A Universal Approach to Translating Numerical and Time Expressions

Mei Tu Yu Zhou Chengqing Zong

National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
{mtu,yzhou,cqzong}@nlpr.ia.ac.cn

Abstract

Although statistical machine translation (SMT) has made great progress since it came into being, the translation of numerical and time expressions is still far from satisfactory. Generally speaking, numbers are likely to be out-of-vocabulary (OOV) words due to their non-exhaustive characteristics even when the size of training data is very large, so it is difficult to obtain accurate translation results for the infinite set of numbers only depending on traditional statistical methods. We propose a language-independent framework to recognize and translate numbers more precisely by using a rule-based method. Through designing operators, we succeed to make rules educible and totally separate from codes, thus, we can extend rules to various language-pairs without re-coding, which contributes a lot to the efficient development of an SMT system with good portability. We classify numbers and time expressions into seven types, which are Arabic number, cardinal numbers, ordinal numbers, date, time of day, day of week and figures. A greedy algorithm is developed to deal with rule conflicts. Experiments have shown that our approach can significantly improve the translation performance.

1. Introduction

Recently, statistical machine translation (SMT) models, especially the phrase based translation models [1], have been widely used and have achieved great improvements. However, there are still some hard problems. One of them is how to translate OOV words. Among all OOV words, the numerical and time expressions (we generally call numbers hereafter) are typically and widely distributed in some corpora. According to our rough statistics in a corpus of travelling domain, there are about 15 percent sentences containing numbers in all 5000 sentences. Theoretically, numbers are innumerable and the forms of numbers vary greatly from universal Arabic numbers to language-dependent number words. For example, "1.234 kg" is an Arabic number with units, the English expression "nineteen eighty-five" consists of cardinal number words, while "1.345 million" is a combination of Arabic number and cardinal number word. Due to the non-exhaustive characteristics and variability of numbers, translating numbers in the traditional SMT framework often suffers from the OOV problem even when the size of training data is very large. Thus we have to seek an efficient way to develop a new module for recognizing and translating of numbers (RTN).

According to the characteristics of numbers, it is intuitive to do RTN work through a framework with rules [2]. Traditionally, rules always depend on the specific languages they are applied to. Researchers have to build specific rule-based framework for each language-pair, thus resulting in low efficiency. Moreover, when the source or target language changes, codes are required to be rewritten accordingly. It costs much time to transplant rules. Considering that RTN is very important for text translations among all languages, we address on designing a uniformed framework to solve the RTN problem.

Based on the analysis above, in this paper we propose a language-independent rule-based approach for RTN. The proposed approach has been successfully applied and verified on bidirectional translation of Chinese-English and other language pairs. The experimental results give a much positive evidence of our work.

The remainder of this paper is organized as follows: Section 2 describes the definition of rules and symbols. Section 3 presents how to apply the rules to recognize and translate numbers. Our experimental results and analysis are presented in Section 4. Section 5 introduces related work. Finally, we give concluding remarks and mention our future work in Section 6.

2. Rules definition

Even though forms of numbers are various, the written manner and usage of number are relatively standardized. When we construct rules, such characteristics contribute a lot, and we also refer to some pervious work on rule-based systems [3-8]. In this section, we will give the details of the definition of the translation rules.

2.1. Overview of the rule-based framework

To depict our RTN module clearly, we use Figure 1 to illustrate the components of rules and how they guide the recognition and translation process.

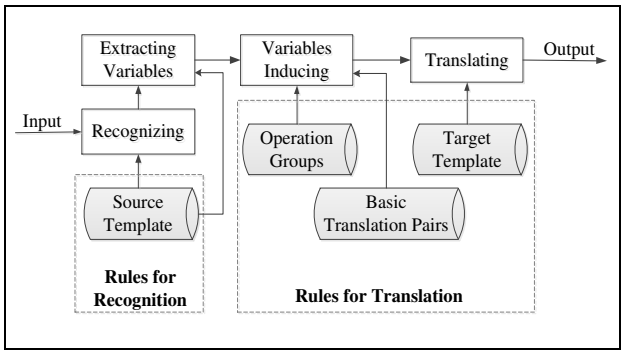


Figure 1: Rules and the workflow of RTN module

As seen in Fig.1, the first step of our module is to recognize numbers in an input sentence under the guide of the database of *Source Template*, which is in forms of regular expressions. *Source Template* consists of variables to be transformed and constants working as anchor words. After recognition, the variables will be used for inducing which is in fact a translating procedure with the assistance of *Operation Groups*

and *Basic Translation Pairs*. *Operation Groups* contain a variety of operations governing the procedure of variable inducing, while the *Basic Translation Pairs* are those translations pairs frequently used. At the final stage of our module, which is after inducing, the *Target Template* will determine the word order for each translated fragment. In order to give clearer explanation of the workflow of our module, we take "he will arrive on the 15th of May" as the input sentence and the Chinese as output language for example. At the first step, "15th of May" will be recognized by our module. And "15th" and "May" are regarded as variables, while "of" are constants. In the stage of inducing, "15th" is transformed to "十五"(fifteenth) and "May" is transformed to "五月"(May) by a series of operations. At last, we reconstruct the transformed variables to the final translation "五月 十五 号 ". In summary, *Source Template*, *Target Template, Operation Group* together with *Basic Translation Pairs* form a rule.

By observing many instances of numbers, we group numbers into seven categories. Rules will be created for each category. The categories and components of rules are described in details in the following sub-sections.

2.2. Types of number

According to the characteristics of numbers, we classify them into seven common used types as follows:

- ***Arabic number***: Arabic numerals are most widely used for counting and measuring in many languages such as Indo-European languages and Chinese. We give some examples of them in Table 1, as well as the following types.
- ***Cardinal number***: Beside Arabic number, there is also another totally different written system of numbers in many languages. Different with Arabic numbers, it is language-dependent. For example, in English, we use "one, two,…., hundred, thousand, …" to represent numbers. In addition, we also put numbers which combine cardinal numbers and Arabic numbers into this type.
- ***Ordinal number***: It represents the rank of something related with the order or position. We put them into a different group from the two types of numbers above because its written form differs from the Arabic and Cardinal numbers in many languages.
- ***Date***: The day, month, and year are always in a fixed expression.
- ***Time of day***: The time of the day often contains following several common types, "XX:XX", time expression in Arabic numbers, in cardinal numbers or the combination of Arabic and cardinal numbers.
- ***Day of week***: It includes words or expressions that represent Monday to Sunday. In some languages, like Chinese, there are several ways to represent them.
- ***Figures:*** Other numbers except above are put in to this group, such as telephone numbers, room numbers, and numbers of product labels.

Table 1: Number examples of types above

Types	Instances
Arabic Number	3.1415 ; 100,000 ; 50%
Cardinal Number	six hundred and eighty-three; 11.3 million; 一千二百(one thousand two hundred);
Ordinal Number	twenty-first ; 第二(the second)
Date	September 3rd ; eighth of August, 2008; 2000 年 1 月 1 号(January 1st, 2000)
Time of day	twelve o'clock ; half past ten a.m. ;7:00; 早八点 (eight a.m.); 八点半(8:30)
Day of week	Monday; Sunday; 星期二(Tuesday); 周六(Saturday)
Figures	telephone number one o o one one two two six ; 幺九二八 (one nine two eight)

2.3. Source template

2.3.1. Regular expression for number recognition

In many sequence searching tasks, regular expressions are chosen to match a certain sequence, for their linear complexity and simplicity. So we adopt it to recognize numbers. For example, in an English text, the regular expression for any day of May is written as follows,

Eg.1:

" (1|2|3){0,1}(1st|2nd|3rd|[4-9]th) of (May)"

We can easily extend the above regular expression to recognize date in other months by adding the alternatives of "May".

One of the most centered questions in recognition is whether the coverage of the regular expression is precise as well as complete. There are three cases in our experiments. Let us use R to represent the real coverage of the regular expression we write, and S to represent the coverage it aims to have. Then we describe the three cases as follows.

Firstly, in most cases, R is exactly equal to S, so we can easily write the regular expression to match numbers such as the double-figure numbers.

Secondly, there are exceptions that $R \succ S$, which means that the sequence extracted by our source template is not a numerical expression that we expect to get, even if it matches our template. For example, the word "second" has two kinds of common meaning. One is the ordinal form of "two" which is an ordinal number, while the other is a unit of time, like "per second", where "second" is not used as a number. Therefore, if there is no explicit anchor word in the surrounding context, like "the second day", to indicate that "second" is an ordinal number, we keep it unrecognized.

The third case is pseudo unequal. Take the regular expression in Eg.1 for example. Our purpose is to match the month-day sequence, which is of course from the first day to the last day in May. But this pattern includes not only 31 days, but also the 32nd to 39th.So if there was "on the 32nd of May" in the text, it would be captured by that pattern. However, "on the 32nd of May" is against common sense, and merely appears in the language, thus we regard this kind of inequality as pseudo inequality and ignore it.

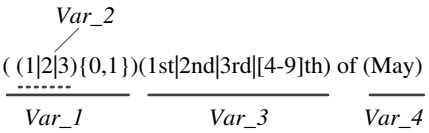
From the analysis above, we conclude that the only difficulty of using regular expression for searching lies in the second situation. In order to ensure the accuracy of our rules, it

is necessary to add more surrounding context to the regular expression.

2.3.2. Variables and constants

After the recognition process has been finished, the next step is to extract variables from the recognized sequences. To distinguish variables and constants clearly, we use brackets "()" which is compatible with the original regular expression to enclose the sequence of variables.

In this paper, we call the sequence enclosed in brackets "Var_N", in which "N" is the rank number. Then a recognized sequence can be divided into variable sequences and constant sequences. Parts of variables are used for being induced in the next stage, and they are what we care most for. We rewrite the Eg.1 pattern in section 2.3.1 as this,



where the variables are marked underlined. Only Var_1, Var_3, and Var_4 will be transformed in the next stage.

2.4. Target template

For each rule, a target template and a source template are built in pairs. And the target template is also constructed with variables and constants, which determine the final translation directly. For example,

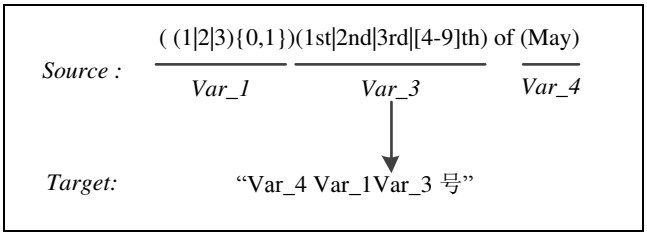


Figure 2: An example of source template and target template pair

Given the source template above, we can write a corresponding target template to convey the same meaning as the source side. The variable in the target template will be replaced with its representing sequence in the final stage of the translation, i.e. we will replace Var_4 with the Chinese translation of "May", similar to Var_1 and Var_3.

2.5. Basic translation pairs

Basic translation pairs provide translations of basic units frequently used. Take the translation of English to Chinese for example. Fig. 3 shows some examples of the basic translation pairs. Each pair is in form of "<A>/<B>", which means sequence A in source side will be translated into B in our RTN module.

We build an index at the beginning of each group to make it clearer and easier to search. The index consists of rule indexes and group number, like "<Date><#1>" which represents the first group of basic translation pairs of Date numbers. Note that the translation pairs we show in Fig.3 can depends on concrete situations, such as the pair "<1>/< 十 >" in "<Arabic><#2>". The Arabic number "1" is actually translated into "一"(one) in Chinese, but when "1" is at the decade position like "12,13,14 …", we use "十"instead of "一" and translate the numbers into "十二,十三,十四…"(twelve, thirteen, fourteen …) in Chinese.

```
<Date><#1>
        <January> /<一 月>
        <February>/<二 月>
        … …
        <May>/<五 月>
        … …
        <December>/<十二 月>
<Arabic><#2>
        <1>/<十>
        <2>/<二十>
        … …
        <9>/<九十>
<Ordinal><#1>
        <1st>/<一>
        <2nd>/<二>
        … …
        <9th>/<九>
<Cardinal><#1>
        <one>/<一>
        <two>/<二>
        … …
        <nine>/<九>
```

Figure 3: Basic translation pairs for each type

2.6. Operation groups

In order to do variable inducing from the source template to the target template, we define a series of operations for variables, which make our templates dynamic and educible, compared to the traditional static methods. Educible templates own the advantage that the rule-makers need to only care about the template and operations, instead of how to make the rules work in codes.

An operation has three terms: a subject variable, an operator, and an object. Its form is designed as,

@Subject_Var_N+Operator+Object

where "@" is a hint symbol to indicate which variable will be transformed. **Subject_Var_N** is an element of {*Var*}, while **Object** can be one of the following forms, the index of a basic translation pair, or a variable, or a sequence of words, which depends on the different operators. In the following, we list all the operators in detail,

- Terminate (T): It is an end mark, which means that all the operations are terminated.
- Join (J): the subject variable will be joined with the object variable. The object can be either another variable or a sequence of words. After joining, the new sequence becomes the subject variable.
- Replace (R): if the object is the index of a basic translation pair, the subject variable will be replaced with its translation. If the object is a sequence of words, then the subject variable is thus replaced with the word sequence.
- Replace Continuously (RC): it is similar to Replace, but the subject variable will be replaced word by word instead of as a whole sequence.

We give some examples with their explanations for each operator in Table 2.

Table 2: The Example of operators

Symbol	Example
T	@Var_1+T+NULL (No operation will be applied to the variable one)
J	@Var_1+J+Var_2 (Var_2 will be jointed to Var_1)
R	@Var_1+R+<Cardinal><#1> (Var_1 will be replaced by the translation given in the basic translation pairs of the index <Cardinal><#1>)
RC	@Var_1+RC+<Cardinal><#1> (Each word of Var_1 will be replaced by the translation given in the basic translation pairs of the index <Cardinal><#1>)

All the operators we define above own two features. First, the result of a piece of operation should still be a variable, which we call it "completeness". Second, the two-argument operator is of non-commutativity. That is why we call the arguments "subject" and "object". Operators are extendable, and we can define many other operators in theory. But in our experiment, the four operators above are enough for inducing in most cases.

After defining the operators, we can transform variables. We use the Eg.1 in section 2.3.1 to explain how the operations work. As the source template for recognition is "((1|2|3){0,1})(1st|2nd|3rd|[4-9]th) of (May)", we write the following operations to transform variables,

@Var_1+R+<Arabic><#2> (1)
@Var_3+R+<Ordinal><#1> (2)
@Var_4+R+<Date><#1> (3)

The Operation (1) translates the decade number of the day to its cardinal form in Chinese. Operation (2) translates the number under 10 to its Chinese expression. At last, the month expressions are transformed to Chinese by Operation (3). After these three operations, all English numbers are translated into Chinese. After that, given the target template as "Var_4 Var_1Var_3 号", we will obtain the Chinese month-day expression finally.

If there is a sentence "he will arrive on the 15th of May" to be dealt with, then the interim results and the final result can be listed as follows,

After "on the 15th of May" is captured by the recognition pattern, the variable inducing starts:

(1): "1" is replaced by "十",
(2): "5th" is replaced by "五"
(3): "May" is replaced by "五 月"

The final Chinese result is "五月 十五号" after substitutions for the variables in the target template.

Several translations for one source sequence are allowed, for which we can design several groups of operations for one recognition pattern. For example, the source sequence " on the 15th of May" can be translated to another kind of expression "5 月 15 日". We only need to put a separator between two groups of operations to let the system know that there is more than one group of transformed operations. Here we use a semicolon as the separator, and two continuous semicolons as the end of all groups of operations.

In the next section, we will describe the matching and integrating strategies.

3. Matching and integrating strategy

When the rules are put into use, the first thing we should care about is how to alleviate the rule conflicts, which is an important problem to use the rules in current SMT systems. In this section, we will describe our strategies in details.

3.1. Matching strategy

Generally speaking, the matching conflicts are caused by two problems: one lies on the inconsistency with tokenization, the other comes from the rule system itself.

As stated above, the recognition pattern on the source side is the regular expression, which is sensitive to the written formats. Consequently, some changes to the expressions or word segmentation in the source text may lead to a different matching result. Some languages, such as Chinese, suffer from the inconsistency of segmentation standard. So for such languages, we have to make our rules as flexible and robust as possible, by adding some alternative spaces. For example, "[0-9][[:space:]]?号" is more capable than "[0-9]号".

For the second problem, when the sequences captured by multiple rules overlap, optimization for the best choice is needed. Let us describe them mathematically. When we use patterns to recognize number sequences in one sentence, we will obtain a group of sequences grouped as {S} which contains *m* elements (sequences), and the corresponding patterns are grouped as {P} with *m* elements too. Among the *m* elements of {S}, *n* of them are under the condition that any one of the n elements overlaps with at least another one of them. Then we say that the n elements are "in conflict". From {S}, there is always a maximum sub set $\{S'\}$ with n elements in conflict, and we re-write the *n* elements as $S'_0, S'_1, ..., S'_{n-1}$, and the corresponding patterns are $P'_0, P'_1, ..., P'_{n-1}$. Then we address the optimization problem as follows,

$$Opt.\begin{cases} C = Max\{\bigcup_{i=0}^{n-1} C_i\} \\ R = Min\{\sum_{j=0}^{n-1} R_j\} \\ O = Min\{\sum_{k<l,l=1}^{n-1} O_{kl}\} \end{cases} \quad (1)$$

Where C_i is the coverage of S'_i, and $R_j = 1$ if S'_j is chosen, otherwise $R_j = 0$. If S'_k and S'_l are both chosen and overlapping, $O_{kl} = 1$, otherwise equals to zero. Our ultimate goal is to cover the longest sequence with fewest rules and fewest overlaps. Thus we adopt three optimization sub-goals, and the first one is more important to us.

For the first and second sub-goals, we design an algorithm based on a greedy method, which controls the complexity in linear time. Considering the optimization of *C* and *R*, we can write the state transition function as follows,

$$f_{k+1} = \max\{f_k \bigcup C'_{k+1}\} \quad (2)$$

$$h_{k+1} = \min\{h_k + R'_k\} \quad (3)$$

where $f_k = Max\{\bigcup_{i=0}^{k} C'_i\}$, $h_k = \min\{\sum_{j=0}^{k} R'_j\}$. We only need to sort the S'_i according to the starting position (the previous word owns higher priority) and coverage length (the longer sequence owns higher priority), and then pick them in order until obtaining the maximum union.

As for the third sub-goal, we need to save the intermediate ending positions so as to allow backtracking to the former state. The pseudo codes of the matching strategy we describe here are given in Figure 4. The captured sequences which contain numbers are saved in NumberSequenceSet in Line 1. Lines 2 and 3 focus on sorting the sequences according to the priority stated in the previous paragraph. Line 4 puts sequences in conflicts into a set. Line 5 is for initialization. The main body of the greedy algorithm is shown in Line 6~13, which is used for searching for the optimized set of sequences to get the widest coverage with the lowest cost (counts of sequences needed).

```
// Greedy algorithm for matching strategy
1:  NumberSequenceSet ← Recognize(srcSentence, Rule)
2:  SortForStartPosition (NumberSequenceSet)
3:  SortForCoverageLength (NumberSequenceSet)
4:  ConfSet ←  NumberSequenceSet.FilterConfront()
5:  CoverageEnd.assign(0); EndPosSet={}; FinalSet={}
6:  For each index in ConfSet:
7:      CurrentEnd = (ConfSet[index]).EndPos
8:      if CoverageEnd.value<= CurrentEnd:
9:          if StartPos(ConfSet[index]) in EndPosSet:
10:             i = EndPosSet.find(StartPos(ConfSet[index]))
11:             FinalSet.delete(i, FinalSet.size()-i)
12:             FinalSet.add(index)
13:             EndPosSet.add(CurrentEnd)
14: FilteredNumberSeqSet ← Output(FinalSet)
```

Figure 4: Pseudo Codes of the greedy algorithm for matching conflict

3.2. Integration approach

It is also a problem to integrate the number recognition and translation module (RTN module) into an SMT system. Traditionally there are three ways. One is in the preprocessing step by translating numbers before putting the source sentence into SMT, while the second way is in the post-editing step by translating numbers after the translation of SMT. Considering that the matching pattern has high requirements about the written formats, we adopt the third way which is more flexible by adding the related number translation knowledge to the translation model. Figure 5 illustrates how to merge the number translation system.

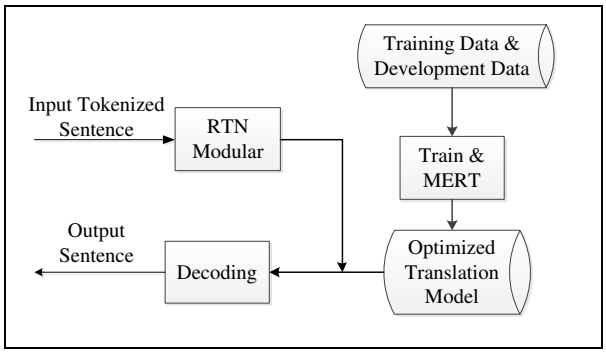


Figure 5: The systematic framework of merging RTN module into SMT

In this framework, we firstly capture the number in the input sentence and then translate those recognized numbers into target translations by the RTN module. Thus we can build a phrase-table of numbers with all the translation probabilities as 1, by considering that we definitely believe our rule-based translations of numbers. After that, the phrase-table of numbers are added into the original optimized translation model to obtain a new united table. Finally, the decoding candidates will be searched from the united table.

4. Experiments

4.1. Experiment setup

We use the IWSLT 2009 (the 6th International Workshop on Spoken Language Translation) corpus for the Chinese-English evaluation task as the bilingual training data, which includes the BTEC, CT-CE and CT-EC corpora. Because there are no test references, we randomly choose part of the development corpus as the testing set and the rest as the development set. The statistics of the training set, the development set and the testing set are listed in Table 3, 4, and 5 respectively.

Table 3: The corpus statistics for BTEC task

Corpus	Size
Training set	19,972 sentence pairs
Development set	1,000 sentences with 16 references
Test set	1,508 sentences with 16 references

Table 4: The corpus statistics for CT-CE task

Corpus	Size
Training set	30,033 sentence pairs
Development set	3,000 sentences with 16 references
Test set	1,447 sentences with 16 references

Table 5: The corpus statistics for CT-EC task

Corpus	Size
Training set	30.033 sentence pairs
Development set	800 sentences with 7 references
Test set	665 sentences with 7 references

GIZA++[1] is used to get alignments from the training corpus with grow-diag-final-and option. We train a 5-gram language model with SRILM[2] on the target part of our training corpus.

[1] http://www.statmt.org/moses/giza/GIZA++.html
[2] http://www.speech.sri.com/projects/srilm/

The translation model is generated by Moses [1] (2010-8-13 version) with default parameter settings. The bestN option is set up to 100 in MERT.

4.2. Experiment results

Table 6 shows the total number of rules we build for each type.

Table 6: The rule counts for each type

Type	En-Ch	Ch-En
Arabic	29	20
Cardinal	41	80
Ordinal	7	7
Date	36	29
Clock	13	15
Week	1	2
Figure	5	18

In our experiments, we find that the most complicated cases are among cardinal numbers and date expressions. Take the English date expression for example. When we say "the third of September", there are different ways to convey the same date, such as "the 3rd of September", "September, 3","Sep. 3" and so on. Fortunately, they are somehow regular and thus easy to write rules for other forms by analogy.

Before we apply all the rules on the translation system, we calculate the statistics manually about the ratio of sentences containing numbers in Table 7.

Table 7: The ratio of sentences with numbers

Corpus	Development	Test
BTEC	9.3%	6.7%
CT-CE	13.5%	13.5%
CT-EC	15.6%	17.8%

From Table 7, we can see that the ratios are different in different tasks. In order to alleviate the interference caused by the distribution difference, we make two kinds of evaluation about the rule contributions. One of them is an evaluation on the original test set, the other is the evaluation on sentences with numbers, which we call the **with-number** sentence set. Next, we will carry out our experiments upon the two sets.

Table 8 shows the performance of using the RTN module to recognize number sequences in the development sets of the BTEC, the CT-CE, and the CT-EC corpus. In the table, the precision is the ratio of correctly captured numbers' counts to all captured ones. The recall is the ratio of correctly captured numbers' counts to the manually marked ones. In fact the performance largely relies on the rule-makers. The more numerical and time expressions they discover the better the performance will be.

Table 8: Performance of automatic recognition by RTN

Corpus	Precision	Recall	F-score
BTEC	0.98	0.90	0.94
CT-CE	0.96	0.93	0.94
CT-EC	0.91	0.84	0.88

4.2.1. Results on development and testing set

Table 9 shows how the translation quality measured by BLEU [9] on the original testing set changes score when we add the additional transferred translation table generated by the RTN module into the phrase-based translation table. The C-E evaluation is based on the case-insensitive BLEU-4 score, and the E-C evaluation is based on the BLEU-4 score of words.

Table 9: BLEU scores of development and testing set

		BTEC	CT-CE	CT-EC
Dev.	Baseline	41.25	33.64	34.54
	Ori + RTN	41.37	33.71	34.84
Test	Baseline	37.67	32.56	33.38
	Ori + RTN	37.79	32.99	33.71

Table 10: BLEU scores of with-number sentence set

		BTEC	CT-CE	CT-EC
With-number Dev.	Baseline	40.71	31.53	39.58
	Ori + RTN	42.05	31.93	41.35
With-number Test	Baseline	30.43	30.15	38.34
	Ori + RTN	31.77	31.81	39.28

In Table 9, Ori stands for Original phrase table generated by the training data, and Ori+RTN means the new translation table after adding the additional transferred translation table generated by the RTN module. Combined with Table 7, results in Table 9 shows an obvious trend in the testing set that the more with-number sentences in testing set, the more improvement of translation performance will be achieved, which can be further confirmed in Table 10 in the experiments on the with-number sentence set.

Comparing the BLEU scores in Table 9 and 10, we can get several hints. First, although the baseline performance seems rather good, our module is still able to improve the translation quality. With the help of precise and exact number translations, the results from machine translation system become more understandable and correct.

Second, the number seems a barrier in Chinese to English translation. The probable reason of that is that numbers in Chinese may be cut into several words instead of a complete word through word segmentation. As is known to us, different segmentation results may lead to different meanings for the computer. For example, the cardinal number "五十六" is a word which means "fifty-six" or "56". But after word segmentation, it is cut into "五 十 六", which becomes three words with three numbers. If the decoder searches translation candidates of this sequence in the phrase-table, there is more than one choice. Because the SMT only depends on the language model and the translation model, the decoder may give a wrong answer "five ten six" instead of "fifty-six".

In order to give a clearer view of the results, we list some typical with-number sentences with their translations in Table 11, where S is short for source sentence, T the translation of S through the baseline translation system without the RTN module, T* the translation of S through Ori+RTN.

The 1st example in Table 11 shows that the rule-based translations are better in word order for the translation of continuous figures. The baseline translation of the 2nd source sentence is wrong, and the currency unit "圆"(yen) is left unknown. Our RNT module helps to translate the number and unit correctly, which also reduce the OOV words in the translation. The remaining examples are from the results of the English to Chinese task. The 3rd sentence is an example to correct the wrong number. The translation in T cannot be understood. After correcting the numbers by the translation from the RTN module, the system is able to generate an

[1] http://www.statmt.org/moses/index.php?n=Main.HomePage

understandable translation. The last sentence is also an example of the reduction of the OOV number words.

The negative effects of OOV words include not only the unknown meaning of themselves, but also a result of confusing word order, as seen in the examples above. Sometimes the reduction of OOV words can contribute a lot to the word order in the final translation [10].

Table 11: Some with-number examples with their translations

1	S	你 的 房间 号码 是 **二 一 零** 。
	T	your room number is **two o one** .
	T*	your room number is **two one zero** .
2	S	就 买 这个 **六 百 三 十 圆** 的 吧 。
	T	i 'll buy this thirty **six hundred** , the **圆** .
	T*	i 'll buy this **six hundred and thirty yen** for that .
3	S	**six hundred eighty three** yen ?
	T	**三 六 八 百** 日元 吗 ?
	T*	**六 百 八 十 三** 日元 吗 ?
4	S	you 'll stay in a **hundred-dollar** room with a bath on the eleventh , and in a **ninety-dollar** room on the twelfth .
	T	你 会 住 在 **hundred-dollar** 带 浴室 的 房间 号 和 **ninety-dollar** 的 房间 十 二 号 。
	T*	你 会 住 在 **一 百 美元** 带 浴室 的 房间 在 十一号 和 **九 十 美元** 的 房间 在 十二号 。

4.2.2. Errors analysis

Translation rules are indeed helpful in our experiments, however, there are still some errors and problems currently remaining unsettled. In the following we list the most common errors.

- Numbers with multiple translations in the target side: in our experiments, the translations are sometimes correct for numbers but wrong for the unit following the number in the target language. For example, When we translate "the thirteenth", we would obtain two translation results through our rules, "在 十三 号" and "第 十三". They are assigned the same probability, and the final choice is determined only by the language model, which may lead to a wrong final choice if they merely occur in the language model. On the other hand, it is likely to omit one or two senses when we create rules.
- Number translations before and after the words "to", or "and" are sometimes inconsistent: The numerical expressions are often not complete, and the same sequence will be omitted, such as "August eleventh, twelfth and thirteenth". It is possible to recognize the days before "and", but the last number is hard to track.
- In our framework, we just have tried integrating the translation of numbers into the SMT system. Although the translations of numbers are corrected by our module, their positions are sometimes wrong. As a matter of fact, there is a complicated but better way to get rid of it. If we replace the numbers with their corresponding types at the training stage, as well as the source sentence at the decoding stage, then the completeness and independence of numbers are guaranteed, which is promising to improve the translation quality much more, which we will test it in the future work.

4.3. Extent experiment

We also did experiments on Inner Mongolian to Chinese (IM-C), Uyghur to Chinese (U-C) and Japanese to Chinese (J-C) for further verification, where CWMT'2011[1] corpora are used as multi-language experimental data. Because of the lack of reference of the testing set, we only observe the improvement on the development set. Table 12, 13, and 14 separately present the corpus statistics, including the with-number sentences which we counted manually.

Table 12: The corpus statistics for Inner Mongolian-Chinese

Corpus	Size
Training set	134,567 sentence pairs
Development set	1000 sentences with 4 references
With-number set	134 sentences

Table 13: The corpus statistics for Uyghur -Chinese

Corpus	Size
Training set	100,000 sentence pairs
Development set	700 sentences with 4 references
With-number set	169 sentences

Table 14: The corpus statistics for Japanese-Chinese

Corpus	Size
Training set	564,996 sentence pairs
Development set	500 sentences with 4 references
With-number set	217 sentences

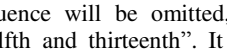
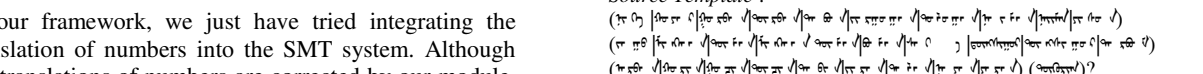
Table 15 gives the experimental results of the development of IM-C, U-C and J-C.

Table 15: BLEU scores of development set and with-number set

		IM-C	U-C	J-C
Dev.	Baseline	24.58	53.42	42.24
	Ori + RTN	24.85	54.12	42.72
With-number	Baseline	24.26	47.40	42.51
	Ori + RTN	26.24	48.86	43.47

We can see that the translation performance improves much both on the development set and on the with-number set. In table 16, we give some samples of rules and the number translations of the baseline system and our system. In the table, the basic translation pairs they use are not presented in detail, due to the space limitation of this paper. But the source template, target template and operation groups are shown.

Table 16: Examples of with-number translation and rules

M-C
S: 
T: 四 一百五十 元
T*:四百五十元
Source Template :

Target Template : var_1 var_2 var_3 var_4
Operation Group：
@var_1+R+<Cardinal><#1>@var_1+R+<Cardinal><#3>
@var_2+R+<Cardinal><#3>@var_3+R+<Cardinal><#6>

[1]http://mt.xmu.edu.cn/cwmt2011/en/index.html

@var_4+R+<Date><#4>;;

U-C
S: 48يـ ا شلار
T: 48ياشلار
T*:48 岁
Source Template :
([0-9]+)(تـ ارماق نى|تد رلد قېكد لوم|ئ ى|يـ ا شلار|ذ ى|ئ ايـ لد ق)
Target Template : var_1 var_2
Operation Group：@var_2+R+<Date><#5>;;

J-C
S: 4 月 3 日
T: 4 月 三 天
T*:4 月 3 日
Source Template :
([1-9]|10|11|12) 月 ((1|2|3){0,1}[0-9]) 日
Target Template : var_1 月 var_2 日
Operation Group：@var_1+T+NULL @var_2+T+NULL;;

5. Related work

In fact, the research on number translations started in early. [2] analyzed cardinal number names in five languages, and implemented the rules governing the construction of number names. The translation from a figure expression into a word expression is also done. [11] did part work of of [2], which only discussed about how to translate number names to Dutch. But they did not separate rules from source codes, which handicaps the scalability of rules. And the number representation they cared about is limited to cardinals with well formats, which is not enough for processing complicated and not so standardized text. [12] proposed an embedded-in-code-rule based framework to do the number recognition and translation task on Chinese-English translation. [13] combined a rule based machine translation system with a statistic based post editor, which showed helpful for a more accurate performance than only using statistical machine translation. However, although the former works on RTN have achieved successes in different ways, as far as we see, works on language-independent and code-separate rule-based number recognition and translation are very rare. Our work may be a big step in the work of number recognition and translation.

6. Conclusions

In this paper, we design a language-independent and rule-based method to translate numbers more precisely. The templates are totally separated from the source codes, which makes multiple language translation of numbers with one framework practical. We design a variety of operators to make the templates educible and dynamic compared to traditional templates, which allows rule-makers to get rid of source code to present the inducing progress and gives more flexible rooms to establish rules. Rule-makers can easily write templates and operations according to their bilingual knowledge of numbers. In our experiments we build groups of rules for bidirectional English-Chinese translation, as well as for some other languages to Chinese. After we apply the rules on the phrase-based translation, the translation quality improved upon both the development set and the testing set. The performance on with-number sentences set are all improved around 1 point BLEU score.

In the future work, we are about to explore more suitable integrating strategy for the RTN module into the SMT system. Rules for other language pairs will be built to improve their translation performance.

7. Acknowledgements

The research work has been partially funded by the Natural Science Foundation of China under Grant No. 60975053 and the Hi-Tech Research and Development Program ("863" Program) of China under Grant No. 2011AA01A207, supported by the External Cooperation Program of the Chinese Academy of Sciences. This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

8. References

[1]Koehn, P., Och, F.J. and Marcu, D., "Statistical phrase-based translation," in the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology Stroudsburg, PA, USA, 2003, pp. 48-54.

[2]Hugo Brandt, Corstius, "Automatic translation between number names," Grammars for number names, vol. 7, p. 103, 1968.

[3]Hayes-Roth, F., "Rule-based systems," Communications of the ACM, vol. 28, pp. 921-932, 1985.

[4]McDermott, J., "R1: A rule-based configurer of computer systems," Artificial intelligence, vol. 19, pp. 39-88, 1982.

[5]Ilgun, K., Kemmerer, R.A. and Porras, P.A., "State transition analysis: A rule-based intrusion detection approach," Software Engineering, IEEE Transactions on, vol. 21, pp. 181-199, 1995.

[6]Ishibuchi, H., Nozaki, K., Yamamoto, N. and Tanaka, H., "Selecting fuzzy if-then rules for classification problems using genetic algorithms," Fuzzy Systems, IEEE Transactions on, vol. 3, pp. 260-270, 1995.

[7]Hammer, J., Garcia-Molina, H., Nestorov, S., Yerneni, R., Breunig, M. and Vassalos, V., "Template-based wrappers in the TSIMMIS system," ACM Sigmod Record, vol. 26, pp. 532-535, 1997.

[8]Yen, J. and Wang, L., "Simplifying fuzzy rule-based models using orthogonal transformation methods," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 29, pp. 13-24, 1999.

[9]Papineni, K., Roukos, S., Ward, T. and Zhu, W.J., "BLEU: a method for automatic evaluation of machine translation," in the 40th Annual Meeting on Association for Computational Linguistics, 2002, pp. 311-318.

[10]Zhang, Jiajun, Zhai, Feifei and Zong, Chengqing, "Handling unknown words in statistical machine translation from a new perspective " presented at the NLPCC, Beijing, 2012.

[11]Corstius, H.B., "Automatic translation of numbers into Dutch," Foundations of Language, vol. 1, pp. 59-62, 1965.

[12]Zhai, Feifei, Xia, Rui and Zong, Chengqing, "An Approach to Recognizing and Translating Chinese & English Time and Number Named Entities," in the 7th China Workshop on Machine Translation (CWMT'2009), Nanjing, 2009.

[13]Terumasa, E., "Rule based machine translation combined with statistical post editor for japanese to english patent translation," 2007, pp. 13-18.