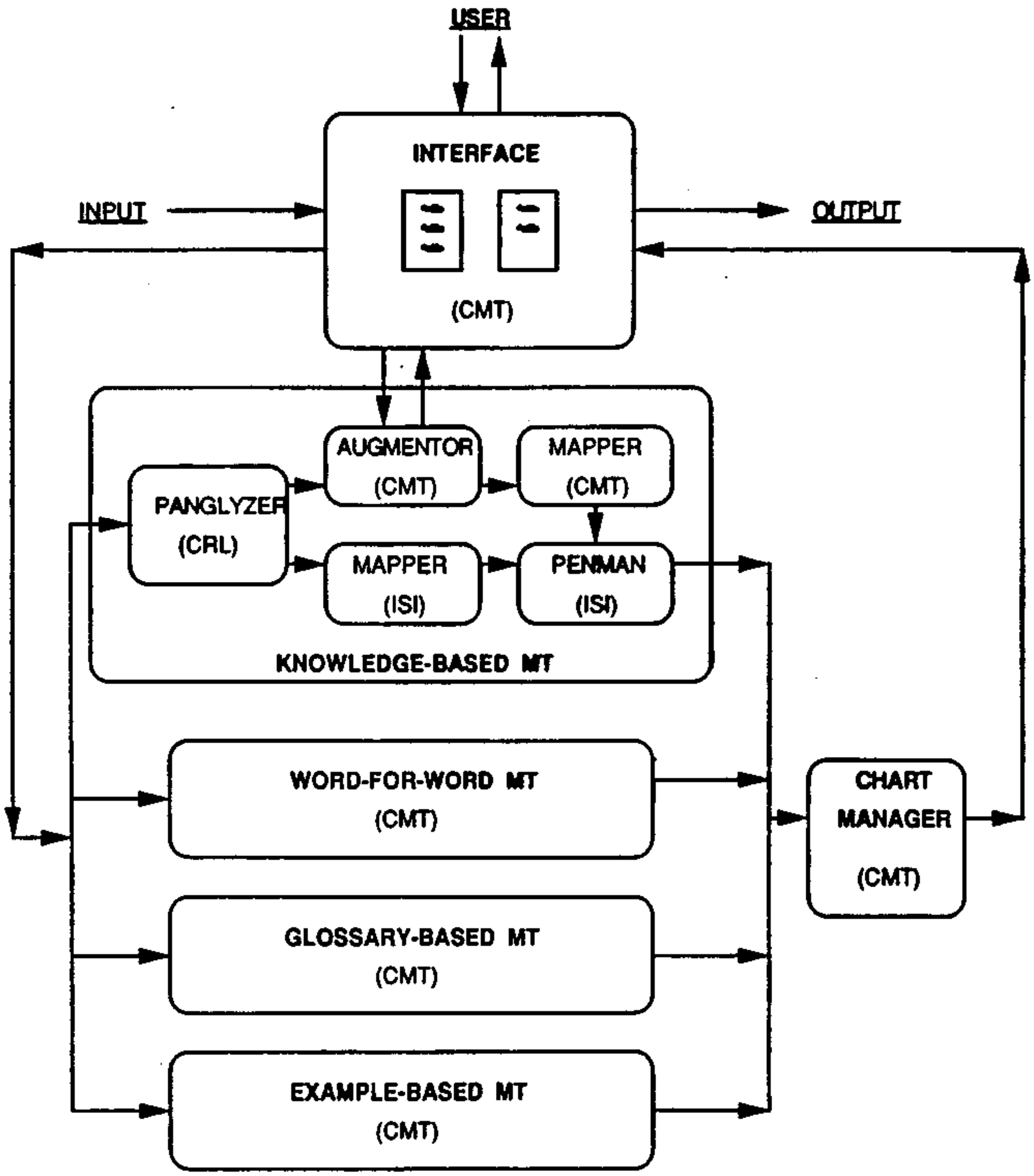
The Structure of the Pangloss System



Pangloss Chart Manager

- **Function:** Keep track of translations from all translation engines; select which to present to user for CMAT editing
- **Input:** Component translations from all translation engines, each marked with type, sentence positions, and (optionally) internal score and other information
- **Output:** List of CMAT component lists
- **Procedures and Resources:** Chart structure, scoring functions, chart walk procedure

Basic Idea

Different MT techniques work better or worse on different parts of a text, so:

- Use multiple MT engines to translate the source text, each using its own segmentation
- Results are all put into one chart
- Each candidate translation is scored
- Best cover of source is selected and presented to user for post-editing
- Other candidates available on request to post-editor

Chart structure

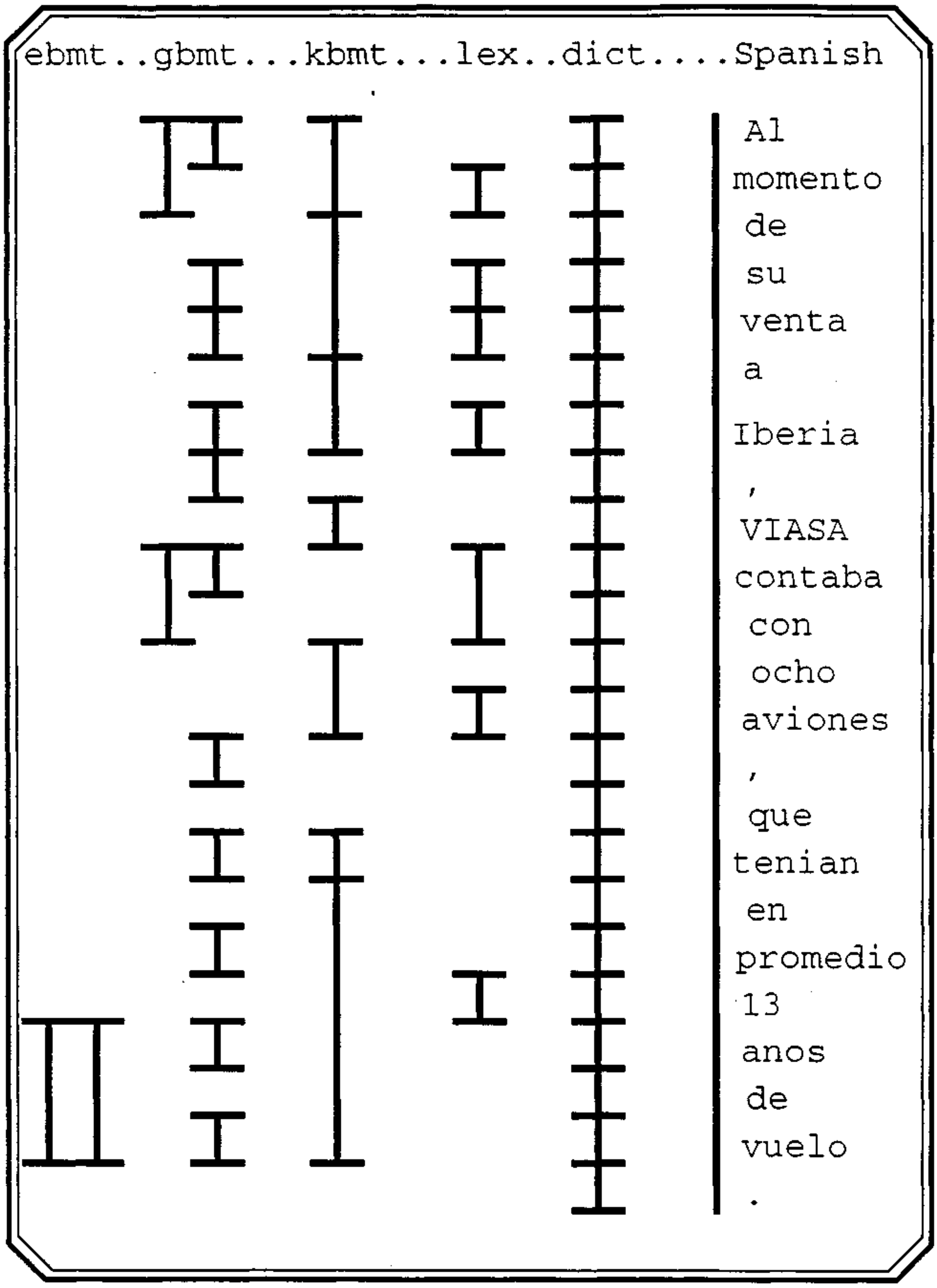
A structure containing

- an array of analyzed source words:

```
#S(WORD :STRING "aviones" :ROOT "avi n"
        :MORPH (NOUN MASCULINE PLURAL))
```

- and an array containing lists of translation edges:

```
#S(CHART-ENTRY :TYPE :DICT :START 12 :END 12 :LENGTH 1
               :INFO
               ("aviones" ("aeroplanes" "planes"
                           "aircrafts" "airplanes"
                           "martins" "hopscotches") 2)))
```



ebmt..gbmt...kbmt...lex..dict....Spanish
Al
momento
de
su
venta
a
Iberia
,
VIASA
contaba
con
ocho
aviones
,
que
tenian
en
promedio
13
anos
de
vuelo
.

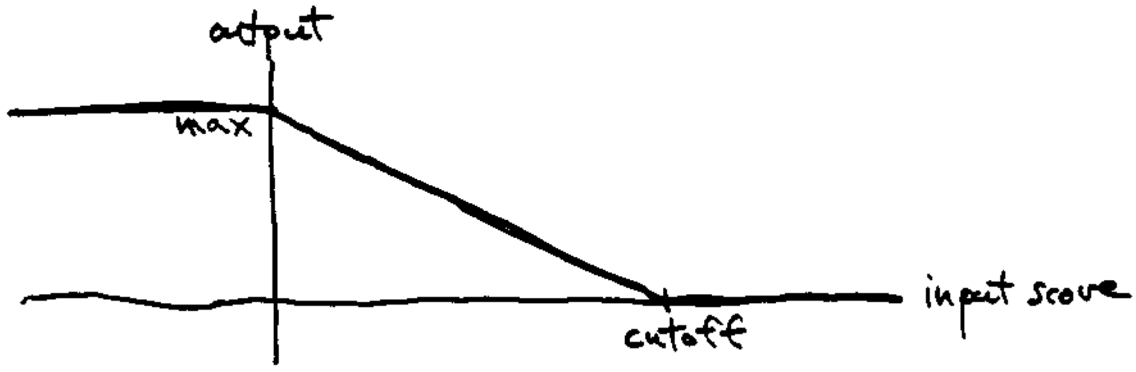
Scoring functions

- Deriving initial scores: length * base score
  - Default base scores:

`*null-dict-score*`	0.5
`*default-dict-score*`	2
`*default-mtlex-score*`	2.5
`*default-gloss-score*`	5
`*converted-number-score*`	15
`*user-selected-arc-score*`	100

  - EBMT scoring: need to linearize, using two constants:

`*max-ebmt-chart-score*`	8
`*ebmt-input-score-cutoff*`	20



  - KBMT scoring: known bad phrases get 0, otherwise decrement by 1
  - **In serious need of tuning!!**
- Combining scores: weighted average (by length) of sub-edges

Chart walk

- Necessitated by need for unique segmentation in CMAT
- Dynamic-programming algorithm (pseudo-recursive):

```
To find best walk on a segment:

if there is a stored result for this segment, return it
    else
    begin
    get all primitive edges from chart 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 primitive and combined edges
    store and return it
    end
```

- Other edges on same interval included, sorted by score, up to a maximum number of CMAT alternatives

.	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	6.75	6.4	5.6	5.57	5.5	5.1	5.1	6.0	5.66	5.42	5.39	5.16	5.15	4.97	4.97	5.5	5.47	5.31	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	4.19	4.24	4.83	4.84	4.7	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	4.3	4.34	4.97	4.97	4.82	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	4.46	4.5	5.16	5.15	4.97	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	4.42	4.46	5.16	5.16	4.97	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	4.38	4.42	5.18	5.16	4.97	5.83	5.61
6							5	5.0	4.0	4.25	6.4	5.66	5.21	5.18	4.83	4.85	4.59	4.62	5.42	5.39	5.16	6.06	5.82
7								5	3.5	4.0	6.75	5.8	5.25	5.21	4.81	4.83	4.55	4.59	5.45	5.42	5.17	6.13	5.87
8									2	3.5	7.33	6.0	5.3	5.25	4.78	4.81	4.5	4.55	5.5	5.45	5.19	6.21	5.93
9										5	10	7.33	6.12	5.9	5.25	5.21	4.81	4.83	5.85	5.77	5.45	6.54	6.21
10											2	2.0	2.16	2.87	2.7	3.08	2.92	3.18	4.5	4.55	4.31	5.58	5.31
11												2	2.25	3.16	2.87	3.3	3.08	3.35	4.81	4.83	4.55	5.91	5.58
12													2.5	3.75	3.16	3.62	3.3	3.58	5.21	5.18	4.83	6.30	5.91
13														5	3.5	4.0	3.5	3.8	5.66	5.57	5.12	6.72	6.25
14															2	3.5	3.0	3.5	5.8	5.66	5.14	6.94	6.39
15																5	3.5	4.0	6.75	6.4	5.66	7.64	6.94
16																	2	3.5	7.33	6.75	5.8	8.09	7.22
17																		5	10.0	8.33	6.75	9.30	8.09
18																			15	10.0	7.33	10.3	8.70
19																				5	3.5	8.84	7.13
20																					2	3.5	3.0
21																						5	3.5
22																							2

Output

```
(("Al momento" :GLOSS ("In a minute" "At once" "A moment"))
 ("de" :GLOSS ("of" "from" "about" "for" "by"))
 ("su" :GLOSS ("his" "her" "its" "one's" "your" "their"))
 ("venta" :GLOSS ("inn" "sale" "selling" "marketing" "countı
 ("a" :GLOSS ("to" "a" "of"))
 ("Iberia" :GLOSS ("Iberia"))
 ("," :GLOSS (","))
 ("VIASA" :GLOSS ("VIASA"))
 ("contaba con" :GLOSS ("was rely on" "rely on" "was count c
 ("ocho" :GLOSS ("eight" "eighth"))
 ("aviones" :GLOSS ("airplane" "aeroplanes" "planes" "aircrɛ
 ("," :GLOSS (","))
 ("que" :GLOSS ("who" "that" "whom" "which"))
 ("ten an" :GLOSS
  ("were have" "have" "were hold" "hold" "were thinking" "tl
   "came"))
 ("en" :GLOSS ("in" "on" "onto" "at" "by"))
 ("promedio" :GLOSS ("average" "mean" "middle" "midpoint" "n
 ("13" :GLOSS ("13"))
 ("a os de vuelo" :GLOSS ("years * experience with space fli
 ("." :GLOSS (".")))
```

Pangloss Spanish-to-English Glossaries

Improved in 3 ways:

- 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, see below

Glossary Variables

To allow matching on "open" patterns, variables were introduced into the glossary entries for the following types of words and phrases:

- proper names, such as individual, company and place names
- numbers
- the various classes of pronouns:
  - nominal
  - possessive
  - reflexive
  - direct object
  - indirect object
  - possessive adjectives

Open-Pattern Matching

Elements of Spanish and English sides of the entry were co-indexed as appropriate. For example,

```
absolver<1> a <dop:2> de <poss:2> promesa
   release<1> <dop:2> from <poss:2> promise
```

Partial list of possible glossary-based translation results would include:

```
I release you from your promise
He released me from my promise
You released her from her promise
etc.
```

Tables of correspondences for feature sets (e.g., person, gender, number) were created to facilitate open pattern matching.

The following table offers some comparison of “old” and “new” glossary performance on a sample text of 347 words.

	Hits	Entries Hit	Hits per Entry Hit	Entries Hit / Glossary Size
1993	341	198	1.72	0.0029
1994	464	258	1.80	0.0015

Example-Based MT

The basic idea of EBMT is simple (cf. 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

- align corpus at sentence level;
- match input chunk with a chunk from the source language part of corpus (intra-language matching);
- find the target language chunk corresponding to the chunk from the source language part of the corpus (inter-language matching);
- combine chunk-level results to obtain the "cover" for the entire text.

Candidate Finder

Input: a Spanish sentence and a corpus of Spanish texts.
Output: a list of of the form

```
((input-substring-1
     ((corpus-string-1-1 score-1-1)
      (corpus-string-1-2 score-1-2)
       ...
      (corpus-string-1-10 score-1-10)))

 (input-substring-2
     ((corpus-string-2-1 score-2-1)
      (corpus-string-2-2 score-2-2)
       ...
      (corpus-string-2-10 score-2-10)))
...

 (input-substring-10
     ((corpus-string-10-1 score-10-1)
      (corpus-string-10-2 score-10-2)
       ...
      (corpus-string-10-10 score-10-10)))
)
```

in which no input substring fully includes any other input substring and the list of ten corpus substrings constitutes the procedure's choice of the ten "best" matches of the input substring by strings from the corpus.

Indexing and Special Cases of Output

Each word in corpus-string-i (with the exception of the members of a special list of "common" or "frequent" words) is referenced through three indices:

```
(file-index-i sentence-index-j word-index-k)
```

where word-index-i is the position of the word (actually, any member of the inflectional paradigm for that word) in the corpus sentence; sentence-index-j refers to the position of the sentence in the corpus file and file-index-k references the corpus file itself.
If a word does not appear in the corpus, an empty list is returned; If a word belongs to a special list of "frequent" words, a special symbol is returned since the corpus was, for efficiency reasons, not indexed for words that are too frequent.

METHOD

A sentence is broken into segments at punctuation marks or unknown words, and a list of all contiguous substrings ("chunks") of a segment is produced.
For every input chunk we look for sentences in the Spanish corpus that contain a matching substring. We assume a relaxed definition of matching in that we allow not ony complete matches but also matches in which

- there are **gaps** between words matching input chunk words, e.g., A X Y B Z C can match input chunk A B C
- the word **order** is different from that in the input chunk, e.g., B C A can match A B C
- a match is found only for a subset of the words in the input chunk, e.g., A D C can match input chunk A B C; (a different, shorter, chunk can eventually prove more appropriate in such a case)
- word correspondence is modulo members of the input word's inflectional paradigm e.g. **gatos** can match **gato**.

For each of the inexact matches, we calculate a penalty for the inexact match (see Nirenburg et al, 1993 for an earlier version of this approach).

Filtering Results

The findings of the candidate finding procedure are filtered to retain only matches whose match scores are above a certain threshold. Match scores are first calculated separately for each of the kinds of incomplete matches listed above. Then a cumulative score is produced. In our current system we set the threshold at 10 best matches. Note that this can be changed to include, for instance, all candidate matches with ratings above a certain threshold.

For efficiency reasons, we carried out the selection process in two stages. First, we filtered out matches that we considered clearly unacceptable. Thus, we did not allow any gaps of length 5 and higher. We expect in future to improve and modify this early filtering.

Scoring Candidates

The following heuristics guide the scoring process:

Unmatched Words Higher priority is given to sentences containing all the words in an input chunk. The penalty for unmatched words is presently set to 10. This penalty is applied for each instance of an unmatched word in the input.

Noise Higher priority is given to corpus sentences which have fewer extra words. The penalty for extra words in the corpus sentence is presently set to 5. This penatly is applied for each instance of an extra word in the corpus sentence.

Order Higher priority is given to sentences containing input words in the order which is closer to their order in the input chunk. The penalty for misordering is presently set to 15. This penalty is applied for each instance of word inversion.

Morphology Higher priority is given to sentences in which words match exactly rather than against morphological variants. If words match identically then no penalty is presently applied. If words match on morphological variants, then we consider whether the word is a content word or a frequently occuring function word. The penalty for morphological matches of content words is presently set to 2 and the penalty for morphological matches of function words is presently set to 1. The appropriate penalty is applied for each word match in the chunk. Note the possibility of false positives, as in the case of the English nominal plural and verbal third person singular - but these would be exact matches and would not be detectable unless the corpus and the input are tagged by parts of speech.

Summary of Intra-Language Matching

Test Type	Condition	Symbol	Penalty
Chunk Level	Identity		0
	Input chunk longer than corpus chunk	I	10
	Input chunk shorter than corpus chunk	G	5
	Word order differences between source chunk and inut chunk	O	15
Word Level: content words	Identity		0
	Morphological variants	M	2
	Part of speech difference	P	10
	Other	R	20
Word Level: others	Identity		0
	Morphological variants	V	1
	Part of speech difference	S	5
	Other	T	10

Combining Mismatch Evidence

Our initial combination of the results of intra-language matching tests was, therefore done according to the following formula:

> overall-chunk-score = 10I + 5G + 15O + 2M + 10P + 20R + V + 5S + 10T

which reflects our intuition that it is more important for the quality of a match for the words to retain their original order and have all input chunk words involved.
We expect to improve the above function both through statistical analysis with feedback and through further honing of heuristics.

Inter-Language Matching

Input: A set of quintuples {IC, CC, SLCS, TLCS, CFS}, where

IC stands for Input Chunk

CC stands for Corpus Chunk

SLCS stands for Source Language Corpus Sentence

TLCS stands for Target Language Corpus Sentence and

CFS is the score of the match, produced by Candidate Finder

IC	representa el nación
SLCS	las naciones unidas han representado todos naciones
TLCS	the united nations represent every country
CC	representado todos naciones
R	9.6

Objective

The goal of inter-language matching is, for each input quintuple, to extract from TLCS that part which is the translation of CC. For the example above, the answer should be:

```
 represent every country
```

Method: Overview

Inter-language matching is performed in several stages.
The major components of this process are:

- Getting English translations for words in SLCS;
- Stemming words in TLCS;
- Finding cross-linguistic correspondences between SLCS and TLCS at word level;
- Finding the longest TLCS substring that can possibly be the translation of CC, given the correspondence data; and
- Finding the "best" substring of the longest TLCS substring, using a special scoring metric.

Method: MRDing

First, all possible English translations of each word in SLCS were obtained from the Collins Spanish-English MRD and store it in a list ("eng-trans-of-spanish-words").
Words belonging to the list of "common" words are not looked up. This is done because we don't want spurious correspondences between common words to cause us to expand the set of possible cross-linguistic matches (see the GET-LONGEST-POSSIBLE-TRANS routine below). Matching on common words is, however, performed when evaluating the candidate alignments (see the SCORE-MATCH routine below).

```
eng-trans-of-spanish-words = NIL
                             nation country
                             unite unity
                             NIL
                             represent
                             every all
                             nation country
```

Method: Stemming

All possible root forms for each of the words in TLCS are obtained using the PC-KIMMO program, version 1.0 and stored in a list ("eng-root-forms"). We only need the root forms because the Collins dictionary contains root forms of translations of the Spanish words. "Common" words are still excluded.

```
eng-root-words = the
                 unite united
                 nation
                 represent
                 every
                 country
```

Method: Finding Cross-Linguistic Word Correspondences

The function used for this is:

```
find-correspondence
  (SLCS                         <1>
   eng-trans-of-spanish-words <2>
   eng-root-words               <3>
   TLCS)                        <4>
```

1. las naciones unidas han representado todas naciones
2. (nation country) (unite unity) NIL represent (every all) (nation country)
3. the (unite united) nation represent every country
4. the united nations represent every country

Method: Finding Cross-Linguistic Word Correspondences

The results are stored in a list of pairs (spa-eng-word-correspondences)

```
spa-word-correspondences =

naciones  nations
naciones  country
unidas  united
representado  represent
todas  every
naciones  nations
naciones  country
```

Utility: Indexing Cross-Linguistic Word Correpondences

For easy access, results of finding word correpondences are stored in a two (temporary) hash table, indexed by source and target language words, respectively.

```
(gethash spanish-word *spanish-corr-hash*)
    -> list of english words
(gethash english-word *eng-corr-hash*)
    -> list of spanish words
```

Thus,

```
(gethash "unidas" *spanish-corr-hash*)
    -> ("united")
(gethash "every" *english-corr-hash*)
    -> ("todas")
```

Also,

```
(gethash "naciones" *spanish-corr-hash*) ->
  ("nations" "country" "nations" "country")
```

During sub-sentential alignment we need to know if there are more than one possible alignments for a given word; this is easily calculated by looking at the length of the list returned from the hash table. It is inconsequential that a target language word was repeated twice in the last example above.

Finding “Longest Possible Translation”

The goal is to find the longest possible substring of the TLCS that could translate the CC.

```
longest-possible-trans =
     get-longest-possible-trans
        (CC
         TLCS
 spa-corr-hash
 eng-corr-hash)
```

Method: Finding "Longest Possible Translation"

First, we find the "smallest-possible-trans." This is the smallest SL substring which contains all the words that unambiguously correspond to words in CC

```
rightmost-pos = 0
leftmost-pos = 10000

for each word in CC
  begin
  eng-trans = gethash(word spa-corr-hash)
  unless length(eng-trans) > 1
         or
         length(eng-trans) = 0
    begin
    position-eng-word =
        find-pos(eng-trans TLCS)
    if position-eng-word < leftmost-pos
      then leftmost-pos =
           position-of-english-word
    if position-eng-word > rightmost-pos
      then rightmost-pos =
           position-of-english-word
    end
  end
```

Method: Finding "Longest Possible Translation" (Cont'd)

```
if leftmost-pos = 10000 then ;failure
  return NIL

smallest-possible-trans =
  get-substring (TLCS
                 leftmost-pos
                 rightmost-pos)
```

Method: Finding "Longest Possible Translation" II

At this stage, we add to the smallest possible substring all the TLCS words to the left and right for which we have not determined which SLCS word it translates. We stop adding on words when we come to a TLCS word that we know translates a SLCS word which is not in CC.

```
for i := leftmost-pos downto 1
  begin
   english-word = get-element(TLCS i)
   spa-word-corrs = gethash(english-word
                                eng-corr-hash)
   if spa-word-corrs = nil
      or
      string-intersection(spa-word-corrs CC)
   then
     leftmost-pos = i
  end
```

Method: Finding “Longest Possible Translation” (Cont’d.)

```
for i := rightmost-pos to length(TLCS)
  begin
   english-word = get-element(TLCS i)
   spa-word-corrs = gethash(english-word
                               eng-corr-hash)
   if spa-word-corrs = nil
      or
      string-intersection(spa-word-corrs CC)
   then
     rightmost-pos = i
  end

return get-substring(TLCS
                     leftmost-pos
                     rightmost-pos)
```

Finding Subsentential Alignment

We assume that the translation of CC is a substring of longest-possible-trans. At this point, we examine all substrings of longest-possible-trans, evaluate each according to a special preference metric and pick the best one(s).

```
best-score = 10000        ;a bad match score
best-translations = NIL
for each substring in
         longest-possible-trans
  begin
  score = score-match(substring
                      CC
                      spa-corr-hash
                      eng-corr-hash)
          + CFS
  if score < best-score then
    begin
    best-score = score
    best-translations = substring
    end
  else if score = best-score then
    begin
    best-translations =
          append(best-translations
                 substring)
    end
  end
return (best-score best-translations)
```

Method: Scoring Inter-Lasnguage Matches

The rate of how well a TLCS substring translates a CC is based on minimizing the estimated number of keystrokes needed to "reduce" each candidate translation to an "optimum" translation (presumably, determined by humans). Thus, the lower the score the better the translation.
The score-match function uses results of seven sperate heuristic tests. The seven values are then combined in a function whose coefficients (weights) are determined with the help of statistical training. The length of CCs also plays a role in determining the weights.

```
score =
    get-weight(0,len)        ;; a constant
    + test1-score * get-weight(1,len)
    + test2-score * get-weight(2,len)
    + test3-score * get-weight(3,len)
    + test4-score * get-weight(4,len)
    + test5-score * get-weight(5,len)
    + test6-score * get-weight(6,len)
    + test7-score * get-weight(7,len)
```

Scoring Inter-Language Matches: Tests

Test 1 returns the number of words in CC that correspond to a word in the TLCS substring. The more correspondences, the more confident we can be of the translation.

Test 2 returns the number of "content" words in CC that do not correspond to any words in the TLCS substring. Each such case lowers our confidence in the translation. We need to ignore common words here because they are not recorded in the correlation hash tables.

Test 3 returns the number of "content" words in TLCS substring that do not correspond to any words in CC.

Test 4 returns the smaller of test2-score or test3-score. The motivation is that if there are unmatched CC words (detected in test2) and unmatched TLCS substring words (detected in test3) then perhaps some of these match each other.

Test 5 attempts to take into account the "commmon" words. (The other tests were not able to do this since the word correlation data excluded common words.) It would be nice if we could return the number of common words that correlated exactly (i.e. *el* with *the*). However, as a **simplification,** we just assume each common word in CC will match one common word in TLCS substring

Scoring Cross-Linguistic Matches: Tests (Cont'd.)

Test 6 returns the number of "content" words in CC that do not correspond to any words in the TLCS substring, but do correspond to words in TLCS. This indicates that some sort of mis-alignment has occurred. Each such occurrence lowers our confidence in the translation.

Test 7 is a harbinger of using syntactic information to help the alignment process. So far, all we do is check whether the TLCS substring is either at the beginning or the end of the sentence. This type of test helps us to gather up "stray" words at the end that do not correspod to any SLCS element but should probably be included in the translation nonetheless. For example, consider

```
SLCS              = (a b c d e f g h i)
CC                = (a b c d)
TLCS              = (z a' b' c' d' m n o p)
TLCS substring    = (z a' b' c' d')
```

Even though **z** does not correspond to anything in SLCS, it quite probably may be a part of the translation of CC.

In the future, if syntactic processing is practical, we will use better defined syntactic features. For example, knowing the boundaries of constituents in all the strings could help cross-linguistic alignment.

Detailed Example: I

Input sentence:

Las inversiones en investigación y desarrollo fueron de 1.210 millones de francos suizos, el 14 por ciento del total de sus ventas.

Corpus facts: number of occurrences of words in corpus

inversiones	307
investigación	793
desarrollo	5074
francos	22
suizos	55
ciento	10
total	709
ventas.	97

Detailed Example: II

CHUNK 1

IC:

investigación y desarrollo

SLCS:

la capacidad end gena abarca más que la habilidad de realizar actividades de investigación y desarrollo, aunque un cierto nivel de investigación y desarrollo puede ser necesario incluso para estar al corriente de las tecnologías e ideas contemporáneas y utilizarlas sabiamente.

TLCS:

endogenous capacity includes more than the ability to conduct research and development, although a certain level of research and development may be necessary even for maintaining awareness of available technologies and ideas and using them intelligently.

Detailed Example III

word correspondences:

capacidad	capacity
capacidad	ability
abarca	includes
habilidad	ability
investigación	research
desarrollo	development
aunque	although
cierto	certain
nivel	level
puede	ability
necesario	necessary
incluso	even
ideas	ideas
utilizarlas	using

result:

1.7 research and development

Detailed Example IV

candidate finder penalties:

```
morph-variants: 0  0.0
words-out-of-order: 0  0.0
word number differences in candidate: 0  0.0
different common words: 0  0.0
```

alignment penalties:

```
constant offset:    10.2
t1: (2)             -7.1
t2: (0)              0.0
t3: (0)              0.0
t4: (0)              0.0
t5: (1)             -1.4
t6: (0)              0.0
t7: (0)              0.0
```

Detailed Example V

CHUNK 2

IC:

inversiones en investigación y desarrollo

SLCS:

las inversiones en investigación y desarrollo de esas tecnologías apropiadas, pese a sus muchas consecuencias para el bienestar de los pobres del mundo en desarrollo y, tal como ha reconocido la comisión brundtland, para el medio ambiente de todo el mundo * han sido penosamente inadecuadas.

TLCS:

investment in the research and development of such appropriate technologies as can have far - reaching impact on the well - being of the developing world's poor and, as the brundtland commission has recognized, on the whole world's environment, has been woefully inadequate.

Detailed Example VI

word correspondences:

investigación	research
desarrollo	development
pobres	poor
mundo	world
tal	such
reconocido	research, recognized
comisi n	commission
brundtland	brundtland
medio	environment
ambiente	environment
todo	whole

result:

6.3 investment in the research and development

Detailed Example VII

candidate finder penalties:

```
morph-variants: 0  0.0
words-out-of-order: 0  0.0
word number differences in candidate: 0  0.0
different common words: 0  0.0
```

penalties due to span-english alignment errors:

```
constant offset:    15.0
t1: (2)             -6.9
t2: (1)              2.2
t3: (1)              1.2
t4: (1)             -1.3
t5: (2)             -3.9
t6: (0)              0.0
t7: (0)              0.0
```