Team SKL's Strategy and Experience in RITE2

Shohei Hattori Satoshi Sato

Graduate School of Engneering, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8603, JAPAN
syohei_h@nuee.nagoya-u.ac.jp, ssato@nuee.nagoya-u.ac.jp

ABSTRACT

This paper describes the strategies and systems of the team SKL in the RITE2 workshop. We implemented three different systems. SKL-01 was designed by two-step classification strategy. Step1 assigns a *default* class to a given text pair by applying a *simple* rule based on an overlap measure; Step2 examines the necessity of *overwriting* the default class by applying heuristic rules. In contrast, SKL-02/03 were designed by a different strategy, which focused on the development of character-based features. SKL-02 is a SVM-based system using these features, and SKL-03 works by hand-tuned decision rules. In the MC subtask of the formal run, our all three systems ranked the top three.

Team Name

SKL

Subtasks

BC, MC, ExamBC (Japanese)

Keywords

two-step classification strategy, overwriting rules, character-based features

1. INTRODUCTION

RITE2 [3] is an evaluation-based workshop aiming to recognize entailment, paraphrase, and contradiction between sentences. Because we had no experience in participating in such workshop, we had no concrete plan when we decided to participate in RITE2 in June 2012.

When the development set was distributed in August, we noticed that the BC subtask of this year is much easier than it of the previous RITE [2] in NTCIR-9. We first examined the classification performance of the character overlap ratio, which will be described later, and found the accuracy is around 80%.

Based on the fact, we determined the following *two-step classification strategy*.

Step1 assigns a *default* class to a given text pair by applying a *simple* rule based on an overlap measure.

Step2 examines necessity of *overwriting* the default class by applying heuristic rules.

Because Step1 archives good performance, Step2 should be carefully applied not to decrease the performance. The SKL-01 was implemented according to this strategy by the first author.

		Surface similarity	
		High	Low
Semantic	High	Case 1 (H/H)	Case 2 (H/L)
similarity	Low	Case 3 (L/H)	Case 4 (L/L)

Figure 1: 2×2 matrix

In contrast, the SKL-02/03, which were implemented by the second author to fill two more runs, were designed by a different strategy, which focused on the development of character-based features. Because they are superb in stability and light calculation, they should be explored exhaustively before moving to exploration of semantically-rich features. Improvement of the classification based on character-based features will simplify the next stage of semantic processing.

Our system development targeted the BC and MC subtasks. Two days before our submission of the formal run, we decided to implement subsystems for the ExamBC subtask, by just adapting the BC subsystems into the ExamBC subtask. The confidence score required in the ExamBC subtask was not seriously examined.

The rest of this paper is organized as follows. Section 2 describes the strategy and details of the SKL-01. Section 3 describes the features and the classifiers used in the SKL-02/03. Section 4 reports our experimental result in the development stage and the formal run.

2. SKL-01

2.1 Strategy

When two sentences have similar meanings, they tend to be similar in appearance. It is natural and well-known that *semantic similarity* and *surface similarity* are correlated.

Figure 1 shows the 2×2 matrix from which we started. In this matrix, Case 1 and 4 correspond to natural cases.

Case 2 corresponds to the case that two sentences look dissimilar but have similar meanings. This happens, for example, when one is a paraphrase of the other.

Case 3 corresponds to the case that two sentences look similar but have the different meanings. This happens, for example, when the small difference is crucial in meaning.

There are several ways to estimate *surface similarity* between two sentences. By combining a similarity function and a threshold value, we can distinguish Case 1/3 from Case 2/4.

In addition, we try to distinguish Case 3 from Case 1 by examining whether the surface difference between two sentences is crucial or not in meaning. By our intuition, separation of Case 3 from Case 1 is much easier than separation of Case 2 from Case 4.

2.2 Generalized Overlap Ratio

First of all, we define two generalized functions of *overlap ratio*. We assume that a set of entities E is given, where entities are, for example, characters, character n-grams, words, and word n-grams. Under this assumption, we define the following function that calculates how many number of E's entities are overlapped between two strings $\mathbf{s}_1$ and $\mathbf{s}_2$.

$$overlap(E; \mathbf{s}_1, \mathbf{s}_2) = \sum_{x \in E} \min(fr(x, \mathbf{s}_1), fr(x, \mathbf{s}_2)) \qquad (1)$$

where a function $fr(x, \mathbf{s})$ calculates the frequency of an entity x in a string $\mathbf{s}$.

By using this function, we define two functions of *overlap ratio* as follows.

$$overlap\_ratio_B(E; \mathbf{s}_1, \mathbf{s}_2) = \frac{2\, overlap(E; \mathbf{s}_1, \mathbf{s}_2)}{\sum_{x \in E} fr(x, \mathbf{s}_1) + \sum_{x \in E} fr(x, \mathbf{s}_2)} \qquad (2)$$

$$overlap\_ratio_D(E; \mathbf{s}_1, \mathbf{s}_2) = \frac{overlap(E; \mathbf{s}_1, \mathbf{s}_2)}{\sum_{x \in E} fr(x, \mathbf{s}_2)} \qquad (3)$$

The first function $overlap\_ratio_B$ is *bidirectional*, i.e.,

$$overlap\_ratio_B(E; \mathbf{s}_1, \mathbf{s}_2) = overlap\_ratio_B(E; \mathbf{s}_2, \mathbf{s}_1) \quad (4)$$

In contrast, $overlap\_ratio_D$ is *directional* because it is normalized by the length of $\mathbf{s}_2$. We use the latter for detecting entailment because entailment is a directional relation. The former is used for detecting contradiction, which is a bidirectional relation.

By using the *generalized* function (3), we can define several different functions by changing E. For example, by using C, the set of all characters used in Japanese texts, we can define the *character overlap ratio* (cor_D) as follows.

$$cor_D(\mathbf{s}_1, \mathbf{s}_2) = overlap\_ratio_D(C; \mathbf{s}_1, \mathbf{s}_2) \qquad (5)$$

This is the same as the function used in the RITE1's baseline method [2], written in a different formalization. Similarly, the *overlap ratio of character bigrams* (bor_D) is defined as follows.

$$bor_D(\mathbf{s}_1, \mathbf{s}_2) = overlap\_ratio_D(C^2; \mathbf{s}_1, \mathbf{s}_2) \qquad (6)$$

2.3 Surface Similarity

In order to find the best similarity function, we examined nine candidates; each of which uses one of the following sets as E in Equation (3):

$$C,\ C^2,\ C^3,\ W,\ W^2,\ W^3,\ L,\ L^2,\ L^3$$

where C is the set of all characters, W is the set of all words in surface form, and L is the set of all words in normalized (lemmatized) form.

Table 1 shows our experimental result of the binary classification (Y: $\mathbf{s}_1$ entails $\mathbf{s}_2$, N: otherwise), where we used both the BC and MC development sets, shown in Table 2. Clearly, the simple character overlap ratio defined in Equation (5) is the best among nine candidates. Based on the result, we decided to use the character overlap ratio to estimate surface similarity.

Table 1: Performance of overlap ratios

E	th.	Y	N	acc.
C	0.69	499/613	472/629	**78.3**
L	0.66	468/613	471/629	75.6
W	0.64	452/613	477/629	74.8
C^2	0.45	484/613	472/629	77.1
L^2	0.35	413/613	487/629	72.5
W^2	0.29	443/613	447/629	71.7
C^3	0.35	410/613	490/629	72.5
L^3	0.19	365/613	472/629	67.4
W^3	0.18	355/613	480/629	67.2

th.: the best threshold, acc.: classification accuracy

Table 2: How to use development sets

task	BC		MC					#
label	Y	N	B		F	C	I	of
#	240	371	83		207	65	193	ins.
direction	f	f	f	b	f	f	f	
Y	✓		✓	✓	✓			613
N		✓				✓	✓	629

f: forward, b: backward

2.4 Overwriting Rules

Next, we proceeded to the development of *overwriting* rules. Careful observation of the BC development set revealed that mismatch of named entities (proper nouns) and numbers causes semantic dissimilarity. Based on this observation, we have implemented two functions: $NE\_mismatch$ and $Num\_mismach$.

$NE\_mismatch$ detects mismatch of named entities of a sentence pair $\langle \mathbf{s}_1, \mathbf{s}_2 \rangle$. When $\mathbf{s}_2$ includes a named entity that does not appear in $\mathbf{s}_1$, the function returns true; otherwise false. In order to detect named entities in sentences, we use JUMAN, a morphological analyzer. When a morpheme (word) whose syntactic category is proper noun or which has the feature of "automatically extracted from Wikipedia"[1], we detect it as a named entity.

Matching of two named entities is not straightforward because of spelling variants, especially when they are written in Katakana. We use a simple heuristic to bridge two different Katakana spellings of the same named entity.

$Num\_mismatch$ is a similar function that focuses on numbers in sentences. Detection of the numbers in sentences is much easier than that of named entities, but still several heuristics are necessary to handle spelling variants, such as Kanji numerals and Arabic numerals.

2.5 BC-01 Subsystem

Figure 2 shows the pseudo-code of the BC-01 subsystem. In this code, the *default* rule is enhanced by using a function kor_D, in order to improve the classification accuracy of sentence pairs around the threshold ($0.73 > cor_D > 0.65$). The function kor_D is another overlap ratio, defined as follows.

$$kor_D(\mathbf{s}_1, \mathbf{s}_2) = overlap\_ratio_D(K; \mathbf{s}_1, \mathbf{s}_2) \qquad (7)$$

where K is the union of Kanji characters and Katakana characters. Because Kanji and Katakana characters are used to

[1] This feature is not included in the data provided by RITE2 organizer.

```
def BC-01($\mathbf{s}_1, \mathbf{s}_2$)
  if (($cor_D(\mathbf{s}_1, \mathbf{s}_2) \geq 0.73$) or
      ($kor_D(\mathbf{s}_1, \mathbf{s}_2) > cor_D(\mathbf{s}_1, \mathbf{s}_2) \geq 0.69$) or
      (($0.69 > cor_D(\mathbf{s}_1, \mathbf{s}_2) > 0.65$) and
       ($kor_D(\mathbf{s}_1, \mathbf{s}_2) - 0.1 > cor_D(\mathbf{s}_1, \mathbf{s}_2)$))))
    if ($NE\_mismatch(\mathbf{s}_1, \mathbf{s}_2)$ or $Num\_mismatch(\mathbf{s}_1, \mathbf{s}_2)$)
      return $\mathsf{N}_{11}$
    else
      return $\mathsf{Y}_{12}$
    end
  else
    return $\mathsf{N}_2$
  end
end
```

Figure 2: BC-01 subsystem

```
def MC-01($\mathbf{s}_1, \mathbf{s}_2$)
  if contradict($\mathbf{s}_1, \mathbf{s}_2$)      then return $\mathsf{C}_1$
  elsif BC-01($\mathbf{s}_1, \mathbf{s}_2$) = $\mathsf{Y}$
    if BC-01($\mathbf{s}_2, \mathbf{s}_1$) = $\mathsf{Y}$ then return $\mathsf{B}_{21}$
    else                             return $\mathsf{F}_{22}$
    end
  else                               return $\mathsf{I}_{23}$
  end
end

def contradict($\mathbf{s}_1, \mathbf{s}_2$)
  (($cor_D(\mathbf{s}_1, \mathbf{s}_2) \geq 0.69$) and
   ($bor_D(\mathbf{s}_1, \mathbf{s}_2) > 0.65$) and
   $Num\_mismatch(\mathbf{s}_1, \mathbf{s}_2)$))              # cond$_1$
  or
  (($0.69 > cor_D(\mathbf{s}_1, \mathbf{s}_2) > 0.59$) and
   $overlap\_ratio_B(C, \mathbf{s}_1, \mathbf{s}_2) \geq 0.5$))  # cond$_2$
  or
  (($0.69 > cor_D(\mathbf{s}_1, \mathbf{s}_2) > 0.5$) and
   $ht\_ratio(\mathbf{s}_1, \mathbf{s}_2) \geq 0.4$))          # cond$_3$
  or
  (($cor_D(\mathbf{s}_1, \mathbf{s}_2) > 0.69$) and
   $ht\_ratio(\mathbf{s}_1, \mathbf{s}_2) \geq 0.75$))         # cond$_4$
end
```

Figure 3: MC-01 subsystem

represent content words in general, kor_D focuses on the content part of two sentences, compared with cor_D.

2.6 MC-01 Subsystem

The MC-01 subsystem is implemented by combining the BC-01 subsystem and additional conditions that detect contradiction. Figure 3 shows the pseudo-code of the MC-01 subsystem. The toplevel of the structure is straightforward.

There are four conditions that detect contradiction, which were implemented based on the observation of the development set. In all conditions, the character overlap ratio (cor_D) is used. In the first condition, the ratio of character bigrams (bor_D) and mismatch of numbers ($Num\_mismatch$) are combined. In the second condition, the bidirectional character overlap ratio ($overlap\_ratio_B$) is combined. The third and fourth conditions use a new function ht_ratio, which is another estimation of surface similarity. We will explain it later, in Section 3.1.3.

```
def ExamBC-01($\mathbf{s}_1, \mathbf{s}_2$)
  if (($cor_D(\mathbf{s}_1, \mathbf{s}_2) \geq 0.73$) or
      ($kor_D(\mathbf{s}_1, \mathbf{s}_2) > cor_D(\mathbf{s}_1, \mathbf{s}_2) \geq 0.69$))
    if ($NE\_mismatch(\mathbf{s}_1, \mathbf{s}_2)$ or $Num\_mismatch(\mathbf{s}_1, \mathbf{s}_2)$)
      return [$\mathsf{N}_{11a}$, 1]
    else
      return [$\mathsf{Y}_{12a}$, $cor_D(\mathbf{s}_1, \mathbf{s}_2)$]
    end
  elsif (($0.69 > cor_D(\mathbf{s}_1, \mathbf{s}_2) > 0.65$) and
         ($kor_D(\mathbf{s}_1, \mathbf{s}_2) - 0.1 > cor_D(\mathbf{s}_1, \mathbf{s}_2)$)))
    if ($NE\_mismatch(\mathbf{s}_1, \mathbf{s}_2)$ or $Num\_mismatch(\mathbf{s}_1, \mathbf{s}_2)$)
      return [$\mathsf{N}_{11b}$, $1 - cor_D(\mathbf{s}_1, \mathbf{s}_2)$]
    else
      return [$\mathsf{Y}_{12b}$, $cor_D(\mathbf{s}_1, \mathbf{s}_2) + 0.1$]
    end
  else
    return [$\mathsf{N}_2$, $1 - cor_D(\mathbf{s}_1, \mathbf{s}_2)$]
  end
end
```

Figure 4: ExamBC-01 subsystem

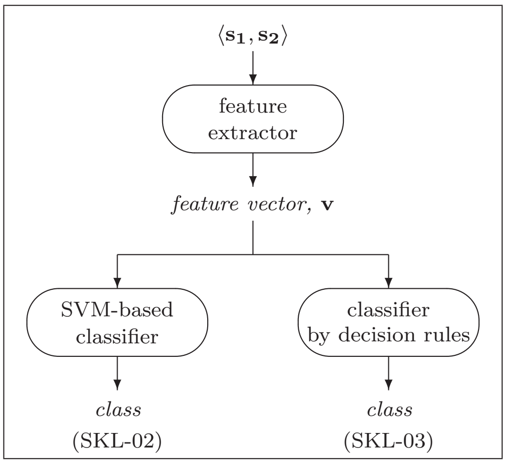

Figure 5: Architecture of SKL-02/03

2.7 ExamBC-01 Subtask

The ExamBC-01 subsystem is the BC-01 subsystem with confidence score estimation, shown in Figure 4. Classification rules are the same but each of N_{11} and Y_{12} is divided into two sub-rules because of different assignment of the confidence score. We just implemented the confidence-score estimation by intuition and did not seriously examined it.

3. SKL-02 AND SKL-03

As mentioned earlier, the development of the SKL-02/03 focused on the exploration of character-based features. Figure 5 shows the architecture of the SKL-02/03. Each subsystem (i.e., SKL-*task*-(02/03)) consists of two modules, *feature extractor* and *classifier*. The former is shared by all subsystems. In contrast, the latter is independent.

3.1 Features

All features used in the SKL-02/03 are character-based; they are extracted directly from a given pair $\langle \mathbf{s}_1, \mathbf{s}_2 \rangle$ with no extra linguistic resources and tools.

Table 3 overviews the features produced by the feature ex-

Table 3: Features extracted from $\langle \mathbf{s}_1, \mathbf{s}_2 \rangle$

	02	03						
ID		BC	MC	EBC	class	description	type	range
$x0$					C_1	$overlap_B(C_1; \mathbf{s}_1, \mathbf{s}_2)$	int.	$0 \leq x0$
$x1$			√		C_1	$\lvert c_1(\mathbf{s}_1)\rvert + \lvert c_1(\mathbf{s}_2)\rvert - 2\,x0$	int.	$0 \leq x1$
$x2$			√		C_2	$overlap_B(C_2; \mathbf{s}_1, \mathbf{s}_2)$	int.	$0 \leq x2$
$x3$			√		C_2	$\lvert c_2(\mathbf{s}_1)\rvert + \lvert c_2(\mathbf{s}_2)\rvert - 2\,x2$	int.	$0 \leq x3$
$x4$	√		√		C_1	$f15 + b15$	int.	
$x5$	√				C_1	$f21 + r21$	int.	$0 \leq x5 \leq 20$
$x6$	√				C_1	$\lvert \mathbf{t} \rvert$	int.	$0 \leq x6$
$x7$	√				C_1	$\lvert shift(\mathbf{s}_2, \mathbf{u}_2) - shift(\mathbf{s}_1, \mathbf{u}_1)\rvert$	real	$0 \leq x7$
$x8$	√				C_1	$\max(\lvert shift(\mathbf{s}_2, \mathbf{u}_2)\rvert, \lvert shift(\mathbf{s}_1, \mathbf{u}_1)\rvert$	real	$0 \leq x8$
$f0$	√		√		C_1	$\lvert c_1(\mathbf{s}_2)\rvert$	int.	$0 < f0$
$f1$	√		√		C_1	$\lvert c_1(\mathbf{s}_2)\rvert / \lvert c_2(\mathbf{s}_1)\rvert$	real	$0 < f1$
$f2$	√		√		C_1	$f0 - x0$	int.	$0 \leq f2$
$f3$	√	√	√	√	C_1	$overlap\_ratio_D(C_1; \mathbf{s}_1, \mathbf{s}_2) = x0/f0$	real	$0 \leq f3 \leq 1$
$f4$	√		√		C_1	$(f2 + b2)/\lvert c_1(\mathbf{s}_2)\rvert$	real	$0 \leq f4$
$f5$	√		√		C_2	$\lvert c_2(\mathbf{s}_2)\rvert$	int.	$0 < f5$
$f6$	√		√		C_2	$\lvert c_2(\mathbf{s}_2)\rvert / \lvert c_2(\mathbf{s}_1)\rvert$	real	$0 < f6$
$f7$	√	√	√		C_2	$f5 - x2$	int.	$0 \leq f7$
$f8$	√		√		C_2	$overlap\_ratio_D(C_2; \mathbf{s}_1, \mathbf{s}_2) = x2/f5$	real	$0 \leq f8 \leq 1$
$f9$	√		√		C_2	$(f2 + b2)/\lvert c_2(\mathbf{s}_2)\rvert$	real	$0 \leq f9$
$f10$				√		$f8 - f3$	real	
$f11$	√	√			C_1	head-character difference	binary	0 or 1
$f12$	√	√	√		C_1	# of Arabic numerals in $\mathbf{u}_2$ (max. 5)	int.	$0 \leq f12 \leq 5$
$f13$	√	√			C_1	# of Roman alphabets in $\mathbf{u}_2$ (max. 5)	int.	$0 \leq f13 \leq 5$
$f14$	√	√	√		C_1	# of Kanji numerals in $\mathbf{u}_2$ (max. 5)	int.	$0 \leq f14 \leq 5$
$f15$	√		√		C_1	$\delta(\mathbf{s}_1, \mathbf{s}_2)$	int.	
$f20$	√	√			C_1	$umc(5, \mathbf{u}_2)$	int.	$0 \leq f20 \leq 5$

$f16$–$f19$ and $f21$–$f23$ are omitted because they are not used by the final version of classifiers.

tractor. A feature vector $\mathbf{v}$ consists of three parts: (1) bidirectional features ($x0$–$x8$), (2) forward features ($f0$–$f23$), and (3) backward features ($b0$–$b23$). Backward features are not shown in Table 3, because they are the same as the forward features calculated from the reverse pair $\langle \mathbf{s}_2, \mathbf{s}_1 \rangle$. From a feature vector $\mathbf{v}$, the feature vector of the reversed pair can be easily produced by swapping the last two parts.

Behind these features, three new operations exist: class-based string reduction, generation of masked strings, and string decomposition into three parts.

3.1.1 Character Class and String Reduction

The Japanese language has several different types of characters. Hiragana, Katakana, and Kanji are three major types but other character types such as Roman alphabet and Arabic numerals are also used. In addition, a dozen symbols such as punctuation marks and quotation marks are included in texts.

By convention, each character type plays a particular role in Japanese writing. Kanji is used to write *content words*. Katakana is used to write *imported words* from other languages. Hiragana is mainly used to write *functional words*. Therefore an average impact per character to content information varies according to character types: Kanji and Katakana are high; Hiragana is low.

To take this into account, we introduce *character class*. In practice, we define three classes. Class0 (the same as C in the previous section) includes all characters. Class1 (C_1) excludes punctuation marks, which bring no content information. Class2 (C_2, which is the same as K in the previous section) consists of only Katakana and Kanji characters, which are primal types of characters that bring content information.

original string	$\mathbf{s} = c_0(\mathbf{s}) =$ 飲むヨーグルトは、酒の一種だ。
class1 string	$c_1(\mathbf{s}) =$ 飲むヨーグルトは酒の一種だ
class2 string	$c_2(\mathbf{s}) =$ 飲ヨーグルト酒一種

Figure 6: Class-based string reduction

Class-based string reduction is the operation that removes all characters that are not included in a given character class. For example, Class1-reduction (function c_1) removes all punctuation marks from a given string; Class2-reduction (function c_2) removes all characters except Kanji and Katakana. We call the produced string by class$N$-reduction *class$N$-string*. Figure 6 shows an example of class-based string reduction.

3.1.2 Masked String

A masked string is produced by replacing some characters into a masked symbol (for which we practically use “.”). It was first introduced for debugging.

Figure 7 shows examples of masked strings. First, from a given pair $\langle \mathbf{s}_1, \mathbf{s}_2 \rangle$, we calculate the overlapped characters. Next, we produced two types of masked strings: $\mathbf{u}_i$ is produced from $\mathbf{s}_i$ by masking the overlapped characters (in other words, characters that are not overlapped remain); $\mathbf{o}_i$ is produced from $\mathbf{s}_i$ by masking the characters that are not overlapped. From these masked strings, we easily know what part of a string is common and different between two strings.

$\mathbf{s}_1$	プロメーテウスは人類に火を渡し張り付けにされた
$\mathbf{s}_2$	プロメテウスは人類に火を齎して罰を受けた
$\mathbf{u}_1$	... ー......... 渡. 張り付. にされ.
$\mathbf{u}_2$	.... 齎. て罰を受..
$\mathbf{o}_1$	プロメ. テウスは人類に火を. し... け... た
$\mathbf{o}_2$	プロメテウスは人類に火を.. し.... けた

Figure 7: Masked strings

	$\mathbf{h}$	$\mathbf{b}_1$	$\mathbf{t}$
$\mathbf{s}_1$	プロメ	ーテウスは人類に火を渡し張り付けにされ	た
$\mathbf{s}_2$	プロメ	テウスは人類に火を齎して罰を受け	た
	$\mathbf{h}$	$\mathbf{b}_2$	$\mathbf{t}$

Figure 8: String decomposition

The value of the feature $f11$ (*head-character difference*) is calculated by using $\mathbf{u}_2$. If the first character (ignoring quotation marks) of $\mathbf{u}_2$ is the masked symbol, then the value is 0; otherwise 1.

The values of $f12$, $f13$, and $f14$ are calculated by counting particular types of characters in $\mathbf{u}_2$: Arabic numerals ($f12$), Roman alphabets ($f13$), and Kanji numerals ($f14$).

The features $f20$ focuses on the last part of $\mathbf{u}_2$. The value is the number of unmasked characters (*umc*) in the last five characters in $\mathbf{u}_2$. This feature was introduced because a predicate (a verb optionally with tense and negation) is located in the last part of a sentence in the Japanese language.

The center of a string is defined as a half of the length. Usually, the masked characters are not distributed uniformly, so the center of the masked string shifts to left or right. A function *shift* calculates this shift, as follows,

$$shift(\mathbf{s}, \mathbf{u}) = \frac{1}{n} \sum_{p=1}^{|\mathbf{s}|} d(\frac{|\mathbf{s}|}{2}, p) \tag{8}$$

$$d(c, p) = \begin{cases} 0 & \text{if } p\text{-th character of } \mathbf{u} \text{ is masked} \\ p - c & \text{otherwise} \end{cases} \tag{9}$$

where n is the number of unmasked characters in $\mathbf{u}$. This function is used to calculate two features, $x7$ and $x8$.

3.1.3 String Decomposition into Three Parts

The last operation is string decomposition into three parts. Figure 8 illustrates this operation. For a given pair $\langle \mathbf{s}_1, \mathbf{s}_2 \rangle$, the longest common prefix ($\mathbf{h}$) and the longest common suffix ($\mathbf{t}$) are determined. By using these substrings, each string decomposes into three parts,

$$\mathbf{s}_1 = \mathbf{h} + \mathbf{b}_1 + \mathbf{t} \tag{10}$$

$$\mathbf{s}_2 = \mathbf{h} + \mathbf{b}_2 + \mathbf{t} \tag{11}$$

where $\mathbf{b}_1$ and $\mathbf{b}_2$ are the different middle parts.

The length of the longest common suffix (i.e., $|\mathbf{t}|$) is used as the value of the feature $x6$. In addition, the feature $f15$ is calculated from this decomposition via a function δ, which is defined as follows.

$$\delta(\mathbf{s}_1, \mathbf{s}_2) = |\mathbf{h}| + |\mathbf{t}| - |\mathbf{b}_2| \tag{12}$$

The function *ht_ratio* in the previous section is also calculated from this decomposition. The definition of *ht_ratio* is as follows.

$$ht\_ratio(\mathbf{s}_1, \mathbf{s}_2) = \frac{2\,(|\mathbf{h}| + |\mathbf{t}|)}{|\mathbf{s}_1| + |\mathbf{s}_2|} \tag{13}$$

```
def MC-02($\mathbf{v}$)
  if SVM-BC($\mathbf{v}$) $> 0$
    if SVM-BC(backward($\mathbf{v}$)) $> 0$ then return $\mathsf{B}_{11}$
    elsif SVM-FC($\mathbf{v}$) $> 0$           then return $\mathsf{F}_{12}$
    else                                    return $\mathsf{C}_{13}$
    end
  else
    if (SVM-nI($\mathbf{v}$) $> 0$ or
        SVM-nI(backward($\mathbf{v}$)) $> 0$) then return $\mathsf{C}_{21}$
    elsif SVM-C($\mathbf{v}$) $> 0$            then return $\mathsf{C}_{22}$
    else                                    return $\mathsf{I}_{23}$
    end
  end
end
```

Figure 9: MC-02 subsystem

3.2 SKL-02 (SVM)

The SKL-02 is a SVM-based system. We have developed six SVMs in total, shown in Table 4. Each line shows the training configuration of a SVM, i.e., what part of the development sets are used as positive examples (+1) and negative examples (-1) in training. We have used the liblinear software package [1] with the default setting. Note that all feature values are normalized into the range between zero to one, in preprocessing.

Each classifier of two binary classification subtasks is a single SVM trained by the development set. This is a straightforward implementation.

In contrast, the MC-02 subsystem uses four SVMs shown in Figure 9. The point of the MC subtask is how to recognize contradiction. For this purpose, we have introduced three SVMs (SVM-FC, SVM-nI, SVM-C) to separate contradiction from other classes, in addition to the SVM-BC, which recognizes entailment from $\mathbf{s}_1$ to $\mathbf{s}_2$.

The confidence score, which is required in the ExamBC subsystem, was not seriously investigated. We just used the following equation.

$$c\_score(\mathbf{s}_1, \mathbf{s}_2; th) = |overlap\_ratio_D(C_1; \mathbf{s}_1, \mathbf{s}_2) - th| \tag{14}$$

where C_1 is the set of Class1 characters, and th is a threshold value, for which we used 0.70. This definition was also used in the ExamBC-03 subsystem.

3.3 SKL-03 (DR)

The SKL-03 uses decision rules for classification, which have been tuned manually. The pseudo-code of the BC-03 subsystem is shown in Figure 10; it of the ExamBC-03 subsystem is shown in Figure 11. In both subsystems, the primal feature is $f3$. In addition, the BC-03 subsystem uses six secondary features, $f11$, $f12$, $f13$, $f14$, $f7$, and $f20$. In contract, the ExamBC-03 subsystem is much simpler; it uses only one secondary feature, $f10$.

The MC-03 subsystem is much more complicated, shown in Figure 12. It calls the BC-03 subsystem as a subroutine, with a different threshold value (0.66). Another subroutine, `count` counts the number of conditions that are satisfied. The list of the 16 conditions (which are omitted in this pa-

Table 4: Six SVMs of SKL-02

	task	BC		MC								ExamBC		# of ins.	
	label	Y	N	B		F		I		C		Y	N	+1	−1
	#	240	371	83		207		193		65		210	330		
	direction	f	f	f	b	f	b	f	b	f	b	f	f		
BC-02	SVM-Y	+1	−1											240	371
MC-02	SVM-BC	+1	−1	+1	+1	+1	−1	−1	−1	−1	−1			613	1094
	SVM-FC			+1	+1	+1				−1	−1			372	130
	SVM-nI			+1	+1	+1	−1	−1	−1	−1	−1			373	723
	SVM-C							−1	−1	+1	+1			130	386
ExamBC-02	SVM-Ex											+1	−1	210	330

f: forward, b: backward

```
def BC-03($\mathbf{v}$, $th := 0.718$)
  if ($\mathbf{v}.f3 > th$ and
      $\mathbf{v}.f11 = 0$ and $\mathbf{v}.f12 < 2$ and $\mathbf{v}.f13 < 2$ and $\mathbf{v}.f14 < 2$)
    if $\mathbf{v}.f7 \geq 7$      then return $\mathsf{N}_{11}$
    elsif $\mathbf{v}.f20 = 0$ then return $\mathsf{N}_{12}$
    else                    return $\mathsf{Y}_{13}$
    end
  else                      return $\mathsf{N}_{2}$
  end
end
```

Figure 10: BC-03 subsystem

```
def ExamBC-03($\mathbf{v}$)
  $cs := c\_score(\mathbf{s}_1, \mathbf{s}_2; 0.70)$
  if $\mathbf{v}.f3 > 0.70$ then return [$\mathsf{Y}_1$, $cs$]
  elsif $\mathbf{v}.f3 > 0.60$ and $\mathbf{v}.f10 > 0.10$ then return [$\mathsf{Y}_2$, $cs$]
  else                                            return [$\mathsf{N}_3$, $cs$]
  end
end
```

Figure 11: ExamBC-03 subsystem

per) were constructed based on the observation of the development set.

4. RESULTS AND DISCUSSION

Table 5 shows our results in the development stage and the formal run [3].

From the result of the formal run, we can observe the followings.

1. In the MC subtask, our all three systems ranked the top three.
2. SKL-02 is the best among three systems in average of three subtasks.
3. In the ExamBC subtask, the value of correct answer ratio (CAR) of SKL-01 is extremely low.

The first fact surprised us because three MC subsystems are quite different. The second fact is as expected because SKL-02 was the best in the development stage. The last fact comes from the poor estimation of the confidence score.

The bottom part of Table 5 shows the performance drop of each subsystem, from the development stage to the formal run. From this part, we can observe the surprising fact that the performance of SKL-MC-01 in the formal run is better than it in the development stage.

```
def MC-03($\mathbf{v}$)
  if BC-03($\mathbf{v}$, 0.66)= $\mathsf{Y}$
    if $\mathbf{v}.x3 \leq 17$
      if ($\mathbf{v}.f15 \geq 10$ or $\mathbf{v}.b15 \geq 10$ or
          $\mathbf{v}.x3 \neq 0$) then return $\mathsf{C}_{111}$
      elsif BC-03(backward($\mathbf{v}$), 0.66) = $\mathsf{Y}_{1121}$
        if $\mathbf{v}.b12 > 0$ then return $\mathsf{C}_{1122}$
        elsif $\mathbf{v}.b14 > 0$ then return $\mathsf{F}_{1123}$
        else return $\mathsf{B}_{1123}$
        end
      elsif ($\mathbf{v}.f3 \leq 0.70$ and
             $\mathbf{v}.b3 \leq 0.55$) then return $\mathsf{B}_{113}$
      elsif $\mathbf{v}.f12 > 0$ and $\mathbf{v}.f15 > -40$ then return $\mathsf{C}_{114}$
      elsif $\mathbf{v}.f6 \geq 1.0$ then return $\mathsf{C}_{115}$
      else return $\mathsf{F}_{116}$
      end
    else
      if $\mathbf{v}.f7 > 10$ then return $\mathsf{I}_{121}$
      else return $\mathsf{F}_{122}$
      end
    end
  else
    if $\mathbf{v}.f3 < 0.55$ then return $\mathsf{I}_{21}$
    elsif (($\mathbf{v}.f3 > 0.70$ or $\mathbf{v}.b3 > 0.70$) and
           $\mathbf{v}.f15 > -40$ and $\mathbf{v}.f3 \leq 17$ and
           $\mathbf{v}.f3 \neq 0$) then return $\mathsf{C}_{22}$
    elsif count($\mathbf{v}$) $\geq 5$ then return $\mathsf{F}_{23}$
    else return $\mathsf{I}_{24}$
    end
  end
end
```

Figure 12: MC-03 subsystem

Table 5: Our results in the development stage and the formal run

		BC			MC			ExamBC			
		acc.	M-F1	#	acc.	M-F1	#	acc.	M-F1	CAR	#
Development	SKL-01	85.43	84.98	(2)	65.15	58.81	(3)	68.04	64.40	42.21	(3)
	SKL-02	86.42	85.93	(1)	70.26	60.58	(2)	69.61	67.80	52.27	(1)
	SKL-03	85.43	84.50	(3)	73.36	65.98	(1)	68.82	67.21	50.00	(2)
Formal Run	SKL-01	78.85	78.61	7	**69.53**	**59.96**	**1**	67.63	61.65	29.63	11
	SKL-02	**79.84**	**79.46**	**3**	68.61	58.25	2	**65.63**	**64.04**	**49.07**	**8**
	SKL-03	77.21	76.40	13	68.07	55.45	3	63.17	60.47	42.59	12
	Best	81.64	80.49	1	69.53	59.96	1	70.31	67.15	57.41	1
	Baseline	63.93	62.53	29	45.44	26.61	16	56.47	54.77	32.41	25
Performance Drop	SKL-01	-6.58	-6.37		**+4.38**	**+1.15**		-0.41	-2.75	-16.58	
	SKL-02	-6.58	-6.47		-1.65	-2.33		-3.98	-3.76	-3.2	
	SKL-03	-8.22	-8.10		-5.29	-10.53		-5.65	-6.74	-7.41	

acc.: accuracy, M-F1: macro F1, CAR: correct answer ratio, #: rank

Table 6: Performance of each rule of three subsystems in the MC subtask

system	c	rule	B	F	C	I	total	recall	prec
SKL-01	B		48 / 44	16 / 7	3 / 2	7 / 8	74 / 61	58% / 63% +	65% / 72% +
	F		14 / 9	142 / 156	21 / 17	19 / 21	196 / 203	69% / 76% +	72% / 77% +
	C		10 / 8	9 / 11	20 / 11	20 / 13	59 / 43	31% / 18%	34% / 26%
		cond_2	7 / 6	7 / 9	9 / 6	19 / 10	42 / 31	14% / 10%	19% / 21% +
		cond_1	0 / 0	1 / 1	6 / 2	1 / 2	8 / 5	9% / 3%	75% / 40%
		cond_4	3 / 2	1 / 1	4 / 1	0 / 1	8 / 5	6% / 2%	50% / 20%
		cond_3	0 / 0	0 / 0	1 / 2	0 / 0	1 / 2	1% / 3% +	—
	I		11 / 9	40 / 31	21 / 31	147 / 170	219 / 241	76% / 80% +	67% / 71% +
SKL-02	B	B_{11}	56 / 44	10 / 6	7 / 2	5 / 5	78 / 57	67% / 63%	72% / 77% +
	F	F_{12}	13 / 7	154 / 157	24 / 21	14 / 29	205 / 214	74% / 77% +	75% / 73%
	C		8 / 15	9 / 10	10 / 7	9 / 10	36 / 42	15% / 11%	28% / 17%
		C_{21}	6 / 14	7 / 7	4 / 7	5 / 8	22 / 36	6% / 11% +	18% / 19% +
		C_{22}	2 / 1	2 / 3	5 / 0	4 / 2	13 / 6	8% / 0%	38% / 0%
		C_{13}	0 / 0	0 / 0	1 / 0	0 / 0	1 / 0	2% / 0%	—
	I	I_{23}	6 / 4	34 / 32	24 / 31	165 / 168	229 / 235	85% / 79%	72% / 71%
SKL-03	B		63 / 43	18 / 15	3 / 3	6 / 5	90 / 66	76% / 61%	70% / 65%
		B_{1113}	59 / 43	14 / 10	3 / 3	5 / 5	81 / 61	71% / 61%	73% / 70%
		B_{112}	4 / 0	4 / 5	0 / 0	1 / 0	9 / 5	5% / 0%	44% / 0%
	F		4 / 12	154 / 161	21 / 23	15 / 35	194 / 231	75% / 79% +	79% / 70%
		F_{122}	0 / 1	90 / 105	11 / 11	11 / 24	112 / 141	44% / 51% +	80% / 74%
		F_{115}	3 / 5	59 / 51	9 / 10	4 / 7	75 / 73	29% / 25%	79% / 70%
		F_{23}	0 / 1	3 / 4	1 / 2	0 / 3	4 / 10	1% / 2% +	75% / 40%
		F_{1112}	1 / 5	2 / 1	0 / 0	0 / 1	3 / 7	1% / 0%	—
	C		6 / 9	4 / 6	17 / 4	4 / 7	31 / 26	17% / 7%	55% / 15%
		C_{22}	2 / 4	2 / 5	5 / 2	2 / 3	11 / 14	8% / 3%	45% / 14%
		C_{111}	3 / 5	1 / 1	6 / 1	1 / 2	11 / 9	9% / 2%	55% / 11%
		C_{113}	0 / 0	1 / 0	3 / 1	0 / 1	4 / 2	5% / 2%	75% / 40%
		C_{1111}	0 / 0	0 / 0	2 / 0	1 / 1	3 / 1	3% / 0%	—
		C_{114}	1 / 0	0 / 0	1 / 0	0 / 0	2 / 0	2% / 0%	—
	I		10 / 6	31 / 23	24 / 31	168 / 165	233 / 225	88% / 79%	72% / 73% +
		I_{21}	6 / 4	12 / 6	12 / 18	108 / 123	138 / 151	57% / 59% +	78% / 81% +
		I_{24}	4 / 2	18 / 17	11 / 13	57 / 40	90 / 72	30% / 19%	63% / 56%
		I_{121}	0 / 0	1 / 0	1 / 0	3 / 2	5 / 2	2% / 1%	—
gold standard			83 / 70	207 / 205	65 / 61	193 / 212	548 / 548		

Note: development / formal run

In order to explore the reason, we have examined the performance of each rule of three subsystems in the MC subtask. Table 6 shows the result. From this table, we can observe that three rules of SKL-MC-01 that detect class B, F, and I work better in the formal run than in the development stage. Because these three rules are bidirectional application of SKL-BC-01, shown in Figure 3, the better performance of SKL-MC-01 comes from the ability of SKL-BC-01.

Contradiction (C) detection is the most difficult in the MC subtask, and SKL-MC-01 has shown relatively high performance (18% recall and 26% precision). Probably this comes from the design principle of the contradiction detection in SKL-MC-01; we have employed the conditions that do not interfere with the entailment detection (i.e., detection of class F and B).

5. CONCLUSION

This paper has reported three systems of the team SKL that participated in the formal run of RITE2. SKL-01, which was designed according to two-step classification strategy, ranked first in the MC subtask. SKL-02, which used only character-based features, ranked second in the MC subtask and third in the BC subtask. These results show that our two different strategies were both effective.

6. REFERENCES

[1] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[2] H. Shima, H. Kanayama, C.-W. Lee, C.-J. Lin, T. Mitamura, Y. Miyao, S. Shi, and K. Takeda. Overview of NTCIR-9 RITE: Recognizing Inference in TExt. In *Proceedings of NTCIR-9 Workshop Meeting*, 2011.

[3] Y. Watanabe, Y. Miyao, J. Mizuno, T. Shibata, H. Kanayama, C.-W. Lee, C.-J. Lin, S. Shi, T. Mitamura, N. Kando, H. Shima, and K. Takeda. Overview of the recognizing inference in text (RITE-2) at NTCIR-10. In *Proceedings of the 10th NTCIR Conference*, 2013.