Three Heads are Better than One

Robert Frederking
Center for Machine Translation
Carnegie Mellon University
Pittsburgh, PA 15213
ref@cs.cmu.edu

Sergei Nirenburg
Computing Research Laboratory
New Mexico State University
Las Cruces, NM 88003
sergei@crl.nmsu.edu

Abstract

Machine translation (MT) systems do not currently achieve optimal quality translation on free text, whatever translation method they employ. Our hypothesis is that the quality of MT will improve if an MT environment uses output from a variety of MT systems working on the same text. In the latest version of the Pangloss MT project, we collect the results of three translation engines — typically, subsentential chunks — in a chart data structure. Since the individual MT systems operate completely independently, their results may be incomplete, conflicting, or redundant. We use simple scoring heuristics to estimate the quality of each chunk, and find the highest-score sequence of chunks (the "best cover"). This paper describes in detail the combining method, presenting the algorithm and illustrations of its progress on one of many actual translations it has produced. It uses dynamic programming to efficiently compare weighted averages of sets of adjacent scored component translations. The current system operates primarily in a human-aided MT mode. The translation delivery system and its associated post-editing aide are briefly described, as is an initial evaluation of the usefulness of this method. Individual MT engines will be reported separately and are not, therefore, described in detail here.

1 INTRODUCTION

Current MT systems, whatever translation method they employ, do not reach an optimal output on free text. In part, this is due to the inherent problems of a particular method — for instance, the inability of statistics-based MT to take into account long-distance dependencies, the difficulty in achieving extremely broad coverage in knowledge-based MT systems, or the reliance of most transfer-oriented MT systems on similarities in syntactic structures of the source and the target languages.

Our hypothesis is that if an MT environment can use the best results from a variety of MT systems working simultaneously on the same text, the overall quality will improve. Using this novel approach to MT in the latest version of the Pangloss MT project, we submit an input text to a battery of machine translation systems (*engines*), collect their (possibly, incomplete) results in a joint chart data structure and select the overall best translation using a set of simple heuristics.

2 INTEGRATING MULTI-ENGINE OUTPUT

In our experiment we used three MT engines:

- a knowledge-based MT (KBMT) system, the mainline Pangloss engine (Frederking et al., 1993b);
- an example-based MT (EBMT) system (see (Nirenburg et al., 1993; Nirenburg et al., 1994b); the original idea is due to Nagao (Nagao, 1984)); and
- a lexical transfer system, fortified with morphological analysis and synthesis modules and relying on a number of databases — a

machine-readable dictionary (the Collins Spanish/English), the lexicons used by the KBMT modules, a large set of user-generated bilingual glossaries as well as a gazetteer and a list of proper and organization names.

The outputs from these engines (target language words and phrases) are recorded in a *chart* whose positions correspond to words in the source language input. As a result of the operation of each of the MT engines, new edges are added to the chart, each labeled with the translation of a region of the input string and indexed by this region's beginning and end positions. We will refer to all of these edges as *components* (as in "components of the translation") for the remainder of this article. The KBMT and EBMT engines also carry a *quality score* for each output element. The KBMT scores are produced based on whether any questionable heuristics were used in the source analysis or target generation. The EBMT scores are produced using a technique based on human judgements, as described in (Nirenburg et al., 1994a), *submitted.*

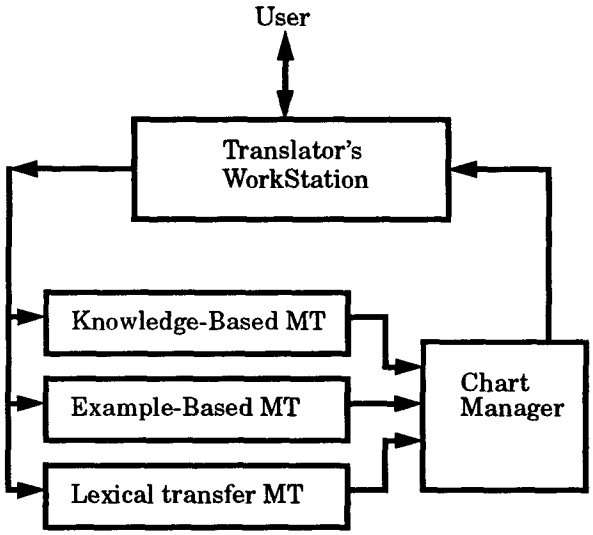

Figure 1: Structure of a multi-engine MT system

Figure 1 presents a general view of the operation of our multi-engine MT system. The *chart manager* selects the overall best cover from the collection of candidate partial translations by normalizing each component's quality score (positive, with larger being better), and then selecting the best combination of components with the help of the *chart walk* algorithm.

Figure 2 illustrates the result of this process on the example Spanish sentence: *Al momento de su venta a Iberia, VIASA contaba con ocho aviones, que tenían en promedio 13 años de vuelo* which can be translated into English as *At the moment of its sale to Iberia, VIASA had eight airplanes, which had on average thirteen years of flight (time).* This is a sentence from one of the 1993 ARPA MT evaluation texts.

For each component, the starting and ending positions in the chart, the corresponding source language words, and alternative translations are shown, as well as the engine and the engine-internal quality scores. Inspection of these translations shows numerous problems; for example, at position 12, "aviones" is translated, among other things, as "aircrafts". It must be remembered that these were generated automatically from an on-line dictionary, without any lexical feature marking or other human intervention. It is well known that such automatic methods are at the moment less than perfect, to say the least. In our current system, this is not a major problem, since the results go through a mandatory editing step, as described below.

2.1 Normalizing the component scores

The chart manager normalizes the internal scores to make them directly comparable. In the case of KBMT and EBMT, the pre-existing scores are modified, while lexical transfer results are scored based on the estimated reliability of individual databases, from 0.5 up to 15. Currently the KBMT scores are reduced by a constant, except for known erroneous output, which has its score set to zero. The internal EBMT scores range from 0 being perfect to 10,000 being worthless; but the scores are nonlinear. So a region selected by a threshold is converted linearly into scores ranging from zero to a normalized maximum EBMT score. The normalization levels were empirically determined in the initial experiment by having several individuals judge the comparative average quality of the outputs in an actual translation run.

In every case, the base score produced by the scoring functions is currently multiplied by the length of the candidate in words, on the assumption that longer items are better. We intend to test a variety of functions in order to find the right contribution of the length factor.

2.2 The chart walk algorithm

Figure 3 presents the chart walk algorithm used to produce a single, best, non-overlapping, contiguous combination (*cover*) of the available component translations, assuming correct component quality scores. The code is organized as a recursive divide-and-conquer procedure: to calculate the cover of a region of the input, it is repeatedly split into two parts, at each possible position. Each time, the best possible cover for each part is recursively found, and the two scores are combined to give a score for the chart walk containing the two best subwalks. These different splits are then compared with each other and with components from the chart spanning the whole region (if any), and the overall best result is

Position L	Position R	Input (Spanish)	Output (English)	E	Q
0	1	Al momento	"In a minute" "At once" "A moment"	G	10
2	2	de	of from about for by	D	2
3	3	su	his her its one's your their	G	5
4	4	venta	inn sale selling marketing "country inn" "small shop" stall booth	G	5
5	5	a	to a of	D	2
6	6	Iberia	Iberia	G	5
7	7	,	,	G	5
8	8	VIASA	VIASA	D	2
9	10	contaba con	"was rely on" "rely on" "was count on" "count on" "was depending on" "depended on" have	G	10
11	11	ocho	eight eighth	D	2
12	12	aviones	airplane aeroplanes planes aircrafts airplanes martins hopscotches	L	2.5
13	13	,	,	G	5
14	14	que	who that whom which	D	2
15	15	tenían	"were have" "have" "were hold" hold "were thinking" thought "were considering" considered "were deeming" deemed "were coming" came	G	5
16	16	en	in on onto at by	D	2
17	17	promedio	average mean middle midpoint mid-point	G	5
18	18	13	13	L	15
19	21	años de vuelo	"years of experience with space flight" "flight activities" "of years"	E	8.8
22	22	.	.	D	2

Figure 2: Chart walk results

used. The terminating step of this recursion is thus getting components from the chart.

```
To find best walk on a region:

if there is a stored result for this region
    then return it
    else
    begin
    get all primitive components for the region
    for each position p within the region
        begin
        split region into two parts at p
        find best walk for first part
        find best walk for second part
        combine into a component
        end
    find maximum score over all primitive
                        and combined components
    store and return it
    end
```

Figure 3: Chart walk algorithm

Without dynamic programming, this would have a combinatorial time complexity. Dynamic programming utilizes a large array to store partial results, so that the best cover of any given subsequence is only computed once; the second time that a recursive call would compute the same result, it is retrieved from the array instead. This reduces the time complexity to $O(n^3)$, and in practice it uses an insignificant part of total processing time.

All possible combinations of components are compared: this is not a heuristic method, but an efficient exhaustive one. This is what assures that the chosen cover is optimal. This assumes, in addition to the scores actually being correct, that the scores are compositional, in the sense that the combined score for a set of components really represents their quality as a group. This might not be the case, for example, if gaps or overlaps are allowed in some cases (perhaps where they contain the same words in the same positions).

We calculate the combined score for a sequence of components as the *weighted* average of their individual scores. Weighting by length is necessary so that the same components, when combined in a different order, produce the same combined scores. Otherwise the algorithm can produce inconsistent results.

The chart walk algorithm can also be thought of as filling in the two-dimensional dynamic-programming array[1]. Figure 4 shows an intermediate point in the filling of the array. In this figure, each element (i,j) is initially the best score of any single chart component covering the input region from word i to word j. Dashes indicate that no one component covers ex-

[1] Note that this array is a different data structure from the chart.

.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0	5	10	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
1		2.5	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
2			2	--	.83	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
3				5	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
4					5	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
5						2	.25	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
6							5	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
7								5	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
8									2	3.5	7.3	--	--	--	--	--	--	--	--	--	--	--	--
9										5	10	7.3	6.1	5.9	5.2	5.2	4.8	4.8	5.8	5.7	5.4	6.5	6.2
10											2	2.0	2.1	2.8	2.7	3.0	2.9	3.1	4.5	4.5	4.3	5.5	5.3
11												2	2.2	3.1	2.8	3.3	3.0	3.3	4.8	4.8	4.5	5.9	5.5
12													2.5	3.7	3.1	3.6	3.3	3.5	5.2	5.1	4.8	6.3	5.9
13														5	3.5	4.0	3.5	3.8	5.6	5.5	5.1	6.7	6.2
14															2	3.5	3.0	3.5	5.8	5.6	5.1	6.9	6.3
15																5	3.5	4.0	6.7	6.4	5.6	7.6	6.9
16																	2	3.5	7.3	6.7	5.8	8.0	7.2
17																		5	10.	8.3	6.7	9.3	8.0
18																			15	10.	7.3	10.	8.7
19																				5	3.5	8.8	7.1
20																					2	3.5	3.0
21																						5	3.5
22																							2

Figure 4: Triangular array filled in through (8,10) by chart walk

.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0	5	10	7.3	6.7	6.4	5.6	5.5	5.5	5.1	5.1	6.0	5.6	5.4	5.3	5.1	5.1	4.9	4.9	5.5	5.4	5.3	5.9	5.7
1		2.5	2.2	3.1	3.6	3.3	3.5	3.7	3.5	3.7	4.8	4.5	4.4	4.4	4.2	4.3	4.1	4.2	4.8	4.8	4.7	5.4	5.2
2			2	3.5	4.0	3.5	3.8	4.0	3.7	3.8	5.1	4.8	4.5	4.6	4.4	4.4	4.3	4.3	4.9	4.9	4.8	5.5	5.3
3				5	5.0	4.0	4.2	4.4	4.0	4.1	5.5	5.1	4.8	4.8	4.6	4.6	4.4	4.5	5.1	5.1	4.9	5.7	5.5
4					5	3.5	4.0	4.2	3.8	4.0	5.5	5.1	4.8	4.8	4.5	4.6	4.4	4.4	5.1	5.1	4.9	5.7	5.5
5						2	3.5	4.0	3.5	3.8	5.6	5.1	4.8	4.8	4.5	4.5	4.3	4.4	5.1	5.1	4.9	5.8	5.6
6							5	5.0	4.0	4.2	6.4	5.6	5.2	5.1	4.8	4.8	4.5	4.6	5.4	5.3	5.1	6.0	5.8
7								5	3.5	4.0	6.7	5.8	5.2	5.2	4.8	4.8	4.5	4.5	5.4	5.4	5.1	6.1	5.8
8									2	3.5	7.3	6.0	5.3	5.2	4.7	4.8	4.5	4.5	5.5	5.4	5.1	6.2	5.9
9										5	10	7.3	6.1	5.9	5.2	5.2	4.8	4.8	5.8	5.7	5.4	6.5	6.2
10											2	2.0	2.1	2.8	2.7	3.0	2.9	3.1	4.5	4.5	4.3	5.5	5.3
11												2	2.2	3.1	2.8	3.3	3.0	3.3	4.8	4.8	4.5	5.9	5.5
12													2.5	3.7	3.1	3.6	3.3	3.5	5.2	5.1	4.8	6.3	5.9
13														5	3.5	4.0	3.5	3.8	5.6	5.5	5.1	6.7	6.2
14															2	3.5	3.0	3.5	5.8	5.6	5.1	6.9	6.3
15																5	3.5	4.0	6.7	6.4	5.6	7.6	6.9
16																	2	3.5	7.3	6.7	5.8	8.0	7.2
17																		5	10.	8.3	6.7	9.3	8.0
18																			15	10.	7.3	10.	8.7
19																				5	3.5	8.8	7.1
20																					2	3.5	3.0
21																						5	3.5
22																							2

Figure 5: Final array produced by chart walk

actly that region. (In rows 1 through 7, the array has not yet been operated on, so it still shows its initial state.) After processing (see rows 9 through 22), each element is the score for the best *set* of components covering the input from word i to word j (the best cover for this substring)[2]. (Only a truncated score is shown for each element in the figure, for readability. There is also a list of best components associated with each element.) The array is upper triangular since the starting position of a component i must be less than or equal to its ending position j.

For any position, the score is calculated based on a combination of scores in the row to its left and in the column below it, versus the previous contents of the array cell for its position. So the array must be filled from the bottom-up, and left to right. Intuitively, this is because larger regions must be built up from smaller regions within them.

For example, to calculate element (8,10), we compute the length-weighted averages of the scores of the best walks over the pair of elements (8,8) and (9,10) versus the pair (8,9) and (10,10), and compare them with the scores of any single chart components going from 8 to 10 (there were none), and take the maximum. Referring to Figure 2 again, this corresponds to a choice between combining the translations of (8,8) *VIASA* and (9,10) *contaba con* versus combining the (not shown) translations of (8,9) *VIASA contaba* and (10,10) *con*. (This (8,9) element was itself previously built up from single word components.) Thus, we compare $(2*1+10*2)/3 = 7.33$ with $(3.5*2+2*1)/3 = 3.0$ and select the first, 7.33. The first wins because *contaba con* has a high score as an idiom from the glossary.

Figure 5 shows the final array. When the element in the top-right corner is produced (5.78), the algorithm is finished, and the associated set of components is the final chart walk result shown in Figure 2.

It may seem that the scores should increase towards the top-right corner. This has not generally been the case. While the system produces a number of high-scoring short components, many low-scoring components have to be included to span the entire input. Since the score is a weighted *average*, these low-scoring components pull the combined score down. A clear example can be seen at position (18,18), which has a score of 15. The scores above and to its right each average this 15 with a 5, for total values of 10.0 (all the lengths happen to be 1), and the score continues to decrease with distance from this point as one moves towards the final score, which does include the component for (18,18) in the cover.

[2]In the actual implementation, the initial components are not present yet in the array, since the presence of an element indicates that the computation has been carried out for this position. They are accessed from the chart data structure as needed, but are shown here as an aid to understanding.

2.3 Reordering components

The chart-oriented integration of MT engines does not easily support deviations from the linear order of the source text elements, as when discontinuous constituents translate contiguous strings or in the case of cross-component substring order differences. We use a language pair-dependent set of postprocessing rules to alleviate this (for example, by switching the order of adjacent single-word adjective and noun components).

3 TRANSLATION DELIVERY SYSTEM

Results of multi-engine MT were fed in our experiment into a translator's workstation (TWS) (Cohen et al., 1993), through which a translator either approved the system's output or modified it. The main option for human interaction in TWS currently is the Component Machine-Aided Translation (CMAT) editor (Frederking et al., 1993a). The user sees the original source language text in one editor window, and phrases marked by double angle brackets in another, each of which is the first translation from a candidate chosen by the chart walk. Menus, function keys and mouse clicks are used to perform both regular and enhanced editing actions.

The most important enhancement provided is the ability to select an alternate translation with a popup menu, and instantly replace the system's initially chosen candidate translation string, which becomes the first alternative in this menu if it is used again. The alternate translations are the other translations from the chosen component[3].

As mentioned above, Figure 2 shows the sets of candidates in the best chart walk that are presented as choices to the human user through the CMAT editor in our example.

4 TESTING AND EVALUATING MULTI-ENGINE PERFORMANCE

Automatically assessing the utility of the multi-engine system relative to the engines taken separately would be a useful development tool. The best method we could find was counting the number of keystrokes in the TWS to convert the outputs of individual engines and the multi-engine configuration to a "canonical" human translation. A sample test on a passage of 2060 characters from the June 1993 evaluation of Pangloss is shown in figure 6.

The difference in keystrokes was calculated as follows: one keystroke for deleting a character; two

[3]The CMAT editor may also include translations from other candidates, lower in the menu, if they have the same boundaries as the chosen candidate and the menu is not too long.

Type of translation	Keystroke difference
human tester (US Government Level 2 translator)	1542
word-for-word lookup in MRDs	1829
lookup in phrasal glossaries	1973
KBMT	1883
Example-Based MT	1876
Multi-engine configuration	1716

Figure 6: Results of keystroke test

keystrokes for inserting a character; three keystrokes for deleting a word (in an editor with mouse action); three keystrokes plus the number of characters in the word being inserted for inserting a word. It is clear from the above table that the multi-engine configuration works better than any of our available individual engines, though it still does not reach the quality of a Level 2 translator.

It is also clear that using keystrokes as a measure is not very satisfactory. It would be much better to make the comparison against the *closest* member of a set of equivalent paraphrastic translations, since there are many "correct" ways of translating a given input. However, this is predicated on the availability of a "paraphraser" system, developing which is not a trivial task.

5 CURRENT AND FUTURE WORK

Ultimately, a multi-engine system depends on the quality of each particular engine. We expect the performance of KBMT and EBMT to grow. We plan to use a standard regression mechanism to modify the scoring system based on feedback from having humans select the best covers for test texts.

The current system is human-aided. We have begun an experiment with a fully-automated mode, with the understanding that the quality will drop. The most important effect of this change is that accurate quality scores become much more important, since the first choice becomes the only choice. Besides improving the KBMT and EBMT scoring mechanisms, we need to provide finer distinctions for the lexical transfer engine's output. As the databases for this are quite large (all together, over 400,000 entries), adding scores to individual entries is, in the short run, prohibitive. We have not as yet discovered any feasible automatic technique for generating such scores. Instead, we are planning to use an English language model on the output, in a manner similar to that done by speech and statistical translation systems (Brown et al., 1990). Statistically generating such a model *is* feasible, since it does not rely on knowing correspondences between source and target languages. It is a weaker approach, but should go some distance in selecting between otherwise indistinguishable outputs.

Another possible direction for future development would be to employ ideas from the area of heuristic search, and only run the highest-quality-score engine on each unit of source text. This assumes that we can reliably estimate scores in advance (not currently true for the expensive engines), and that the engines can be run on fragments. A less ambitious version of this idea would be to run the low-scoring engines only where there are gaps in the normally high-scoring engines.

References

Brown, P., K. Cocke, S. Della Pietra, V.J. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer and P.S. Roossin. "A statistical approach to Machine Translation", *Computational Linguistics* 16, pp.79-85, 1990.

Cohen, A., Cousseau, P., Frederking, R., Grannes, D., Khanna, S., McNeilly, C., Nirenburg, S., Shell, P., Waeltermann, D. *Translator's WorkStation User Document*, Center for Machine Translation, Carnegie Mellon University, 1993.

Frederking, R., Grannes, D., Cousseau, P., and Nirenburg, S. "An MAT Tool and Its Effectiveness." In Proceedings of the DARPA Human Language Technology Workshop, Princeton, NJ, 1993.

Frederking, R., A. Cohen, P. Cousseau, D. Grannes and S. Nirenburg. "The Pangloss Mark I MAT System." Proceedings of EACL-93, Utrecht, The Netherlands, 1993.

Nagao, M. "A framework of a mechanical translation between Japanese and English by analogy principle." In: A. Elithorn and R. Banerji (eds.) *Artificial and Human Intelligence*. NATO Publications, 1984.

Nirenburg, S., C. Domashnev and D.J. Grannes. "Two Approaches to Matching in Example-Based Machine Translation." Proceedings of TMI-93, Kyoto, 1993.

Nirenburg, S., S. Beale and C. Domashnev. "A Full-Text Experiment in Example-Based Machine Translation." Submitted to the International Conference on New Methods in Language Processing, Manchester, September 1994.

Nirenburg, S., S. Beale, C. Domashnev and P. Sheridan. "Example-Based Machine Translation of Running Text." *In preparation.*