Smoothing Statistic Databases in a Machine Translation System

Keh-Yih Su[*], Mei-Hui Su[**] and Li-Mei Kuan[**]

[*]Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan, R.O.C.

[**]BTC R&D Center
28 R&D Road II, 2F
Science-based Industrial Park
Hsinchu, Taiwan, R.O.C.

ABSTRACT

In a Machine Translation (MT) system, it is necessary to be able to determine the most likely structure among the ambiguities. This can be accomplished by using the probability as a selection basis for the well-formedness of each structure. However, this method requires a very large set of training data for the probabilistic database in order to obtain an acceptable degree of selection appropriateness.

In ArchTran English-Chinese Machine Translation System, a probability-based approach to automatizing the structure selection process is adopted. Although this method performs satisfactorily for structures already in the database, it performs rather poorly for structures not in the database. This is the problem with a sparse database. Therefore, in this paper, we propose to improve the prediction power of the database by a technique called **Database Smoothing**. Briefly, there are two smoothing methods that can be adopted. The first method is to employ a flattening constant to smooth the empty probability cells of the database. The second method is to incorporate additional information from another database into the one to be smoothed. We have conducted a simulation on the smoothed database and an improvement of 13.1 percent is observed for the open test samples. This is very encouraging because it shows improvements can be achieved for all database applications that employ a smoothed probabilistic model.

MOTIVATION

In a Machine Translation (MT) system, it is natural to have more than one interpretation for most input sentences. These ambiguous interpretations are attributable not only to the over-generative grammar adopted by the system but also to the inherent characteristics of the source language. Since the main purpose of a MT system is to produce a single appropriate interpretation for an input sentence in order to reduce the work for post editor, it is therefore desirable that the system provides a fast and competent mechanism to single out the correct interpretation.

In order to minimize the time spent on selecting the correct parse trees, we constructed several statistical databases (SDBS) as the means to automatize the tree selecting process [SU 88]. These databases contain the tree structures that are successfully parsed and selected by the linguist. With these databases, the well-formedness, in terms of score, of every ambiguous parse tree can be calculated for an input sentence. Afterwards, the parse tree with the highest score is selected as the preferred interpretation over all the other ambiguities.

We reported an experiment in regard to SDBS's prediction accuracy in [SU 88]. We found that with the database size of 1468 sentences, the accuracy rate for the close test can reach as high as 85%. However, the result is less accurate for the open test. The reason for this difference is because the training data for the database is not large enough. Consequently, the variety of sub-structures that can be found in the database is not extensive enough. Because of this, even if the structure is correct if its sub-structures do not match any corresponding entry in the database, its likelihood probability approaches zero. This is a serious problem for using database that is sparse in a MT system. In this paper, we propose to adopt the database smoothing technique that maintains the high accuracy rate for the structures already in the database and improves the prediction accuracy for structures outside the database.

There are two general approaches in smoothing a sparse database for improving the selection result of an open test. The first method is to smooth the cells of a database by a small flattening constant [FIEN 72]. The second method is to include information from a database that might not perform as well as the database to be smoothed but is less sparse. In the later sections, the approaches adopted for database smoothing will be presented.

Aside from structure selection, database smoothing can also be extended to other database applications. For instance, the truncation parsing mechanism in ArchTran also employs a probability database to direct the parsing of the input sentences. Information from this database is used to predict whether a path will eventually succeed or not. If a path receives a low prediction value, it will be truncated and the time will be saved. Similar to structure selection, the truncation mechanism will also fall short of its function if its database is sparse and database smoothing is not used. Therefore, it is obvious that database smoothing is required for improving the reliability of the applications that use databases.

In the following sections, we will briefly discuss how the well-formedness of a structure is measured; how the databases in ArchTran are constructed and their shortcomings. Then, the mechanism of database smoothing will be described, followed by the result of our testing on the smoothed database. Last but not least, we will discuss some limiting factors that will affect the result of the database smoothing.

SCORE

The degree of well-formedness of a structure can be measured in terms of the syntactic well-formedness ($SCORE_{syn}$), the semantic well-formedness ($SCORE_{sem}$) and the lexical well-formedness ($SCORE_{lex}$) of the structure [SU 88]. According to [SU 88], for a structure X, its score can be reduced to $SCORE(X) = SCORE_{syn}(X) * SCORE_{sem}(X) * SCORE_{lex}(X)$. So for a sentence with more than one ambiguous structure, the most appropriate structure should be the one with the highest score. Since the semantic score and the lexical score have similar formulation as the syntactic score, they will not be discussed here.

The syntactic score of a structure can be generalized as the product of the conditional probability of its reduction sequences. Take the syntax tree in Fig.1 as an example. In this tree, *n* and *v* are the lexical categories of the input words, and *S*, *NP* and *VP* are the grammatical symbols. For this tree, written in the form of context sensitive rules with one right lookahead and one left context symbol, the reduction sequences of a LR derivation are : *(ϕ n v => ϕ NP v), (NP v ϕ => NP VP ϕ)*, and *(ϕ NP VP ϕ => ϕ S ϕ)*, where ϕ is the null symbol.

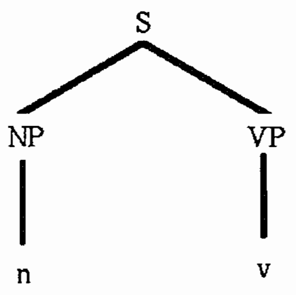

Fig. 1 A syntax tree

For these reduction sequences, the conditional probabilities are : *P(NP/ ϕ n v), P(VP/ NP v ϕ)* and *P(S/ ϕ NP VP ϕ)*, respectively. From [SU 88], the syntactic score for *S* is

$$SCORE_{syn}(S) = P(S|\phi\ NP\ VP\ \phi) * P(VP|NP\ v\ \phi) * P(NP|\phi\ n\ v\ ).$$

Based on the structural well-formedness defined above, we can construct probability databases for selecting the most likely structure among the ambiguities for an input sentence.

DATABASES AND THEIR SPARSE DATA PROBLEM

In this section, we will first briefly describe the databases constructed for structure selection in ArchTran. Next, the sparse data problem of these databases will be discussed and the possible solutions will be presented.

Currently, we have ten independent databases that store the conditional probabilities of different types of reduction sequence. They are : L3, L2R1, L2, L1R2, L1R1, L1, R3, R2, R1 and N (no context information), where the numbers following L and R designate the number of left context symbols and right lookahead symbols referenced. These databases differ in that they incorporate different scopes of context information during their construction. For example, the L2R1 database is constructed with two left context symbols and a right lookahead symbol.

The problem with using a probability-based approach to select the most appropriate structure is that it can not do well for structure that is outside the scope of the database. This sparse data problem which can be decomposed into two parts. The first is the proliferation of empty cells (every possible reduction sequence occupies a cell in a database) because the training sample is small relative to all possible reduction sequence in the analysis grammar. The second is a special instance of the sparse data problem [JELI 80] when the samples in a set of databases are not large enough. As a result, some databases will be more reliable but have less statistics support, while other databases, are less reliable but have more samples to produce significant statistics. Under such circumstances, one database may perform better in some cases but less favorably in other cases.

The empty cell problem will affect the prediction performance of the database when most cells in the database are essentially empty. And the effect is that most of the cell queries will be zero during structure selection. Since the probability estimation of small values will not reflect the true probabilistic model, it could not be trusted as noted in [NADA 85]. Therefore, these cells must be filled. The most obvious solution is to enter as much sampling data into the SDBS as possible. But this is a very time-consuming long-term task whose affect is not immediately felt. The reason is that the man power needed to find those correct sampling structures that will completely cover all possible derivations of an analysis grammar for a natural language is simply too enormous to even consider. A more feasible alternative is to adopt the flattening constant method suggested in [FIEN 72]. A more detailed description of this method will be presented in the next section.

Next, we will address the second aspect of the sparse data problem. The performance of different databases differs because the context reference in building a database also serves as a constraining factor in building the entries and in matching the sub-structures of a parse tree during structure selection. For example, the L2R1 database might support the linguistic model more accurately than the L1R1 database, but the variance of L2R1 is larger than L1R1. Therefore, L2R1 has less statistics for the open test samples and the prediction on of these samples is lowered. The following example will demonstrate this problem more concretely.

$$L_2\ L_1\ A\, R_1\ \rightarrow L_2\ L_1\ B\, R_1$$

Let the above equation be a sub-structure included in the SDBS, where A is the symbol that reduces to B; L_2 and L_1 are the left context symbols; and R_1 is the right context symbol. Then, there will be an entry of $L_2\ L_1\ A\ \ R_1\ \rightarrow\ L_2\ L_1\ B\ \ R_1$ in the L2R1 database. At the same time, there will be an entry of $L_1\ A\ \ R_1\ \rightarrow\ L_1\ B\ \ R_1$ in the L1R1 database. If a given sub-structure to be matched is $L_2'\ L_1\ A\ \ R_1\ \rightarrow\ L_2'\ L_1\ B\ \ R_1$, this will not match any entry in L2R1 but it will match $L_1\ A\ \ R_1\ \rightarrow\ L_1\ B\ \ R_1$ in L1R1. This means with L1R1, this sub-structure will have a value for its likelihood but not so with the L2R1. This shows that with a small training sample, L1R1 has more matchable entries than L2R1. In other words, for a structure outside a database's training data, it is more likely to obtain some usable information from database that is not as context-sensitive.

Following this logic, we can claim that if there is a database with no context restriction, any sub-structure will be most likely to match some entry in this database. But from [SU 88], it is shown that the accuracy rate for less restrictive database is lowered for selecting structures already in the database. The reason for this is the context-sensitiveness of the natural language. As the context information is discarded, the prediction power deteriorates. Therefore, switching a database to a less restrictive one (i.e. from L2R1 into L1R1) will not improve the selection result in general.

There are two ways to resolve this problem. The first method, is to enter as much sampling data into the SDBS as possible. Again, the required man power and time are the limiting factors for adopting this method. The second method, an extension of an existing technique in signal processing [LEE 88], is to smooth the database with information from another database that is less context restricted. This technique of the database smoothing will be discussed in the next section.

SMOOTHING

To compensate for the inadequacy of not well-trained database in selecting structures outside database, we are adopting two methods of database smoothing to improve the prediction accuracy of a database.

The first is to smooth the databases with a flattening constants. In order to explore the extent of empty cells in the database, we did a tentative check. With 182 English sentences from the open test sample, all the ambiguity structures are broken down into database queries. And the result is tabulated in the following table.

Number of Queries \ Databases	L2R1	N
Total Queries	53019	53019
Total Empty Cells Queried	20329	6401

Table 1. Database queries of the open test sentences

It is obvious from the table above that most cell queries from sentences in the open test sample are empty and therefore flattening constant is needed. The inclusion of flattening constant can be summarized in three simple steps. If we let the flattening constant be α, then the steps for smoothing entries with empty cells are as follows :

[1] For every empty cell, let the cell value be equal to α.

[2] For every non-empty cell, increment the cell value by α.

[3] For every cell, calculate the probability of each cell by cell value / total occurrences in the entry.

From [FIEN 72], we choose to set α equal to 1/2 and we will demonstrate these steps with the following example. Let two original entries in a database be that shown in Fig. 2.

	Left Context	Current State	Lookahead	Reduced To	Occurrences	Probability
				A	500	500/501
1.	L2 L1	S	R1	B	1	1/501
				C	0	0
	Total Occurrences = 501					
				A'	1	1/1
2.	L2' L1'	S'	R1'	B'	0	0
				C'	0	0
	Total Occurrences = 1					

Fig. 2 Two Entries in L2R1 Database

In the above figure, each entry consists of three cells (or the number of possible reduction sequences) and each cell is followed by its number of occurrences and its conditional probability.

From this example, an additional problem of using a simple probability model can be observed. In Fig.2, the first reduction cell of the first entry has a probability value of 500/501 and the first reduction cell of the second entry has a value of 1/1. Consider the number of occurrences, it is obvious that the first instance of the first entry should be more likely than the first instance of the second entry. But the values of 500/501 vs. 1/1 do not reflect this observation. We will see that with the flattening constant added, this will be remedied. In the following figure, the entries are modified with the flattening constant.

	Left Context	Current State	Lookahead	Reduced To	Occurrences	Probability
				A	500+1/2	1001/1005
1.	L2 L1	S	R1	B	1+1/2	3/1005
				C	1/2	1/1005
	Total Occurrences = 502+1/2					
				A'	1+1/2	3/5
2.	L2' L1'	S'	R1'	B'	1/2	1/3
				C'	1/2	1/3
	Total Occurrences = 2+1/2					

Fig. 3 Two Entries in L2R1 Database with α

Now, the empty cells of these entries are filled with values relative to the total number of occurrences of the entry. It should be noted that, the original value of 500/501 is replaced by 1001/1005 and the original value of 1/1 is replaced by 3/5. This new set of values now reflects their real relative probability state.

The second smoothing method is to smooth the database with another database that is less sparse. So, the score of a tree is not the conditional probability calculated from just a single database. Instead, it is the interpolated conditional probability calculated from several databases.

In order to acquire a modeling for our databases, we devised a reward function *y* that rates how well the correct structures are selected. The reward function is such that after all the ambiguities of a sentence are ranked by the score from a database, if the correct structure falls at the first place, a reward of 5 is added. If the correct structure falls at the second place, a reward of 2 is added. For any place beyond, no reward is added. Now, we can show how different databases perform with this reward function. In the following figure, the open test sentences are grouped according to the percentage of empty cells they have queried. The numbers in the square brackets are the number of sentences in each group. For each group, the average reward is found and plotted against the group.

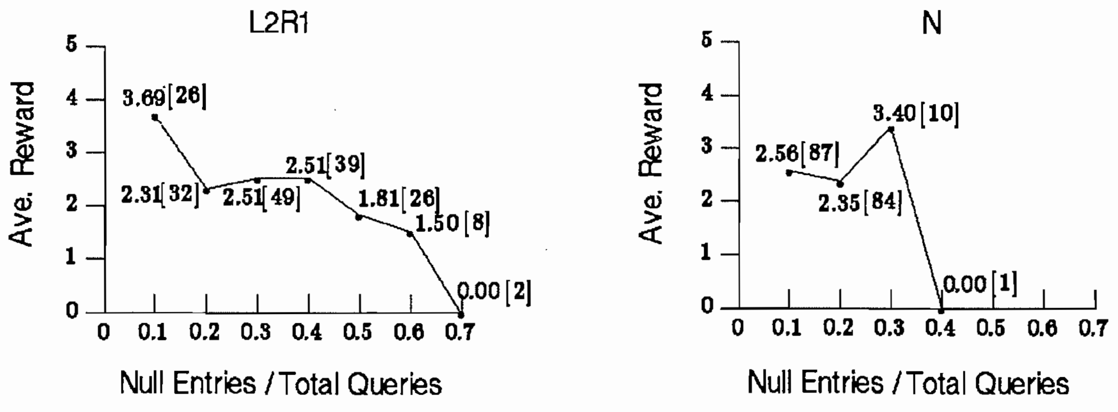

Fig. 4 Ave. Reward vs. Null Entries

From the figure above, it can be seen that the performance of L2R1 database deteriorates as the percentage of empty cells increases. But it is actually the opposite for the N database. Therefore, if we smooth the L2R1 database with the N database, the prediction of those sentences whose database queries are mostly zero will improve.

If we let P_s be the interpolated sum and P_i be the conditional probability calculated from the ith database, then $P_s=c_1P_1+c_2P_2$, with P_1 from L2R1 and P_2 from N. The coefficients are subject to $c_1+c_2=1$. The reason L2R1 database is selected as the one to be smoothed is because it exhibits the highest prediction rate for the close test. As for the N database, the reason why it is selected for smoothing is because it has most entries.

The P_s equation can be further modified with additional weighting functions. The reason for these functions is that the trustworthiness of a probability should be dependent on the total occurrences of all cells within the same entry. Therefore, the new equation is $P_s=c_1h_1(x)P_1+c_2h_2(x)P_2$, where $h_1(x)$ and $h_2(x)$ are the weighting functions such that $x=n/t$ (n is the number of total occurrences for this entry; t is the number of cells in this entry). The need for the weighting functions can be justified from the curve in Fig.5.

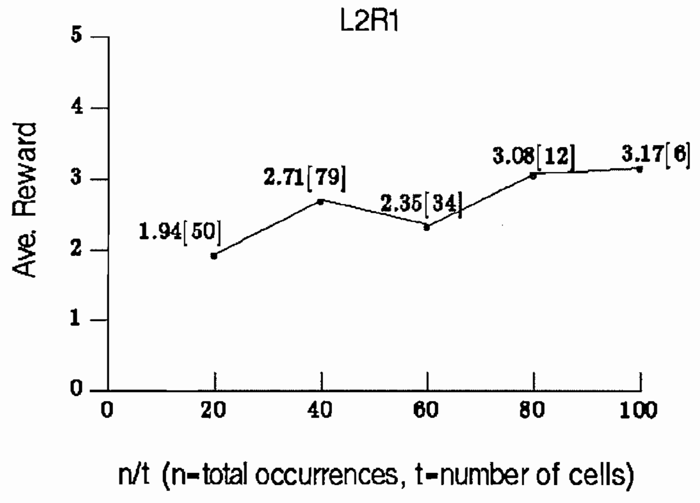

Fig. 5 Ave. Reward vs. n/t

In Fig. 5, the test sentences are divided into several groups according to the average of total occurrences divided by the number of cells of each database query (n/t). Afterwards, each group is plotted against the the average reward value of the group. From the curve, it can be seen that as the n/t value increases, the corresponding average reward increases. Therefore, there is a direct link between the accuracy of a probability and its n/t. With this curve, we can define the weighting function for L as $h_1(x)=a(1-e^{-xb})+1$, where a and b are tunable variables for matching the current state of the database. For simplicity's sake, the weighting function for N database is set to 1. The final equation for P_s is as follows :

$$P_s = c_1 h(x) P_1 + c_2 P_2,$$

$$Where \quad c_1 + c_2 = 1,$$

$$h(x) = a\left(1 - e^{-bx}\right) + 1, \; x = \frac{n}{t},$$

$$n = number\ of\ occurrences, \quad t = number\ of\ cells,$$

$$a\ and\ b\ are\ tunning\ variables,$$

$$P_1\ and\ P_2\ are\ probabilities\ from\ two\ different\ databases.$$

In the following section, we will present the result of the simulation we conducted for testing the new Ps equation derived above.

TEST

We conducted a simulation with 182 sentences as the open test samples. For these test sentences, the reward value for using L2R1 database is 428 and for using N database is 418. The purpose of the simulation is to find out to what degree the reward value increases for the smoothed database.

During the simulation, we encountered two problems. First, our original databases did not record the empty cells because they will take up too much space. So, we have to expand the databases to include the empty cells for adding the flattening constant. But, it is simply impossible to generate all possible sub-structure of an over-generative analysis grammar for a natural language. As a result, we resort to expand the databases with just the ambiguous structures we have collected in the past. The second problem we encountered is that the reward function *y* does not have an analytic formula. So, all we can do is to observe the improvement of *y* as c_1 and c_2 make small deviations. Note that the reward function is not the same as the smoothed score function. The reward function is a measuring function of how well the smoothed score function is, that is, how well it predicts the correct structure of an input sentence.

Now, it is the question of finding a best set of *a*, *b*, c_1, and c_2 for the smoothed database such that the reward value is the greatest. This can be seen as an optimization problem for the nonlinear reward function with certain constraints. We have devised an iteration method for finding these coefficients. Briefly, with some pre-selected values for *a* and *b*, we start with a set of initial coefficients, $C^0=[c_1, c_2]$. The next set of coefficients are found by shifting each coefficient slightly in a direction such that the reward function increases, $C^{i+1}=C^i + \Delta C^i$. This iteration process continues until an optimal value is found for the reward function.

In the simulation, we selected several sets of initial values for *a*, *b*, c_1, and c_2. The results after several iterations are tabulated in the following table.

Inputs & Results / Data Sets	a	b	c1	c2	Final Reward	Improvement (%)
1	0	~	0.8	0.2	475	11%
2	1	1	0.8	0.2	476	11.2%
3	1	1	0.6	0.4	483	12.8%
4	1	2	0.9	0.1	482	12.6%
5	1	100	0.7	0.3	475	11%
6	100	1	0.8	0.2	484	13.1%

Table 2. Open test results of the smoothed L2R1 database with different sets of inputs

As can be seen, the highest value we have achieved so far is 484. Compared with 428, it is an improvement of 13.1 percent. We also conducted a close test which consists of 50 sentences on the smoothed database. The open test results are tabulated in Table 3 with entries corresponding to the data sets in Table 2.

Results / Data Sets	Reward Value	Deterioration (%)
1	208	3%
2	210	2.5%
3	207	3.7%
4	210	2.5%
5	208	3%
6	213	0.9%

Table 3 Close test results on smoothed L2R1 database

Comparing the results in Table 3 with the reward value of 215 for the original L2R1 database, it is obvious that the result of the close test has not deteriorate much. All in all, the result of the open test is very encouraging with the few points we tried. In the future, we would like to conduct a more extensive search for a even better set of values.

LIMITING FACTORS IN DATABASE SMOOTHING

In this section, we would like to discuss three factors that might influence the outcome of a smoothed database.

First, if the number of iterations is not large enough in looking for c_is, it is questionable whether or not we have arrived at the best choice of all maximums. The embedded problem is that the analytic reward function is not known and its stability is dependent on the training sample of the databases. But there is an additional action that can be taken to minimize the effect of this problem. One can take some coefficient vectors that are more distant from the current maximum and start other searching iterations. When different end results are compared, if the current point is still the maximum then it can be certain that it is a relatively good maximum.

Second, if the test sentence sample is not large or random enough, then not every sentence type outside the database is compiled into the sample. As a consequence, the prediction power might not have improved for some sentences outside the database. Ideally, if it is possible to compile every possible sentence structure into the test sample, then a nearly perfect database can be constructed.

Third, if the test sample for the smoothing mechanism is too small then the variance in the smoothed database will be so large that it will affect the selection of structures that are within the database. Therefore, it is better to do the smoothing iteration with a test sentence sample consists of sentences from both inside and outside the database.

These factors are intended to serve as a reminder when employing the technique of database smoothing.

CONCLUSION

In a MT system, it is a time-consuming task to manually select the correct interpretation for a sentence among all generated ambiguities. Therefore, the idea of employing a statistic database as a tool to automatizing the structure selection evolves. But when the database has a small training sample, its prediction accuracy is not good enough for the open test. In this paper, we proposed to overcome this deficiency with the technique of database smoothing. This includes the adding of a flattening constant and the incorporating of additional information from another database.

We have conducted an open test of 182 sentences on the smoothed database. The result of a few trial tests shows an improvement of 13.1 percent. This encouraging result has prompted a more extensive testing planned in the near future.

ACKNOWLEDGEMENT

We are indebted to Prof. Anne Chao for her valuable suggestions. In addition, we would like to thank Jing-Shin Chiang for his helpful comments on this paper.

REFERENCE

[FIEN 72] Fienberg, S.E and P.W. Holland, "On the Choice of Flattening Constants for Estimating Multinomial Probabilities," *Journal of Multivariate Analysis*, Vol. 2, PP. 127–134, 1972.

[JELI 80] Jelinek, F. and R.L. Mercer, "Interpolated Estimation of Markov Source Parameters from Sparse Data," In E.S. Gelsema and L.N. Kanal (eds.) : *Pattern Recognition in Practice*, North-Holland Publishing Company, Amsterdam, Netherlands, PP. 381–397, 1980.

[LEE 88] Lee Kai-Fu, "Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System," Doctoral thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1988.

[NADA 87] Nada, A., "On Turing's Formula for Word Probabilities," *IEEE Transaction on Acoustics, Speech, and Signal Processing*, Vol. ASSP-33, No. 6, PP. 1414–1416, Dec. 1985.

[SU 88] Su K.Y. and J.S. Chang, "Semantic and Syntactic Aspects of Score Function," *Proceedings of the 12th international conference on Computational Linguistic*, Budapest, Hungary, PP. 642-644, 1988.