3 features
- Case 4 : Use f_1, f_2, f_3 features

In this examination, we used "RITE2_JA_dev_bc.xml", "RITE2 _JA_dev_exambc.xml" and "RITE-VAL_JA_testlabel _system-val.xml" as data source, divide them into equal size training set and test set randomly. Then, adjust hypothesis parameter by training set and calculate macro F1 by test set for each cases. We repeat these operation 100 times, examined mean difference between Case0 and others by t-test. We use R version 3.1.0 to calculate p-value. Our examination result is table 6.

In this result, p-values of Case1-3 are very small value, it suggests that Case1-3's mean of macro F1 aren't same as Case0. Moreover, Case1-3's macro F1 mean are smaller than Case0, it indicates Case1-4 lack some features. On the other hand, p-value of Case4 is sufficiently large to decide Case4's mean of macro F1 is same as Case0. Hence, dominant features of our system are f_1, f_2, f_3

Case	Mean of Macro F1	p-value
Case 0	0.726	–
Case 1	0.699	$> 2.2 \times 10^{-16}$
Case 2	0.706	$> 2.2 \times 10^{-16}$
Case 3	0.707	$> 2.2 \times 10^{-16}$
Case 4	0.727	0.4021

Table 6: Feature Comparison

6.2 Search methods

We applied two search methods shown in section 2.3 to FV subtasks datasets of dev and testlabel. Experimental results of each methods are following:

1. Unit of Search Index
   For both datasets of dev and testlabel, it is very effective that making a search index by a paragraph unit. It illustrates that a paragraph is appropriate unit of search index, and the proximity between search keywords is valid for weighting.

2. Search Query
   The search query which weights named entity 5 times was effective for FV dev datasets, however, it was not good for FV test datasets. It illustrates that if the word groups of named entity is stable, the search query will obtain good results to some extent.

7. CONCLUSIONS

In this paper, we described approach of NUL System at NTCIR RITE-VAL tasks. On SV and FV tasks, our system brought good results. In FV-JA task, our result was the best though we searched for documents related to each H text from textbooks without using retrieved documents. Considering the results, operations described above (especially search strategy) contributed to the result on this search. In SV-JA task, our result was also the best. To solve this RTE task, we were able to obtain good results in simpler way. If the quality of extracted Named Entity becomes good, it was suggested that better results can be achieved.

8. REFERENCES

[1] S. Matsuyoshi, Y. Miyao, T. Shibata, C.-J. Lin, C.-W. Shih, Y. Watanabe, and T. Mitamura. Overview of the NTCIR-11 Recognizing Inference in TExt and Validation (RITE-VAL) Task. In *Proceedings of the 11th NTCIR Conference*, 2014.

[2] R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, 1996.

[3] T. M. Ran Tian, Yusuke Miyao and H. Komatsu. Bno at ntcir-10 rite: A strong shallow approach and inference-based textual entailment recognition system. In *Proceedings of NTCIR-10*, 2013.

[4] S. Watanabe. *Algebraic Geometry and statistical Learning Theory*. Cambridge university press, 2009.

[5] 狩野芳伸. 大学入試センター試験歴史問題の自動解答. In **人口知能学会誌**, 2014.

[6] 稲田和明. 日本語文における論理式変換の実装と課題調査. In *Bachelor's thesis, Tohoku University, Department of Information and Intelligent Systems*, 2013.

APPENDIX

A. SQL QUERY

We describe the concrete example of sql query for the f_6 at the section 2.2.6.

```
select * from synset where synset not in (
 select synset2 from ancestor
  where synset1='04916342-')
 and synset<>'04916342-n'
 and synset in (
  select synset2 from ancestor
   where  synset1 in (
   select synset2 from synlink
    where link='attr' and synset1 in (
    select synset from synset
     where pos='a' and synset in (
     select synset from sense
      where wordid in (
      select wordid from word
       where lemma= "Content Word" ))
    union
    select synset2 from synlink
     where link='sim' and synset1 in (
     select synset from synset
      where pos='a' and synset in (
      select synset from sense
       where wordid in (
       select wordid from word
        where lemma= "Content Word" )))))
  union
  select synset2 from synlink
   where link='attr' and synset1 in (
   select synset from synset
    where pos='a' and synset in (
    select synset from sense
     where wordid in (
     select wordid from word
      where lemma= "Content Word" ))
  union
  select synset2 from synlink
   where link='sim' and synset1 in (
   select synset from synset
    where pos='a' and synset in (
    select synset from sense
     where wordid in (
     select wordid from word
      where lemma= "Content Word" ))))))
```