INTEGRATING TRANSLATIONS FROM MULTIPLE SOURCES WITHIN THE PANGLOSS MARK III MACHINE TRANSLATION SYSTEM

Robert Frederking[1], Sergei Nirenburg[2], David Farwell[2], Stephen Helmreich[2], Eduard Hovy[3], Kevin Knight[3], Stephen Beale[3], Constantine Domashnev[1], Donalee Attardo[1], Dean Grannes[1], Ralf Brown[1]

1) Center for Machine Translation
Carnegie Mellon University
Schenley Park
Pittsburgh, PA 15213-3890

2) Computing Research Laboratory
New Mexico State University
Box 30001 / 3CRL
Las Cruces, NM 88003

3) Information Sciences Institute
of the University of S. California
4676 Admiralty Way
Marina del Rey, CA 90292

Abstract

Since MT systems, whatever translation method they employ, do not reach an optimum output on free text; each method handles some problems better than others. The PANGLOSS Mark III system is an MT environment that uses the best results from a variety of independent MT systems or engines working simultaneously within a single framework on the same text. This paper describes the method used to combine the outputs of the engines into a single text.

1 Introduction

PANGLOSS Mark III is a multi-engine Spanish-to-English MT system. The PANGLOSS team is distributed at three sites — the Computing Research Laboratory of New Mexico State University, the Information Sciences Institute of the University of Southern California and the Center for Machine Translation of Carnegie Mellon University. Originally, PANGLOSS was supposed to be a pure knowledge-based machine translation (KBMT) system implemented in a version of the interlingua architecture. The project, however, evolved toward a more eclectic approach, mostly due to the necessity to perform well during periodical external evaluations whose timing and frequency was established after the project started. As KBMT systems rely on extensive knowledge bases, their gestation periods are typically longer than those of other kinds of systems. The first system configuration, the PANGLOSS Mark I system, was pure KBMT and did not perform well on the first evaluation in Summer 1992. The project team then decided to channel some of the resources into developing an interim evaluation system which would show an immediate improvement in output quality, while in parallel continuing to develop the "mainline" KBMT system. The idea was that at a certain stage the KBMT system would supplant the interim system. Thus, PANGLOSS Mark II was a simple lexical transfer system based on phrasal bilingual glossaries and machine-readable dictionaries. The evaluation results were better than during the first evaluation (White and O'Connell, 1994).

As our understanding of the MT task continued to evolve, it was decided not to discard the "interim" system in favor of the "mainline" one but rather to make these systems coexist and even include additional MT systems. In fact, our current system, PANGLOSS Mark III, differs from other MT systems because it employs not a single translation engine but a set of several engines whose results are integrated for the best overall output.

PANGLOSS Mark III contains three distinct MT engines:

- a KBMT system, the mainline Pangloss engine;
- an example-based MT (EBMT) system (Nirenburg et al., 1993; the original idea is due to 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.

In what follows, we first describe the multi-engine architecture and the nature of engine integration in PANGLOSS Mark III. Then we describe the individual engines, focusing primarily on KBMT. Finally we discuss two extensions to the project, the handling of Japanese to English translation and a project to the quality of language analysis via the development of computational-linguistic microtheories.

2 Multi-Engine System Architecture

2.1 Obtaining and Scoring the Outputs of MT Engines

A number of proposals have come up in recent years for hybridization of MT. Current MT projects — both "pure" and hybrid, both predominantly technology-oriented and research-oriented — are single-engine projects, capable of one particular type of source text analysis, one particular method of finding target language correspondences for source language elements and one prescribed method of generating the target language text.

It is common knowledge that MT systems, whatever translation method they at present employ, do not reach an optimum 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 or the reliance of most transfer-oriented MT systems on similarities in syntactic structures of the source and the target languages. Another crucial source of deficiencies is the size and quality of the static knowledge sources underlying the various MT systems — particular grammars, lexicons and world models. Thus, in knowledge-based MT the size of the underlying world model is typically smaller than necessary for secure coverage of free text.

The PANGLOSS Mark III system of January 1994 is an MT environment which uses the best results from a variety of MT systems working simultaneously on the same text. In this novel multi-engine MT approach, we submit an input text to a battery of independent machine translation systems (engines). The results obtained from the individual engines (target language words and phrases) are then recorded, jointly, in a chart whose initial edges correspond to words in the source language input. New edges are added to the chart and labeled with the translation of a segment of the input string and indexed by this segment's beginning and end positions. In addition, a quality score is added to each edge. Using the normalized quality score, the chart manager selects the overall best cover from a collection of candidate partial translations by the "chart-walk" algorithm.

The KBMT and EBMT engines provide a score for each output element, based on their internal confidence in the quality of its translation. For the Lexical Transfer Engine, the score for each glossary is a constant based on the reliability of the glossary. The scores are also normalized so as to be comparable. Finally, during chart construction, the base score produced by the scoring functions is multiplied by the length of the candidate in words, on the assumption that longer items are better.

2.2 The Chart-Walk Algorithm

Once the edges are scored, the cover is produced using a simple dynamic programming algorithm. The figure below presents the chart-walk algorithm used to produce a single, best, non-overlapping, contiguous combination of the available component translations. The algorithm uses dynamic programming to find the optimal cover (a cover with the best cumulative score), assuming correct component quality scores. The code is organized as a recursive divide-and-conquer procedure: for each position within a segment, the sentence is split into two parts, 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. This primitive step is repeated for each possible top-level split of the input sentence, compared with each other and with any simple edges (from the chart) spanning the segment, and the overall best result is used.

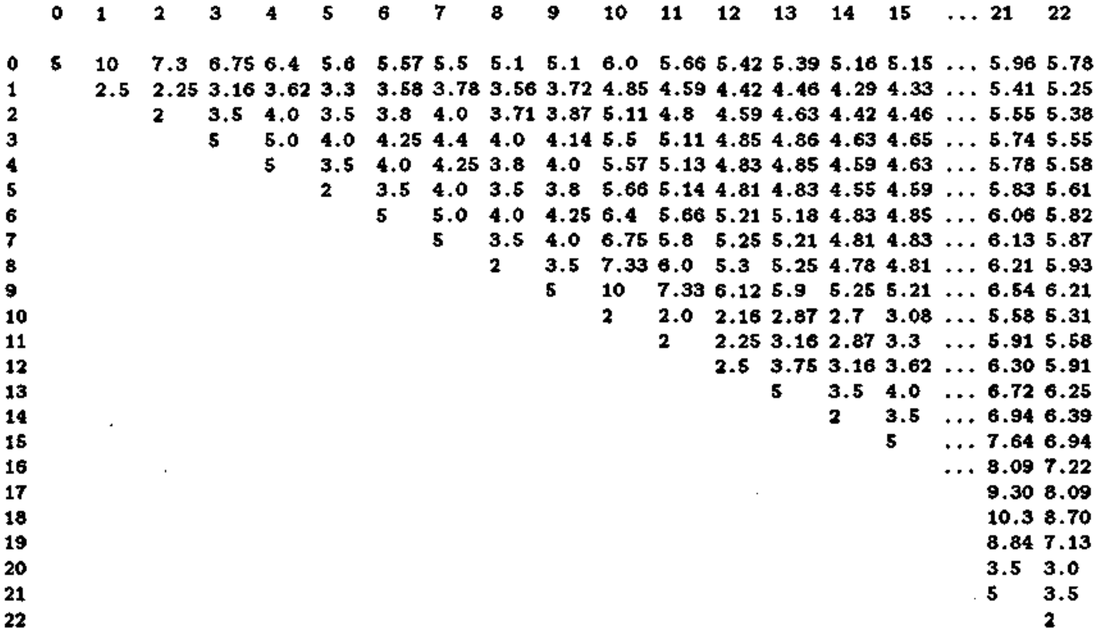
To find best walk on a segment:

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

Without dynamic programming, this algorithm 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 $0(n^3)$, and in practice it uses an insignificant part of total processing time.

The combined score for a sequence of edges is the weighted average of their individual scores. Weighting by length is necessary so that the same edges, when combined in a different order, produce the same combined scores. In other words, whether edges a, b and c are combined as ((a b) c) or (a (b c)), the combined edge must have the same score, or the algorithm can produce inconsistent results.

The chart-walk algorithm can also be visualized as a task of filling a two-dimensional array. The array for our example sentence is shown in the figure below. Element (i,j) of the array is the best score for any set of edges covering the input from word i to word j. (The associated list of edges is not shown, for readability.) For any position, the score is calculated as a weighted average of the scores in the row to its left, in the column below it and the previous contents of the array cell for its position. So to calculate element (1,4), we compare the combined scores of the best walks over (1,1) and (2,4), (1,2) and (3,4), and (1,3) and (4,4) with the scores of any chart edges going from 1 to 4, and take the maximum. When the score in the top-right corner is produced, the algorithm is finished and the associated set of edges is the final chart-walk result.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...	21	22
0	5	10	7.3	6.75	6.4	5.6	5.57	5.5	5.1	5.1	6.0	5.66	5.42	5.39	5.16	5.15	...	5.96	5.78
1		2.5	2.25	3.16	3.62	3.3	3.58	3.78	3.56	3.72	4.85	4.59	4.42	4.46	4.29	4.33	...	5.41	5.25
2			2	3.5	4.0	3.5	3.8	4.0	3.71	3.87	5.11	4.8	4.59	4.63	4.42	4.46	...	5.55	5.38
3				5	5.0	4.0	4.25	4.4	4.0	4.14	5.5	5.11	4.85	4.86	4.63	4.65	...	5.74	5.55
4					5	3.5	4.0	4.25	3.8	4.0	5.57	5.13	4.83	4.85	4.59	4.63	...	5.78	5.58
5						2	3.5	4.0	3.5	3.8	5.66	5.14	4.81	4.83	4.55	4.59	...	5.83	5.61
6							5	5.0	4.0	4.25	6.4	5.66	5.21	5.18	4.83	4.85	...	6.06	5.82
7								5	3.5	4.0	6.75	5.8	5.25	5.21	4.81	4.83	...	6.13	5.87
8									2	3.5	7.33	6.0	5.3	5.25	4.78	4.81	...	6.21	5.93
9										5	10	7.33	6.12	5.9	5.25	5.21	...	6.54	6.21
10											2	2.0	2.16	2.87	2.7	3.08	...	5.58	5.31
11												2	2.25	3.16	2.87	3.3	...	5.91	5.58
12													2.5	3.75	3.16	3.62	...	6.30	5.91
13														5	3.5	4.0	...	6.72	6.25
14															2	3.5	...	6.94	6.39
15																5	...	7.64	6.94
16																	...	8.09	7.22
17																		9.30	8.09
18																		10.3	8.70
19																		8.84	7.13
20																		3.5	3.0
21																		5	3.5
22																			2

It may seem that the scores should increase towards the top-right corner. In our experience, however, this has not generally been the case. Indeed, the system suggested a number of high-scoring short edges, but many low-scoring edges had to be included to span the entire input. Since the score is a weighted AVERAGE, these low-scoring edges pull it 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 and the score continues to decrease with distance from this point as one moves towards the final score, which does include (18,18) in the cover.

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-segmental substring order differences. Following a venerable tradition in MT, we used a target language-dependent set of postprocessing rules to alleviate this problem (e.g., by switching the order of adjectives and nouns in a noun phrase if it was produced by the word-for-word engine).

2.4 Translation Delivery System

In normal operation, results of multi-engine MT are fed in PANGLOSS into a translator's workstation (TWS) (Cohen et al., 1993) through which a translator either approves the system's output or modifies it. The main option for human interaction in TWS currently is the Component Machine-Aided Translation (CMAT) editor (Frederking et al., 1993), in which the user can use menus, function keys, and mouse clicks to change the system's initially chosen candidate translation string, as well as perform both regular and enhanced editing actions.

3 Knowledge-Based MT Engine

The mainline engine of PANGLOSS is the knowledge-based engine. It consists of an analyzer, called the PANGLYZER and a generation module centered on the generator called PENMAN. Unfortunately, given space limitations, we can provide only a brief overview here; please see (PANGLOSS 1994) for a more comprehensive account.

3.1 Panglyzer: Spanish Language Analysis System

The function of the Spanish analysis component of the PANGLOSS system, or Panglyzer, is to provide for each clause in the input text a set of possible meaning representations ranked on the basis of likelihood. The system is described in more detail in (Farwell et al. 1994). For this effort the focus is placed on Spanish language newspaper articles in the area of finance, specifically mergers and acquisitions. Currently, the output is a sequence of sets of ranked partial meaning representations corresponding at times to the words, at times to the phrases and at times to the clauses of the input text. The output is passed to a mapper which converts Panglyzer representations to representation appropriate for generation.

The approach has been to develop the system in a bottom up manner, providing layer after layer of increasingly abstract analysis in a multi-pass process. Each level of analysis is based on a focused type of knowledge and, to the extent possible, exploits proven techniques. Since a high premium has been placed on robustness, that is, producing some throughput even if it is partially correct, an iterative approach to design has been used which relies on rapidly producing an initial prototype and then following a short test and revise cycle. Thus, at this point, all but the deepest level of analysis produces throughput and the on-going objective is to improve the accuracy of that throughput from one test cycle to the next.

3.2 Panglyzer-to-Penman Mapper

In the eventual PANGLOSS, semantic analysis will constitute a significant module in the pathway from source text to Interlingua form, involving several tasks, including mapping of syntactic and semantic information as produced by the Panglyzer into basic TMR propositions, reference resolution, metonymy, discourse structure building at the paragraph level, and so on. At present, however, mainly the basic proposition construction portion has been addressed, using a unification-based bottom-up chart parser with approx. 500 rules.

3.3 Penman: English Sentence and Phrase Generation

PENMAN is of the largest English sentence generator programs available. A detailed overview of language generation in Penman can be found in Matthiessen and Bateman (1991).

Penman consists of a number of components. Nigel, the English grammar, is the heart of the system. Based on the theory of Systemic Functional Linguistics (a theory of language and communication, and used in various AI applications, such as in SHRDLU, Nigel is a network of over 700 nodes called "systems", each node representing a single minimal grammatical alternation. In order to generate a sentence, Penman traverses the network guided by its inputs and default settings. At each system node, Penman selects a feature until it has assembled enough features to fully specify a sentence. After constructing a syntax tree and choosing words to satisfy the features selected, Penman generates the English sentence. The Nigel grammar is described in, among others, Matthiessen (1984).

The taxonomy of generalizations is called the Penman Upper Model and can be seen as a very general taxonomic model of the entities, objects, qualities and relations in the world (Bateman et al., 1989). This taxonomy acts to link the terms in a user's application domain to the terms used and decision made within Penman. Within PANGLOSS, the Upper Model is embedded in the Ontology Base and ensures that any Ontology term used in an input to Penman will be handled correctly in creating an appropriate English sentence or phrase.

4 Example-Based MT Engine

The basic idea of EBMT is simple (Nagao, 1984): given an input passage S in a source language and a bilingual text archive, where text passages S' in the source language are stored, aligned with their translations into a target language, passages T', S is compared with the source-language "side" of the archive. The "closest" match for passage S' is selected and the translation of this closest match, the passage T' is accepted as the translation of S.

EBMT steps include the following:

1. align corpus at sentence level;
2. find chunks from the source language part of corpus which are best candidates for matching an input chunk (intra-language matching);

3. find the target language chunk corresponding to the chunk from the source language part of the corpus (inter-language matching);
4. combine chunk-level results to obtain the "cover" for the entire text.

5 Lexical Transfer MT Engine

PANGLOSS uses a simple and traditional lexical transfer MT engine as a safety net. Lexical transfer is carried out using a number of bilingual resources: the lexicons developed as an aid in the KBMT engine; a machine readable dictionary (Spanish-English Collins) and a set of manually produced glossaries. PANGLOSS Spanish-to-English glossaries were improved in 3 ways between the May 1993 and January 1994 evaluations:

- Pure size: glossaries grew from 68,000 to 174,000 entries between October and December 1993;
- Cleaning: mass effort by the glossary acquisition team to rid all glossaries of useless grammatical information, correct inaccurate entries, etc.;
- Variables: mass effort by the glossary acquisition team to allow the glossary entries to use coindexed variables.

To allow matching on "open" patterns, variables were introduced into the glossary entries for proper names, such as individual, company and place names; numbers; and various classes of pronouns.

6 Conclusion and Additional Activities

Ultimately, a multi-engine system depends on the basic quality of each particular engine. We expect the performance of some of the individual engines (especially, KBMT and EBMT) to grow. Consequently, the multi-engine environment will improve, as larger static knowledge sources are added and the scoring mechanism is further adjusted. In addition, we are currently working on adding a statistical English language model, to allow PANGLOSS to operate as a fully-automatic MT system.

PANGLOSS is being extended not only from the standpoint of system architecture. Starting in December 1993, considerable effort has been devoted at ISI on the development of Japanese parsing capabilities for PANGLOSS. The overall goal is to test and strengthen the interlingual nature of the Ontology and to demonstrate the utility of the PANGLOSS framework for the incorporation of new languages. The work on Japanese falls into two major areas: KBMT system module development and resource construction. A mixture of statistical and symbolic methods are used to attain coverage, robustness, and quality in a short amount of time. This work is described in (Knight et al., 1994).

The Mikrokosmos project is an extension of PANGLOSS devoted to a study of semantic and pragmatic aspects of texts. It is not realistic to hope for the development of a single all-encompassing theory of computational linguistics. However, high-quality applications require knowledge about a large number of language and language use phenomena. A natural way of combining the diverse knowledge into a single entity is to allow for the various phenomena to be treated by separate computational-linguistic "microtheories" united through a system's control architecture and knowledge representation conventions. We perceive the following microtheories as central for the support of knowledge-based machine translation (and other high-demand applications): lexical-semantic dependency, aspect, time, modality and other speaker attitudes, discourse relations, reference, style, and spatial description.

References

[Bateman et al. 89] Bateman, J.A., R.T. Kasper, J.D. Moore and R.A. Whitney. A General Organization of Knowledge for Natural Language Processing: The Penman Upper Model. Unpublished research report, USC/Information Sciences Institute, Marina del Rey, 1989.

[Cohen et al. 93] Cohen, A., P. Cousseau, R. Frederking, D. Grannes, S. Khanna, C. McNeilly, S. Nirenburg, P. Shell and D. Waeltermann. Translator's Workstation User Document, Center for Machine Translation, Carnegie Mellon University, 1993.

[Frederking et al. 93] Frederking, R., D. Grannes, P. Cousseau, and S. Nirenburg. An MAT Tool and Its Effectiveness. *Proceedings of the DARPA Human Language Technology Workshop,* Princeton, NJ, 1993.

[Knight et al. 94] Knight, K., I. Chander, M. Haines, V. Hatzivassiloglou, E. Hovy, M. Iida, S. Luk, A. Okumura, R. Whitney, K. Yamada. Integrating Knowledge Bases and Statistics in MT. *Proceedings of the 1st AMTA Conference,* Columbia MD, October 1994.

[Matthiessen 84] Matthiessen, C.M.I.M. Systemic Grammar in Computation: The Nigel Case. *Proceedings of 1st Conference of the European Association for Computational Linguistics,* Pisa. Also available as USC/ISI Research Report RR-84-121, 1984.

[Matthiessen & Bateman 91] Matthiessen, C.M.I.M. and J.A. Bateman. *Systemic-Functional Linguistics in Language Generation: Penman.* 1991.

[Nagao 84] 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 et al. 93] Nirenburg, S. C. Domashnev and D.J. Grannes. Two Approaches to Matching in Example-Based Machine Translation. *Proceedings of TMI-93,* Kyoto, 1993.

[PANGLOSS 94] The Pangloss Machine Translation System. Joint Technical Report, Computing Research Laboratory (New Mexico State University), Center for Machine Translation (Carnegie Mellon University), Information Sciences Institute (University of Southern California). 1994.

[White & O'Connel 94] White, J.S. and T. O'Connell. Evaluation in the ARPA Machine Translation Program: 1993 Methodology. *Proceedings of the ARPA HLT Workshop.* Plainsboro, NJ. March, 130-133,1994.